N° d'ordre: 2017 53 Étude comparative des codeurs CABAC et CAVLC et contribution à l'implémentation d'une technique de compression vidéo sans perte

Donia Ammous Damak

République Tunisienne
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Ecole Doctorale
Sciences et Technologies

Thèse de *DOCTORAT*

Génie électrique

N° d'ordre: 2017– 53

THESE

Présentée à

L'École Nationale d'Ingénieurs de Sfax

En vue de l'obtention du

DOCTORAT

Dans la discipline Génie Electrique

Par

Donia AMMOUS DAMAK
(Mastère en électronique option multimédia)

Étude comparative des codeurs CABAC et CAVLC et contribution à l'implémentation d'une technique de compression vidéo sans perte

Soutenu le 6 décembre 2017, devant le jury composé de :

M.	**Abdennaceur KACHOURI** (Professeur à l'ENIS)	Président
M.	**Mohamed Ali BEN AYED** (Professeur à l'ENET'COM)	Rapporteur
M.	**Adel M. ALIMI** (Professeur à l'ENIS)	Rapporteur
M^{me}	**Alima DAMAK MASMOUDI** (Maître de Conférences à la FSS)	Examinateur
M.	**Fahmi KAMMOUN** (Professeur à la FSS)	Directeur de Thèse

DEDICACE

Á mes chers

Je dédie ce travail

REMERCIEMENTS

Ce travail a été développé au sein de l'équipe Circuit et Système ***(C&S)*** *du Laboratoire d'Electronique et des Technologies de l'Information* ***(LETI)*** *dirigé par Monsieur le Professeur* ***Nouri MASMOUDI****.*

Mes sincères remerciements s'adressent à Monsieur : ***Fahmi KAMMOUN****, Professeur à la Faculté des Sciences de Sfax pour son aide précieuse, ses conseils bienveillants, sa direction, son soutien moral, ses encouragements illimités, ses compétences, ses grandes expériences et ses qualités humaines qui m'ont été d'une grande importance et qui m'ont permis de mener à bien ce travail. Qu'il trouve ici l'expression de mes profondes reconnaissances.*

Je tiens à exprimer mes hautes gratitudes et mes remerciements aux membres des jurys : Monsieur ***Abdennaceur KACHOURI****, Professeur à l'ENIS et membre de L'équipe* ***MEEM*** *(Microélectronique et Electronique Médicale) du laboraroire* ***LETI*** *(Laboratoire d'Electronique et des Technologies de l'Information), pour l'honneur qu'il me fait de présider le jury de ce mémoire.*

Monsieur ***Mohamed Ali BEN AYED****, Professeur à l'ENET'COM et membre dans l'unité de recherche (New Technologies and Telecommunication Systems Research Unit), d'avoir accepté de juger mon travail et d'y apporter leurs critiques constructives et leurs suggestions valeureuses.*

Monsieur ***Adel M. ALIMI,*** *Professeur à l'ENIS et directeur de recherche du laboratoire* ***REGIM*** *(Research Groups in Intelligent Machines), pour accepter d'être rapporteur de cette thèse.*

Madame ***ALIMA DAMAK MASMOUDI****, Maître de Conférences à la Faculté des Sciences de Sfax et membre de l'équipe* ***CIELS*** *(Computers Imaging Electronics and Systems Group) du laboraroire* ***CEMLab*** *(Advanced Control and Energy Management Laboratory) pour accepter d'examiner ce travail.*

Il m'est agréable de saisir cette occasion pour adresser mes sincères remerciements et ma gratitude la plus profonde à Monsieur ***Nouri MASMOUDI****, Professeur à l'ENIS, pour ses conseils et remarques constructives, ses critiques, son soutien moral, ses qualités humaines et son suivi incessant.*

Mes remerciements s'adressent ainsi à Madame ***Amina Kessentini Hachicha****, Assistant à l'ISIMG et mademoiselle* ***Naziha KHLIF*** *pour l'intérêt qu'elles ont porté à mon travail. Leurs remarques et leurs conseils utiles ont été très bénéfiques pour ma formation et pour la réussite de ce projet.*

Je remercie également tous les membres de l'équipe ***C&S*** *du laboratoire* ***LETI*** *pour leurs encouragements et pour leurs coopérations.*

Sommaire

Liste des figures

Liste des tableaux

Liste des abréviations

A

AVC : Advanced Video Codec.

AGSP : Accurate Gradient Selective Prediction.

APP : Approche Pyramidale Prédictive.

ASCII : American Standard Code for Information Interchange.

B

BL : Base Layer.

BPC : Bit Plane Coding.

BPL : Bit-Plane.

C

C&S : Circuits et Systèmes.

CIELS : Computers Imaging Electronics and Systems Group.

CEMLab : Advanced Control and Energy Management Laboratory.

CALIC : Context-based, Adaptive, Lossless Image Coding.

CABAC : Context-Adaptive Binary Arithmetic Coding.

CAVLC : Context-Adaptive Variable Length Coding.

CIF : Common Intermediate Format.

CU : Coding Unit.

D

DCT : Direct Cosine Transform.

DPCM : Differential Pulse Code Modulation.

E

EL : Enhancement Layer.

F

FRExt : Fidelity Range Extensions.

FLC : Fixed Length Code.

FRC : First-layer Residual Coder.

G

GAP : Gradient Adjust Predictor.

GLICBAWLS : Grey Level Image Compression By Adaptative Weighted Least Squares.

H

HEVC : High Efficiency Video Coding.

Huffyuv : Huffman –compressed YUV.

I

IWT : Integer Wavelet Transform.

ICT : Integer Cosine Transform.

J

JPEG-LS : Joint Photographic Experts Group-Lossless Standard.

JPEG : Joint Photographic Expert Group.

JVT : Joint Video Team.

L

LGM : Local Gradient Magnitude.

LPOSTC : Linear Prédiction Over Spatial Temporel Corrélation.

LAR : Locally Adaptive Resolution.

LETI : Laboratoire d'Electronique et des Technologies de l'Information.

M

MEEM : Microélectronique et Electronique Médicale.

MSU : Moscow State University.

MED : Median Edge Detector.

MPEG : Moving Picture Expert Group.

MSE : Mean Square Error, " Erreur Quadratique Moyenne ".

MSL : Most Significant Bit-plane Level.

O

OPTA : Optimum Performance Theoretically Attainable.

P

PNG : Portable Network Graphics.

PSNR : Peak Signal to Noise Ratio, "rapport crête signal sur bruit ".

PCM : Pulse code modulation.

PU : Prediction Unit.

R

REGIM : Research Groups in Intelligent Machines.

RCT **:** Reversible Color Transform.

RO : Rate Optimization.

RIPC : Residual Image Prediction and Coding.

S

SAO : Sample Adaptive Offset.

SRC : Second-layer Residual Coder.

SAP : Sample Angular Prediction.

SWP : Sample-based Weighted Prediction.

SELC : Sample-based weighted prediction for Enhancement Layer Coding.

T

TRC : Two-layered residual coding.

V

VLC : Variable Length Code.

Introduction Générale

Pour une bande de transmission ou de mémoire limitée, la vidéo numérique aborde beaucoup de problèmes pour le traitement ou le stockage. Ceci contribue à une demande plus élevée de compression vidéo.

La compression vidéo fait partie intégrante de plusieurs domaines d'applications tel que la télévision par satellite, l'imagerie médicale (télémédecine), la vidéo conférence, l'archivage de vidéo ou de recherche scientifique et la sécurité vidéo. Il existe deux types de compression: sans perte (Lossless) et avec perte (Lossy).

La compression sans perte : (compactage ou compression conservative)

La compression sans perte consiste à réduire la taille des données sans les dégrader. La suite de bits obtenue après la compression et la reconstitution est strictement identique à l'originale. Il n'y a aucune perte dans l'information. Cette dernière est seulement réécrite d'une manière plus concise. Les algorithmes de cette catégorie ne garantissent pas la réduction de la taille de toutes les données présentées à l'entrée. En d'autres termes, il peut y a voir des données qui ne sont pas comprimées (c'est-à-dire une forme de redondance non supprimée).

La compression avec perte : (compression irréversible ou compression non conservative)

La suite de bits obtenue après la compression et la reconstitution est différente de l'originale, mais l'information reste sensiblement la même. Elle est réservée aux données «perceptibles», en général sonores ou visuelles, qui peuvent subir une modification, parfois importante, sans que cela ne soit perceptible par un humain. La perte d'information est irréversible. Il est impossible de retrouver les données d'origine après une telle compression. Pour cela, la compression avec perte est appelée compression non conservative.

Sur les images fixes, Les deux techniques de compression ci-dessus ont été appliquées. En fait, les normes telles que JPEG (Joint Picture Expert Group)et JPEG2000 (J2K) offrent les deux possibilités. La norme JPEG2000 utilise l'ondelette 9/7 de Daubechies pour une compression avec perte et l'ondelette 5/3 de Le Gall pour le cas conservative. La norme PNG (Portable Network Graphics) effectue également une compression sans perte.

Sur les séquences vidéo, les techniques et les normes de compression des séquences vidéo font généralement partie de la deuxième catégorie, permettant un taux de compression relativement élevé. Les travaux dans ce domaine sont nombreux et les techniques existantes sont relativement puissantes et normalisées. Cependant, il y a eu un regain d'intérêt pour un certain nombre d'utilisateurs des techniques conservatives.

Le choix de la compression appropriée est effectué selon le besoin de l'utilisateur et le domaine d'applications. La compression vidéo doit s'adapter aux nombreux applications et

aux multiples besoins telle que la qualité requise, la gamme de débit et la nécessité d'une bande passante limitée.

Le but de ce travail est constitué de deux parties :

Notre but consiste à cerner les différentes méthodes de compression vidéo sans perte qui existent dans la littérature, à étudier leurs faisabilités, à analyser leurs performances, à modifier les paramètres de codage de l'une de ces approches (celle de Wei-Da Chien et AL) afin d'améliorer ses résultats en termes du taux de compression et à contribuer à implémenter un algorithme de compression de séquences vidéo sans perte en vue d'atteindre des taux de compression intéressants.

La deuxième partie de la thèse est consacrée à une étude comparative entre les codeurs entropiques CABAC (Context-Adaptive Binary Arithmetic Coding) et du CAVLC (Context-Adaptive Variable Length Coding) sur l'algorithme JM18.3. Cette étude s'intéresse principalement au taux de compression et à la complexité du calcul.

Ce manuscrit comporte trois chapitres. Dans le premier chapitre, nous présentons un état de l'art des différentes méthodes de compression vidéo sans perte. Ces méthodes sont variées et sont basées sur des principes et des schémas différents de ceux de la compression avec perte.

Dans le deuxième chapitre, nous introduisons les différents résultats d'une évaluation comparative entre les codeurs CABAC et CAVLC et nous décrivons en détail le traitement de nos travaux de modification des paramètres du logiciel JM de la norme H.264/AVC (Advanced Video Codec).

Dans le dernier chapitre, nous expliquons la conception et la mise au point d'une nouvelle technique de compression basée sur l'approche hiérarchique (à deux couches) par réarrangement des coefficients résiduels à l'entrée du codeur entropique.

Ce rapport s'achève par une conclusion dans laquelle nous donnons une vision sur des travaux avancés.

CHAPITRE I

Compression sans perte

1. Introduction

La vidéo numérique prend une place de plus en plus importante dans notre vie quotidienne. Actuellement, nos téléphones portables sont capables d'acquérir, de stocker et de transmettre des images numériques. Les médecins utilisent de plus en plus les images, sous différentes formes, pour réaliser des diagnostics plus fiables et plus précis. Les satellites nous envoient des images dont la qualité et la résolution ne cessent de croître. Ces applications utilisent un ensemble de techniques qui permettent de manipuler et de traiter ces images. Ces techniques sont basées entre autres sur la compression vidéo.

2. Outils pour la compression: Rappel de la théorie de l'information

2.1. *Définition (Source d'information simple, loi de probabilité)*

On suppose qu'un signal est la suite de N réalisations d'une variable aléatoire discrète X, appelée source d'information simple. X prend ses valeurs$\{x_0, x_1, x_2, \ldots.. x_{S-1}\}$ de S symboles et suit une certaine loi de probabilité imposée.

$$\{p(x_i) = Prob(X= x_i)=p_i \ , \forall i = 0 \ldots S-1\} \quad (1)$$

La loi de probabilité indique la fréquence d'apparition du symbole x_i dans le signal, c'est-à-dire le rapport entre le nombre d'occurrences de x_i dans le signal et le nombre N de symboles dans le signal.

2.2. *Mesure de l'information : l'entropie*

2.2.1. *Définition de l'information propre d'un symbole x_i*

C'est la quantité d'information amenée par l'apparition du symbole x_i. Elle indique le nombre de bits minimum qu'il faut en moyenne pour stocker un pixel de l'image, c'est-à-dire qu'elle mesure l'information utile présente dans un message. Alors l'information propre du symbole x_i est :

$$I(x_i) = log_2 \frac{1}{p(X_i)} \quad (2)$$

On remarque que plus la probabilité du symbole x_i est petite, plus la quantité d'informations propre sera grande (Plus un symbole est probable, moins il amène d'information).

2.2.2. *Définition de l'entropie d'une source X*

L'entropie H d'une variable aléatoire X de S symboles $\{x_0, x_1, x_2, \ldots\ldots x_{S-1}\}$ et suivant une loi de probabilité $p(x_i)$ pour chaque symbole, est donnée par :

$$H(X) = \sum_{i=0}^{S-1} p_i I(x_i) = -\sum_{i=0}^{S-1} p_i \log_2(p_i) \quad bit/symbole \tag{3}$$

L'entropie d'un message est la somme des entropies des symboles de ce message. Elle est d'autant plus grande que la quantité d'informations est importante.

H(X) est la limite inférieure du nombre moyen de bits nécessaires au codage d'un signal, appelée OPTA (Optimum Performance Theoretically Attainable). L'objectif est de se rapprocher de la limite théorique de compression sans pertes, donnée par l'entropie de la source X[1].

3. Les types de compressions

La compression consiste à réduire la taille physique des données en retirant la redondance qui existe entre ses éléments (voir Figure 1). Cette opération permet de réduire l'espace mémoire de stockage et les temps de lecture et de transmission, ce qui permet le fonctionnement en temps réel. Il existe deux types de compression: avec perte et sans perte: Le choix de la méthode est lié à la nature des données à traiter.

Figure I. 1. *Compression\ décompression des données*

4. Les types des artefacts de codage

Les algorithmes de compression avec perte utilisés dans diverses normes de codage vidéo sont assez semblables. La plupart d'entre eux comptent sur la compensation de mouvement et la transformé DCT à base de bloc avec quantification ultérieure des coefficients et aux erreurs de calcul sur la transformation non réversible.

De nombreux artefacts de codage qui sont introduits dans les séquences vidéo comprimées, donc, la vidéo est altérée par un ou plusieurs artefacts de compression, comme l'effet de bloc,

le flou, les saignements de couleur, des bourdonnements, les faux bords, le mouvement déchiqueté et la chrominance vacillante [2, 3, 4, 5] :

L'effet de bloc se réfère à une configuration de bloc dans la séquence compressé. Elle est due à la quantification indépendante de blocs individuels (généralement de 8×8 pixels) dans des systèmes de codage DCT à base de blocs, conduisant à des discontinuités aux frontières de blocs adjacents. L'effet de bloc est souvent la distorsion visuelle la plus importante dans une séquence comprimé en raison de la régularité et de l'étendue du motif. C'est probablement la distorsion la plus étudiée. L'effet de bloc est habituellement défini comme le résultat des discontinuités aux frontières de blocs adjacents dans une image reconstruite, car il est fréquent de réduire la mesure en prenant seulement en compte les pixels aux limites de bloc.

L'effet de flou consiste en une perte de précision spatiale et une réduction de la netteté des bords sur les régions d'activité spatiales élevées de images, tels que dans les zones plus ou moins texturées ou autour objets de la scène bords. Elle est due à la suppression des coefficients de hautes fréquences par une quantification grossière pour des macro-blocs intra-codés. Pour des macro-blocs codés inter-trames, est principalement la conséquence de l'utilisation d'un macro-bloc prédit à un manque de détails spatiaux.

Le saignement de couleur est l'effet de flou sur les informations de chrominance. Il en résulte dans le barbouillage de couleurs entre les zones de chrominance fortement différentes. Elle est due à la suppression de coefficients de hautes fréquences des composantes de chrominance, et en plus de sous-échantillonnage de la chrominance, le saignement de la couleur se prolonge sur tout un macro-bloc.

L'Effet d'escalier peut apparaitre sur les contours obliques. Cet effet est dû au fait que les fonctions de base de la DCT sont plus adaptées à la représentation des contours horizontaux et verticaux. Les contours, dont l'orientation n'est ni horizontale ni verticale, nécessitent davantage de coefficients hautes fréquences pour obtenir une représentation précise. La quantification trop importante de ces coefficients introduit des irrégularités sur ces contours obliques (*jagged*). Après quantification grossière, la troncature des contributions se traduit par l'échec de reconstruire les bords en diagonale. Ainsi, ces bords sont convertis dans le sens horizontal ou vertical. La forte quantification de ces coefficients provoque l'apparition de lignes obliques et déchiqueté.

L'effet d'ondulation (*ringing*) est fondamentalement associé au phénomène de Gibbs (oscillation de reconstruction d'un signal discontinu avec une somme de signaux continus). Il est donc plus marqué sur les contours à fort contraste que sur les zones uniformes. Cet effet

est un résultat direct de la quantification provoquant des irrégularités dans la reconstruction. L'effet de *ringing* se produit à la fois sur la luminance et sur la chrominance.

L'effet de mosaïque Ceci est une conséquence de l'inégalité de représentation dans le dictionnaire, les blocs homogènes (présentant une faible activité) sont dominants. Le résultat visuel de cet effet est une déformation des contours de l'image.

Les faux contours résultent de la quantification directe des valeurs de pixels. Il se produit dans les zones à variation lente de luminance et chrominance en douceur d'images contenant une transition graduelle de la valeur des pixels sur une surface donnée. Cet artefact est directement attribué soit à un nombre insuffisant de niveaux de quantification pour la représentation de la zone, ou de leur mauvaise répartition.

Les contours fantômes sont une conséquence du transfert par compensation de mouvement de la discontinuité de la frontière d'un bloc, induite par un effet de bloc, depuis une image de référence vers une image prédite.

Le mouvement saccadé (*jagged motion*) peut être dû à une mauvaise estimation de mouvement. Les estimateurs de mouvement fonctionnant par bloc sont plus efficaces lorsque les mouvements de tous les pixels d'un macro-bloc sont identiques. Lorsque l'erreur résiduelle d'estimation de mouvement est importante, elle peut être quantifiée grossièrement.

La quantification de mouvement est souvent réalisée seulement sur la luminance et le même vecteur est utilisé pour les composantes de chrominance. Il peut en résulter une «*chrominance mismatch*» sur un macro-bloc.

L'effet de moustique « mosquito » est une distorsion temporelle observée principalement sur les zones faiblement texturées comme des fluctuations de la luminance (ou de la chrominance) autour des contours à fort contraste ou des objets en mouvement. Il est la conséquence de la variation du choix des paramètres de codage d'une même zone d'une scène dans les images successives d'une séquence.

Le papillotement (*flickering*) apparait principalement dans les zones texturées. La texture des blocs de ces zones est compressée avec des pas de quantifications variant au cours du temps, ce qui a pour effet de créer des papillotements dans ces zones.

Autre source importante des dégradations est la transmission du flux de bits sur un canal bruité. Deux types de déficiences différentes peuvent se produire. Les paquets peuvent être corrompus et jetés, ou peuvent être retardés et ont perdu la tranche pour le décodage. Les pertes peuvent affecter soit le contenu, provoquant une sorte d'erreur qui peuvent être transmises sur des trames consécutives; ou de données globale qui est encore plus dommageable.

Mis à part les artefacts de compression et les erreurs de transmission, la qualité de séquences vidéo numériques peut être affectée par n'importe quel stade de pré- ou de post-traitement dans le système comme des conversions entre le numérique et le domaine analogique; le sous-échantillonnage de chrominance; la conversion de fréquence d'images.

5. Utilités de la compression vidéo sans perte

On utilise la compression sans perte pour avoir une haute qualité de multimédia. L'utilisation de la technique de la compression sans perte est présente dans nombreuses applications telles que la recherche scientifique, l'archivage des films, le cinéma numérique, les œuvres d'art, les services militaires (fabrication d'armes, trace des avions militaires, radar militaire) et l'imagerie médicale (télémédecine) (voir Figure 2).

Figure I. 2. Diverses applications de la compression sans perte

6. Utilités de la compression vidéo sans perte dans l'imagerie médicale

Une image médicale non compressée prendra plus de temps pour se transférer dans un réseau car le stockage des données volumétriques pour diagnostics prend une mémoire énorme. Face à l'augmentation des données en imagerie médicale, une diminution de la taille des données par compression vidéo est nécessaire. Cette dernière permet en effet de minimiser l'espace de stockage ou l'archivage des données et de réduire le temps de transfert des données et la largeur de bande dans le cas de traitement à distance (télémédecine). Mais, la compression vidéo introduit des distorsions dans le contenu des données des rapports médicaux. D'où, cette solution n'est pas satisfaisante, Nous devons compresser l'image avec un nombre réduit de bits, sans déformer et dégrader la qualité de l'image originale. De plus, lors de la transmission, le bruit canal peut aussi introduire des erreurs qui risquent d'erroné le diagnostic [6, 7, 8, 9, 10].

Pour stocker et transmettre des rapports d'imagerie de patients avec garantir l'intégrité des données, nous utilisons les techniques de compression sans perte. En effet, le médecin exige la haute précision et la haute qualité des images pour prendre la décision exacte du diagnostic. Dans certains cas, la faute décision du docteur de la maladie entraînera un effet dangereux pour le patient. Par exemple, pour le cancer de l'endomètre utérin au stade de début parfois ne sera pas visualisé par une sonde échographie pelvienne, il faut une sonde endovaginale ou un exéma endoscopique (hystéroscopie). Pour les tumeurs non infiltrant, l'aspect échographique peut être tout à fait similaire à celui observé dans l'hyperplasie de l'endomètre (voir Figure 3).Il faut plutôt un appareil plus sophistiqué (échographie 3D).

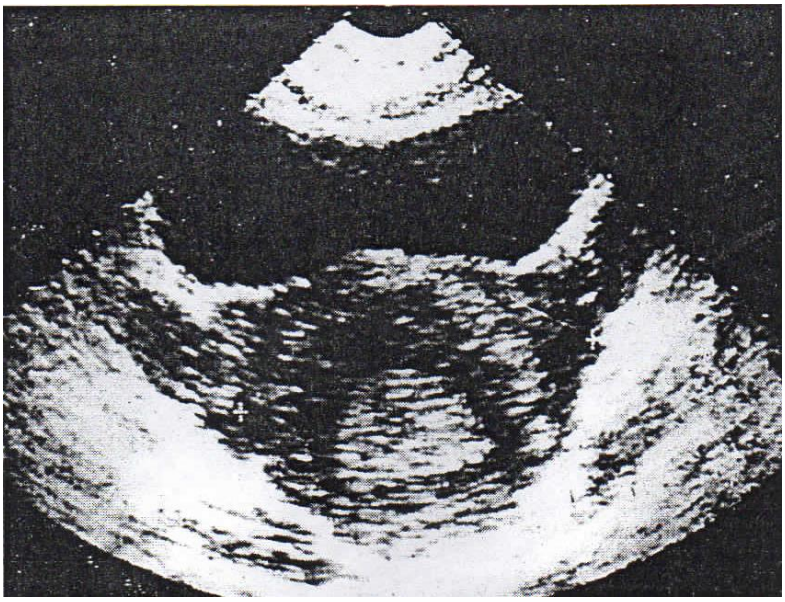


Figure I. 3. Tumeur bénigne ou Hyperplasie de l'endomètre

C'est pour cela, la netteté d'images facilite le diagnostic de la maladie. Le médecin lorsqu'il trouve l'image floue. Il refait la fibro ou il émet un colorant pour colorer la lésion. Il y a plusieurs types d'appareils de fibroscopie et d'échographie (Pentax, Olympics, LOGI C5, SIUI, Mindray) qui sont menés des différentes versions de caméra. Donc, il faut un moyen d'exploration de haute précision pour ne pas erroné le diagnostic d'une maladie.

La technologie de la télémédecine est étendue dans plusieurs hôpitaux. Par exemple, lorsque deux médecins font une réunion à distance (communication sans fil), ils permettent tous les deux les informations (vidéo (c'est-à-dire fibro)) de la passion. Par conséquent, ces données (quel que soient fibro, écho, etc.) vont subir des pertes lors la transmission et du stockage (au niveau du flash disk : les données seront compressées). On obtient donc une dégradation des qualités des images (images floues).

Les applications (la fibroscopie bronchique, la fibroscopie " gastro ", nasofibroscopie, l'échographie, la coloscopie virtuelle (détail très fin à l'échelle de mm) et l'imagerie en coupe " IRM, scanner " utilisent des instruments médicaux avancés afin de faire un bon diagnostic de la maladie. Certains instruments médicaux contiennent un appareil nommé sonde, ces sondes sont menues d'une caméra. Ces deux images sont captées de deux types d'appareils d'échographie (LOGI C5, SIUI) (voir Figure 4). La différence de la clarté des deux images est due à la version d'appareils utilisés plutôt à la caméra utilisée.

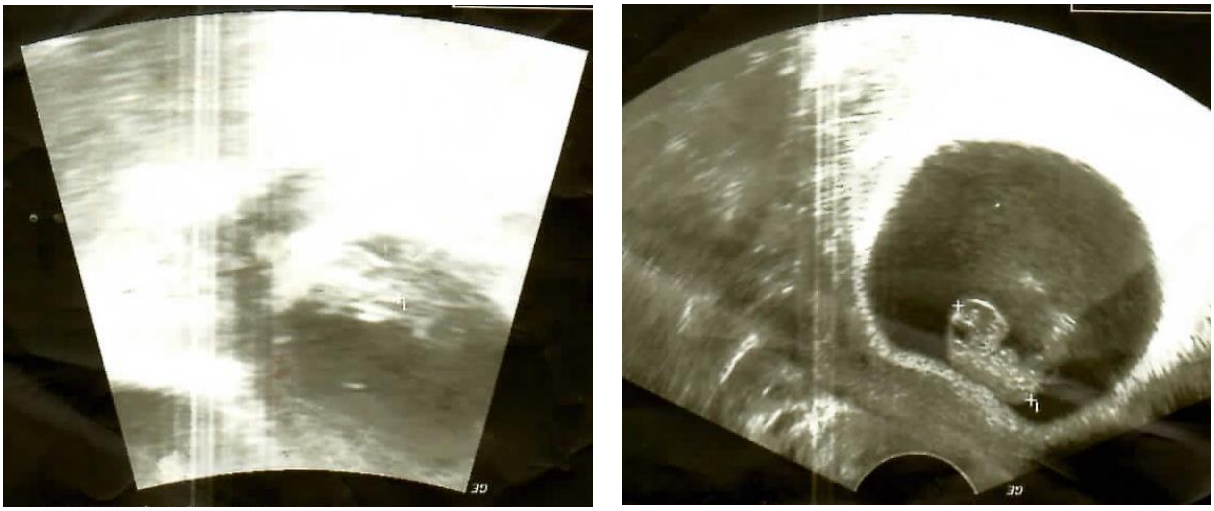


Figure I. 4. Deux images captées de deux types d'appareils d'échographie

Dans un workshop médical ou dans une salle d'opération, le médecin explique les détails de l'opération aux résidanats (voir Figure 5). La clarté de la vidéo joue un rôle important surtout lorsque l'opération est très délicate et il faut observer les organes à l'échelle de mm.

Figure I. 5. Les conditions de déroulement de l'opération

Par exemple, cette opération est effectuée sur la main gauche. Une plaie ouverte de la main gauche entrainant une section de deux tendons et une section d'un nerf. Une suture pratiquée sous microscope aboutit à la réparation des deux tendons et un nerf (voir Figure 6).

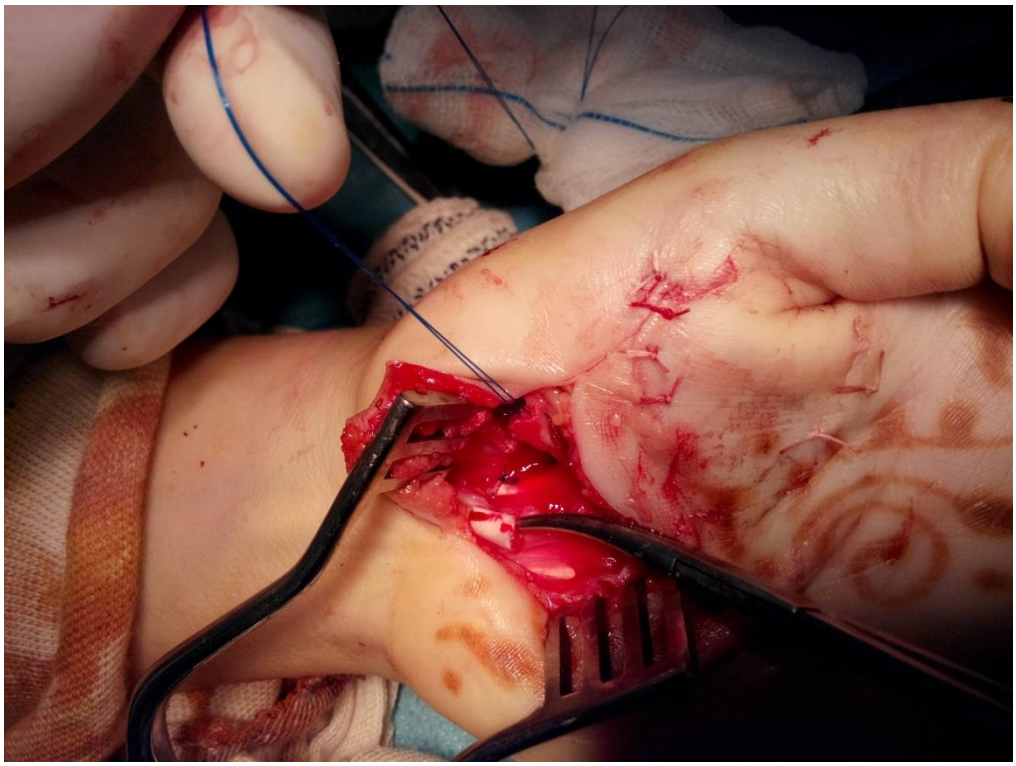


Figure I. 6. Une étape d'opération

7. Classification des méthodes de compression sans perte

Les méthodes de compression sans perte sont évaluées par leur taux de compression et par leur temps d'exécution. Il existe trois grandes familles de méthodes de compression vidéo sans perte (Figure 7). La première famille [11-15] comprend des méthodes qui sont gardées secrètes à des buts commerciaux. La seconde [16-18, 22-24] inclut une variété de méthodes innovantes, appelées aussi non conventionnelles, tandis que la troisième [25 –30] se compose d'un ensemble de méthodes basées sur des algorithmes et des normes liées à la compression avec perte d'images fixes telles que JPEG-LS (Joint Photographic Experts Group-Lossless Standard) et JPEG2000, et des séquences vidéo comme la norme H.264/AVC et HEVC (High Efficiency Video Coding).

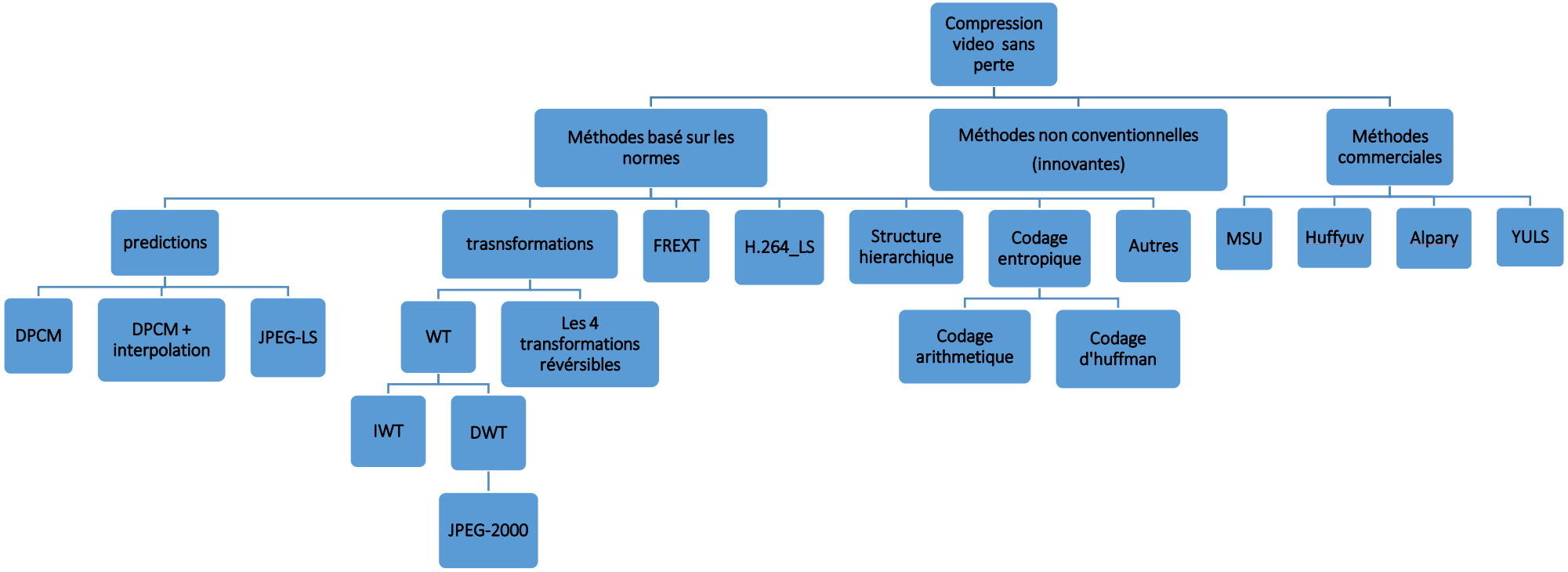


Figure I. 7. *Les techniques de compressions vidéo sans perte*

7.1. *Méthodes non conventionnelles*

7.1.1. *La compression vidéo sans perte basée sur l'ondelette en utilisant une prédiction adaptative et le mouvement de rapprochement*

Cette méthode a été présentée par Simon Jeyakumar et S.Sundaravadivelu en août 2009 [16].
Le schéma de principe de la méthode proposée est illustré dans la Figure suivante :

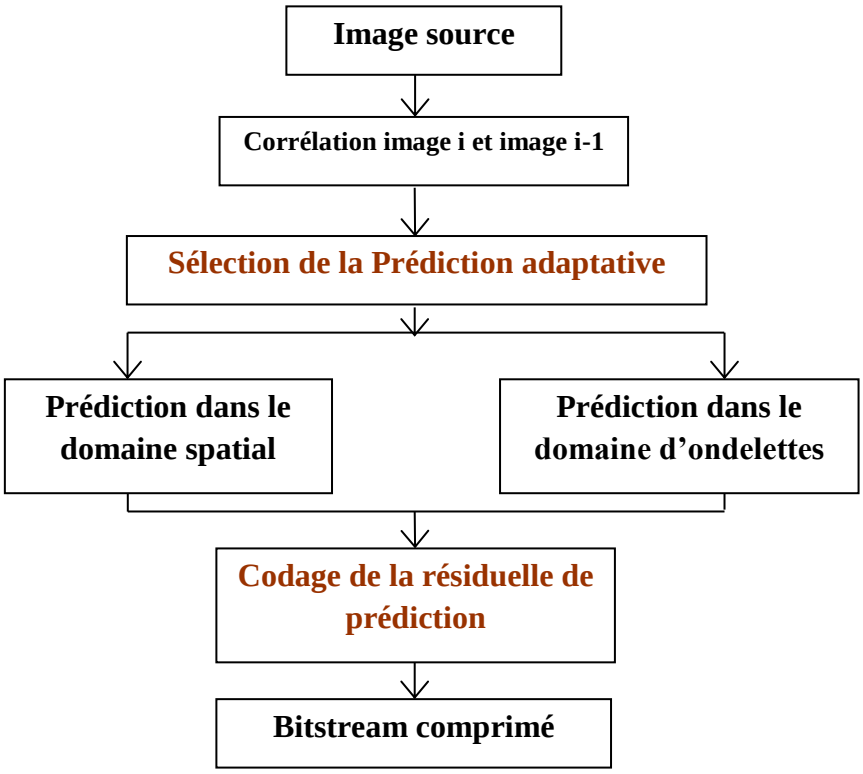


Figure I. 8. *Diagramme de la méthode proposée*

Cette technique de compression sans perte des séquences vidéo en utilisant un codage prédictif vise à améliorer l'efficacité de compression. L'algorithme proposé est un procédé d'adaptation permettant de basculer entre la prédiction dans le domaine spatial et la prédiction dans le domaine des ondelettes en fonction de la corrélation temporelle.
La quantité de la redondance temporelle est estimée par le coefficient de corrélation d'inter-frame de la séquence vidéo. Le coefficient de corrélation d'inter-frame $C_{n,n-1}$ entre les frames "n" et "n-1"peut être calculé par l'équation suivante:

$$C_{n,n-1} = \frac{\sum_x \sum_y \left(i_n(x,y) - \overline{i_n}\right) \left(i_{n-1}(x,y) - \overline{i_{n-1}}\right)}{\sqrt{\left(\sum_x \sum_y \left(i_n(x,y) - \overline{i_n}\right)^2 \sum_x \sum_y \left(i_{n-1}(x,y) - \overline{i_{n-1}}\right)^2\right)}} \quad (4)$$

où x×y est la taille d'une image, i_n est la valeur de pixel de l'image courante à la position (x,y) et i_{n-1} est la valeur de pixel de l'image de référence à la position (x,y).

Si le coefficient de corrélation d'inter-frame est plus petit que le seuil prédéfini de 0,99, la séquence est susceptible d'être une séquence vidéo de haut mouvement. Dans ce cas, la compensation de mouvement et le codage de la partie résiduelle seraient inefficaces dans le domaine d'ondelettes et par conséquent il est sage d'agir sur la séquence vidéo dans le domaine spatial.

Dans le cas contraire, si le coefficient de corrélation d'inter-frame est plus grand que le seuil prédéfini de 0,99, les images ont plus de similitudes entre elles avec très peu de mouvement, elles sont codées en utilisant la prédiction temporelle dans le domaine d'ondelettes entières. Toutes les étapes précédemment illustré dans la Figure ci-dessus sont obtenues par la réalisation de l'algorithme suivant :

```
Begin
Read image Frame i and Frame i-1
Correlate Fame I and Frame i-1
If Correlation Coefficient > 0.99 then
Apply S wavelet transform to each frame
Divide the frames into macro blocs of
Equal size m x n
For each bloc in Frame i,
Set search range as –W to + W in Frame i-1
Map the bloc of Frame i with respective
Non overlapping blocs of Frame i-1
Calculate minimum abs (difference)
Motion vector = (x,y) positions of the bloc
Whose MAD is minimum
Predict motion vector of (x,y) using motion
Vectors of left-upper neighboring
Pixels (x-1,y), (x-1,y-1),(x,y-1),(x+1,y-1)
Calculate MV residual= Actual MV – Predicted MV
Calculate Frame residual = Actual pixel value– Predicted pixel value
Encode the error residuals
End for
End
```

7.1.2. *La prédiction linéaire à l'aide de la projection 3D pour la compression vidéo sans perte*

Cette méthode a été proposée par Daejung Bang, Haijiang Tang et Sei-ichiro Kamata du "Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan" en juillet 2009 [17].Le but de cette approche est d'augmenter l'efficacité de la prédiction pour la compression vidéo sans perte. La méthode proposée est une prédiction à trois dimensions, étendue de la méthode conventionnelle LGM (Local Gradient Magnitude), qui utilise des gradients spatio-temporels. Le gradient spatio-temporel est une donnée spatiale qui résulte de la projection du prisme triangulaire composé de pixels voisins.

7.1.2.1. Prédiction à deux dimensions

GAP (Gradient Adjust Predictor) de CALIC (Context-based, Adaptive, Lossless Image Coding) et AGSP (A Gradient Spatio-Temporal) sont des techniques de prédiction bidimensionnelle. CALIC prédit en utilisant GAP qui est basé sur la relation entre le pixel courant x et les pixels voisins (Figure 9).

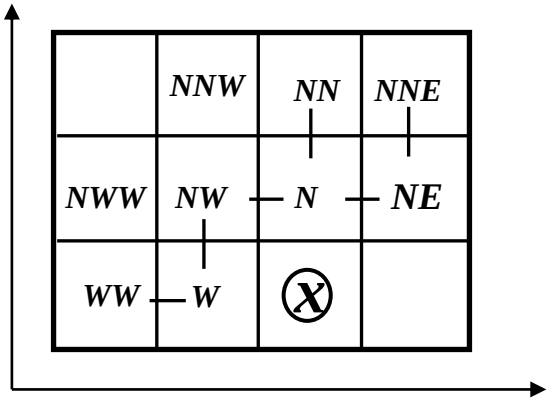


Figure I. 9. *Les pixels voisins du pixel courant « GAP »*

GAP est évalué en utilisant les gradients de luminance du pixel courant x et des pixels voisins. Les gradients verticaux d_v et les gradients horizontaux d_h de GAP sont comme suivant :

$$d_v = |W - NW| + |N - NN| + |NE - NNE| \quad (5)$$

$$d_h = |W - WW| + |N - NW| + |N - NE| \quad (6)$$

7.1.2.2. Prédictions à trois dimensions (LGM)

Le gradient spatio-temporel tridimensionnel est adopté pour améliorer les méthodes conventionnelles de compression d'image telles que GAP et MED (Median Edge Detector) qui sont des prédictions bidimensionnelles basées sur des gradients horizontaux et verticaux.

Le terme « spatio-temporel » signifie le domaine à trois dimensions qui est constitué par le domaine d'image à deux dimensions et le temps de la vidéo, c'est pour cela qu'on peut définir la vidéo comme un ensemble d'images fixes dans un ordre chronologique. Vu que la relation entre les images est liée au temps, le domaine de la vidéo est présenté comme x, y, t. LGM utilise six pixels n, w, nw, n', w' et nw' pour estimer le pixel cible x (voir Figure 10).

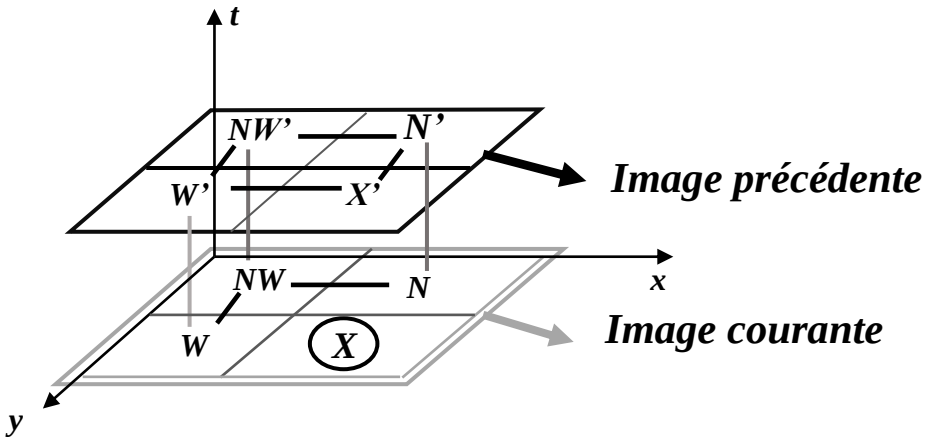


Figure I. 10. *Structure de LGM*

La méthode habituelle pour estimer un LGM dans une dimension est de calculer la somme des valeurs absolues des différences entre les pixels choisis le long de cette dimension, comme dans CALIC.

7.1.2.3. Prédiction linéaire utilisant une projection 3-D

La méthode proposée se base sur la prédiction en utilisant la projection 3-D qui améliore la technique conventionnelle (LGM) [17]. Le pixel cible est fortement corrélé avec les pixels voisins. Pour prédire ce pixel à l'aide de cette corrélation optimale, On transforme l'information des pixels à 3D en 2D en utilisant une projection. En effet, les données des pixels voisins 3-D sont obtenues à l'aide d'une projection et par la suite ces données sont exploitées pour la prédiction du pixel cible. Pour l'obtention des données dans 2D, la méthode proposée divise la structure 3D aux trois prismes triangulaires (prisme triangulaire 1, 2 et 3) comme dans la Figure 11.

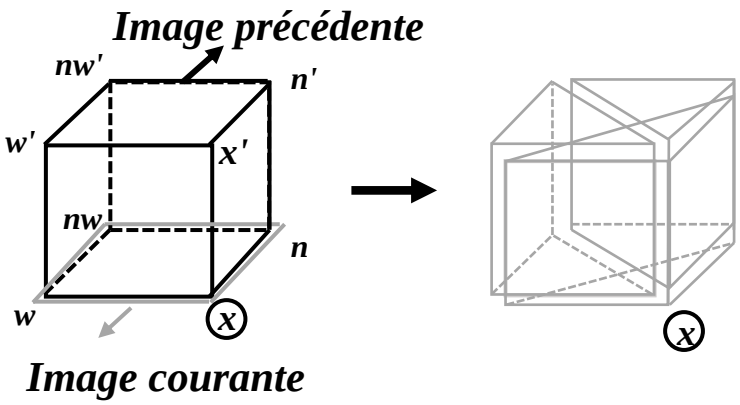


Figure I. 11. *La division spatiale par l'algorithme proposé*

Ensuite, elle fait la projection de chaque prisme triangulaire afin d'obtenir les données des pixels voisins comme dans la Figure 12. Par conséquent, la valeur du pixel cible est le résultat du calcul du gradient entre les pixels voisins.

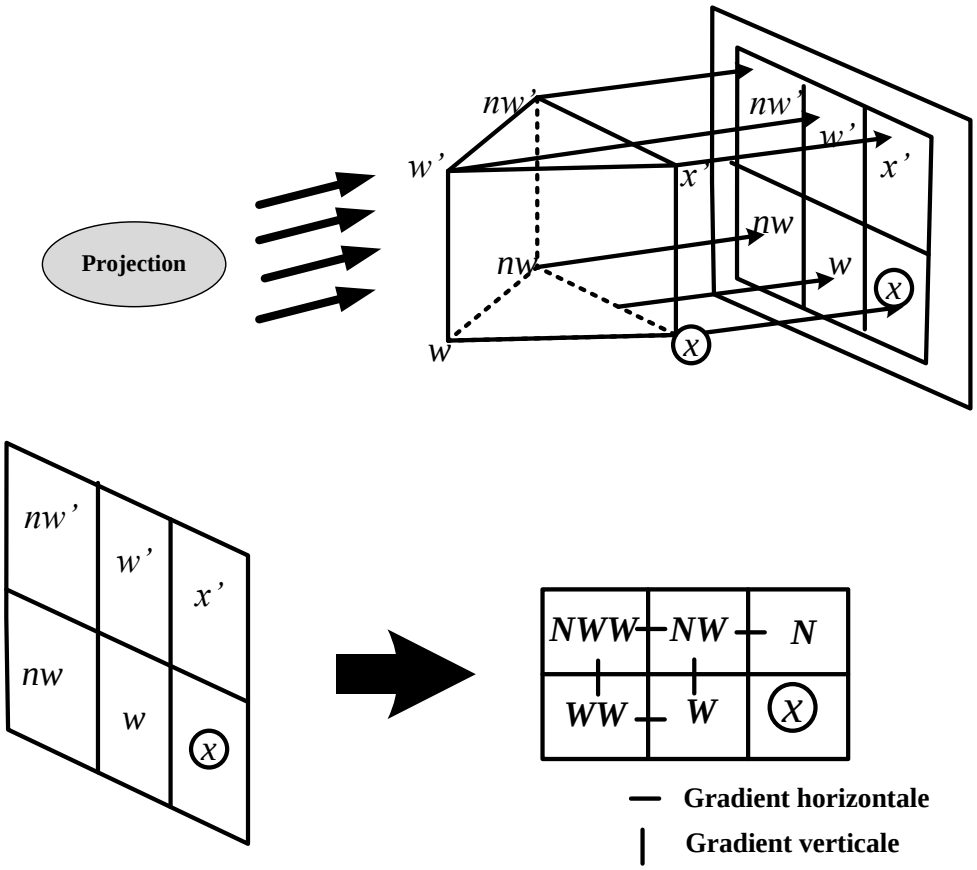


Figure I. 12. *Projection du prisme triangulaire 1 et mise en œuvre des pixels voisins du pixel cible*

Dans la Figure 12, le pixel cible est évalué par la corrélation des pixels voisins. Tout d'abord, les valeurs absolues de la différence entre les pixels de voisinage sont calculées et sommées suivant les expressions :

$$G_h = |NW - N| + |WW - W| + |NWW - NW| \quad (7)$$

$$G_v = |NWW - WW| + |NW - W| \quad (8)$$

7.1.3. *La compression vidéo sans perte utilisant une prédiction optimale spatio-temporelle*

Cette méthode a été suggérée par Stefano Andriani, Giancario Calvagno et Gian Antonio Mian du département d'Ingénierie d'Informations, Université de Padova Italie et publié en Septembre 2005 [18]. Cette méthode est basée sur la prédiction optimale qui exploite la corrélation temporelle. Cette solution a apporté une amélioration du taux de compression par rapport aux méthodes qui l'ont précédé, mais l'algorithme nécessite beaucoup de calcul. En conséquence, une méthode alternative est proposée pour réduire la complexité globale du calcul.

7.1.3.1. Algorithme de GLICBAWLS

GLICBAWLS est un acronyme de "Grey Level Image Compression By Adaptative Weighted Least Squares". C'est un codeur basé sur la prédiction, présenté par Meyer et Tischer. Son code source est le langage C [19-21].

7.1.3.2. Principe de GLICBAWLS

Cet algorithme est basé sur la théorie de la prédiction optimale et le calcul du coefficient de prédiction qui est le noyau de l'algorithme de GLICBAWLS (voir la Figure 13).

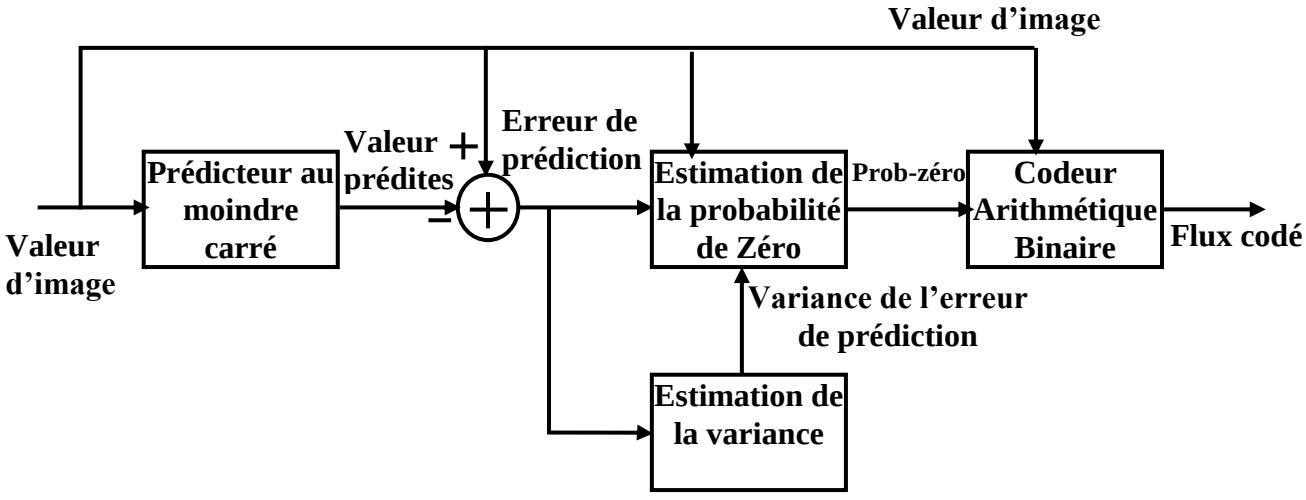


Figure I. 13. *Schéma de description du codeur GLICBAWLS*

GLICBAWLS construit la prédiction linéaire optimale du pixel courant à la position (x_c,y_c) en utilisant les six pixels causaux voisins avec la distance de Manhattan inférieure ou égale à deux (l'algorithme de GLICBAWLS est désigné sous le nom de P6), voir Figure 14.

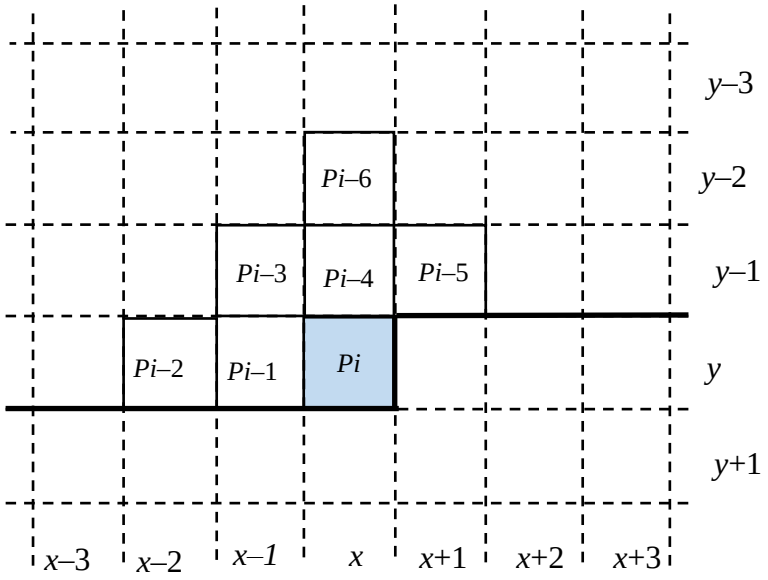


Figure I. 14. *La notation des pixels utilisée par l'algorithme proposé dans l'image courante*

7.1.3.3. Algorithme de LPOSTC

L'algorithme de GLICBAWLS utilise seulement la corrélation spatiale et il est efficace dans le codage d'image fixe. Pour la vidéo, la corrélation temporelle est très utile afin d'augmenter le taux de compression. Toutefois, la corrélation temporelle est inutile quand la scène change. Donc, les auteurs estiment que les algorithmes basés sur l'estimation de mouvement ne sont pas efficaces. Les auteurs proposent une nouvelle technique de compression qui peut non seulement exploiter la corrélation temporelle entre les images adjacentes, mais également être robuste contre les changements possibles d'une scène.

7.1.3.4. Principe de LPOSTC

LPOSTC est une abréviation de Linear Prédiction Over Spatial Temporel Corrélation. Son principe se résume par l'intégration de l'information temporelle à l'intérieur de la prédiction spatiale présentée dans l'algorithme de GLICBAWLS en incluant dans l'image précédente une matrice (3 x 3) centrée à la même position du pixel prédit (Figure 15).

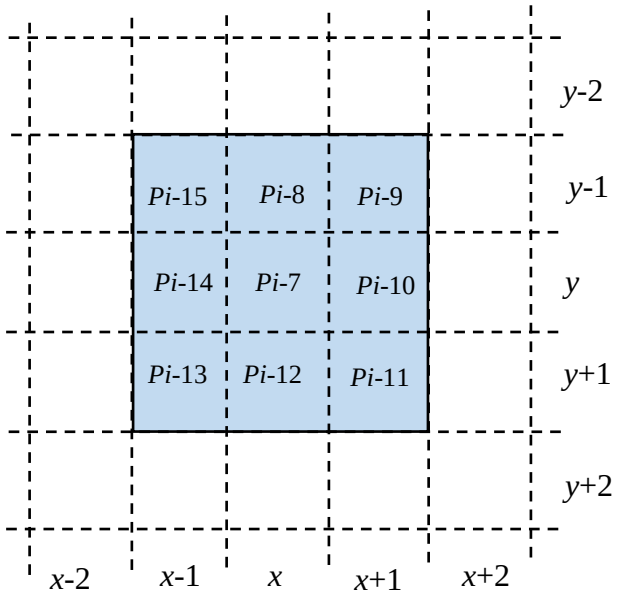


Figure I. 15. *La notation des pixels utilisée par les algorithmes LPOSTC dans l'image précédente*

7.1.4. *Codage vidéo sans perte à fort degré de scalabilité*

Le LAR (Locally Adaptive Resolution) est une méthode de compression scalable élaborée par Olivier Déforges [22]. Elle repose sur l'idée qu'une image est la superposition de deux informations complémentaires : une image basse résolution dite " image des blocs ", et une image de texture (voir Figure 16).

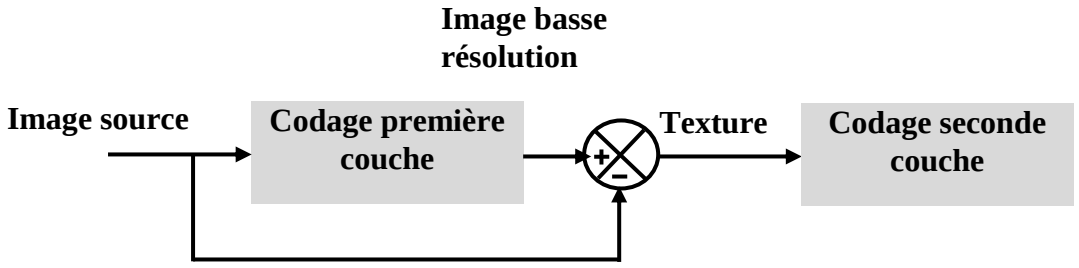


Figure I. 16. *Schéma général du codeur LAR : deux couches de codage*

La méthode s'est enrichi de nombreuses fonctionnalités telles que la scalabilité en résolution via l'approche pyramidale prédictive (APP). LAR-Vidéo repose sur l'extension à la vidéo d'une méthode de codage scalable existant pour les images fixes (LARAPP).

7.1.4.1. Principe du codeur vidéo sans perte "LAR-Vidéo"

Propre au codage vidéo, une étape d'estimation de mouvement est effectuée afin de produire une image d'erreur résiduelle. L'idée ici c'est d'appliquer sur cette erreur, une décomposition pyramidale issue d'une méthode de codage scalable d'images fixes, le LAR-APP (voir Figure 17).

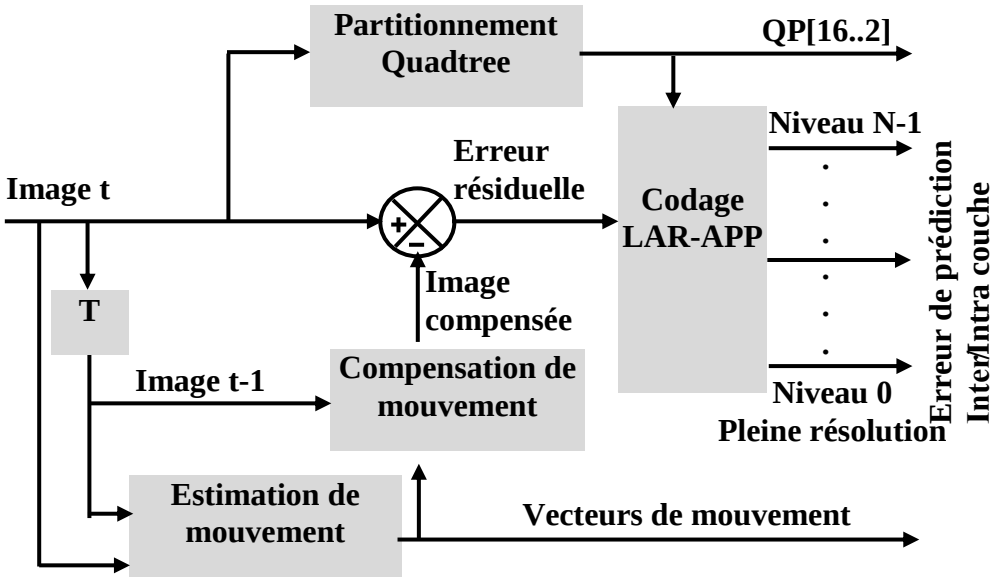


Figure I. 17. *Schéma général du codeur vidéo sans perte "LAR-Vidéo" avec compensation de l'erreur résiduelle sur N niveaux par le LAR-APP*

7.1.4.2. Principe du décodeur vidéo sans perte "LAR-Vidéo"

Le décodeur peut ainsi reconstruire les images de la séquence de façon scalable par niveau de résolution spatiale (voir Figure 18).

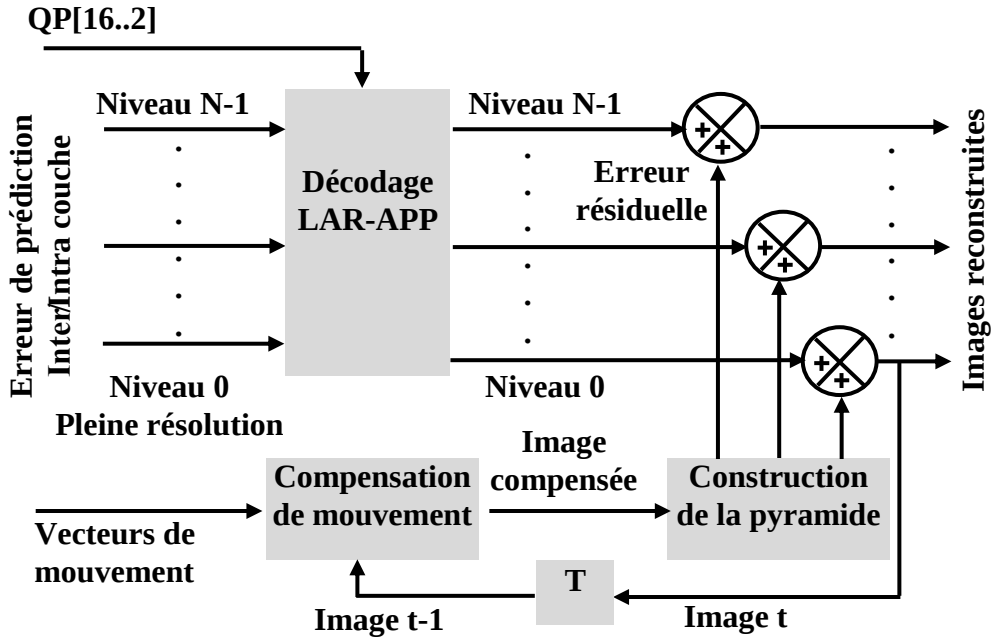


Figure I. 18. *Schéma général du décodeur vidéo sans perte vidéo sans perte "LAR-Vidéo" avec une décomposition de l'erreur résiduelle sur N niveaux par le LAR-APP*

7.1.5. *La compression vidéo sans perte utilisant une transformation entière en ondelettes*

7.1.5.1. Principe

Chaque image de la séquence vidéo RGB subit une transformation RCT (Reversible Color Transform) pour exploiter la redondance spectrale (voir Figure 19). On sélectionne ensuite pour chaque composante de couleur, la transformée IWT (Integer Wavelet Transform) qui aboutit à la plus basse valeur de l'entropie [23, 24].

L'IWT sélectionnée parmi les quatre transformations suivantes [31] est transmise pour le décodeur :

Transformation S.
Transformation (2,2).
Transformation (2+2,2).
Transformation (4,4).

Chaque image est divisée en macroblocs et ce dernier est partitionné en blocs. Une prédiction spatiale MED et une prédiction temporelle sont effectuées pour chaque bloc. Le buffer d'images est utilisé pour stocker l'image précédente. e1 et e2 sont les résiduelles de la prédiction spatiale et de la prédiction temporelle, respectivement.

Dans le bloc "Sélection adaptative du bloc", le mode de prédiction entre la prédiction spatiale et la prédiction temporelle pour chaque bloc est sélectionné en utilisant le SAD (Sum of Absolute Difference). Par la suite ce mode de prédiction est transmis pour le décodeur.
Dans le bloc "Sélection de mode de prédiction du macrobloc", le mode de prédiction pour un macrobloc est déterminé utilisant l'entropie des résiduelles obtenue par "Sélection adaptative du bloc" et l'entropie des coefficients d'ondelettes. Ensuite ce mode de prédiction est transmis pour le décodeur.
Dans le bloc "Sélection adaptative de l'IWT", la meilleure approche entre l'utilisation de l'IWT ou le non utilisation de l'IWT est déterminée.
Enfin, un codage arithmétique est utilisé pour coder l'erreur résiduelle ou les coefficients d'ondelettes selon le mode de prédiction pour chaque macrobloc.

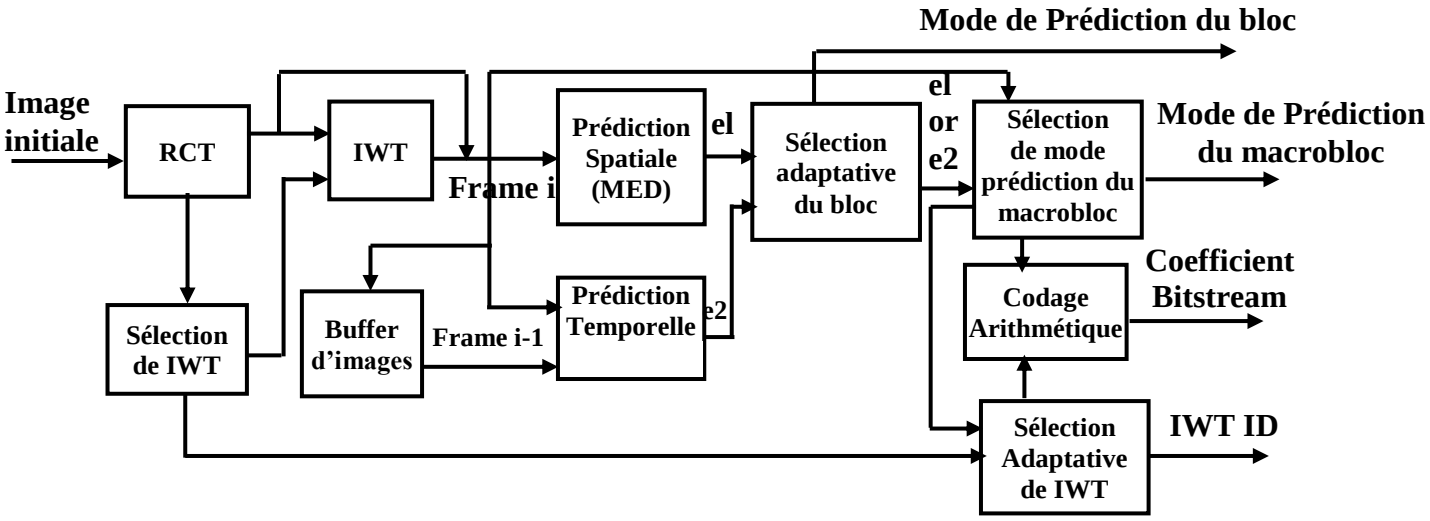


Figure I. 19. *Schéma de principe du codeur vidéo sans perte*

Au niveau du décodeur, la même procédure est effectuée dans l'ordre inverse pour récupérer les pixels d'origine. Le procédé de décodage peut être réalisé plus rapidement que le processus de codage puisque chaque mode d'opération est inclus dans le bitstream.

7.1.5.2. Transformation entière en ondelettes (IWT)

La transformée en ondelettes a été utilisée comme un outil efficace pour la compression vidéo et image. Dans la compression sans perte, l'IWT est appliquée sur les pixels à valeurs entières afin d'aboutir à des coefficients entiers.
IWT peut récupérer les valeurs originales de l'image sans perte. La performance de l'IWT dépend fortement du contenu de la vidéo et de l'image et du filtre d'ondelette utilisé. Une

approche choisie pour construire l'IWT est l'utilisation du système du lifting qui se fait dans le domaine spatial. Le système de lifting se compose de trois étapes : Split, Predict et Update (voir Figure 20).

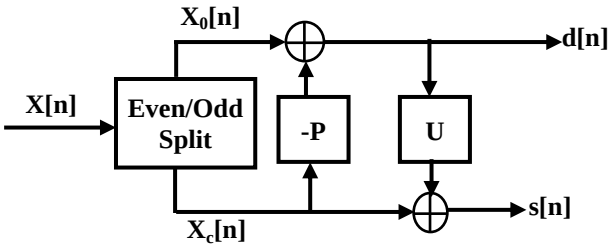


Figure I. 20. *Schéma de principe du système de Lifting*

7.2. *Méthodes commerciales*

7.2.1. *MSU LossLess video codec*

Graphic & Media Lab Video Group of Moscow State University a été développé un codeur vidéo sans perte nommé MSU (Moscow State University). Le but de développement de ce codeur était la production de vidéo sans perte à un taux de compression élevé. MSU supporte les formats de couleurs suivantes RGB, YUY2 et YV12 et fournit le meilleur taux de compression parmi les autres codeurs commerciaux sans perte.

7.2.2. *Huffyuv*

Ben Rudiak a créé un codeur vidéo sans perte appelé Huffyuv (Huffman –compressed YUV). L'algorithme d'Huffyuv est similaire à celui de JPEG sans perte. Son code source est le langage C++. Huffyuv utilise un codage entropique d'Huffman, qui est à la base de plusieurs algorithmes de compression sans perte (le codage d'Huffman est utilisé dans Zip, RAR, etc.). Ce codeur gère le RGB et YUY2. Il compresse chaque image en Zip. Par conséquent, la compression de vidéo en Huffyuv est rapide alors que la décompression est lente.

7.2.3. *Alpary*

La compagnie Alparysoft a développé un codeur vidéo sans perte nommé Alpary. C'est un codeur dédié pour la compression et la décompression de vidéo. Il supporte les formats de couleurs suivantes RGB24, YUY2 et YV12.

7.2.4. *YULS*

La compagnie YUVsoft a été élaboré un codeur vidéo sans perte YULS. YULS est efficace en termes du taux de compression, mais il est lent en temps d'exécution. Il est dédié pour les formats de couleurs suivantes RGB 24, YV12 et YUY2.

Une étude de comparaison [11, 15] a été effectuée entre les différents codeurs vidéo pour différents séquences vidéo. Les codeurs MSU et YULS sont les meilleurs en termes du taux de compression (voir Figure 21 et 22).

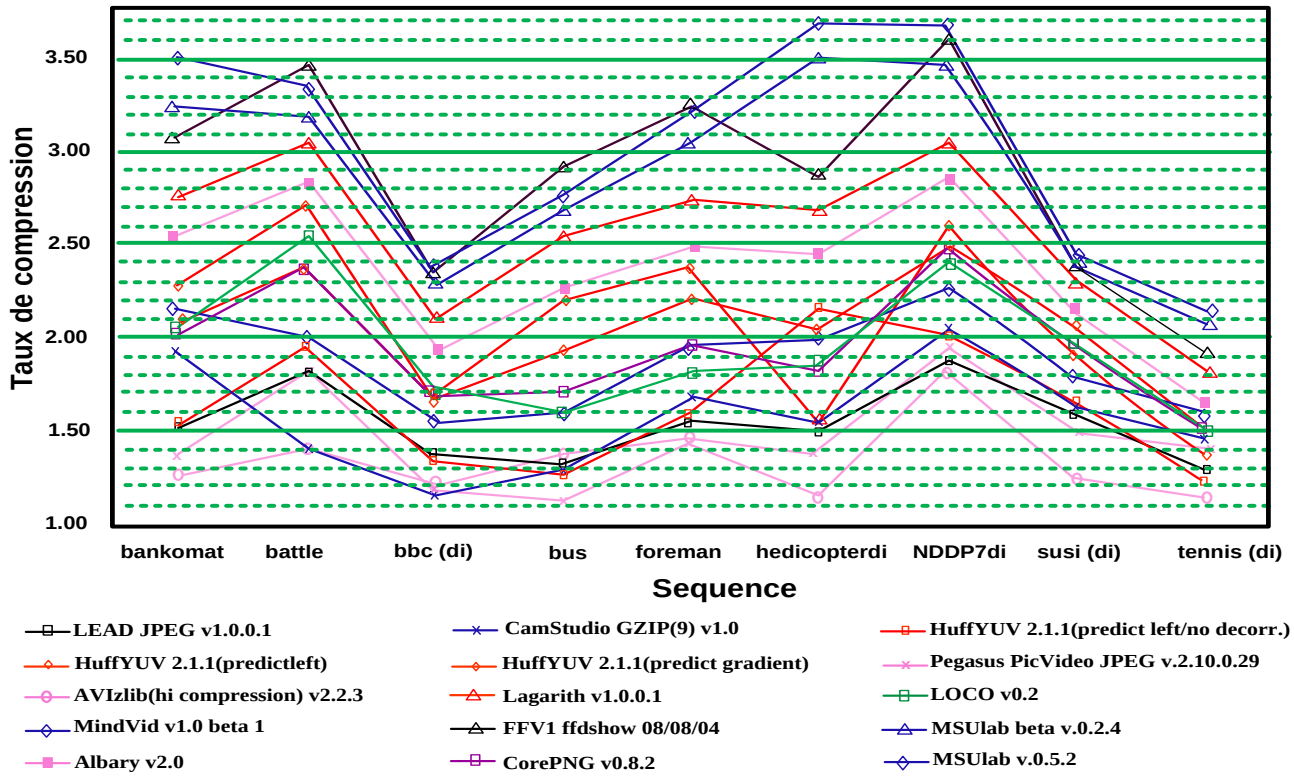


Figure I. 21. *Comparaison des différents codeurs [11]*

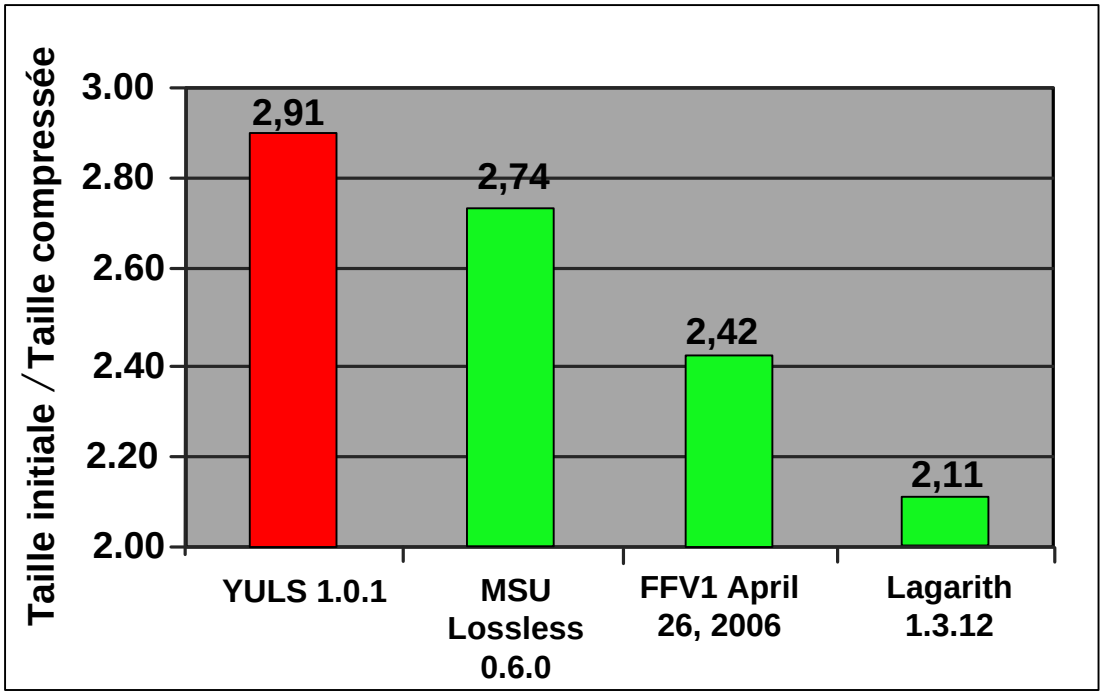


Figure I. 22. *Taux de compression de la séquence vidéo "Foreman" (format YV12)[15]*

REMARQUE.— Alpary (v2.0), AVIzlib (v2.2.3), CamStudio GZIP (v1.0), CorePNG (v0.8.2), FFV1 ffdshow (08/08/04), HuffYUV (v2.1.1), Lagarith (v1.0.0.1), LEAD JPEG (v1.0.0.1), LOCO (v0.2), MindVid (v1.0 beta 1), MSUlab beta (v0.2.4), MSUlab (v0.5.2) et PicVideo JPEG (v.2.10.0.29) sont des méthodes proposées sans citer leurs principes

Les codeurs sans perte sont illustrés dans le Tableau ci-dessous.

Tableau I. 1. *Liste des codeurs vidéo commerciaux sans perte*

Codeurs	Producteur /Auteur	Espaces de couleurs		
		RGB24	YUY2	YV12
Alpary	Alparysoft	✓	✓	✓
ArithYuv	JulienMuchembled	-	✓	-
AVIzlib	Kenji Oshima	✓	-	-
CamStudioGZIP	RenderSoft	✓	-	-
CorePNG	Jory Stone	✓	✓	✓
FastCodec	VideoSoft	✓	✓	-
FFV1	M.Niedermayer	✓	✓	✓
Huffyuv	Ben Rudiak Gould,ffdshow team	✓	✓	✓

Lagarith	**BenGreenwood**	✓	✓	✓
LOCO	**M. Rezaei**	✓	✓	✓
LZO	**AbrahamMaciasParedes**	-	-	✓
MSU Lab	**MSUGraphics &Media Lab**	✓	✓	✓
PICVideo	**PegasusImagingCorporation**	✓	-	-
Snow	**MichaelNiedermayer**	-	-	✓
x264	**x264 team**	-	-	✓
YULS	**YUVsoft**	✓	✓	✓

7.3. *Méthodes basées sur les normes*

7.3.1. *Méthodes de compression basée sur la norme MPEG-2*

Cette méthode est basée sur l'utilisation de MPEG-2 (Moving Picture Expert Group) comme plate-forme fondamentale. La compression vidéo sans perte est essentiellement réalisée par estimation et compensation de mouvement pour éliminer la redondance temporelle et par le codage entropique pour éliminer la redondance spatiale (voir Figure 23) [25].
Comme le codage entropique joue un rôle important dans l'optimisation des performances de compression vidéo sans perte, un codage arithmétique a été adopté pour cette approche.

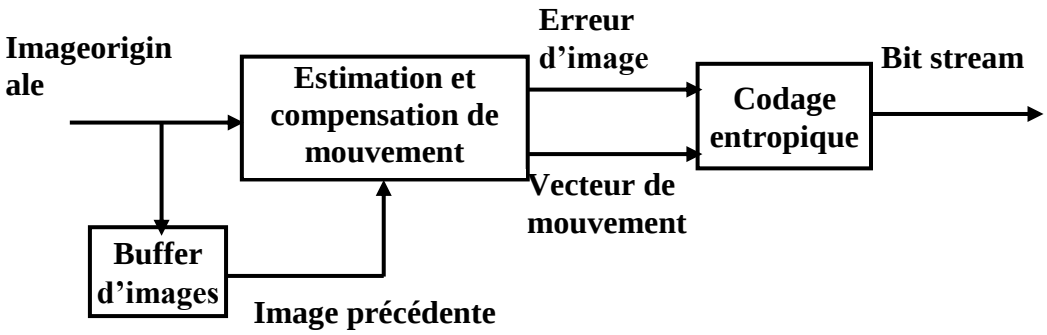


Figure I. 23. *La structure générale de la plate-forme proposée pour la compression vidéo sans perte basée sur la norme MPEG2*

7.3.2. *Méthodes de compression basée sur la norme H.264/AVC*

7.3.2.1. Codage Sans perte de H.264 FRExt [JM 8.2]

H.264 FRExt (Fidelity Range Extensions) supprime la transformation et la quantification et applique directement le codage entropique aux erreurs de prédiction (voir Figure 24) [32,33].

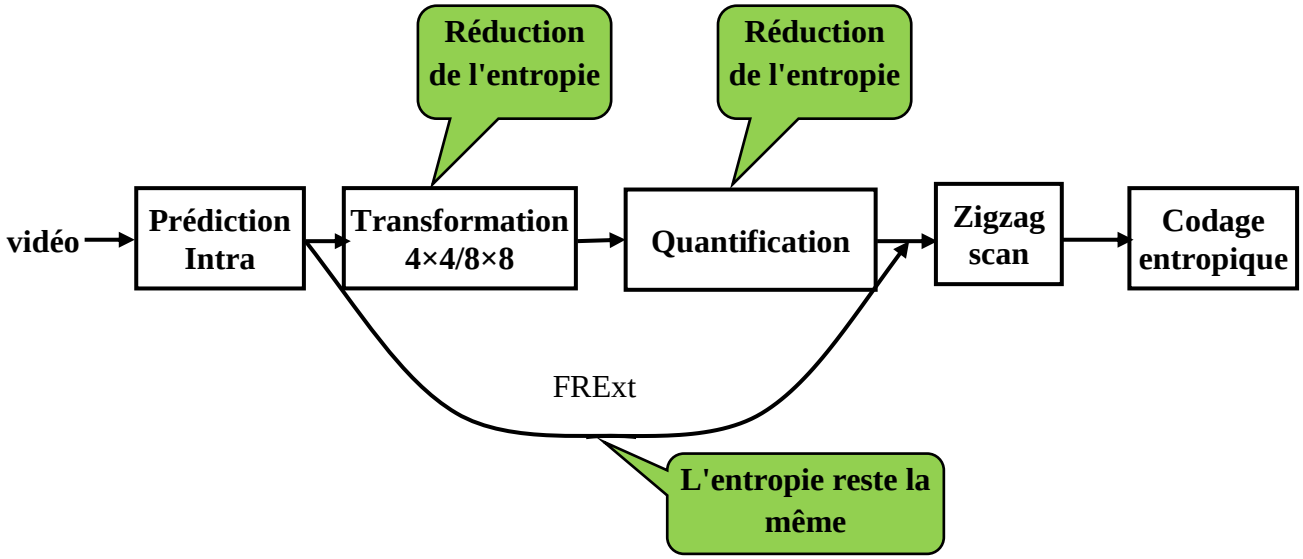


Figure I. 24. *Structure de la méthode FRExt*

Amélioration de FRExt

Au lieu de non-fonctionnement (FRExt), une transformation réversible est appliquée pour conserver l'entropie et réduire la corrélation qui existe entre les coefficients de la prédiction (voir Figure 25).

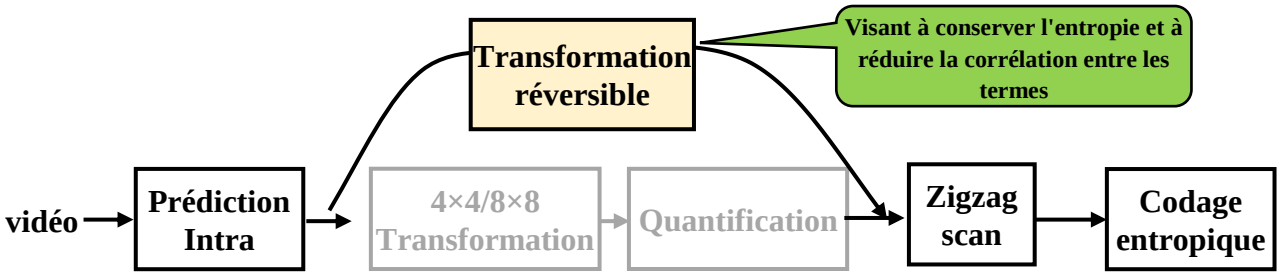


Figure I. 25. *Amélioration de FRExt*

Quatre transformations sont proposées:

Transformation entière 4×4 de la norme H.264 (réversible si non quantifié).

Piecewise Scaled Transform (décomposition de la matrice de transformation).

DCT (Direct Cosine Transform) réversible.

Prédiction linéaire :

– Version non-adaptative.

– Version adaptative.

7.3.2.2. Codage vidéo sans perte basé sur le DPCM [JM 14.2]

Une nouvelle méthode de codage intra sans perte basée sur le DPCM (Differential Pulse Code Modulation) est présentée comme une amélioration de l'algorithme de codage sans perte FREXT dans la norme H.264/AVC (Figure 26) [34,35].

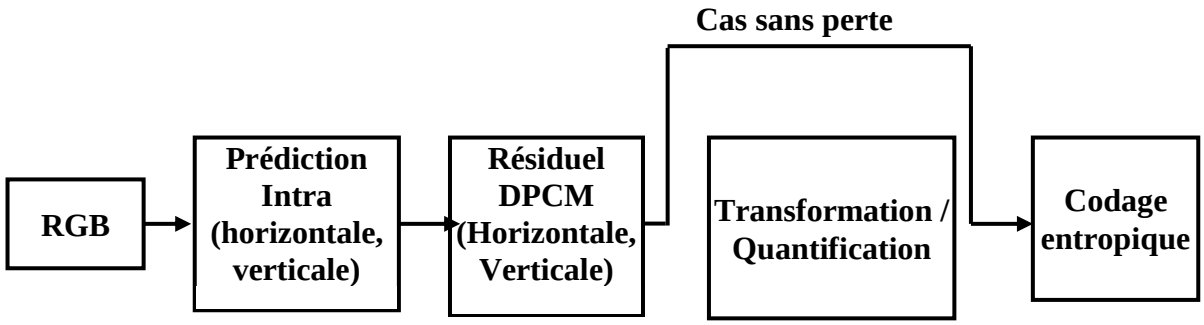


Figure I. 26. *Codage sans perte adopté dans FREXT avec DPCM de la norme H.264/AVC*

Les étapes de transformation et de quantification sont omises puisque la transformation DCT et la quantification pourrait causer des pertes de données par les opérations d'arrondissage ou de décalage. Ces erreurs ne peuvent pas être récupérées lors de la reconstruction dans le décodeur. Par conséquent, le signal sera dégradé. La technique DPCM est appliquée dans la prédiction du mode horizontale et verticale de la norme H.264. Ensuite, les données résiduelles sont directement codées par un codage entropique.

Principe du DPCM

Le DPCM est effectué utilisant une prédiction pixel par pixel au lieu de prédiction basée sur des blocs (voir Figure 27) [36,37]. Dans la méthode proposée de codage intra sans perte, les résiduelles de prédiction spatiales sont transformés aux résiduelles DPCM.

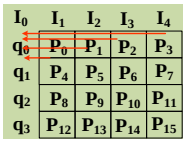
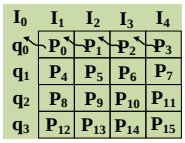
<table>
<tr><th colspan="4">Pour le mode 1 (horizontale)</th></tr>
<tr><th colspan="2">Prédiction basée sur des blocs</th><th colspan="2">Prédiction basée sur DPCM</th></tr>
<tr>
<td>$r_0=p_0-q_0$
$r_1=p_1-q_0$
$r_2=p_2-q_0$
$r_3=p_3-q_0$</td>
<td>  </td>
<td>$r_0=p_0-q_0$
$r_1=p_1-p_0$
$r_2=p_2-p_1$
$r_3=p_3-p_2$</td>
<td>  </td>
</tr>
<tr><td colspan="2">(a)</td><td colspan="2">(b)</td></tr>
</table>

Figure I. 27. *Déroulement du concept DPCM*

Par exemple pour le mode 1, Nous avons noté les résiduelle par r_0 à r_3 et nous avons indiqué les échantillons par p_0 à p_3. Dans les versions ordinaires de la norme H.264, chaque direction de prédiction utilise les pixels déjà codé des blocs frontières voisins pour prédire les pixels dans le bloc 4 × 4 actuels (voir la Figure (a)). Cependant, dans l'application DPCM, les pixels du bloc 4 × 4 sont prédits à partir des pixels voisins situés au-dessus et / ou à gauche du bloc 4 × 4 actuels. Autrement dit, j'utilise le pixel le plus proche de ce dernier. La Figure 27 montre la différence entre le calcul du résiduelle dans H264 ordinaire et le calcul du résiduelle dans H264 avec application du DPCM (voir la Figure b)).

Tableau I. 2. *Les deux versions du DPCM*

La version originale (DPCM)	**La seconde version (DPCM + interpolation des pixels)**
DPCM → Mode 0 et 1 L'efficacité du codage est fortement améliorée.	même DPCM que celui de la version originale.
DPCM → Mode 3 et 4 Les valeurs prédites de ces deux modes ne sont pas assez précises et les probabilités sont basses.	Mode 3 et 4 utilisent trois pixels voisins interpolés peuvent augmenter les probabilités des prédictions de direction diagonales (voir Figure 23).
les cinq modes de prédictions sont restés les mêmes comme dans la norme H264/AVC originale.	Deux pixels voisins sont interpolés pour des modes 5, 6, 7 et 8. Quatre pixels voisins sont interpolés pour le mode 2.

Pour améliorer la précision de prédiction et atteindre une meilleure performance dans la prédiction intra du H.264 / AVC, les directions de prédiction diagonales en bas à droite et en bas à gauche sont modifiées par un concept d'interpolation simple (voir Tableau 2 et Figure 28).

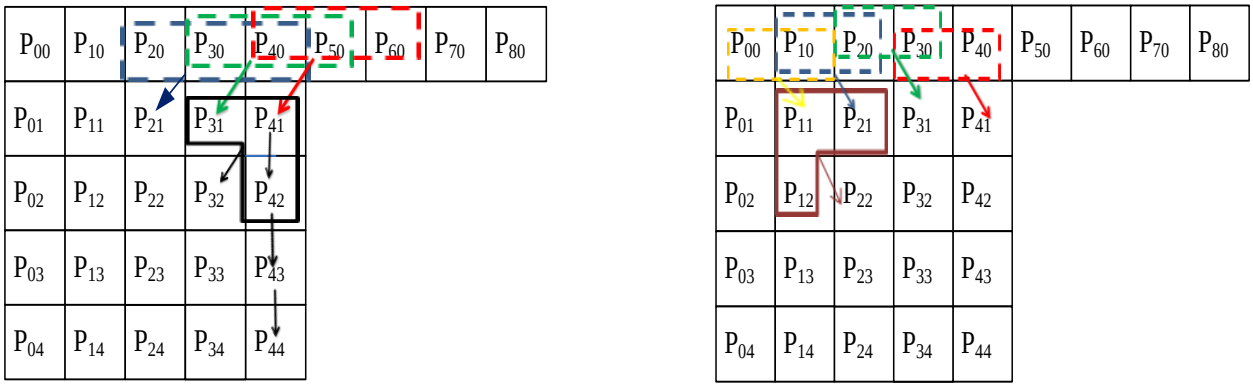


Figure I. 28. *Concept d'interpolation pour les modes 3 et 4*

7.3.2.3. Système de codage sans perte hiérarchique [JM 15.1]

La quantification et la transformation engendrent une distorsion. Mais en contrepartie, ces deux procédures permettent de réduire considérablement le débit. C'est pour cela qu'une nouvelle méthode a été développée. Elle maintient la puissance du codage avec perte en termes du taux de compression: c'est une structure à deux couches basée sur la norme ordinaire H.264/AVC (Figure 29) [28]. Cette nouvelle architecture nommée aussi système de codage sans perte hiérarchique.

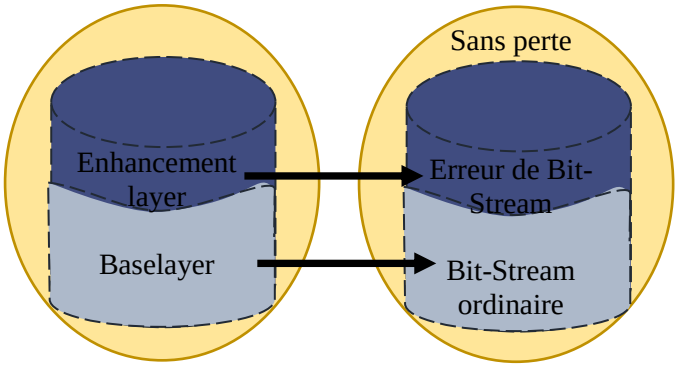


Figure I. 29. *Structure à deux couches basée sur la norme H.264/AVC*

Dans le système de codage sans perte hiérarchique, le flux de bits codé par la seconde couche est utilisé pour compenser les distorsions du flux de bits codé de la première couche afin de reconstruire la vidéo originale sans perte (voir Figure 29 et Figure 30).

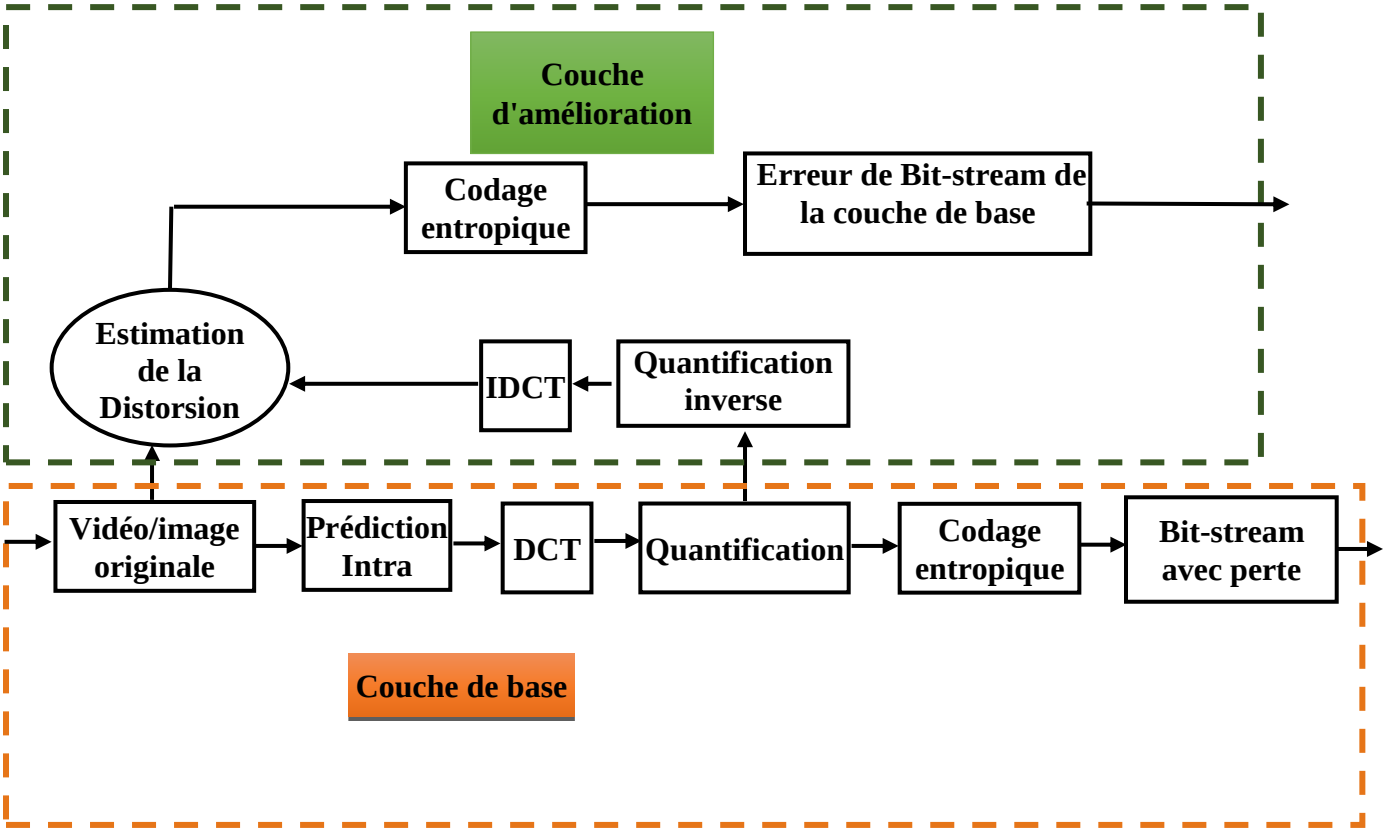


Figure I. 30. *Schéma détaillé de Codage Sans perte Hiérarchique*

La première couche BL (Base Layer) est le système de codage avec perte H.264/AVC, et la seconde couche EL (Enhancement Layer) est le système de codage entropique sans perte utilisé pour coder l'erreur faite par la DCT et la quantification.

7.3.2.4. Méthode basée sur la norme H.264/AVC FRExt [JM18.3]

Le groupe JVT (Joint Video Team) a été élaboré une extension à la norme H.264 / AVC, connue sous le nom FRExt ou nommée aussi en anglais "high profile encoder"[29, 38,39]. Le codeur H.264/AVC FRExt supporte quatre nouveaux profils appelé le High Profile qui sont essentiellement des étendus du main profiles. Ces profils sont fournis un ensemble de nouveaux outils de codage en termes de "sample bit depth" et de "color format"(Figure 31).

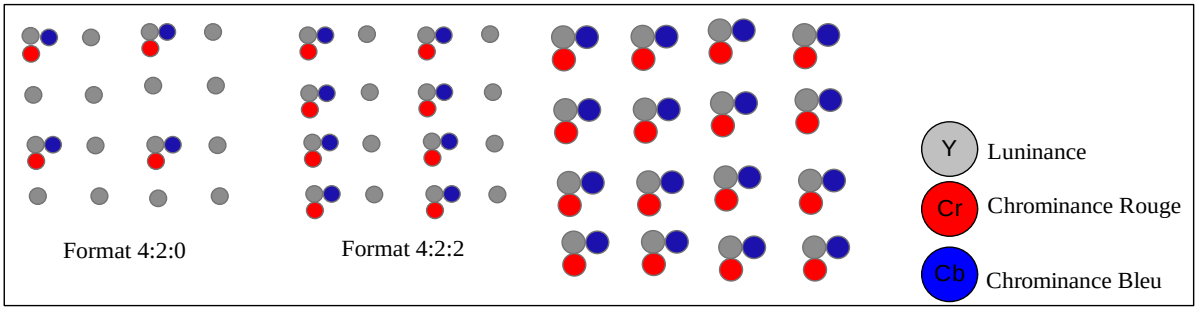


Figure I. 31. *La chrominance YCrCb 4:4:4*

H.264 / AVC High Profile fournissent des fonctionnalités supplémentaires comme "8×8 transform usage", "8×8 spatial luma prédiction", "scaling matrices" et "lossless coding" etc…(voir Tableau 3).

Tableau I. 3. *Les outils de codage de FREXt*

Outils de codage	**High**	**High 10**	**High 4:2:2**	**High 4:4:4**
Main Profile	X	X	X	X
8x8 vs 4x4 Transform Adaptivity	X	X	X	X
Quantization Scaling Matrices	X	X	X	X
Separate Cb Cr QP control	X	X	X	X
Monochrome video format	X	X	X	X
8×8 spatial luma prédiction	X	X	X	X
9 and 10 Bit Sample Bit Depth		X	X	X
4:2:2 Chroma Format			X	X
11 and 12 Bit Sample Bit Depth				X
4:4:4 Chroma Format				X
Residual Color Transform				X
Predictive Lossless Coding				X

7.3.3. *Méthodes de compression basée sur la norme HEVC*

La méthode de base pour le codage sans perte au sein du HEVC est de supprimer la transformation et la quantification dans le codeur et la transformation inverse et la

quantification inverse dans le décodeur car ces quatre phases introduisent une dégradation dans le signal de sortie (Figure 32) [30].

Le filtre SAO (Sample Adaptive Offset) et le filtre anti-bloc sont aussi contournés dans le codage sans perte parce qu'ils introduisent aussi des dégradations dans le signal reconstruit.

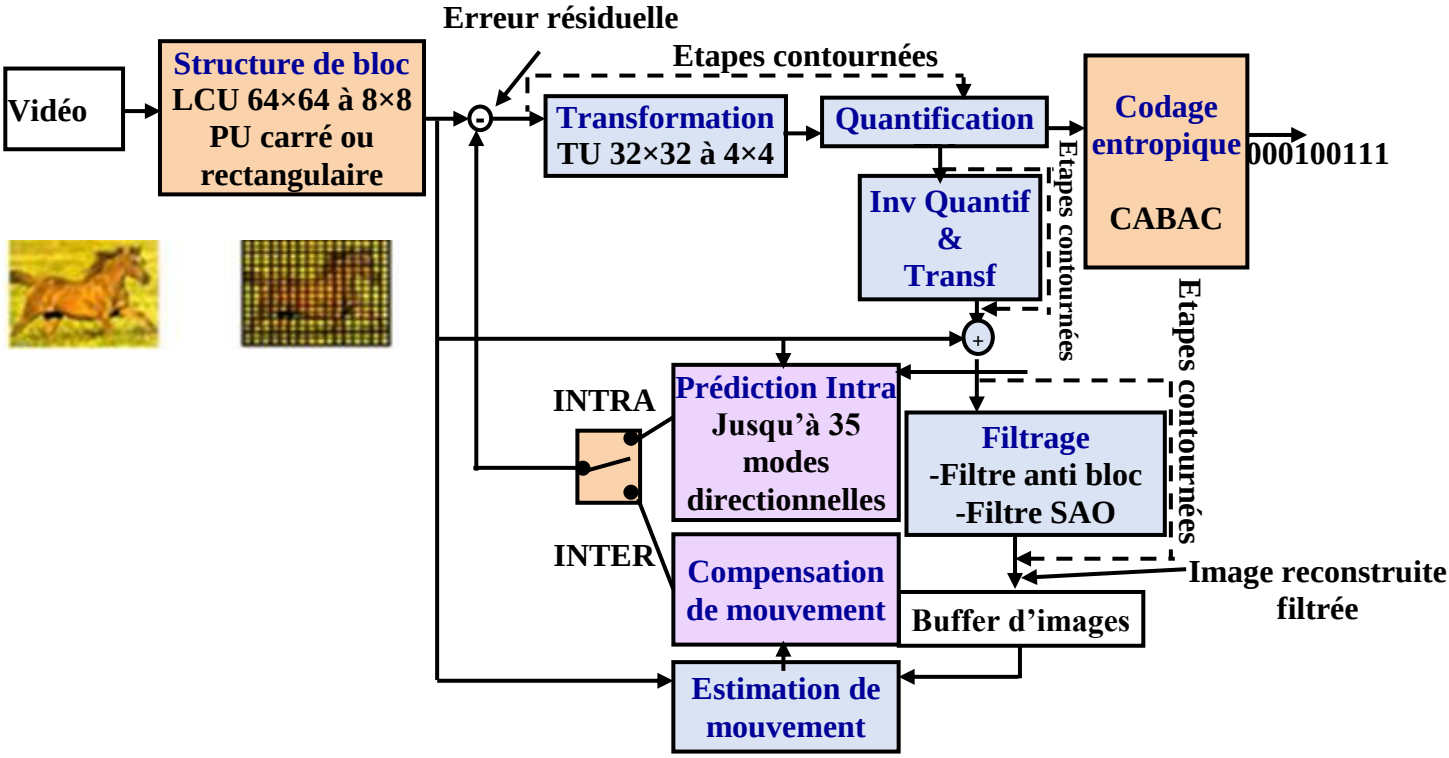


Figure I. 32. *La structure globale du codage sans perte de la norme HEVC*

8. Conclusion

Le grand intérêt des chercheurs pour la compression vidéo sans perte a débouché sur un ensemble de techniques de compression évoluées. Certains de ces techniques atteignent une maturité manifestée sous forme de standards internationaux. Cependant, les besoins et la variété des applications en imagerie exigent davantage d'améliorations des performances de la qualité de compression. De telles améliorations sont possibles en exploitant les codages entropiques.

Le codage entropique est très intéressant dans l'optimisation des performances de compression vidéo sans perte. C'est pour cette raison, je vais expliquer en détail le fonctionnement de deux codages entropiques CABAC et CAVLC dans le chapitre suivant. Ensuite, je vais faire une étude comparative entre les deux codages entropiques en fonction de leur taux de compression, la qualité de la reconstitution et de leur temps d'exécution.

CHAPITRE II

Modification de la Compression à deux couches et étude Comparative entre CABAC et CAVLC

1. Introduction

La communication numérique est présente dans de nombreuses applications en raison de sa fiabilité, son chiffrage (cryptage) et son efficacité de traitement. La question la plus fondamentale face à des archivistes est de savoir comment maintenir l'essentiel du contenu média (audiovisuel) dans un espace mémoire le plus petit possible. Ainsi, la compression vidéo est apparue comme une solution adéquate pour de nombreux problèmes. Cette compression vidéo est nécessaire pour la transmission de données vidéo numériques sur les réseaux actuels avec une bande passante limitée et pour des applications de capacités de stockage limitées.

Les techniques de compression peuvent être classées en deux catégories: la première contient les méthodes conservatrices (Lossless), tandis que la deuxième inclut des méthodes non conservatrices (Lossy) [1]. Il est bien connu que la deuxième catégorie aboutit à un taux de compression plus élevé. Ce qui justifie la grande popularité de ses méthodes, en particulier dans les images fixes et les séquences vidéo. Cependant, il existe des applications qui ne tolèrent aucune distorsion, en particulier l'imagerie médicale (télémédecine) [2], les travaux d'art, la recherche scientifique, l'archivage des films, le cinéma numérique (systèmes multimédia de haute qualité) et le service militaire (fabrication d'armes, trace d'avion militaire et le radar militaire).

Dans ce chapitre, nous sommes plutôt intéressés par les techniques qui modifient la norme H.264/AVC pour la rendre conservative. Le reste de ce chapitre est organisé comme suit : dans la deuxième section, on présente le principe du CABAC et du CAVLC. Dans la troisième section, les algorithmes de compression sans perte sont décrits. Dans la quatrième section, deux études comparatives sont présentées: la première a porté sur une comparaison de notre travail avec des méthodes antérieures de la littérature, tandis que la seconde compare les codeurs CABAC et CAVLC. Finalement, nos conclusions ont été tirées dans la dernière section du chapitre.

2. Présentation du CABAC et CAVLC

La norme H.264/AVC comporte deux méthodes de codages entropiques : le CABAC et le CAVLC [3,4]. Dans le codeur vidéo, le choix du codage entropique (CAVLC ou CABAC), qui convertit les données vidéo codées en une chaîne de bits organisée sous forme de flux binaire (bitstream), est la dernière étape avant la transmission de données sur le canal.

2.1. *Principe du CABAC*

Le processus de CABAC se compose de trois étapes: binarisation, la modélisation de contexte et le codage arithmétique (voir Figure 1) [5,6].

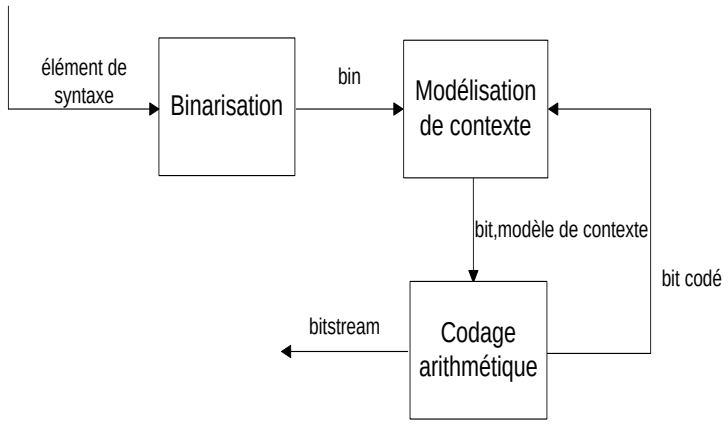


Figure II. 1. *Schéma général du codage entropique CABAC*

Binarisation

En utilisant un processus de binarisation, les éléments de syntaxe (par exemple, les vecteurs de mouvement, les modes de prédiction, les coefficients) sont transformés en symboles binaires (chaîne de bins). Les formes diverses de binarisation, comme Exp-Golomb, le codage à longueur fixe sont utilisées dans H.264/AVC. La norme H.264/AVC définit le type de binarisation utilisé pour chaque élément de syntaxe.

Modélisation de contexte ou sélection de contexte

Dans la modélisation du contexte, pour chaque bin, un modèle de probabilité est choisi en se fondant sur les éléments de syntaxe précédemment codés. Il existe plusieurs centaines de modèles de contexte différents utilisés dans H.264/AVC. Par conséquent, une machine d'état est nécessaire pour sélectionner le contexte approprié pour chaque bin. En outre, la probabilité estimée du modèle de contexte sélectionné est mise à jour après que chaque symbole binaire soit codé ou décodé.

Codage arithmétique

En utilisant le codage arithmétique, les bins sont compressés en bits (c'est-à-dire que plusieurs bins peuvent être représentés par un seul bit); ceci permet aux éléments de syntaxe d'être représenté par un nombre fractionnaire de bits, ce qui améliore l'efficacité de codage.

Afin de compresser efficacement les bins en bits, la probabilité des bins doit être estimée avec précision. Ensuite, le codage arithmétique encode chaque valeur binaire avec son modèle de

probabilité associé. Tous les bins ne sont pas codés en utilisant une probabilité estimée. En effet, les bins correspondant au signe peuvent être également codés en utilisant le codage bypass (c'est-à-dire une probabilité uniforme égale à 0,5 est associée aux bins). La structure du codage de CABAC pour un bloc 4×4 de coefficients transformés quantifiés est représentée dans la Figure 2.

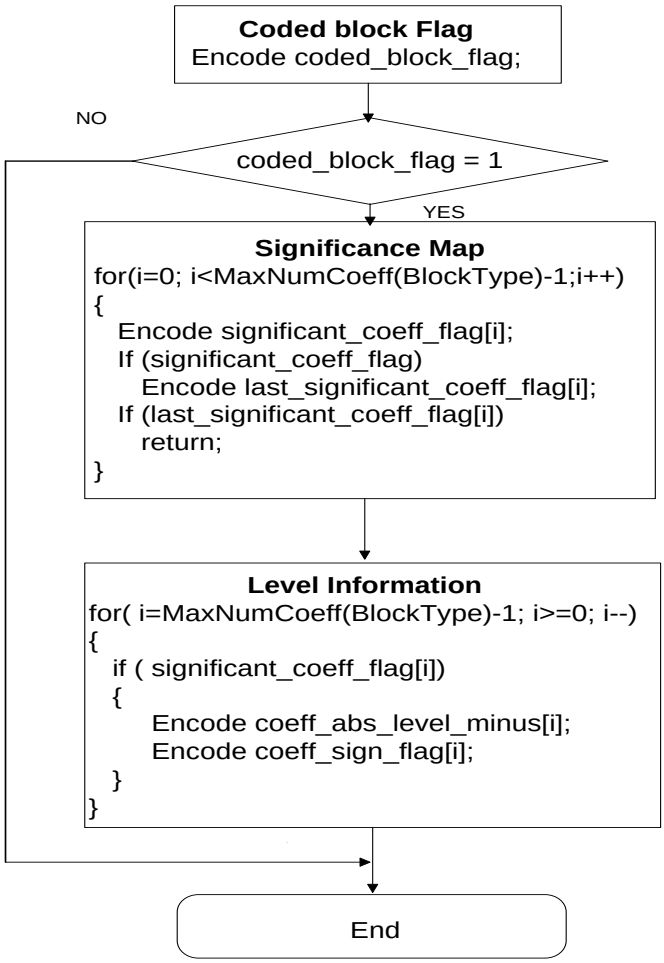


Figure II. 2. *La structure de codage CABAC pour les données résiduelles [7]*

Pour les données résiduelles dans un bloc4×4, CABAC a utilisé des éléments de syntaxe tels que coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, coeff_abs_level_minus1 et coeff_ sign_ flag.

Tout d'abord, pour chaque bloc4×4, un symbole de bit appelé coded_block_flag est transmis pour indiquer si le bloc 4×4a des coefficients ou nom. Si le coded_block_flag est nul (c'est-à-dire, coded_block_flag indique que le bloc 4x4actuel n'a pas de coefficient), le traitement du bloc actuel peut être arrêté.

Dans le cas contraire, si le coded_block_flag indique l'existence de coefficients, les processus de codages «significance map» et «level information» sont codés séquentiellement. Le

processus «significance map» détermine l'emplacement de tous les coefficients significatifs (non nul) dans un bloc. Dans le codage «significance map», un symbole binaire est associé pour chaque coefficient (c'est-à-dire un élément de syntaxe à 1 bit : significant-coeff-flag) indique si le coefficient est significatif ou pas à la position actuelle.

Si le significant_coeff_flag est égale à 1, c'est-à-dire un coefficient non nul existe à cette position, un autre élément de syntaxe à 1 bit : last_significant_coeff_flag est codé. Cet élément de syntaxe indique si le coefficient significatif actuel est le dernier coefficient à l'intérieur du bloc ou non.

Dans le processus «level information», les valeurs des coefficients significatifs sont codées en utilisant deux éléments de syntaxe: coeff_abs_level_minus1 et coeff_sign_flag. L'élément de syntaxe coeff_sign_flag est codé par un symbole de 1 bit et la méthode de binarisation(UEG0) Unary/0th-order Exponential Golomb est utilisée pour coder la valeur de coeff_abs_level_minus1 qui représente la valeur absolue du level minus1.

2.2. *Principe du CAVLC*

CAVLC est utilisé pour coder des données résiduelles. Le codage de longueur variable adaptatif basé sur le contexte (CAVLC) a été conçu pour profiter de plusieurs caractéristiques des données résiduelles dans un codage avec perte [8,9] :

1) Après la transformation et la quantification, les blocs contiennent généralement de nombreux zéros qui se concentrent dans les régions à hautes fréquences.

2) En contre parti, les niveaux des coefficients non nuls sont petits.

3) Les niveaux de coefficients non nuls tendent à être plus importants vers les régions de basses fréquences.

Par conséquent, en tenant compte des caractéristiques ci-dessus des coefficients transformées quantifiés balayés en zigzag pour un bloc 4×4, CAVLC utilise cinq étapes de codage [10] (comme le montre la Figure 3).

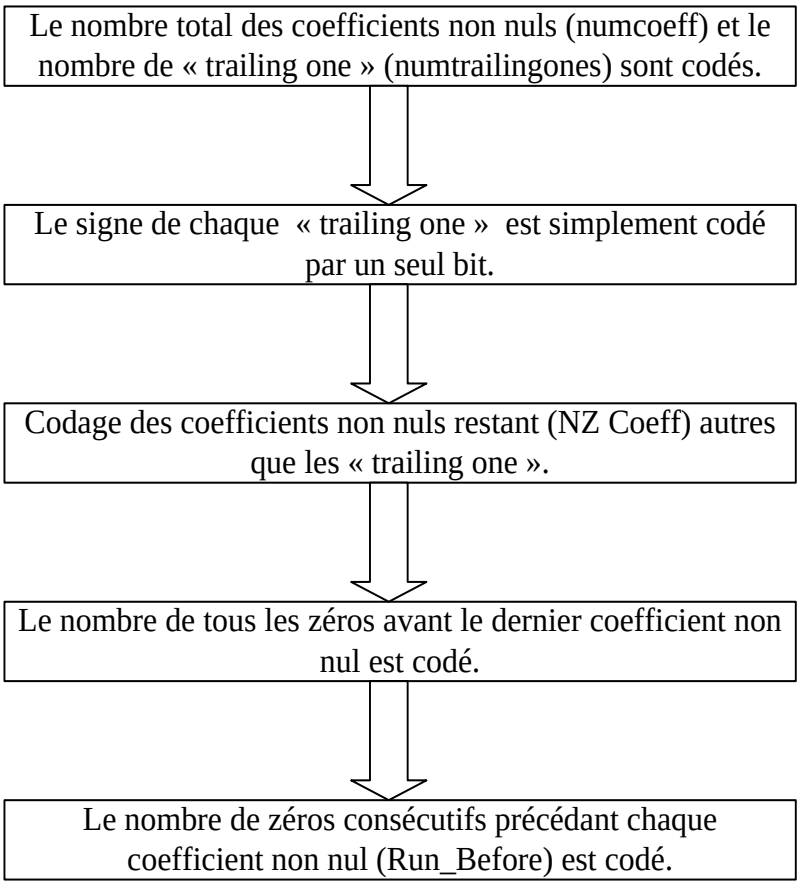


Figure II. 3. *Étapes de codage CAVLC*

3. Les algorithmes de compression sans perte modifiant la norme H.264/AVC

La structure (codeur/décodeur) de la vidéo numérique contient une paire complémentaire de systèmes: un compresseur (codeur) et un décompresseur (décodeur). Le codeur convertit la vidéo source dans un format compressé avant la transmission ou le stockage, alors que le décodeur le convertit en une représentation de la vidéo originale ou l'approximation de la séquence source.

Ainsi, si la séquence reconstituée correspond entièrement à celui d'origine, alors le processus de codage est sans perte. Si la séquence décodée diffère de l'originale, le processus de codage est avec perte. Le schéma de compression/décompression est illustré dans la Figure 4.

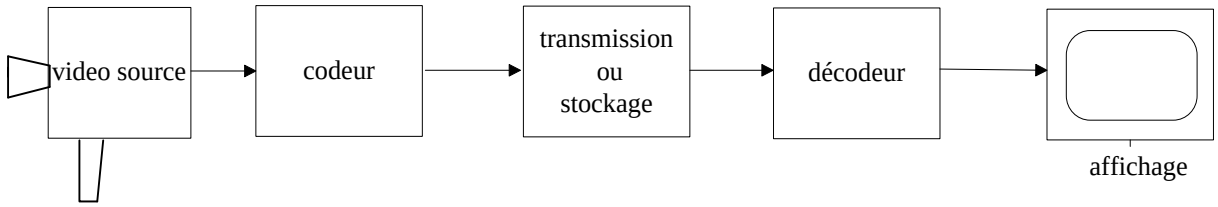


Figure II. 4. *La structure codeur / décodeur de la vidéo numérique*

La norme H.264/AVC permet l'encodage d'une vidéo numérique qui se compose d'images successives. L'encodeur H.264/AVC contient une prédiction spatio-temporelle (prédiction intra/inter) en plus d'autres modules : la transformation, la quantification, le codage entropique, l'estimation de mouvement et le filtre anti-bloc (comme indiqué dans la Figure 5).

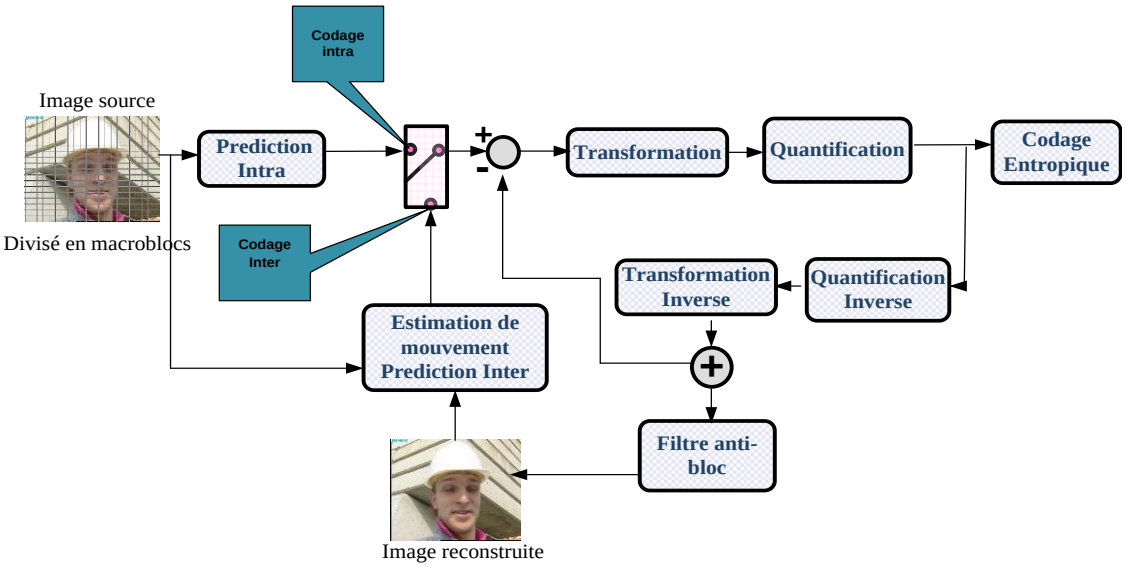


Figure II. 5. *La chaîne de codage vidéo de la norme H.264/AVC*

La première étape consiste à diviser chaque image de la vidéo en macroblocs (voir Figure 6). Les macroblocs de la première image sont prédits en utilisant un processus de prédiction intra. Cependant, les autres images sont codées en intra ou inter prédiction. La différence entre le macrobloc original et sa prédiction, appelé l'erreur de prédiction résiduelle, subit alors une transformation en cosinus discrète entière ICT (Integer Cosine Transform).

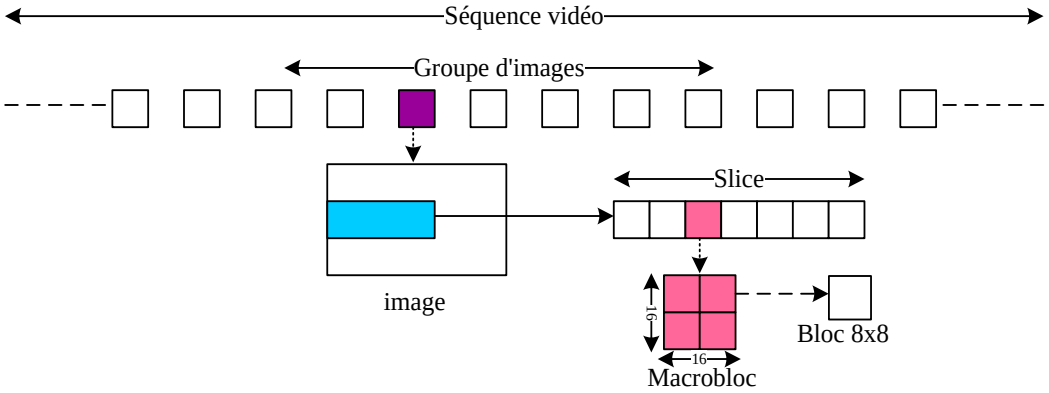


Figure II.6. *Hiérarchie des données d'une séquence vidéo*

La transformée ICT appliquée à une matrice A (bloc 4×4) est définie par :

$$B = (C_F A C_F^{\,T}) = \left\{ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad [A] \quad \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right\} \qquad (1)$$

B est la résultante de la transformée.

Les coefficients transformés vont ensuite subir une quantification. La quantification sert à diviser les coefficients du MB résiduel déjà transformés par un pas de quantification (QP) variable de 1 à 52. Ce processus réduit les hautes fréquences et augmente le nombre de coefficients nuls. Ceci provoque une perte d'information mais abouti à une amélioration du taux de compression. L'équation de la quantification est comme suit:

$$W_{ij} = \left[\frac{Z_{ij}}{Q_P}\right] \qquad (2)$$

Z_{ij} est un coefficient obtenue suite à la transformée.

Q_P est le pas de quantification.

W_{ij} est le coefficient quantifié.

Avant d'appliquer le codage entropique, les coefficients transformés quantifiés sont balayés dans un ordre zigzag. Un flux binaire (bitstream) pour la vidéo compressée est obtenu en sortie.

Pendant le processus de quantification, des opérations d'arrondissement ou de décalages binaires introduisent des erreurs qui peuvent affecter le signal d'erreur de prédiction. Ces erreurs ne peuvent pas être récupérées lors de la reconstruction dans le décodeur. En conséquence, la vidéo reconstruite sera dégradée et nous obtenons des images avec des artefacts de blocs (voir la Figure 7).

Figure II. 7. *Illustration des artefacts de blocs d'une image de la séquence "vidyo4"*

Afin d'assurer une bonne compression vidéo qui garantit une réduction du nombre de bits sans la dégradation de la qualité visuelle, une compression vidéo sans perte est adoptée. Par conséquent, la norme H.264/AVC a été modifiée pour la rendre conservative.

Le codage vidéo sans perte a été obtenu en ignorant le processus de quantification, puisque ce processus causerait de grandes distorsions [11-13, 19-21] (voir la Figure 8). La nouvelle chaîne de codage serait donc la suivante:

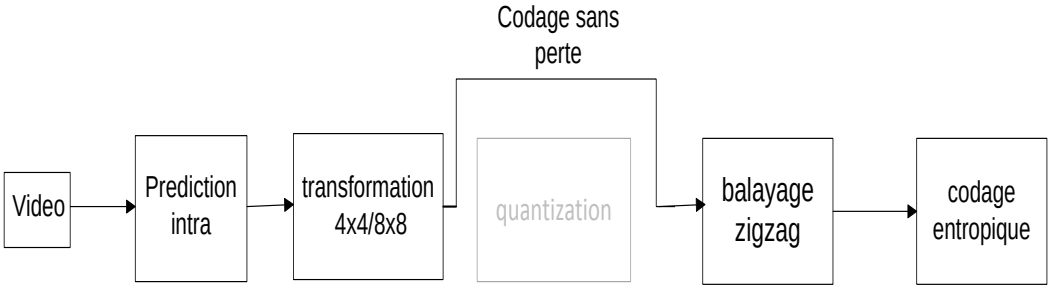


Figure II. 8. *H.264 / AVC_LS*

3.1. *Les techniques basées sur la transformation*

Le codage basé sur la transformation : applique une transformation réversible sur l'image. En effet, l'image source est divisée en quatre sous-bandes ou plus. La sous-bande située aux fréquences inférieures contient la majeure partie de l'énergie du signal, alors que les autres, à de plus hautes fréquences, incluent une petite partie d'énergie d'image. Ce genre de codage est typiquement utilisé en travaillant dans le mode lossy et lossless, comme dans le JPEG

2000. Les techniques de l'état d'art [11-15] visent à réduire les erreurs possibles dans la transformation en remplaçant les transformations classiques par d'autres qui sont réversibles.

3.1.1. Méthode de Wei et al

1ére version :

Sans le processus de la transformation, la méthode de codage ne peut pas prendre l'avantage de compactage des données spatiales. Pour résoudre ce problème, Wei et al. [16] ont proposé un système de codage vidéo sans perte qui utilise la transformation Hadamard et la troncature adaptative pour remplacer la TCD entière et la quantification. Comme le montre la Figure 9, les coefficients transformés non quantifiés sont divisés en deux parties quotient et reste. La première partie est directement codée par les codeurs entropiques de la norme H.264 (CABAC et CAVLC) puisque le compactage d'énergie est fourni par la transformation Hadamard. La deuxième partie est codée par le codeur à une structure de bits fixes.

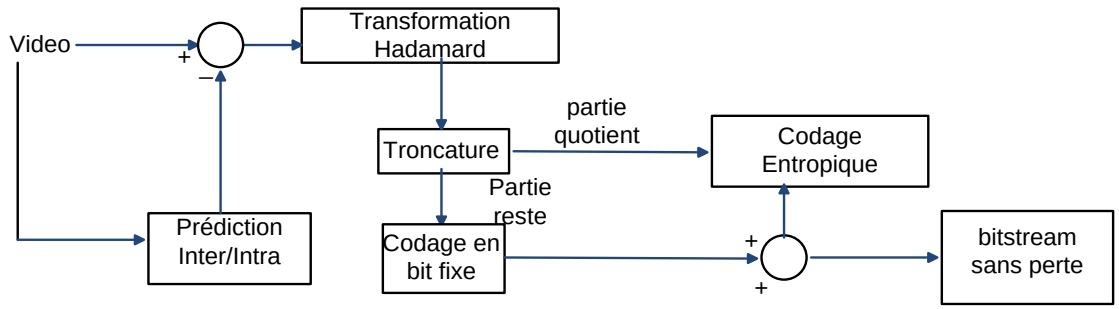


Figure II. 9. *Les procédures de codage modifiées avec la transformation de Hadamard pour la norme H.264/AVC sans perte [16]*

2éme version :

Les outils de codage sans perte proposés pour coder les données résiduelles sont représentés dans la Figure 10 [17]. La transformée de Hadamard est appliquée en premier lieu aux coefficients résiduels. Les redondances dans les coefficients transformés sont supprimées dans la troncature compacte. Un algorithme de troncature compacte est utilisé pour enlever un nombre adéquat de bits supplémentaires des coefficients dans la partie reste [16-18]. En effet, basé sur les corrélations entre les coefficients des blocs voisins, l'algorithme de troncature adaptative détermine le nombre de bits à supprimer.

L'unité de réglage ajuste les coefficients compactés. Pour améliorer les performances de codage sans perte de la norme H.264 / AVC, les coefficients ajustés sont codés par un codeur CAVLC amélioré pour obtenir le bitstream.

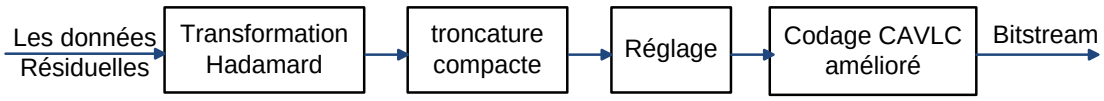


Figure II. 10. *Procédure de codage de données résiduelles dans l'algorithme de codage sans perte proposée par Wei et al [17]*

Les coefficients résiduels sont compactés et ajustés. Ils sont ensuite codés par CAVLC pour générer le bitstream dans l'algorithme proposé. Cinq éléments syntaxiques, à savoir « coeff_token », « trailing_ones_sign_flag », « level », « total_zeros » et « run_before », sont codés par CAVLC. Ces éléments de syntaxe sont conçus en fonction des propriétés statistiques des coefficients quantifiés dans le codage avec perte. La plupart des éléments syntaxiques fonctionnent correctement lorsqu'ils sont utilisés pour les coefficients résiduels dans l'algorithme classique avec perte. Cependant, l'un des éléments syntaxiques, « level », ne peut pas être codé efficacement par CAVLC dans le mode sans perte. En effet, L'élément « level» inclut le signe et l'amplitude de chaque coefficient non nul restant. Le code binaire pour chaque élément « level » est composé d'une partie de préfixe « level_préfixe » et d'une partie de suffixe « level_suffixe ».La variable « suffixLength », allant de 0 à 6, est utilisée pour déterminer la plage d'amplitude correspondante de « level_suffix ».Une grande valeur de « suffixLength » convient à un élément « level » ayant une amplitude élevée, tandis qu'une petite valeur de « suffixLength » est appropriée pour un élément « level » de faible amplitude. D'où, quand la valeur de « suffixLength » n'est pas correctement mise à jour après le codage de chaque élément « level », les performances de codage de l'élément « level » sont considérablement réduites.

D'une manière générale, la séquence de codage des éléments de syntaxe dans CAVLC ne peut pas être modifiée à cause des dépendances de données dans le décodeur. Par exemple, « coeff_token » doit être codé avant « trailing_ones_sign_flag » et « level ». Néanmoins, il n'y a aucune dépendance de données entre les éléments « level » et les éléments « total_zeros » et « run_before ». Par conséquent, l'ordre de codage de ces éléments de syntaxe peut être changé. Dans ce cas, la position de chaque élément « level » dans un bloc 4 × 4 est connue lorsque l'élément « level » est décodé. Par conséquent, les performances de codage de CAVLC avec l'ordre de codage modifié des éléments de syntaxe peuvent être améliorées.

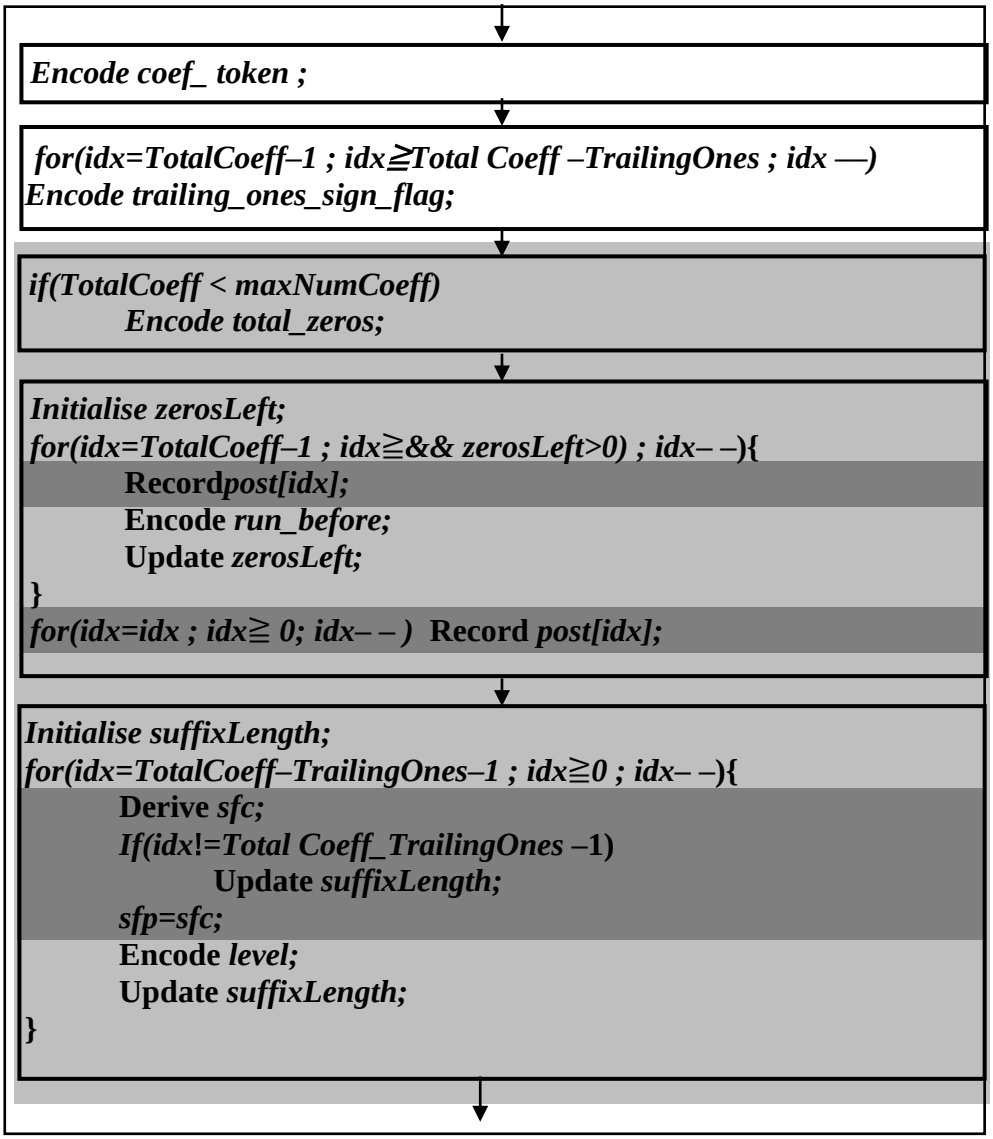


Figure II. 11. *Codeur CAVLC amélioré basé sur l'ordre de codage modifié d'éléments de syntaxe proposée par Wei et al [17]*

L'algorithme du codeur CAVLC amélioré basé sur l'ordre de codage modifié des éléments de syntaxe est représenté dans la Figure 11. Tout d'abord, les éléments de syntaxe « coeff_token » et « trailing_ones_sign_flag » sont codés, comme dans le cas classique de CAVLC, suivi de « total_zeros ». Ensuite, chaque élément « run_before » indexé par « idx» est codé, où la position de l'élément « level » correspondant dans un bloc est calculé et noté comme :

pos[idx] = idx + zerosLeft (3)

Chaque élément « level » est codé et la position de chaque élément « level » enregistré dans l'étape « run_before » est utilisée pour traiter une mise à jour supplémentaire de la valeur « suffixLength » avant le codage de chaque élément « level ».

Concernant la mise à jour supplémentaire, la valeur du facteur de réduction pour l'élément « level » courant, dénoté par « sfc », est d'abord tirée de la position de l'élément « level » comme :

$$\text{sfc} = \begin{cases} 0 & pos\,[idx]=0 \\ 1 & 1 \le pos\,[idx] \le 3 \; ou \; pos\,[idx]=5 \\ 2 & pos\,[idx]=4 \; ou \; 6 \le pos\,[idx] \le 9 \; ou \; pos\,[idx]=11 \\ 3 & pos\,[idx]=10 \; ou \; 12 \le pos\,[idx] \le 14 \\ 4 & pos\,[idx]=15 \end{cases} \quad (4)$$

La valeur de « suffixLength » est alors mise à jour comme :

suffixLength = max (0, min (suffixLength + sfp − sfc, 6)) (5)

Où « sfp » est la valeur du facteur de réduction pour l'élément « level » précédent enregistré pendant le codage de l'élément « level » précédent. La valeur « suffixLength » est utilisée pour coder l'élément actuel « level ». La mise à jour régulière de la valeur « suffixLength » après le codage de chaque élément « level » est également traitée. Les performances du codage sont améliorées car les effets des facteurs de réduction sont considérés.

3.2. *Les techniques basées sur la prédiction*

Dans le codage de prédiction, un pixel de l'image est prédit en utilisant la corrélation spatiale. L'image résiduelle (ou l'image d'erreur de prédiction) est codée. Si la prédiction est adéquatement conçue, l'entropie de l'image résiduelle est inférieure à l'entropie de l'image originale et par conséquent le taux de compression s'améliore.

Aux niveaux des techniques des références [19-21], le processus de transformation a été ignoré et le concept DPCM a été introduit dans le processus du prédiction intra pour obtenir une meilleure performance.

3.2.1. *Principe du DPCM*

Prédiction horizontale:

La première colonne sera déplacée comme elle est dans la matrice temporelle puis une opération de soustraction est appliqué à la deuxième colonne pixel par pixel jusqu'à la fin de

la matrice. L'opération de soustraction est représentée dans la Figure12. La valeur de chaque pixel est soustraite par le pixel précédent (déjà codé).

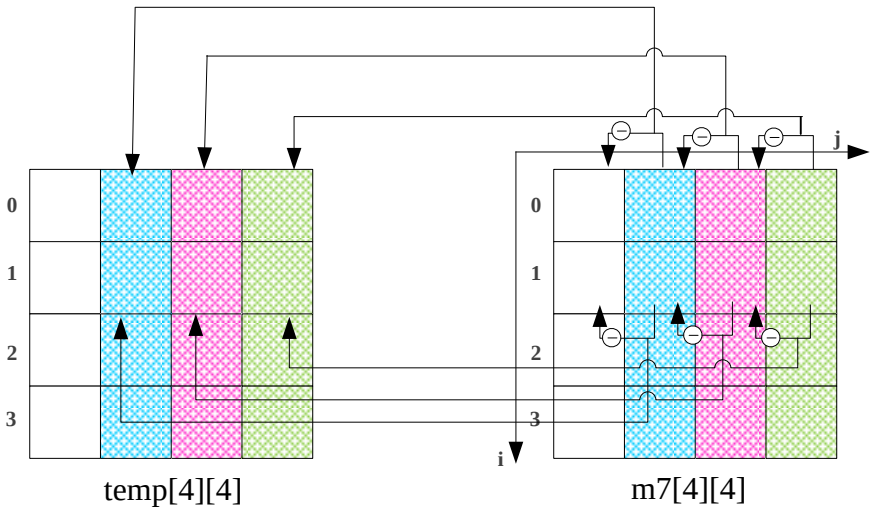


Figure II. 12. *Résiduel DPCM 4×4 : Prédiction horizontale*

Prédiction verticale:

La première ligne sera déplacée comme elle est dans la matrice temporelle puis une opération de soustraction est appliqué à la deuxième ligne pixel par pixel jusqu'à la fin de la matrice. L'opération de soustraction est représentée dans la Figure13. La valeur de chaque pixel est soustraite par le pixel précédent (déjà codé). Autrement dit, pour prédire un pixel, j'utilise le pixel le plus proche de ce dernier.

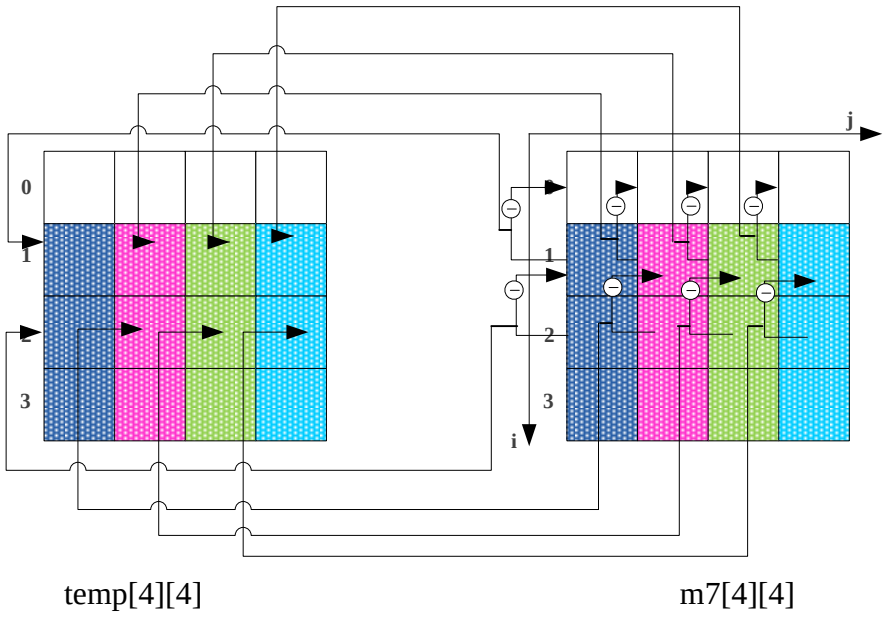


Figure II. 13. *Résiduel DPCM 4×4 : Prédiction verticale*

Concernant l'étape de la reconstruction, une opération d'addition est appliqué à la deuxième colonne pixel par pixel jusqu'à la fin de la matrice (voir Figure 14).

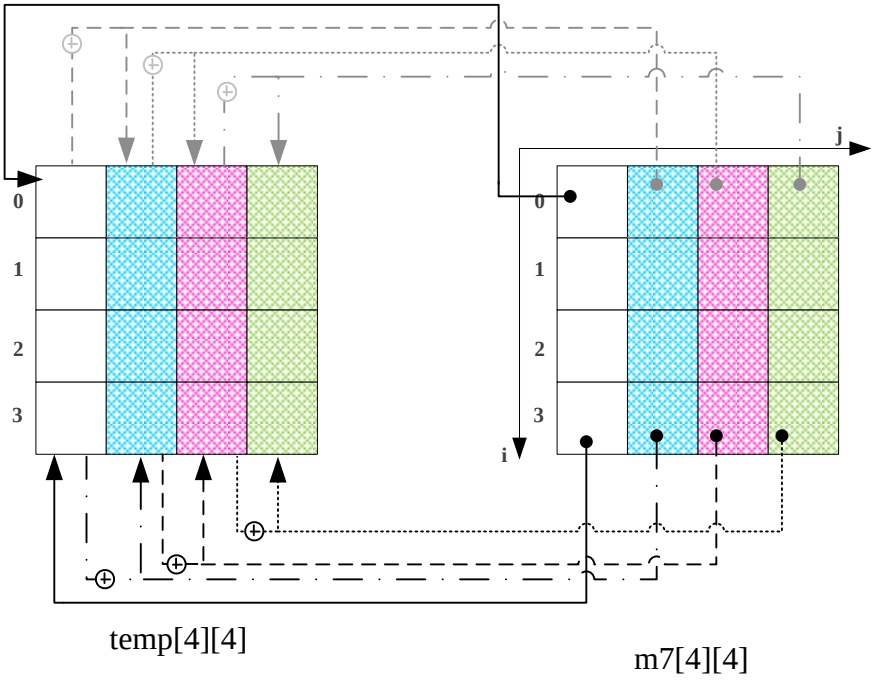


Figure II. 14. *Inverse Résiduel DPCM 4×4 : Prédiction horizontale*

Dans le décodage, pour la prédiction verticale, une opération d'addition est appliqué à la deuxième ligne pixel par pixel jusqu'à la fin de la matrice (voir Figure 15).

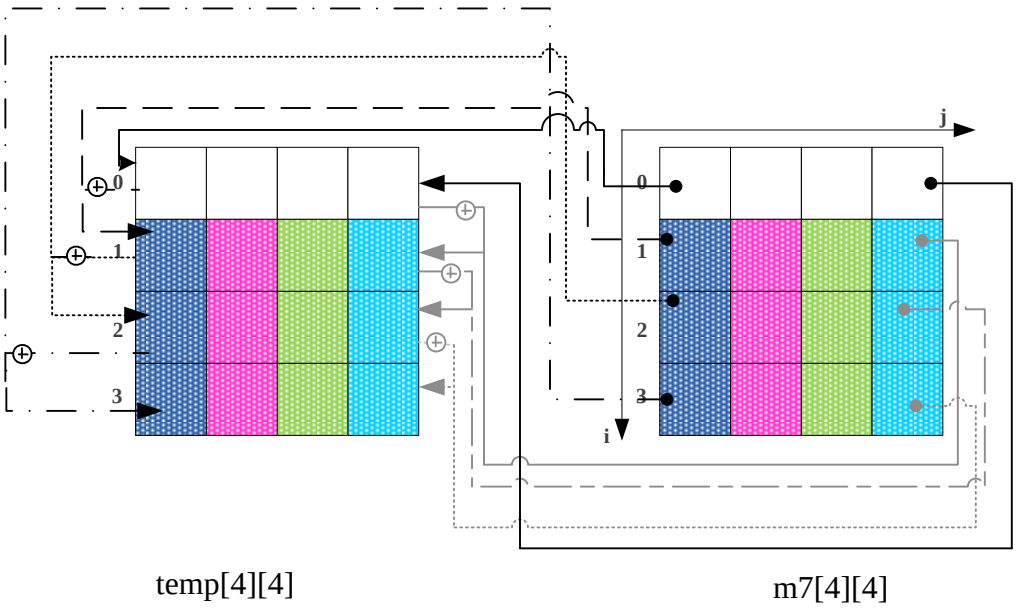


Figure II. 15. *Inverse Résiduel DPCM 4×4 : Prédiction verticale*

Pour améliorer la précision de prédiction et atteindre une meilleure performance dans la prédiction intra du H.264 / AVC, la technique DPCM est adoptée pour les blocs 8×8.

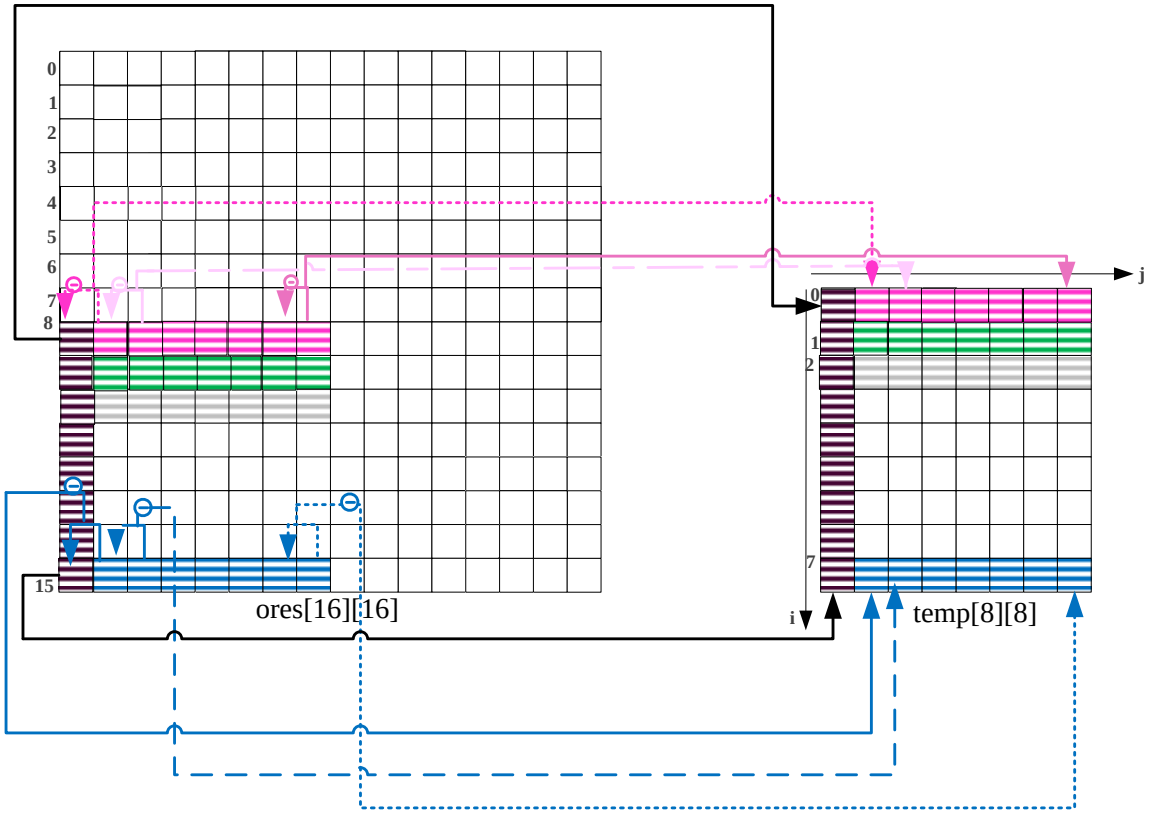


Figure II. 16. *Résiduel DPCM 8×8 : Prédiction horizontale*

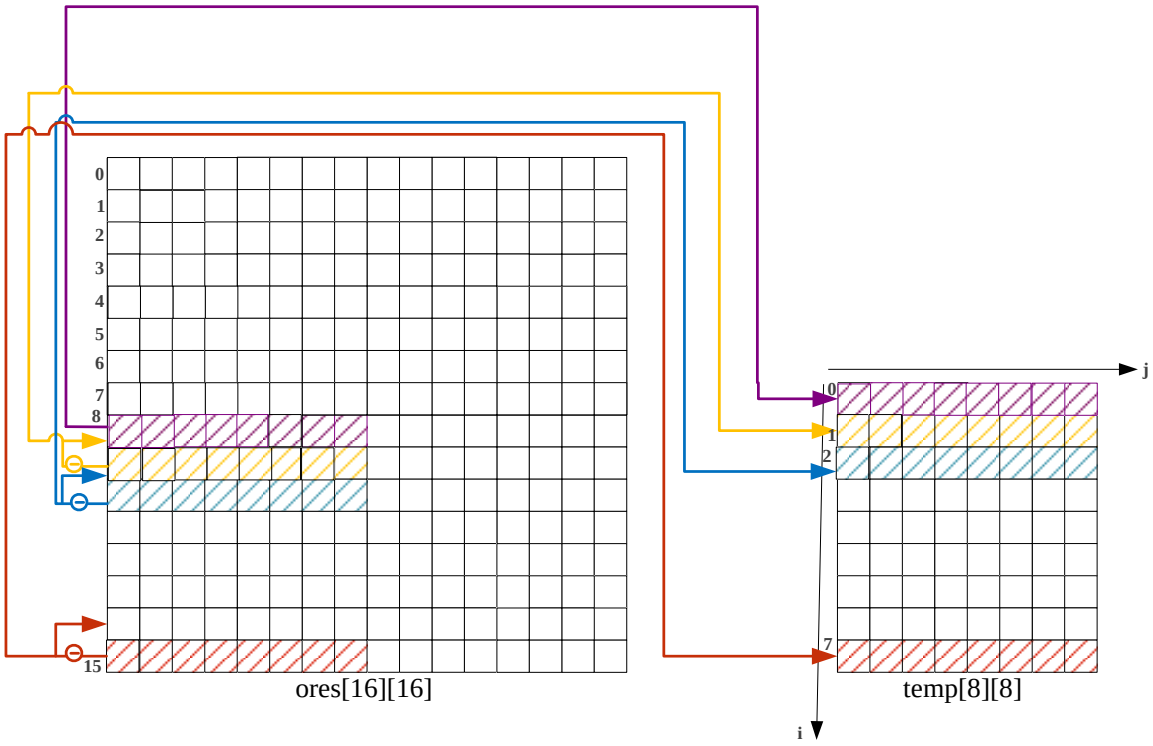


Figure II. 17. *Résiduel DPCM 8×8 : Prédiction verticale*

Dans le décodage, pour les blocs 8×8, une opération d'addition est appliqué à la deuxième colonne pixel par pixel jusqu'à la fin de la matrice (voir Figure 18).

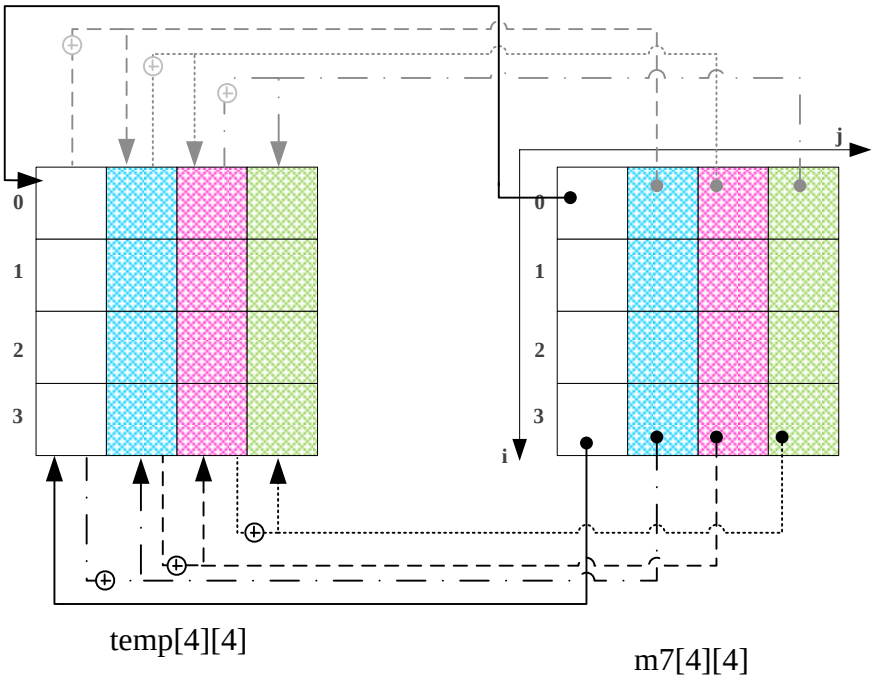


Figure II. 18. *Inverse Résiduel DPCM 8×8 : Prédiction horizontale*

Pour la Prédiction verticale des blocs 8×8, la reconstruction des pixels est réalisée par des opérations d'addition (voir Figure 19).

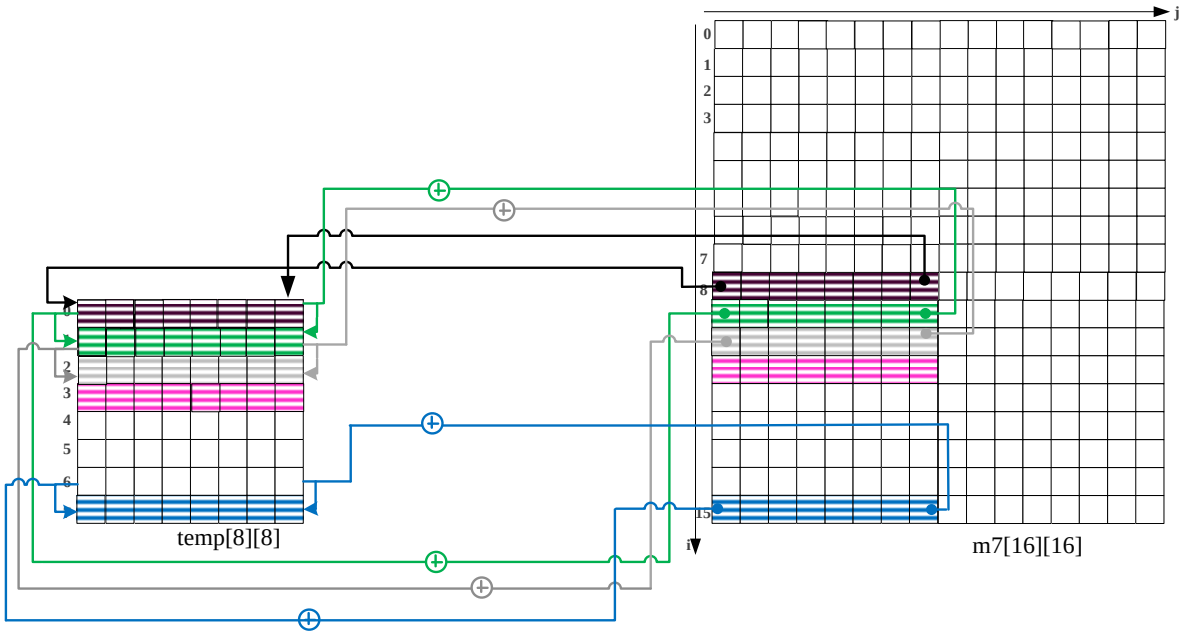


Figure II. 19. *Inverse Résiduel DPCM 8×8 : Prédiction verticale*

3.3. *Autres techniques*

Récemment, comme les demandes pour les domaines d'applications tels que post-traitement, l'édition de studio, la diffusion de contenu et le stockage vidéo à haute définition ont augmenté, le groupe JVT a développé une extension à la norme H.264 / AVC, appelée FREXt [22, 23].

Le FREXT supporte quatre nouveaux profils : " high profile ", " high 10 profile ", " high 4:2:2 profile ", " high 4:4:4 profile ". Le dernier type de profil offre des fonctionnalités supplémentaires telles que le codage sans perte et la transformation entière de couleur résiduelle pour le codage des séquences vidéo RGB. Il supporte le format de chrominance 4:4:4 avec 12 bits de précision (comme indiqué dans la Figure 20).

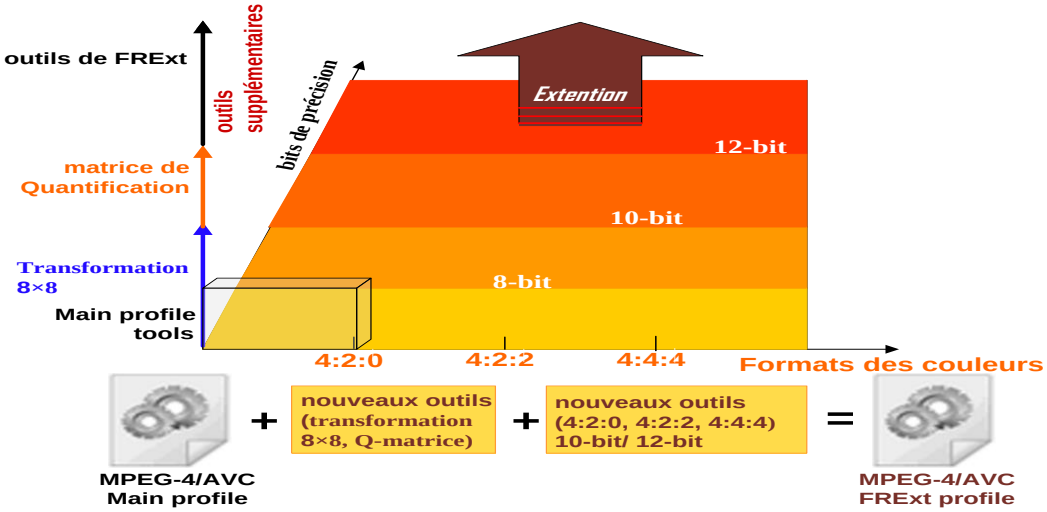


Figure II. 20. *Les outils supplémentaires du norme H.264 / AVC FRExt*

Dans le profil "high 4:4:4 profile", neuf modes de 8 ×8 luma sont ajoutés à l'algorithme intra. L'encodeur "high 4:4:4 profile" améliore la section de transformation en ajoutant une transformation entière 8×8, ce qui permet de préserver les détails fins des images trop texturées. L'adoption de la transformation 8 × 8 est très bénéfique pour les applications à haute résolution et à débit élevé.

Afin d'augmenter la performance du codage intra sans perte dans le standard H.264 / AVC FRExt, JVT a adopté une nouvelle méthode en utilisant le DPCM basée sur l'interpolation des pixels [24]. Dans cette méthode, les étapes de transformation et de quantification sont toutes contournées et chaque échantillon est prédit par un pixel voisin dans la direction déterminée par la prédiction DPCM. Les résiduelles sont alors directement codés par un codage entropique.

4. Résultats expérimentaux et discussion

4.1. *Analyse et Etude du codec JM*

Le téléchargement de JM est accessible du site web [25]. L'extraction du fichier "jm 18.3.zip" abouti à plusieurs fichiers solutions d'extension ".sln" et plusieurs fichiers de configuration et plusieurs dossiers. La solution "jm_vc7.sln" pour visual studio 2003, "jm_vc8.sln" pour visual studio 2005 et "jm_vc9.sln" pour visual studio 2008. La sélection de la solution utilisée dépend de la version de "visual studio" installée.

Il existe deux types de fichiers de configurations :

Les fichiers de configurations qui décrivent en détail les séquences à coder.

Les fichiers de configurations principaux qui gèrent la solution complète.

La sélection du fichier de configuration se fait selon le besoin de l'utilisation. Ce choix est effectué suivant le profil utilisé (main, baseline, extended), les formats de la vidéo (format yuv 4:4:4, yuv 2:2:2, yuv 4:2:0 et plusieurs autres caractéristiques.

La solution se compose de 4 programmes :

lencod qui encode la vidéo.

ldecod qui la décode.

rtpdump qui permet d'analyser le contenu des paquets RTP.

rtp_loss qui fait la simulation des paquets RTP.

Il y a trois types de dossiers :

Chaque programme a un dossier approprié qui inclut les fichiers sources et headers.

Un dossier de documentation du JM.

Un dossier bin contenant "lencod.exe", " ldecod.exe " et les séquences d'entrée et de sortie.

4.2. *Description des paramètres du codage du code JM*

Le fichier de configuration contient un très grand nombre de paramètres. Donc, je ne vais pas les commenter tous (ils sont bien détaillés dans le dossier de documentation associé du JM) mais je vais expliquer le plus important dans le tableau suivant :

Tableau II. 1. *Les paramètres du codage du code JM*

Paramètres	Commentaires
InputFile = "foreman_part_qcif.yuv"	# séquence à coder.
StartFrame = 0	# première frame à coder, on peut par exemple, commencer le codage du frame 3.
FramesToBeEncoded = 3	# Nombre de frame total à coder.
FrameRate = 30.0	# Nombre de frames par seconde (doit être compris entre 0.1 et 100.0). la valeur par défaut est 30.0.

SourceWidth = 176	# Largeur de l'image de la séquence originale.
SourceHeight = 144	# Hauteur de l'image de la séquence originale.
ReconFile = "test_rec.yuv"	# Le nom de la séquence reconstruite ou tu peux mettre l'emplacement de la séquence reconstruite.
OutputFile = "test.264"	# Le nom de la séquence de sortie.
ProfileIDC	# on peut choisir le type de profil baseline, main, extented, FRExt profile.cette valeur est également en corrélation avec le format yuv. on peut mettre par exemple 244 pour le profil High 4:4:4 et 122 pour le High 4:2:2.
QPISlice	# Valeur du paramètre de quantification de la frame I (doit être entre 0 et 51).
QPPSlice	# Valeur du paramètre de quantification de la frame P (doit être entre 0 et 51).
QPBSlice	# Valeur du paramètre de quantification de la frame B (doit être entre 0 et 51).
SymbolMode	# le codage entropique utilisé CABAC ou CAVLC.
FrameSkip	# Nombre de frames sautés lors du codage.
NumberBFrames	# Nombre de frames B
PartitionMode = 0	# donne l'information de partitionner ou nom en slice.si ce paramètre égale à 0 donc il n'y a pas de partitions, si ce paramètre égale à 1 signifie le nombre de partitions en slice égale à 3.
YUVFormat	# type de format yuv. 1 correspond au format 4:2:0, 2 correspond au format 4:2:2, et 3 correspond au format 4:4:4.
IDR Period	# période des images IDR dans la séquence codée. Une valeur de 0 implique que seule la première image sera codée comme IDR.
INTRA Period	# période des images codés I (non IDR) dans la séquence coder.
Transform8x8Mode = 1	# la valeur de ce paramètre par défaut égale à 0 donc on utilise seulement la transformation 4×4. La modification de ce paramètre à 1 permet l'utilisation supplémentaire de la transformation 8×8. si ce paramètre égale à 2, on utilise

	seulement la transformation 8×8.
DFParametersFlag = 0	# le filtre sera désactivé
UseRDOQuant	# ce paramètre permet de maintenir la valeur de QP d'entrée.
FastCrIntraDecision	# Avant de déterminer le mode de codage final, une décision de mode intra chroma séparée a été effectué. Ce paramètre fournit une vitesse d'encodage significative.
CbQPOffset	# Définition du paramètre de quantification (QP) qui sera utilisé pour coder les composants Cb. La valeur peut être dans la plage (-51 .. 51). Ce paramètre est spécifique seulement pour le profil FRExt. Pour les autres profils, on utilise "ChromaQPOffset".

Pour faire les différents tests sur JM, on doit tout d'abord initialiser les paramètres dans le fichier de configuration.

Afin de s'assurer que ces paramètres sont effectivement modifiés, il faut vérifier la console de sortie " lencod.exe "(Figure 21) :

```
Setting Default Parameters...
Parsing Configfile encoder.cfg.....................
.....................................................................
Warning: Hierarchical coding or Referenced B slices used.
       Make sure that you have allocated enough references
in reference buffer to achieve best performance.
------------------------------ JM 18.6 (FREext) -------------------------------
Input YUV file                 : foreman_part_qcif.yuv
Output H.264 bitstream         : test.264
Output YUV file                : test_rec.yuv
YUV Format                     : YUV 4:2:0
Frames to be encoded           : 3
Freq. for encoded bitstream    : 30.00
PicInterlace / MbInterlace     : 0/0
Transform8x8Mode               : 1
ME Metric for Refinement Level 0  : SAD
ME Metric for Refinement Level 1  : Hadamard SAD
ME Metric for Refinement Level 2  : Hadamard SAD
Mode Decision Metric           : Hadamard SAD
Motion Estimation for components  : Y
Image format                   : 176x144 (176x144)
Error robustness               : Off
Search range                   : 32
Total number of references     : 5
References for P slices        : 5
References for B slices (L0, L1)  : 5, 1
Sequence type (I-B-P-B-P)              : Hierarchy (QP: I 28, P 28, B 30)
Entropy coding method          : CABAC
Profile/Level IDC              : (100,40)
Motion Estimation Scheme       : EPZS
EPZS Pattern                   : Extended Diamond
EPZS Dual Pattern              : Extended Diamond
EPZS Fixed Predictors          : Aggressive
EPZS Aggressive Predictors     : Disabled
EPZS Temporal Predictors       : Enabled
EPZS Spatial Predictors        : Enabled
EPZS Threshold Multipliers     : (1 0 2)
EPZS Subpel ME                 : Basic
EPZS Subpel ME BiPred          : Basic
Search range restrictions      : none
RD-optimized mode decision     : used
Data Partitioning Mode         : 1 partition
Output File Format             : H.264/AVC Annex B Byte Stream Format
---------------------------------------------------------------------------
Frame    Bit/pic   QP   SnrY    SnrU    SnrV    Time(ms) MET(ms) Frm/Fld Ref
---------------------------------------------------------------------------
00000(NVB)    320
00000(IDR)  21392   28  37.638  41.571  43.231     303       0   FRM   3
00002( P )   8928   28  37.242  41.102  42.738     639     239   FRM   2
00001( B )   2560   30  36.634  41.296  42.970    1343     888   FRM   0
---------------------------------------------------------------------------
Total Frames:  3
Leaky BucketRateFile does not have valid entries.
Using rate calculated from avg. rate
Number Leaky Buckets: 8
    Rmin    Bmin    Fmin
  328800   21392   21392
  411000   21392   21392
  493200   21392   21392
  575400   21392   21392
  657600   21392   21392
  739800   21392   21392
  822000   21392   21392
  904200   21392   21392
------------------ Average data all frames  -----------------------------------
Total encoding time for the seq. :   2.286 sec (1.31 fps)
Total ME time for sequence       :   1.128 sec
Y { PSNR (dB), cSNR (dB), MSE }  : {  37.171,  37.152,  12.52913 }
U { PSNR (dB), cSNR (dB), MSE }  : {  41.323,  41.318,   4.79993 }
V { PSNR (dB), cSNR (dB), MSE }  : {  42.980,  42.975,   3.27751 }
Total bits                     : 33200 (I 21392, P 8928, B 2560 NVB 320)
Bit rate (kbit/s)  @ 30.00 Hz  : 332.00
Bits to avoid Startcode Emulation : 18
Bits for parameter sets        : 320
Bits for filler data           : 0
---------------------------------------------------------------------------
Exit JM 18 (FREext) encoder ver 18.6
```

Fichier de configuration utilisé

Séquence d'entrée et de sortie

Format YUV

Nombre de frames I et P à coder

Nombre de frames B à coder

Fréquence pour le codage. Cette valeur est égale au frame rate (nombre de frame codé par seconde)

Forme de la séquence et rappel des QP pour chaque type de frame

Nombre total de frames codées

Temps total de codage

Temps total de calcul de l'EM

Valeur du PSNR pour chaque composante

Nombre total de bit (bits utilisés pour les frames I, P, B et l'entête)

Débit

Figure II. 21. *La console de sortie " lencod.exe "*

4.3. *Critères d'évaluations*

Les évaluations ont été obtenues en termes de pourcentage de gain sur la taille totale de la séquence "total bits", le gain de temps et le taux de compression. Ces critères sont calculés comme suit:

$$\text{Gain en taille (\%)} = \frac{\text{Total bits}_{\text{CAVLC}} - \text{Total bits}_{\text{CABAC}}}{\text{Total bits}_{\text{CAVLC}}} \times 100 \quad (6)$$

$$\text{Gain en temps (\%)} = \frac{\text{Temps d'encodage}_{\text{CABAC}} - \text{Temps d'encodage}_{\text{CAVLC}}}{\text{Temps d'encodage}_{\text{CABAC}}} \times 100 \quad (7)$$

$$\text{Taux de compression} = \frac{\text{Taille de fichier d'origine (octet)}}{\text{Taille de fichier compressé (octet)}} \quad (8)$$

4.4. *Les conditions expérimentales*

Les expériences ont été effectuées sur le code JM du standard H.264 "JM 18.3" [25, 26]. Nous avons utilisé un ensemble de 5 séquences CIF (Common Intermediate Format) et un autre de 18 séquences pour cinq résolutions différentes (416×240 /832×480 /1280×720/1920×1080/2560×1600) [27, 28]. Les détails sur les séquences de test et les classes de séquence qui ont été utilisées pour les comparaisons suivantes sont résumés dans le Tableau 2.

Chaque image dans les séquences de test est codée en mode intra. Les expériences ont été effectuées sur plusieurs formats YUV 4: 4: 4 après l'utilisation d'un outil YUVtool (version 3.0) pour la conversion de YUV 4: 2: 0 à YUV 4: 4: 4.

YUVtool est utilisé pour la comparaison entre les deux séquences (c'est à dire la séquence initiale et la séquence codée) en termes de qualité métrique PSNR (Peak Signal to Noise Ratio).

MSU Video Quality Measurement Tool (version 2.01 BETA) est aussi utilisé pour le calcul du PSNR et MSE (Mean Square Error).

Tableau II. 2. *Les séquences de test*

Classes	Séquences	Nombre d'images	"Frame rate" (fps)	Résolution	"Length"	Bit de précision	"LevelIDC"
A	PeopleOnStreet	150	30	2560×1600	5s	8	50
	Traffic	150	30				50
B	Kimono1	240	24	1920×1080	10s	8	42
	ParkScene	240	24				42
	Cactus	500	50				42
	BQTerrace	600	60				42
	BasketballDrive	500	50				42
C	RaceHorses	300	30	832×480	10s	8	40
	BQMall	600	60				40
	PartyScene	500	50				40
	Basketball Drill	500	50				40
D	RacesHorses	300	30	416×240	10s	8	40
	BQSquare	600	60				40
	BlowingBubbles	500	50				40
	BasketballPass	500	50				40
E	vidyo1	600	60	1280×720	10s	8	32
	vidyo3	600	60				32
	vidyo4	600	60				32

4.5. Les étapes d'implémentation du filtre de prédiction de Calvagno et al. amélioré

Dans cette section, je vais présenter les étapes d'implémentation du filtre de prédiction. Notre but est d'intégrer cet algorithme de prédiction dans la référence software JM et évaluer ces performances par rapport à la technique DPCM de la littérature précédemment décrit. Nous avons commencé par le calcul de la matrice A_i qui est définie par la formule :

$$A_i = p_i \, p_i^T \tag{9}$$

Nous avons fixé à deux la distance relative au voisinage du pixel courant lors du calcul de la matrice Ai, ce choix aboutit à un voisinage de six pixels (voir Figure 22).

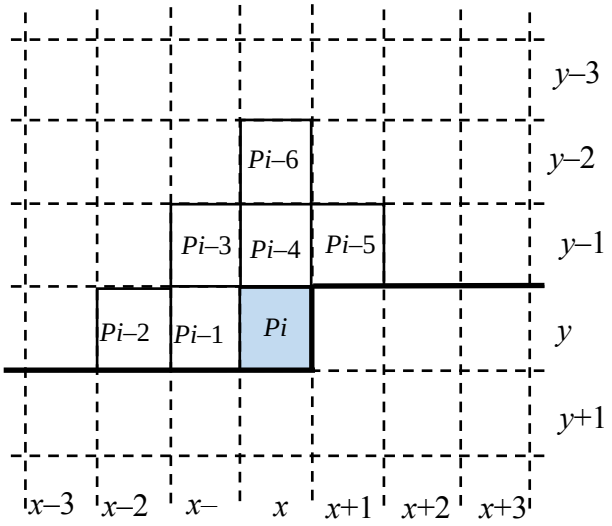
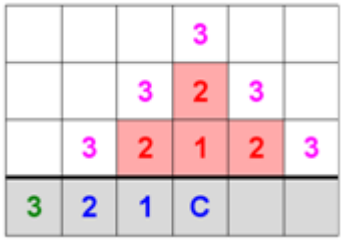


Figure II. 22. *La notation des pixels utilisés*

L'une des premières difficultés que nous avons rencontrée est la gestion des effets de bord au voisinage de six pixels lors du calcul de vecteur pi.

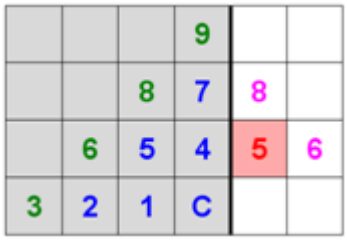
Avec $p_i = [p_{i-1},\ldots..p_{i-6}]^T$ (10)

Pour la gestion des frontières, nous avons remplacé les pixels qui manquent avec la valeur des pixels les plus proches connus (voir Figure 23).

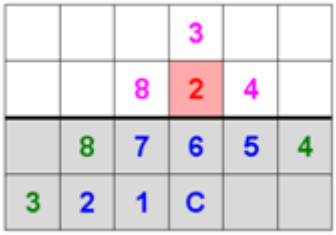


(a)*Bordure du haut extrême*

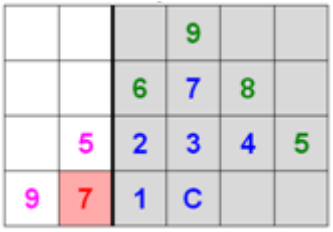
(b) *Bordure de côté de gauche extrême*



(c) *Bordure du côté de droite*



(d) *Bordure du haut non extrême*



(e) *Bordure de côté de gauche non extrême*

Figure II. 23. *La gestion des cinq configurations de bords*

En rouge : distance de Manhattan de deux (c'est notre cas).

En rose : distance de Manhattan de trois (c'est la version originale de GLICBAWLS).

Pixels en gris : valeurs initialement existantes dans l'image.

Pixels blancs : valeurs ajoutées à l'extérieur de l'image.

En fait, la gestion de frontières peut être choisie arbitrairement:

- ✓ Pour les grandes images, elle n'affecte pas trop le taux de compression final, mais elle le fait pour des petites images.
- ✓ Seuls les pixels déjà codés peuvent être utilisés.

Selon la Figure 23, nous avons divisé l'image de format CIF (352×288) sous forme de six cas possibles de voisinage comme suit :

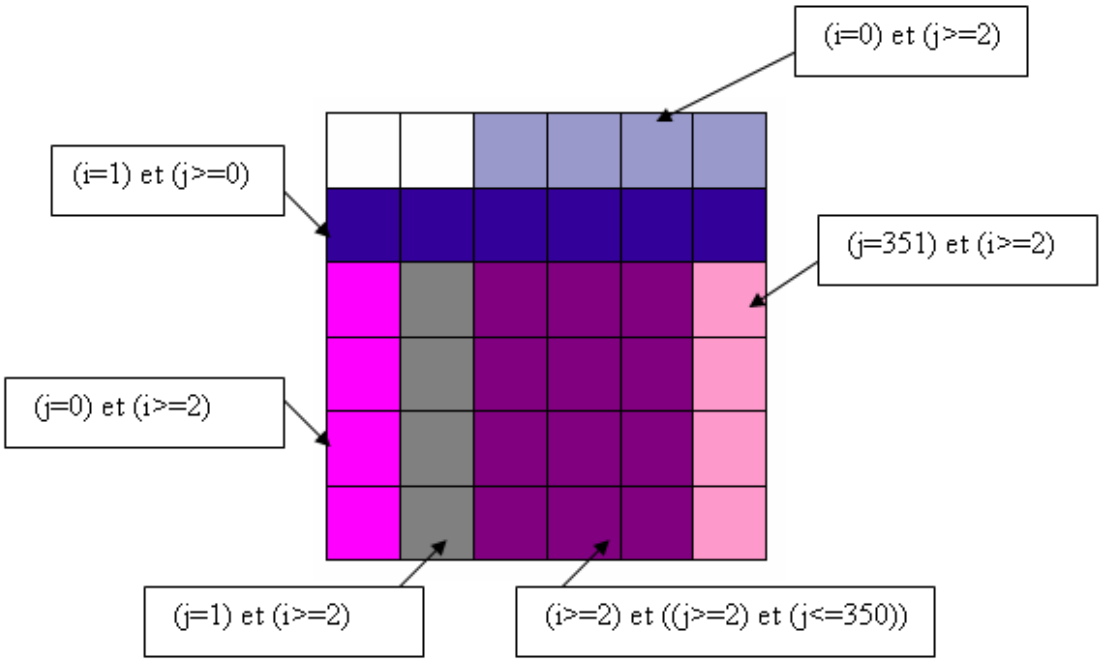


Figure II. 24. *Les six cas possibles de voisinage*

Ensuite, nous traitons cas par cas afin de déterminer le vecteur $p_i = [p_{i-1}, \ldots, p_{i-6}]^T$, pour chaque pixel, qui dépend du format de filtre de prédiction selon la Figure 22 et par conséquent on déduit la matrice A_i .

Nous avons géré les problèmes relatifs au calcul de la matrice d'autocorrélation A_c qui nécessite la sommation de 30 matrices A_i pour chaque pixel prédit et les problèmes de tailles importantes de mémoire requise pour le stockage des matrices Ai.

Finalement, nous avons effectué le calcul de la matrice A_c à partir de la sommation de 4 matrices A_i qui correspond aux 4 pixels voisins du pixel courant pour la première version. La complexité de calcul sera inférieure par rapport à la sommation de 30 matrices Ai. D'où la matrice A_c est définie comme :

$$A_c = \sum_{i=1}^{N=4} 0.8^{|x_C - x_i| + |y_C - y_i|} A_i \qquad (11)$$

Et le vecteur :

$$b_C = \sum_{i=1}^{N=4} 0.8^{|x_C - x_i| + |y_C - y_i|} b_i \qquad (12)$$

où le poids $0{,}8^{|x_c - x_i| + |y_c - y_i|}$ est utilisé pour diminuer l'influence des pixels les plus lointains de la position actuelle (xc,yc).

Dans les équations (11) et (12), N est le nombre de pixels précédemment codés.

L'algorithme de l'annexe 1 montre la procédure de calcul de la matrice A_c selon le filtre de quatre pixels voisins du pixel courant (voir Table 1 dans l'annexe 1). Ce filtre nécessite l'inversion de la matrice A_c (de taille 6×6) pour chaque pixel.

Plusieurs méthodes d'inversion de matrice et de résolution des systèmes d'équations peuvent être utilisées : Après une étude de ces méthodes, nous avons opté pour la méthode la plus rapide et la plus simple à implémenter : il s'agit de la méthode de Gauss Jordan. Une partie de cette technique se manifeste dans la Table 2 de l'annexe 2.

Méthode de Gauss Jordan

C'est la méthode la plus utilisée. Pour la présenter, nous allons prendre l'exemple d'un système de 4 équations à 4 inconnues :

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \quad (13)$$

Dans la méthode du pivot, on choisit successivement chaque ligne comme ligne pivot ; le pivot étant le 1er élément non nul de la ligne [29].

Ainsi, on divise la ligne n° 1 du système par a_{11} :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} & a'_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y'_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \quad (14)$$

On annule le 1er terme de chacun des autres lignes : à la 2ème ligne, on retranche la 1ère multipliée par a_{21}, à la 3ème ligne, on retranche la 1ère multipliée par a_{31}, à la 4ème ligne, on retranche la 1ère multipliée par a_{41}.

Le système devient :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} & a'_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{pmatrix} \quad (15)$$

La 2ème ligne est considérée maintenant comme une ligne pivot, et a'_{22} comme un élément pivot. On répète sur cette 2ème ligne les opérations précédentes, et on obtient après division de cette ligne par a'_{22} :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} & a'_{14} \\ 0 & 1 & a''_{23} & a''_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y'_1 \\ y''_2 \\ y'_3 \\ y'_4 \end{pmatrix} \quad (16)$$

On annule les autres termes de la seconde colonne ; c'est à dire : à la 1ère ligne, on retranche la seconde multipliée par a'_{12}, à la 3ème ligne, on retranche la 2ème multipliée par a'_{32}, à la 4ème ligne, on retranche la 2ème multipliée par a'_{42}.
On obtient :

$$\begin{pmatrix} 1 & 0 & a''_{13} & a''_{14} \\ 0 & 1 & a''_{23} & a''_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & a'_{43} & a''_{44} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y''_1 \\ y''_2 \\ y''_3 \\ y''_4 \end{pmatrix} \quad (17)$$

On considère ensuite la 3ème ligne comme pivot, puis la 4ème ligne ; ce qui donne :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1^{(4)} \\ y_2^{(4)} \\ y_3^{(4)} \\ y_4^{(4)} \end{pmatrix} \textbf{Soit la solution du système :} \begin{cases} x_1 = y_1^{(4)} \\ x_2 = y_2^{(4)} \\ x_3 = y_3^{(4)} \\ x_4 = y_4^{(4)} \end{cases} \quad (18)$$

NOTE. —Cette méthode peut servir à inverser la matrice A_c de départ en appliquant les différentes étapes à une matrice identité.

Les termes des matrices sont introduits en réels double précision (8 octets par terme). Nous avons vérifié les résultats obtenus en utilisant les utilitaires Matlab (version 7.6) et Mapple (version 12.0). Il n'est pas possible de connaître les méthodes adoptées par ces utilitaires pour effectuer l'inversion des matrices. Les résultats obtenus sont proches mais pas identiques.
Nous avons réalisé l'implémentation en langage C avec Microsoft visual C++ 6.0 sur une machine Pentium 4 de fréquence d'horloge 2GHz et de capacité RAM 256 Mo. Le temps d'exécution de la méthode de Gauss Jordan est 420 ms. Le temps d'exécution de l'algorithme de prédiction jusqu'au calcul de b_c est 50.16 minutes. Il nous reste à déduire les coefficients de prédiction w_i qui sont calculés de manière adaptative en résolvant le système linéaire :

$$A_c w = b_c \quad (19)$$

avec w est le vecteur constitué des N termes w_i.
Par conséquent, les valeurs de l'image sont prédites en utilisant l'expression suivante :

$$\hat{p}_i = \sum_{k=1}^{6} W_k P_{i-k} \quad (20)$$

où $\hat{p}_i$ est la valeur prédite, p_{i-k}sont les pixels voisins du pixel courant p_i avec une distance inférieure ou égale à deux.

Afin d'obtenir une qualité d'implémentation précise et standard de n'importe quel besoin d'utilisateur, il faut ajouter un code qui calcule le déterminant de chaque matrice A_c pour savoir est ce que la matrice est inversible ou pas.

4.6. *Comparaison avec des travaux précédents*

La première version du codage sans perte de la norme H.264/AVC (H.264-LS) est une technique PCM (pulse code modulation) où les échantillons de séquence d'origine sont envoyés directement au codage entropique sans effectuer les étapes de prédiction, de transformation et de quantification. Beaucoup de recherche ont essayé d'introduire des améliorations à cette méthode de H.264-LS pour améliorer ses performances.

En fait, H.264-LS-DPCM (Differential Pulse Code Modulation) est une méthode de codage sans perte améliorée de la norme H.264-LS. Dans cette version, le processus de prédiction est effectué avec la technique DPCM au lieu de la prédiction basée sur les blocs. Pour obtenir plus de précision dans l'étape de prédiction, le concept DPCM est appliqué où chaque échantillon est prédit par ses échantillons voisins immédiats.

L'algorithme de codage en deux couches (la méthode de Chien et al.) [30] a été proposé pour améliorer encore les performances de la norme H.264 basé sur le codage sans perte (H.264-LS-DPCM). Cette méthode est basée sur le codage vidéo ordinaire avec perte du standard H.264 avec une autre couche pour compenser le bitstream avec perte.

En effet, cette technique est constituée de deux couches : une couche EL sert à coder les erreurs d'informations réalisées par transformation et quantification et la couche BL inclut la chaine ordinaire du codage avec perte de la norme H264/AVC. Ainsi, combinant les outils puissants de codage avec perte dans le système de codage sans perte serait plus efficace.

Une modification a été appliquée dans cette technique pour obtenir de meilleures performances (résultats dans le Tableau 4) [1]. Dans notre travail, nous avons suggéré de modifier les paramètres de codage de JM. Ainsi, nous avons ajusté la référence software JM de la norme H.264 / AVC : la configuration a adopté les paramètres résumés dans le Tableau 3.

Tableau II. 3. *Les paramètres de codage pour la norme H.264 / AVC*

Paramètres	**JM_18.3**
ProfileIDC	244 (High 4:4:4)
YUVFormat	3
RDOptimization	1 (enabled)
Transform8x8Mode	1
DFDisableRefISlice	0 (enabled)
FastCrIntraDecision	1 (enable)
CbQPOffset	0
LosslessCoding	1 (lossless)
QP Quantification parameter (QP)	0(lossless)
UseRDOquant	0
Deblockingfilter	Non
OutFileMode	0
IDRPeriod	0
IntraPeriod	1 (only Intra Coding)
NumberBFrames	0
FrameSkip	0
Optimisations Software	Les optimisations de visual studio: Augmenter la vitesse , En faveur d'un code rapide, Débogage multithread…etc
SymbolMode	1 (CABAC) / 0 (CAVLC)

La référence software (JM 18.3) est utilisée pour le standard H.264/AVC FRExt "high 4:4:4 profil". La configuration de l'encodeur de profil high 4:4:4 [31, 23] et la configuration du codage sans perte sont mentionnés dans le Tableau 3.

La configuration du codage sans perte est la suivante : l'encodeur se transforme en codage sans perte lorsque le paramètre "lossless coding" est modifié à 1 et le terme QP est réglé à 0. Un problème est apparu: la valeur de QP (paramètre de quantification) ne reste pas stable à zéro pour toutes les images. En effet, ce paramètre change de manière aléatoire en passant d'une frame à la suivante durant toute la séquence puis revient à la valeur correcte. Ce problème a été résolu en donnant à la variable « UseRDOquant » la valeur 0 pour assurer le mode sans perte.

L'utilisation du variable "UseRDOquant" égal à zéro aboutit à quatre avantages :

• au cours de la séquence entière, la valeur de QP reste stable à zéro.

• Une diminution de l'erreur quadratique moyenne (MSE) : pour chaque image, le MSE tend vers zéro.

• L'augmentation du rapport crête signal sur bruit (PSNR): pour chaque image, le PSNR est pratiquement égal à 100.

• Une réduction légère du nombre total de bits.

En conséquence, on déduit que "UseRDOquant" influe sur la qualité de l'image.

« AddCabacZeroWords » est une fonction optionnelle intégré dans le code JM. Cette fonction permet d'envoyer des bits supplémentaires associés à la syntaxe élément du codeur CABAC. Grâce aux tests que je fais cette fonction fournit une augmentation du taux de compression. En effet, nous atteignons une réduction considérable du nombre total de bits lorsque nous ignorons la fonction "addCabacZeroWords" dans les fichiers source JM. Selon nos résultats, nous pouvons conclure que la fonction "addCabacZeroWords" n'affecte pas la qualité, mais elle change seulement le nombre total de bits.

Le filtre a joué le rôle de lissage du contenu vidéo. Comme l'opération de filtrage modifie la valeur du pixel (les valeurs des pixels sont très proches), le processus du filtrage sera désactivé. Le variable « OutFileMode » est mis à 0 pour obtenir en sortie le format de bitstream lisible.

Le paramètre "LevelIDC" dépend de la résolution supportée et du nombre d'images par secondes "FrameRate". Nous avons changé "LevelIDC" pour chaque résolution parce que nous avons utilisé des résolutions différentes des séquences dans notre test.

La liste de tous les levels de la norme H.264 avec leur résolution adéquate et le "FrameRate" correspondant est définie dans le support de la référence software JM.

Le mode du codage intra est effectué en mettant "IDRPeriod" à 0 et "IntraPeriod" à 1. Certains des paramètres du mode du codage inter, affichés dans le Tableau 3, ont été utilisés pour s'assurer que l'algorithme fonctionne pour le mode intra. En effet, "NumberBFrames" est le nombre des images codées B. Cette valeur doit être inférieure ou égal à celui de "FrameSkip". Lors du codage en mode intra, le nombre total d'images et le nombre des images à coder doivent être égaux. Pour cette raison, le " NumberBFrames " a été fixé à 0 selon la formule suivante :

$$[(F_{encoded} - 1)\,(N_{BFrames} + 1)] + 1 = T_{number\ of\ frames} \quad (21)$$

Avec :

$F_{encoded}$ = "FramesToBeEncoded".

$N_{BFrames}$= "NumberBFrames".

$T_{number\ of\ frames}$ = "TotalNumberOfFrames".

"FrameSkip" est le nombre des images à sauter lorsque la séquence d'entrée est codée dans le mode intra (c'est à dire les tranches B non utilisées). S'il est égal ou supérieur à 1, il permet la création des images B et la réduction de la fréquence d'images d'origine de moitié ("Framerate"). Quand le paramètre "FrameSkip" est mis à zéro, la valeur originale de "FrameRate" est maintenue et la création des images B et le saut des images sont évités. Ainsi, "FrameSkip" a deux avantages principaux.

Afin d'améliorer le temps d'exécution, la configuration de visual studio (version 2008) a été modifiée de DEBUG à RELEASE et certains des paramètres de configuration RELEASE ont été ajustés comme suit :

Optimisation : Augmenter la vitesse (/O2)

Privilège à la taille ou à la vitesse : En faveur d'un code rapide (/Ot)

Bibliothèque Runtime : Débogage multithread (/MTd)

Optimisation de l'ensemble du Programme : Oui (/GL)

Activation des Optimisations à Fibres sécurisées : Oui (/GT)

Expansion des Fonctions Inline : Tout ce qui est possible (/Ob2)

Activation des Fonctions Intrinsèques : Oui (/Oi)

Format des Informations de Débogage : Compatible C7 (/Z7)

Niveau d'avertissement : Niveau 3 (/W3)

Activation de la régénération minimale : Oui (/Gm)

Vérifications de base à l'exécution : Par défaut

Nous avons comparé nos résultats à ceux de H.264/AVC_LS, H.264 / AVC_LS_DPCM et de la méthode de Chien et al [19, 24, 30]. À partir du Tableau 4, nous pouvons constater que le gain en taille de la méthode H.264 / AVC_LS _DPCM atteint 4,6% mieux que H.264 / AVC_LS, et la méthode de Chien et al. présente une amélioration du gain que H.264 / AVC_LS environ 10,97%.Les résultats de simulation montrent que notre méthode proposée aboutit à un gain plus élevé que la méthode de Chien et al. et la méthode H.264 / AVC_LS _DPCM. Une grande amélioration a été remarquée dans le nombre total de bits grâce à l'approche améliorée (Tableau 4). En fait, cette approche améliorée aboutie à une réduction du nombre total de bits d'environ 35%, ce qui constitue une amélioration remarquable par rapport à H.264 / AVC_LS.

Tableau II.4. *Améliorations du "taille totale" (KB). [30]*

Séquences 50 images	Nombre des images	H.264/AVC_ LS	H.264/AVC_LS_ DPCM		Méthode de Chien et al.		Nos résultats	
		"Taille totale" (KB)	"Taille totale" (KB)	Gain en taille(%)	"Taille totale" (KB)	Gain en taille(%)	"Taille totale" (KB)	Gain en taille(%)
Foreman	300	8323	7955	4.42	7707	7.4	4894.49	41.193
Mobile	300	12346	11611	5.95	10573	14.36	7595.608	38.477
Akiyo	300	6805	6432	5.48	5944	12.65	4130.764	39.298
FootBall	260	8310	7929	4.58	7515	9.57	5695.575	31.461
WaterFall	260	10484	10213	2.58	9453	9.83	7390.84	29.503
Moyenne				4.6		10.76		35.986

4.7. *Une étude comparative entre CABAC et CAVLC*

Dans nos expériences, l'efficacité de codage de CABAC a été comparée à celle de CAVLC. Les Tableaux (5-7) résument les résultats de la comparaison entre les deux codages entropiques en termes de gain en taille, le gain de temps et le taux de compression. Une étude expérimentale a été détaillée pour démontrer l'importance du gain obtenu par CABAC en termes d'efficacité de compression et cette performance de gain atteint de 15 à 19% pour les hautes résolutions (Tableau 6) et de 15 à 17% pour le format CIF (Tableau 5).

Le MSE a été vérifié pour être égal à 0 pour toutes les séquences. Une MSE égale à 0 signifie que la séquence reconstruite correspond entièrement à celle originale. En d'autres termes, lorsque les données sont compressées et ensuite décompressées, les informations d'origine contenues dans ces données sont restituées sans aucune modification. Par conséquent, la méthode de compression sans perte est une opération complètement réversible.

Tableau II.5. *Pourcentage de gain en "Taille totale" pour le format CIF avec les codeurs entropiques CAVLC et CABAC*

Séquences	Nombre des images	"Taille totale"(octet) CAVLC	"Taille totale"(octet) CABAC	Gain en taille (%)
Foreman	300	31388086	26616879	15.200
Mobile	300	48622663	41196763	15.272
Akiyo	300	24798732	20442994	17.564
FootBall	260	27126181	23023533	15.124
WaterFall	260	39188147	32262732	17.672

Tableau II.6. *Pourcentage de gain en "Total bits" pour les 6 classes de séquences avec les codeurs entropiques CAVLC et CABAC*

Séquences	Taille de fichier d'origine (octet)	Taille de fichier compressé (octet) (avec CAVLC)	Taux de compression	Taille de fichier compressé(octet) (avec CABAC)	Taux de compression	Gain en taille (%)
PeopleOnStreet	921600000	582152405	1.583	488805102	1.885	16.034
Traffic	921600000	618014270	1.491	503971195	1.828	18.453
Kimono1	746496000	484486185	1.540	390948855	1.909	19.306
ParkScene	746496000	564843204	1.321	461550014	1.617	18.287
Cactus	1555200000	1276773427	1.218	1033631687	1.504	19.043
BQTerrace	1866240000	1481350916	1.259	1214641431	1.536	18.004
BasketballDrive	1555200000	1089261250	1.427	881251825	1.764	19.096
RaceHorses	179712000	130896177	1.372	109295312	1.644	16.502
BQMall	359424000	252607785	1.422	210517695	1.707	16.662
PartyScene	299520000	263474312	1.136	222329788	1.347	15.616
BasketballDrill	299520000	229435436	1.305	191205928	1.566	16.662
RaceHorses	44928000	34600695	1.298	29232016	1.536	15.516
BQSquare	89856000	75850373	1.184	64931726	1.383	14.394
BlowingBubbles	74880000	69171870	1.082	57880024	1.293	16.324
BasketballPass	74880000	46740051	1.602	39607468	1.890	15.260
vidyo1	829440000	425068349	1.951	342353045	2.422	19.459
Vidyo3	829440000	406951817	2.038	331108042	2.505	18.637
Vidyo4	829440000	395462740	2.097	321158659	2.582	18.789

Pour les vidéos non texturées telle que «Kimono1», «vidyo1», «Vidyo3» et «Vidyo4», nous pouvons voir dans le Tableau 6 que ces séquences donnent un taux de compression plus élevé (de 2,582 à 1,909). Le taux de compression est plus élevé pour les séquences vidéo: "vidyo1", "Vidyo3", "Vidyo4" et "Kimono1", car ces séquences présentent des redondances et des zones

uniformes. En effet, l'arrière-plan n'a pas changé pour ces quatre séquences. Ces séquences contiennent peu de mouvement qui peut être résumée dans les expressions faciales et les mouvements de la tête, du corps. Ces séquences montrent : un homme parle devant une caméra dans son bureau ("Vidyo1"), un enseignant parle face à une caméra dans une salle ("Vidyo3"), une réunion de trois personnes parlant devant une caméra dans une entreprise ("Vidyo4"), une femme marche lentement devant une caméra dans le jardin d'une maison ("Kimono1"). Par contre, les deux séquences "BasketballPass" et "BasketballDrive" ont des valeurs inférieures en termes du taux de compression car ces séquences sont très mouvementés et présentent de changement de scène (caméra bouge).

Tableau II.7. *Comparaison des temps de codage entre CABAC et CAVLC*

Séquences	**Temps de codage (CABAC) (sec)**	**Temps de codage (CAVLC) (sec)**	**Gain en temps (%)**
PeopleOnStreet	7188.351	4481.006	37.662
Traffic	7280.485	4696.652	35.489
Kimono1	5618.058	3646.393	35.095
ParkScene	6068.423	3821.441	37.027
Cactus	13669.341	8168.087	40.245
BQTerrace	15331.947	9643.101	37.104
BasketballDrive	17108.704	7760.484	54.640
RaceHorses	1399.408	881.127	37.035
BQMall	2833.678	1760.447	37.874
PartyScene	2637.322	1572.243	40.384
BasketballDrill	2503.462	1572.457	37.188
RaceHorses	378.126	226.123	40.199
BQSquare	740.884	451.165	39.104
BlowingBubbles	669.063	405.616	39.375
BasketballPass	3330.455	358.670	89.230
vidyo1	5621.100	3896.345	30.683
Vidyo3	5477.942	3648.314	33.399
Vidyo4	5417.386	3526.081	34.911

Le temps de codage de chaque codage entropique est donné dans le Tableau 7. Les deux séquences "BasketballPass" et "BasketballDrive" ont des valeurs très importantes en gain en temps de 89.2306 et 54.6401%, respectivement. Ils ont pris beaucoup de temps en utilisant CABAC comme codage entropique car ils ne contiennent pas trop de redondances et l'image est envoyée telle quelle. CAVLC s'adapte bien aux caractéristiques de ces séquences et optimise le temps d'exécution. Ils sont à la fois des séquences trop texturées et très

mouvementées. L'arrière-plan de ces séquences est variant. Ils constituent aussi une vue prise par une caméra placée à une grande distance.

CABAC fournit une efficacité de codage plus élevée que le CAVLC. Cependant, CAVLC fournit un temps d'exécution très court. D'autre part, cette excellente efficacité du CABAC a été réalisée au détriment d'une complexité de calcul significativement plus élevée. Pour cette raison, nous pouvons dire que CABAC est plus complexe, mais aussi plus efficace que CAVLC. CABAC peut généralement être utilisé pour des applications qui exigent les hautes résolutions, et ainsi c'est le seule codage entropique utilisé dans la norme vidéo HEVC [32].

5. Conclusion

La perte d'informations est le critère le plus important pour les algorithmes de compression, car elle affecte directement le débit et la qualité de l'image ou de la vidéo. C'est pourquoi une grande partie de la recherche s'est concentrée sur l'étude de la compression sans perte: elle engendre une augmentation du bitrate sans dégrader la qualité visuelle.

Cette chapitre a abordé le codage vidéo sans perte pour la norme H.264 / AVC avec une étude comparative des deux codages entropiques: CABAC et CAVLC. En fait, ce travail vise à cerner les différentes méthodes de compression vidéo sans perte basées sur des modifications de la norme H.264/AVC et la maitrise des problèmes pratiques relatifs à l'une de ces méthodes (c'est dire la méthode hiérarchique). Dans ce cadre, il est impératif de distinguer la limite pratique entre compression vidéo sans perte ; pouvoir manipuler le code source du JM et ses paramètres. La méthode améliorée fournit une amélioration de près de 35% de la taille totale du fichier compressé par rapport aux travaux antérieurs. En outre, grâce au codage vidéo sans perte, la performance de gain de CABAC atteint de 15 à 19% par rapport à celle de CAVLC.

CHAPITRE III

Compression vidéo sans perte hiérarchique améliorée

1. Introduction

La vidéo prend une place important dans les diverses applications multimédias, médicales, de sécurité, etc. c'est pour cela, la qualité des images utilisées est devenue un critère primordial. De ce fait, l'évaluation de la qualité des images est un élément clé de la définition de la performance du codage.

De nos jours, il est devenu nécessaire de juger la qualité des images traitées ou produites dans de nombreuses applications multimédia, par rapport à l'image initiale au niveau d'une chaine de codage.

La compression vidéo sans perte résout le problème de la qualité des images. Dans ce chapitre nous nous sommes intéressées à la méthode de compression vidéo sans perte hiérarchique et nous pouvons y introduire des améliorations.

2. Analyse du code LETI

2.1. *Code du LETI*

Notre équipe Circuits et Systèmes(C&S) du Laboratoire d'Electronique et des Technologies de l'Information (LETI) a développé un codeur vidéo. Le codeur étudié est écrit en langage C. Il est semblable au JM, mais plus optimisé et plus maniable. Comme c'est le cas pour le JM, ce code permet aux utilisateurs d'introduire leurs choix et leurs options de codage comme le nombre des frames à coder, le type de frames I ou P, le pas de quantification (QP), le loop filter, etc. Je vais utiliser ce codeur pour introduire le codage entropique CAVLC dans notre architecture proposé. La structure du code LETI et la suivante (Figure 1) :

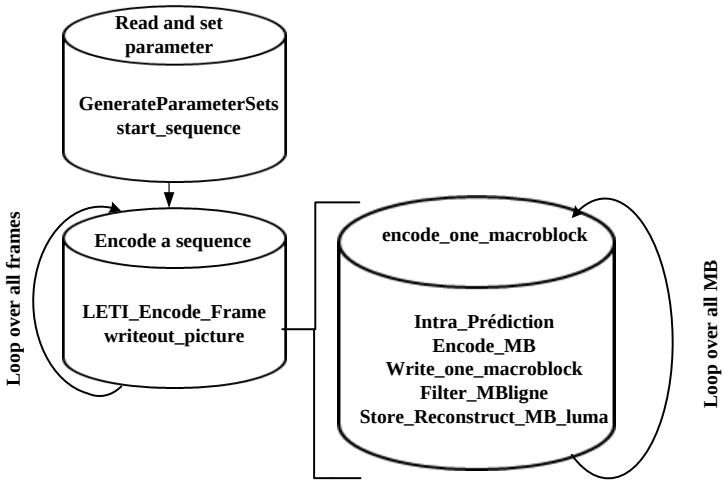


Figure III.1. *Structure du code LETI*

2.2. *Les étapes de déroulement du codage CAVLC*

Pour chaque bloc 4 × 4, l'algorithme CAVLC se déroule en 5 étapes [1, 2].

Etape 1 : Le nombre total des coefficients non nuls «Tot_Coeff» et le nombre des derniers coefficients de hautes fréquences ayant une valeur égale à ±1 «Trailingones» sont codés en utilisant un code binaire combiné. «Trailingones» nommées «T1s» constituent un ensemble parmi trois coefficients non-nuls consécutifs à la fin d'un balayage de coefficients non nuls, ayant une valeur absolue égale à 1.

«Tot_Coeff» est un nombre pouvant prendre une valeur de 0 à 16. «T1s» est un nombre pouvant prendre une valeur de 0 à 3. Les autres coefficients de valeurs (+1) et (-1) seront codés en tant que «Level». Pour coder «Tot_Coeff», il faut utiliser un tableau de taille 17 x 4.

On applique 3 tableaux de longueurs variables (Variable Length Code : VLC) et un tableau de longueur fixe (Fixed Length Code : FLC) pour coder «Tot_Coeff». Ces quatre tableaux d'Huffman sont donnés en annexe 3 (Table 3). Le choix du tableau utilisé se fait selon la valeur du paramètre N (voir Tableau 1).

Tableau III. 1. *Choix de la table de codage de Tot_Coeff*

N	Table de codage de Tot_Coeff
0, 1	Num-VLC0
2, 3	Num-VLC1
4, 5, 6, 7	Num-VLC2
≥ 8	FLC

Le paramètre N dépend du nombre de coefficients des blocs au-dessus et à gauche du bloc considéré. La valeur du paramétré N est définit par :

$N= (N_U+N_L)/2$ (1)

Avec :

N_U et N_L nombre de coefficients non nuls haut et gauche respectivement

Etape 2 : Le signe de chaque «Trailingones» est simplement codé par un seul bit tout en commençant par la plus haute fréquence contenant un «Trailingones».

Etape 3 : Les coefficients non nuls restant (NZ Coeff) appelés Level autres que les «Trailingones» sont codés dans un ordre inverse utilisant sept tables VLC (VLC0 … VLC6) (voir Table 4dans l'annexe 4).

La différence entre ces tables est la longueur de code. Les valeurs les plus faibles situés en hautes fréquences sont traités en premier lieu, la table sélectionnée est celle dont les codes sont les plus courts, généralement VLC0. Si «Tot_Coeff» >10 et «T1s» <3, alors VLC1 est la table de départ.

Si le module du coefficient à codé dépasse un seuil, fixé par le standard, la table de Huffman sera remplacée par celle qui suit. Le tableau suivant donne les seuils de passage d'une table à une autre :

Tableau III. 2. *Seuil de passage de table de Level*

Table VLC courante	Seuil de passage de tableau
VLC0	0
VLC1	3
VLC2	6
VLC3	12
VLC4	24
VLC5	48
VLC6	N /A (table plus élevé)

Etape 4 : Le nombre de tous les zéros avant le dernier coefficient non-nul «Tot_Zeros» est codé en utilisant 15 tables de Huffman.
Les tables de Huffman pour «Total zeros» sont présentées en annexe 5 à la fin du manuscrit (Table 5).
Etape 5 : Le nombre de zéros consécutifs précédant chaque coefficient non nul «Run_Before» est codé dans un ordre inverse.
Les valeurs nécessaires pour les «Run_Before» sont définit par la Table 6dans l'annexe 6.

3. Logiciel JM

Le groupe JVT a créé un logiciel de référence JM (Joint Model) dont les sources sont librement téléchargeables sur son site [3,4]. Il est écrit en langage C. Afin d'expérimenter les fonctionnalités de JM, de nombreux paramètres peuvent être modifiés dans leur fichier de configuration, tel que le nombre d'images à coder, le mode intra\inter, le nombre d'images de référence, le choix du codage entropique (CABAC ou CAVLC), les modes de partitionnement d'un macrobloc, le pas de quantification (QP), etc. Dans ce travail, j'ai utilisé la version JM 18.3 pour valider l'algorithme proposé.
J'ai effectué une analyse détaillée du software JM de la norme H.264/AVC (voir la Figure 2). Cette analyse a pour but de comprendre globalement le déroulement de chaque module du chaine de codage et ensuite d'extraire des données du JM (en d'autre terme l'image d'entré du module filtrage).

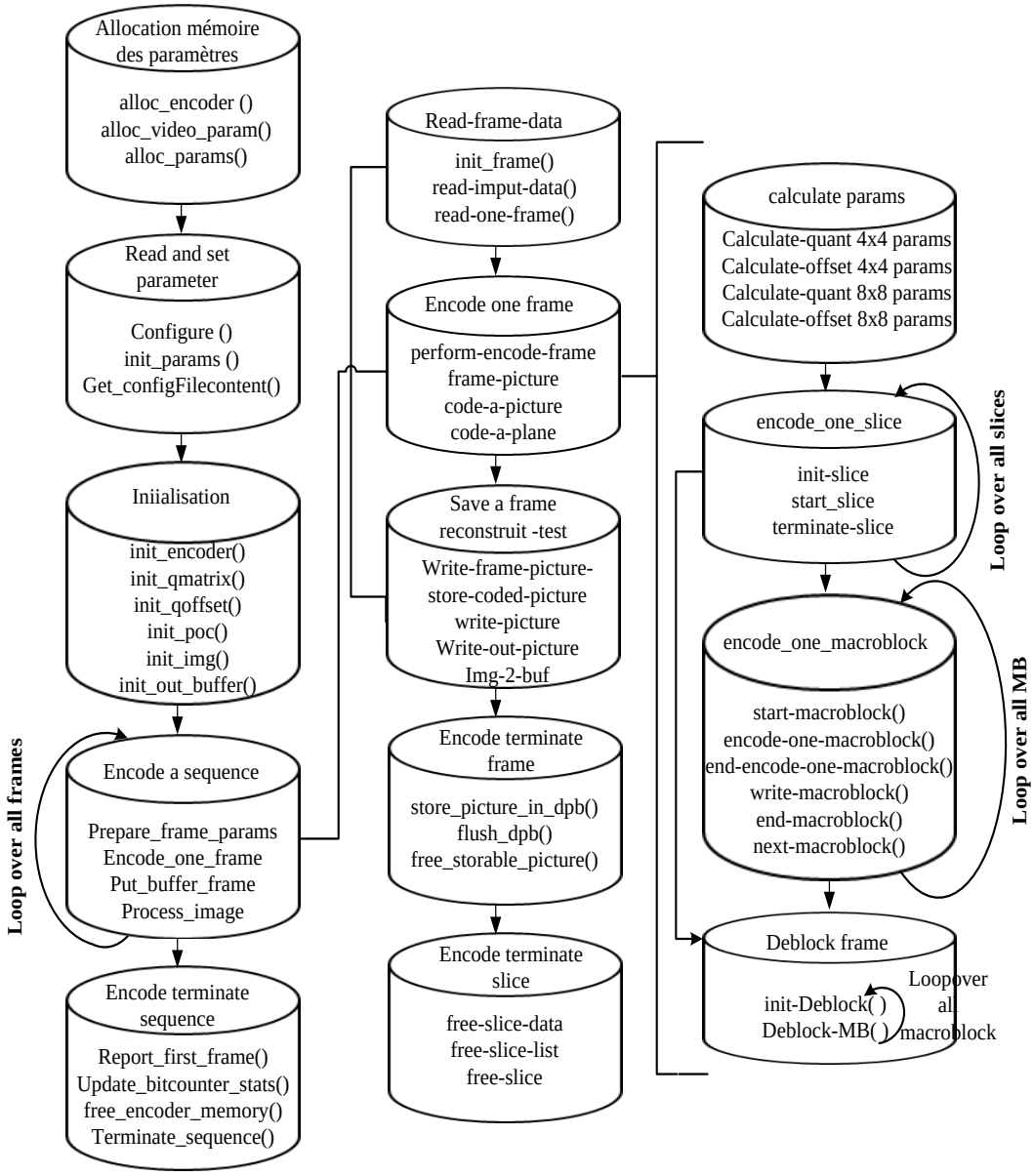


Figure III. 2. *Structure du code JM*

4. L'état de l'art de la structure hiérarchique

4.1. *Méthode de Jun-Ren Ding*

En général, la transformation et la quantification ne peuvent pas être utilisées pour le codage des images et des vidéos sans perte, telles que JPEG-LS, H.264-LS et H.264-LS_DPCM, car elles entraîneraient des erreurs de transformation et de quantification. Si nous voulons transmettre un bit-stream, qui peut fournir des images sans pertes et avec pertes en même temps pour différents clients, JPEG-LS ou H.264-LS ne peut pas offrir une telle capacité.

Pour résoudre ce problème d'accès universel, Jun-Ren Ding et al [5] proposent une méthode de codage vidéo sans perte basée sur le codage avec perte du standard H.264 comme le montre la Figure 3. Ils peuvent contrôler le flux binaire de compression vidéo avec perte en contrôlant les paramètres de quantification (QP). Pour les applications sans perte, ils utilisent le codage entropique existant pour coder la différence entre les images originales et reconstruites après l'exécution de la DCT inverse et de la quantification inverse obtenue à partir du flux binaire avec pertes. Comme le montre la Figure 3, le D bitstream est utilisé pour représenter les résultats de codage de la différence des images après l'exécution de CABAC. Les auteurs peuvent réaliser une compression sans perte grâce à une combinaison entre D bitstream et le bitstream avec perte [5].

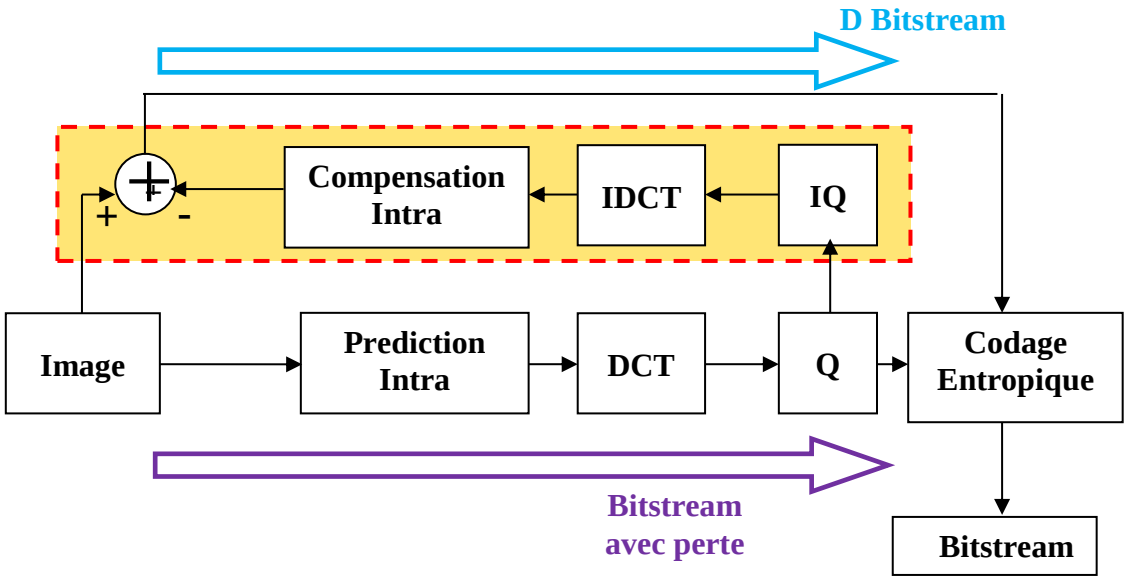


Figure III.3. *Diagramme de bloc de H.264-LS proposé basé sur le codage vidéo avec perte du standard H.264 [5].*

4.2. *Méthode de Wei-Da-Chien (deuxième version):*

Les procédures de transformation et de quantification, qui produiraient des erreurs, ne peuvent pas être directement utilisées dans le codage vidéo sans perte. Cependant, les procédures de transformation et de quantification peuvent considérablement réduire les bits de codage de la redondance spatiale. Ainsi, un système de codage sans perte hiérarchique à deux couches a été proposé en incluant le codeur H.264 / AVC dans la couche de base et le codeur Rice dans la couche d'amélioration pour le codage d'erreur résiduelle pour améliorer l'efficacité de compression et la flexibilité d'application. Pour explorer la quantification optimale, il faut

reconcevoir la couche d'amélioration [6]. En effet, La couche de base fournit une bonne performance de codage avec perte, et la couche d'amélioration corrige les erreurs provoquées par la couche de base. Finalement, We-Da-Chien et al. ont réalisé la vidéo sans perte en combinant les flux binaires des couches de base et d'amélioration dans l'architecture de codage proposée. We-Da-Chien et al. ont proposé un système de codage vidéo sans perte efficace avec l'architecture de codage à deux couches pour fournir simultanément une vidéo sans perte et avec perte. Les différences de cette méthode avec la méthode de Jun-Ren Ding et al. est en premier lieu l'utilisation du codeur rice par rapport à un codage entropique CABAC et en second lieu l'ajout du module de la sélection de QP.

Le système proposé à deux couches de codage vidéo sans perte représenté dans la Figure 4 est composé de trois composants principaux: (i) codeur H.264 / AVC; (ii) un codeur de rice QP-adaptatif; et (iii) le mécanisme de sélection QP dans le système de codage proposé. Les descriptions détaillées de ces trois composantes sont traitées dans la figure suivante:

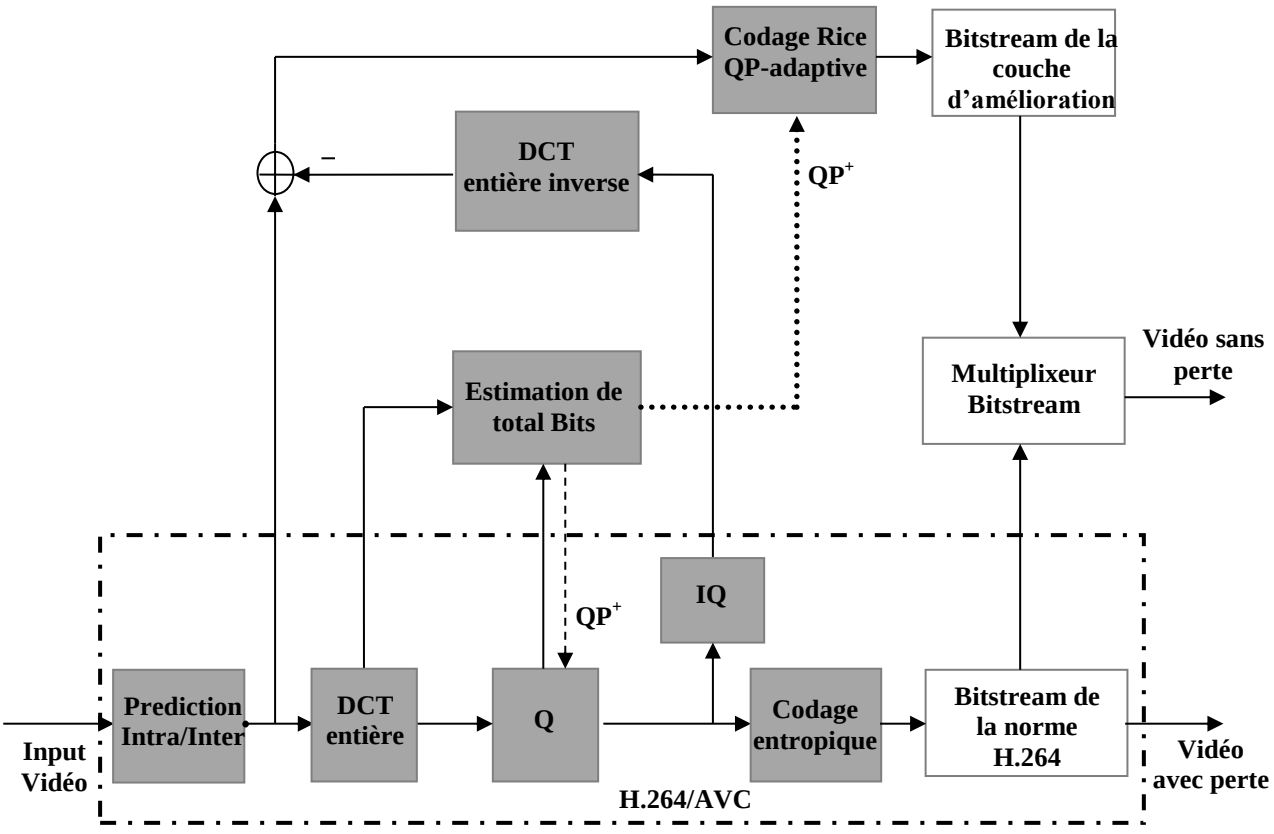


Figure III.4. *Système de codage hiérarchique sans perte à deux couches proposé par Wei-Da-Chien et al [6]*

4.3. *Méthode de Li-Li Wang et al.*

Dans l'algorithme proposé de Li-Li Wang et al, deux contributions ont été faites. La première est que les échantillons dans un macrobloc (MB) / bloc sont prédits hiérarchiquement au lieu d'utiliser la prédiction basée sur des blocs dans son ensemble. Plus spécifiquement, quatre groupes sont extraits des échantillons dans un macrobloc / bloc. Les échantillons du premier groupe sont d'abord prédits en se basant sur la méthode intra prédiction directionnelle et ensuite les échantillons des autres groupes sont prédits en utilisant les échantillons du premier groupe comme références. En conséquence, l'information dans le bloc résiduel peut être réduite puisque les échantillons peuvent être précisément prédits en utilisant des références plus proches. L'autre contribution est que deux modes de codage sont conçus pour coder efficacement le bloc résiduel résultant. Un meilleur mode de codage peut être choisi en fonction de l'optimisation du débit RO (rate optimization).En outre, une amélioration de la méthode de codage entropique CAVLC plus efficace a été proposée pour coder les flags des schémas de codage pour un MB [7].

La Figure 5 montre le diagramme de bloc du processus de sélection du schéma de codage [7]. En effet, il existe deux schémas de codage: les schémas de codage à une et à deux couches, conçus pour coder les résiduelles. Dans le schéma de codage à une couche, les coefficients résiduelles sont directement balayés et codés par codage entropique par la méthode CAVLC. Alors que le schéma de codage à deux couches inclut des processus de codage avec perte et sans perte. Le bloc résiduel est d'abord transformé et quantifié pour obtenir les coefficients QDCT dans le processus de codage avec pertes. Les coefficients QDCT sont balayés et codés par codage entropique. Tandis que pour le processus de codage sans perte, la quantification inverse et la DCT inverse sont appliquées aux coefficients QDCT pour obtenir le bloc résiduel reconstruit **r'**. La différence entre le bloc résiduel original **r** et le bloc résiduel reconstruit **r'** est balayée et codée par le codage entropique. Le choix de la sélection d'un meilleur schéma de codage à partir des deux schémas candidats est un problème critique pendant le codage.

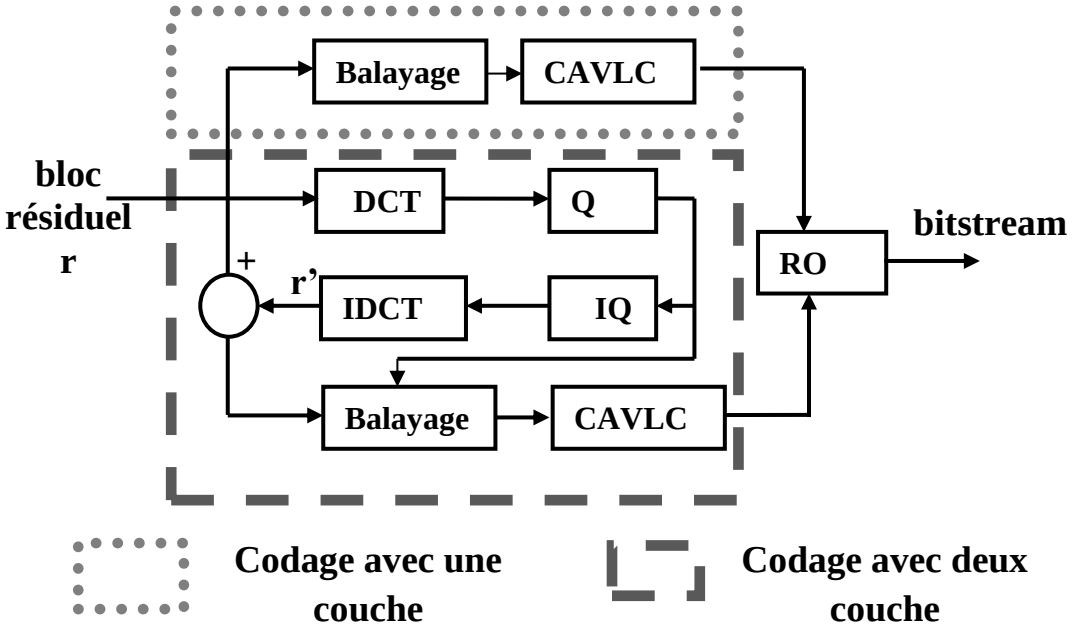


Figure III.5. *Sélection du schéma de codage basé sur l'optimisation du débit [7].*

La méthode hiérarchique est aussi appliquée sur les fichiers médicaux. On peut citer Comme exemple la méthode de Zhongwei Xu et al. [8].

4.4. *Méthode de Seung-Hwan Kim et al.*

Dans la méthode TRC (two-layered residual coding), la première couche FRC (first-layer residual coder) utilise le codage résiduel du standard conventionel H.264 / AVC, qui consiste des étapes de transformation et de quantification avec un paramètre de quantification (QP). La seconde couche SRC (second-layer residual coder) adopte une méthode BPC (bit plane coding) [9].

La structure de la méthode TRC est illustrée dans la Figure 6 (voir première étape en jaune, deuxième étape en bleu et troisième étape en rouge). L'image d'entrée est codée en trois étapes : 1) prédiction intra de la norme H.264/AVC ; 2) DCT, quantification, DCT inverse, et quantification inverse dans la première couche FRC ; et 3) la méthode de codage BPC de la seconde couche SRC. Ils sont détaillés ci-dessous.

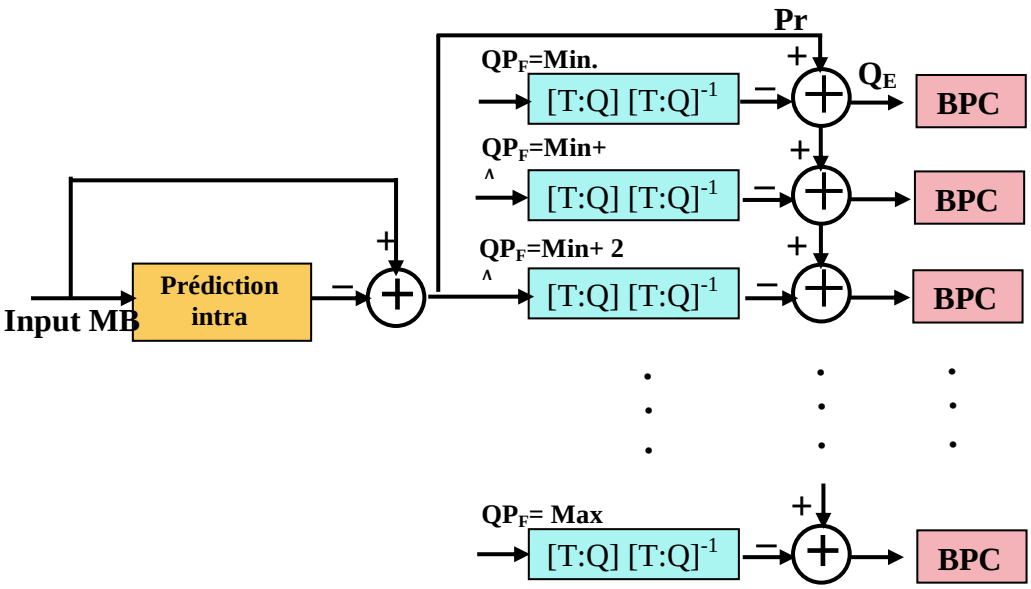


Figure III.6. *Structure de la méthode TRC [9]*

Après la prédiction intra du codeur avec perte H.264 / AVC, un schéma avancé a été proposé pour le codage des résiduels. Il est constitué de deux codeurs résiduels en cascade. Le codeur résiduel de la première couche est conduit via une transformation et une quantification avec un paramètre de quantification donné. Le codeur résiduel de la deuxième couche est une méthode de codage BPC (bit plane coding)

La prédiction intra de la norme H.264 / AVC [10] a été appliquée pour chaque macrobloc d'entrée (MB) et l'image résiduelle a été calculée en conséquence. L'image résiduelle de prédiction, notée Pr (x, y), peut être écrite comme suit:

$$Pr\,(x, y) = I(x, y) - P(x, y) \quad (2)$$

Où I (x, y) et P (x, y) sont des échantillons de l'image originale et de l'image prédite obtenus par prédiction intra de la norme H.264 / AVC, respectivement. Le codeur de transformée standard a été adopté pour implémenter le FRC comme indiqué dans la deuxième étape de la Figure 6, où le bloc résiduel est transformé et quantifié de sorte que l'énergie sur les basses fréquences peut être codée efficacement. Une QP plus grande, notée QP_F, est utilisée pour quantifier les coefficients de transformation et la taille de bloc de transformation est choisie de manière adaptative entre 4 × 4 et 8 × 8. Il est clair que le FRC est un codeur avec perte et l'image résiduelle après le FRC peut être obtenue comme :

$$Q_E\,(x, y) = Pr\,(x, y) - \overline{Pr\,(x, y)} \quad (3)$$

Où Pr(x, y) représente l'échantillon d'image reconstruit à partir de la FRC. Puisque l'image résiduelle dans (2) est principalement causée par la quantification, elle est appelée l'image résiduelle de quantification pour plus de commodité.

En raison de la nature de perte de la FRC, il faut un deuxième codeur résiduel pour obtenir le résultat de codage sans perte. Pour implémenter le SRC, une méthode BPC a été proposée. Puisque le FRC est exécuté par l'unité MB, l'image résiduelle de quantification Q_E est également divisée en l'unité MB et codée indépendamment.

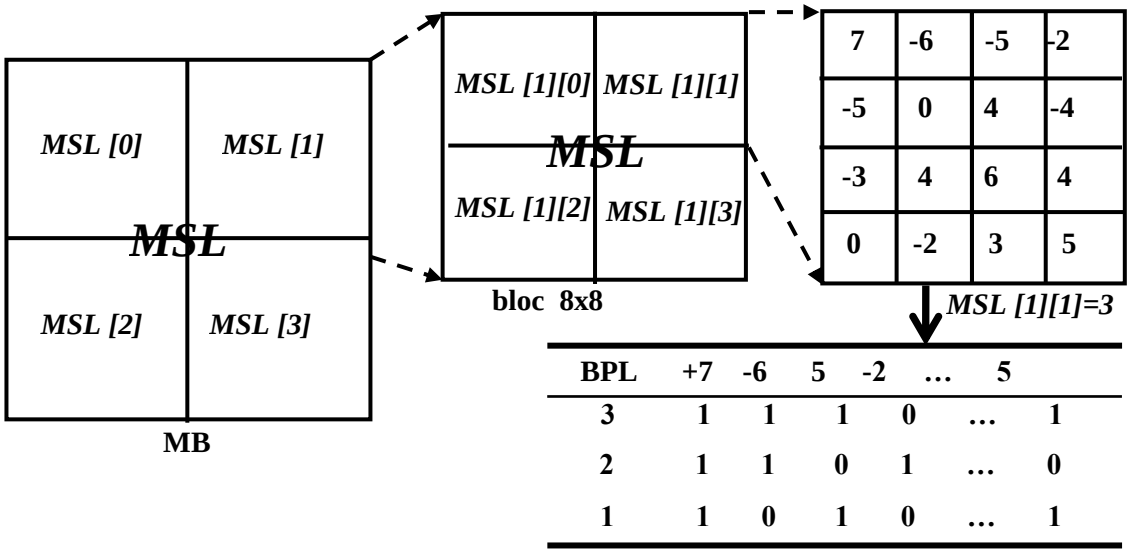


Figure III.7. *Structure du BPC*

Dans la Figure 7, la structure BPC proposée a été représentée pour une erreur MB donnée. Il se compose de deux parties de codage: 1) le codage de MSL (most significant bit-plane level) et 2) le codage des données BPL (bit-plane) Pour coder efficacement les informations MSL, un schéma de codage similaire à celui de MPEG-4 a été utilisé [11]. Chaque MB a été classé en trois types de blocs (c'est-à-dire des blocs 4 × 4, 8 × 8 et 16 × 16) en fonction de la taille du bloc. Les informations MSL correspondantes sont alors définies comme suit.

1) MSL: le MSL de l'erreur de quantification pour un MB de taille 16 × 16.

2) MSL [j]: MSL pour un bloc 8 × 8 dans un MB (j = 0, 1, 2, 3).

3) MSL [j] [k]: MSL pour un sous-bloc 4 × 4 dans un bloc 8 × 8 dans un MB (k = 0, 1, 2, 3).

Après le codage des informations MSL comme indiqué dans la Figure 7, un codage des données BPL a été traité. Les valeurs d'erreur de quantification dans un sous-bloc de 4 × 4 sont ordonnées par balayage raster et plusieurs valeurs BPL sont formées en fonction de la MSL[j][k] donnée. Deux méthodes de codage des données BPL sont présentées. Pour la première méthode BPC, les valeurs binaires de BPL et les informations de signe sont directement transmises sans aucune modélisation complexe et codage entropique. Ceci sert

d'un point de référence de performance. Pour la seconde méthode BPC, les valeurs binaires BPL sont codées par le codage arithmétique binaire contextuel [12].

Les paramètres de codage pour le logiciel de référence H.264 / AVC [13] et le TRC proposé ont été établis comme suit :

1) ProfileIDC= 244 (high 4: 4: 4).
2) IntraPeriod = 1 (seulement codage intra).
3) QPISlice = 0.
4) SymbolMode = 0 ou 1 (0: CAVLC, 1: CABAC).
5) QPPrimeYZeroTransformBypassFlag = 1 (sans perte).
6) Le nombre des images codés = 100.
7) Min. QPF =10, Max. QPF = 40, $\Delta = 2$ (comme illustré dans la Figure 6).

4.5. *Méthode de Qi Zhang et al.*

Les techniques de compression sans perte sont constituées généralement de deux phases indépendantes: la prédiction basée sur la modélisation et le codage entropique des résidus de prédiction [14]. Beaucoup de recherches de codage sans perte s'orientent vers le schéma de prédiction qui minimise l'erreur de prédiction. De ce fait, il existe trois schémas de prédiction bien connus développés auparavant: 1) CALIC, 2) JPEG-LS et 3) prédiction intra dans le standard H.264 / AVC sans perte (H.264-LS).

Il faut étudier d'abord les différents schémas de prédictions utilisés dans le codage sans perte, et observer que leurs résidus ne sont pas blancs mais sous forme de bruit granulaire. Cette étude révèle que les schémas de prédiction sans perte actuels ne peuvent pas prédire efficacement la présence de bruit granulaire dans les contenus HD et il est nécessaire de traiter l'image résiduelle avec le bruit granulaire soigneusement pour améliorer l'efficacité du codage. Alors, un nouveau schéma sans perte a été proposé pour encoder efficacement les images résiduelles. Puisque cette idée est basée sur la prédiction et le codage d'image résiduelle (RIPC), la méthode de codage est appelée RIPC.

Après la prédiction, les erreurs de prédiction (ou les images résiduelles) sont codées par codage entropique pour produire le bitstream final. Bien que les trois schémas de prédiction ci-dessus adoptent des règles différentes, leurs résidus de prédiction sont visuellement similaires comme le montre la Figure 8 (b) - (d).

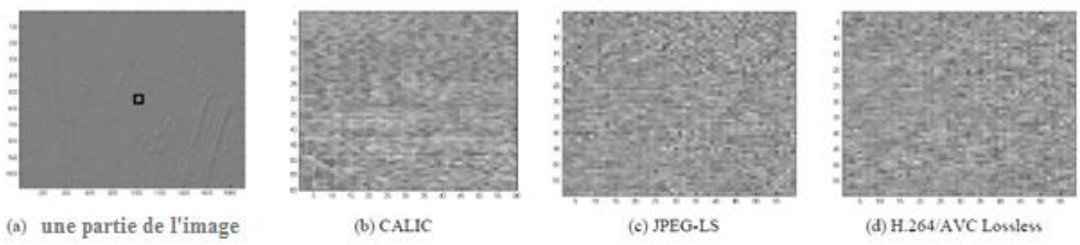


Figure III.8. *Les images résiduelles de la séquence rush hour HD 1920x1080 par différents schémas de prédiction utilisés dans plusieurs codeurs sans perte*

Comme indiqué dans la Figure 8. L'image résiduelle contient des textures non compressées, et la majorité apparaît sous forme de bruit granulaire. La propriété de couleur non-blanche implique qu'il existe un peu de redondance dans l'image résiduelle qui n'a pas été bien éliminée dans la phase de prédiction. La propriété des images résiduelles non blanches empêchent le codeur entropique d'atteindre un bon taux de compression [16]. Cette observation se manifeste aussi dans les histogrammes des résidus de prédiction de la Figure 9(a) - (c).

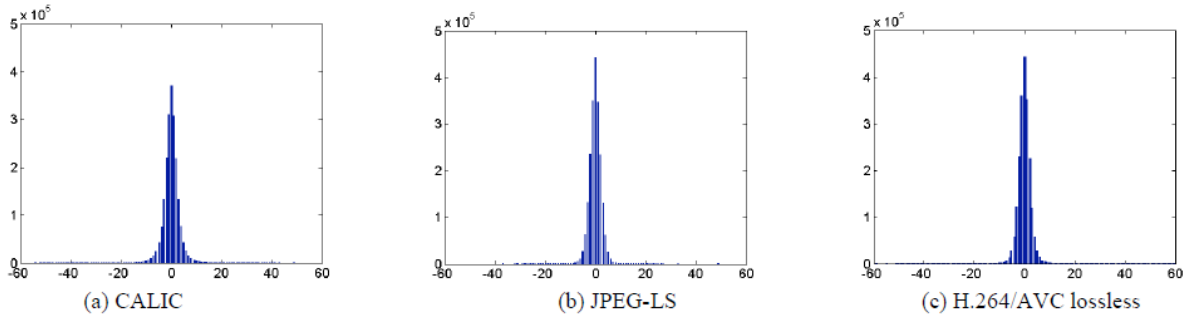


Figure III.9. *Les histogrammes d'erreurs de prédiction pour la séquence rush hour HD 1920x1080*

Des études dans [15] confirment que l'image résiduelle peut être vue comme un bruit granulaire. Le bruit granulaire n'est pas important dans la vidéo SD et même moins perceptible dans les vidéos à basse résolution telles que CIF et QCIF. Cependant, ces variations superficielles dans la vidéo HD sont visibles. La méthode de Qi Zhang et al. tente d'exploiter la corrélation spatiale et temporelle du bruit granulaire par un autre niveau de prédiction afin de réduire la redondance.

Le système de codage vidéo HD sans perte est représenté dans le schéma fonctionnel de la Figure 10. L'image d'entrée est décomposée en deux parties par une technique de débruitage. Ensuite, ces deux parties peuvent être codées indépendamment. Ils sont intégrés de nouveau dans la fin de décodeur. Donc, la méthode de compression vidéo sans perte proposée est basée

sur la prédiction d'image résiduelle et le codage RIPC (residual image prediction and coding) [16]. En effet, deux schémas de prédiction différents sont effectués pour le contenu et le bruit granulaire, et leurs résidus sont codés par un codage entropique différent (comme illustré dans la Figure 10).

Dans le premier schéma de prédiction, pour une image d'entrée F, un processus de codage vidéo avec perte est utilisé d'abord avec un QP donnée. Ensuite, la différence entre l'image reconstruite F' et l'image originale F est l'image de bruit extraite notée par N. Enfin un codage entropique comme CABAC est appliqué aux images N. Il existe un compromis entre les bits affectés au codage de F' et le codage de N, en fonction de QP sélectionné.

Cette première partie d'image résiduelle utilise le profil de base de la norme H.264 / AVC pour garder le codage avec une complexité de calcul minimale. Le paramètre de quantification a été défini par 25, le nombre d'images de références était 1, le processus RDO est réglé à une valeur pour trouver une faible complexité, la recherche de sous-pixels est désactivée et le codage entropique utilisé est CABAC. Dans ce processus de prédiction d'image résiduelle, le bloc de prédiction dans l'image résiduelle était fixé à une taille de 4×4 et la plage de recherche était limitée à 32 pour les directions horizontale et verticale.

Concernant la deuxième partie d'image résiduelle, une combinaison des méthodes de prédiction spatiales et temporelle a été utilisée pour effectuer une prédiction par blocs [17] dans l'image résiduelle. Ensuite, les erreurs de prédiction ont été codées par les codes Exp_Golomb. Contrairement à d'autres systèmes de codage sans perte ou avec perte, le système RIPC peut produire des résultats de codage sans perte et avec perte, sans modification du matériel (hardware). Il peut également atteindre l'adaptabilité avec des modifications mineures.

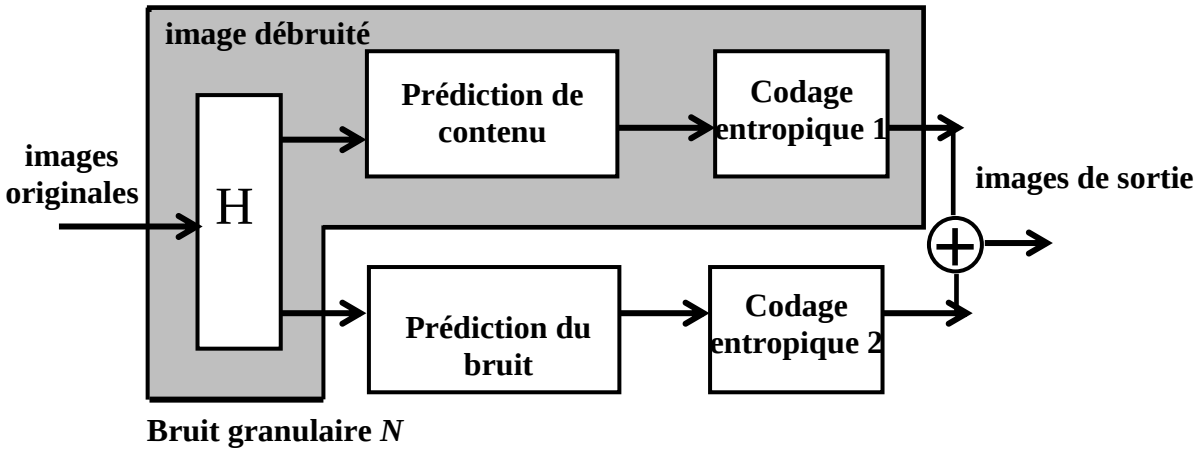


Figure III.10. *Codage sans perte pour les contenus HD [16]*

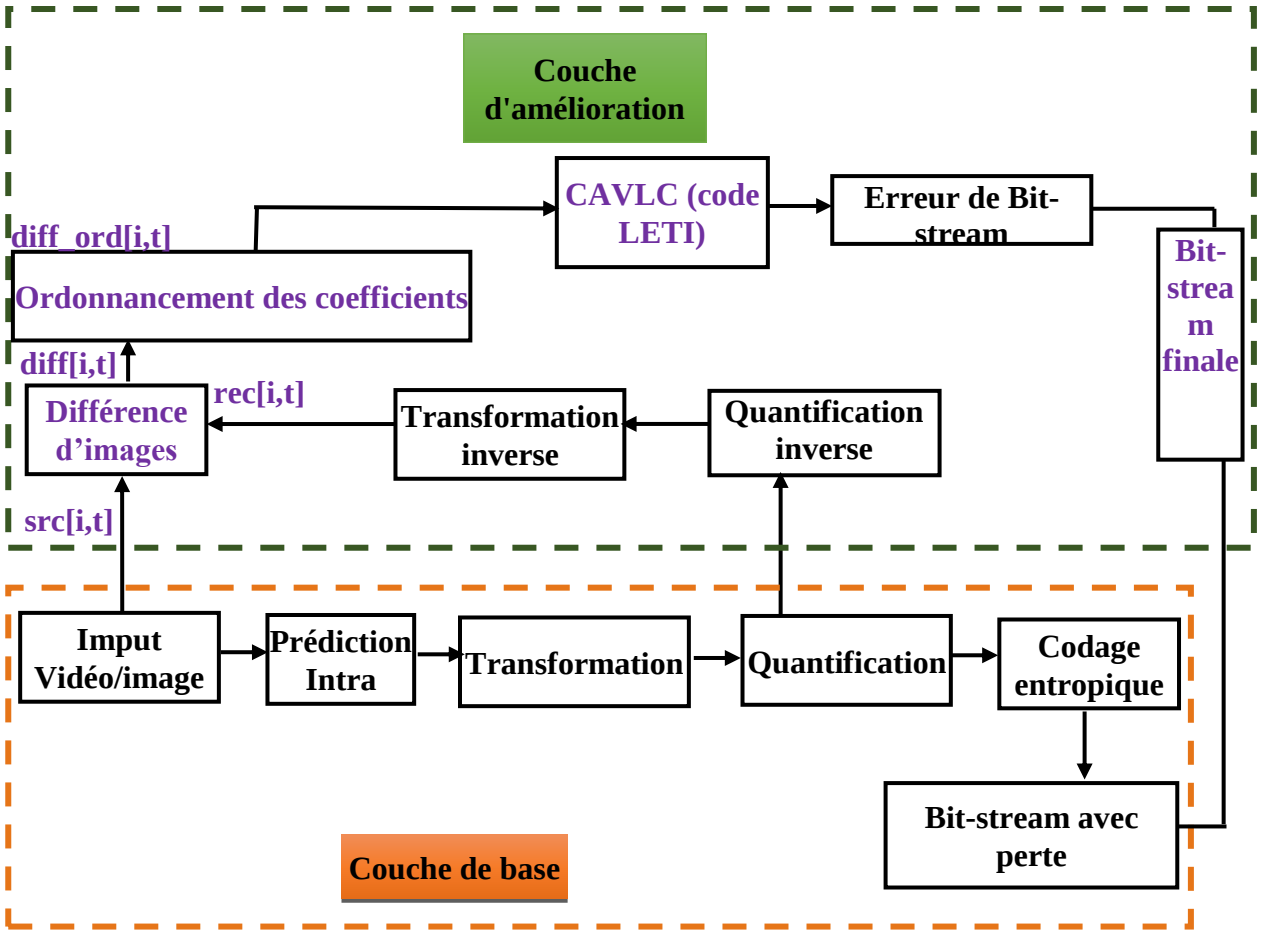
5. Méthode de Wei Da Chien

Dans le chapitre1, j'ai déjà présenté la méthode de compression vidéo sans perte hiérarchique. Cette technique est basée sur 2 couches. Une couche de base génère un bitstream relatif à une compression avec perte et l'autre couche corrige ce dernier par un second bitstream afin d'obtenir un bitstream final qui permet une reconstitution sans perte [5,18-22]. La couche EL de Wei Da Chien comporte quatre parties : quantification inverse, transformation inverse, estimation de la distorsion et codage entropique (voir Figure 30 du chapitre1).

6. Méthode proposée

6.1. *Présentation*

J'introduis des améliorations principalement sur la couche EL. Je propose une approche qui inclut trois étapes. Pour cela, j'ajoute trois blocs "différence d'images", "ordonnancement des coefficients" et "CAVLC (code LETI) ". En effet, la chaine de codage de la norme H.264/AVC génère deux sorties : l'image reconstruite et le bitstream avec perte BL. Je calcule en un premier lieu la différence entre l'image reconstituée avec perte (de la couche BL) et l'image originale pour corriger le bitstream avec perte (voir Table 7 dans l'annexe 7). En second lieu, j'intègre un code d'ordonnancement des coefficients selon un critère d'apparition des distorsions suit du code de différence dans le code JM. La Figure 11 donne un aperçu détaillé de ce système.



src[i,t] représente l'image source
rec[i,t] désigne l'image reconstruite
diff[i,t] indique l'image de différence
diff_ord[i,t] est référé l'image de différence ordonnée

Figure III.11. *Structure générale du codage vidéo sans perte hiérarchique amélioré*

Le but de ce code se résume comme suit : La perte des données existe surtout dans les frontières des blocs 4×4 de l'image (effet de blocs). L'idée consiste à ordonnancer les coefficients de l'image de différence avant de les envoyer au codage entropique CAVLC du code LETI. Pour la composante de la luminance, les coefficients des frontières des blocs 4×4 sont envoyés en premier lieu. Les coefficients des deux composantes de chrominance sont envoyés avec un ordonnancement similaire en second lieu. Enfin, le reste des coefficients ("cores" des blocs 4×4) est transmis. Cet ordonnancement des coefficients vise à améliorer les performances du codeur entropique CAVLC et augmenter ainsi le taux de compression.

Avant de faire intégrer le CAVLC du code LETI dans le code JM, il faut faire une étude préliminaire du codage entropique CAVLC du JM pour éviter la confusion du compilateur lors de la simulation (voir Figure 12). Il y a des fonctions et des structures qui sont similaires

au code LETI telque : write uvlc2 buffer, symbol2vlc, seq-parameter-set-rbsp-t et bitstream,..etc. J'ai changé les noms de ces fonctions et de ces structures similaires du code LETI pour résoudre le problème.

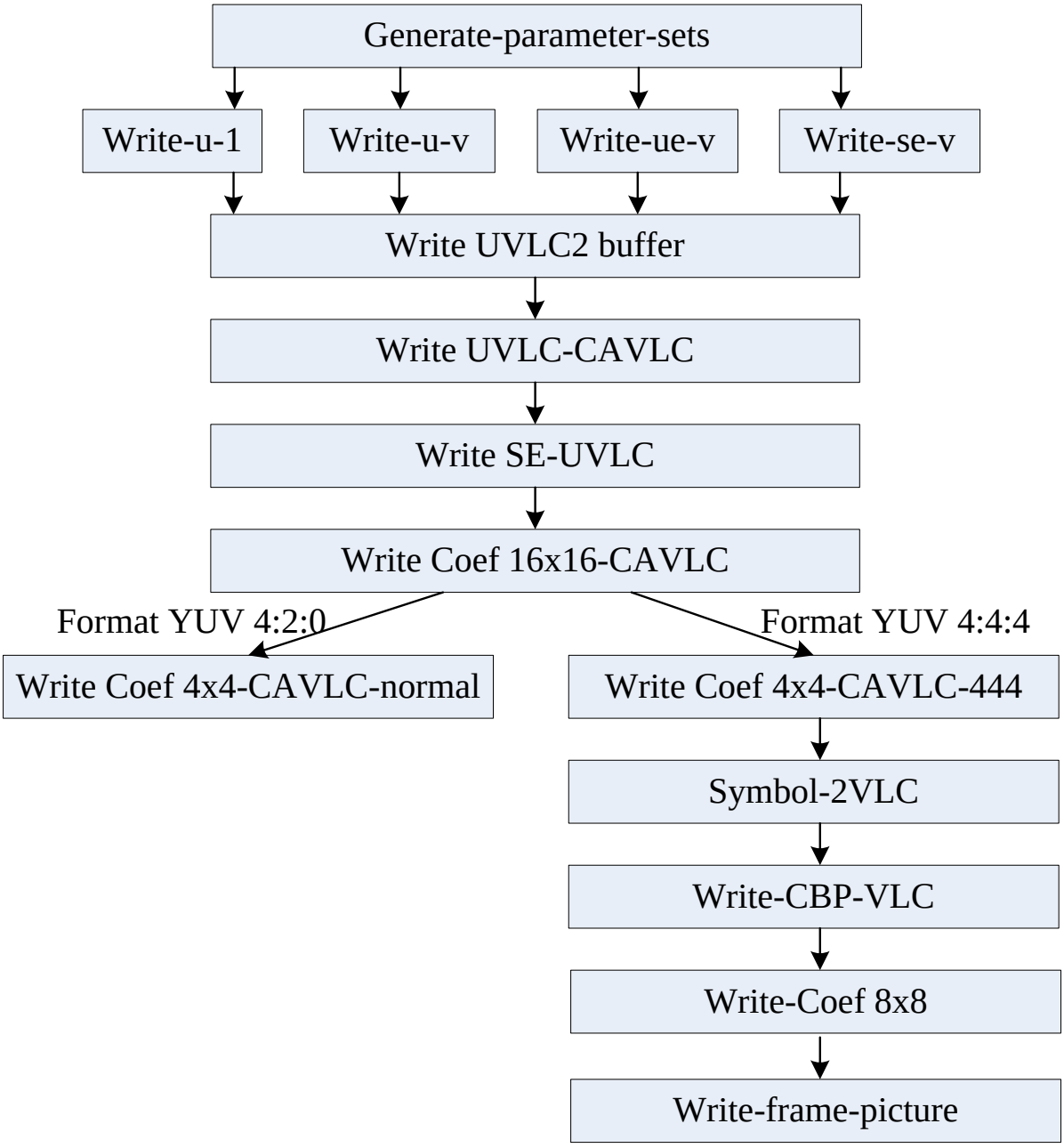


Figure III.12. *Organigramme des étapes préliminaires pour le codage CAVLC du code JM*

6.2. *Déroulement de l'algorithme*

L'algorithme d'ordonnancement des coefficients de l'image reconstruite se fait en deux grandes parties différentes (voir Figure 13) :

Ordonnancement des coefficients situés dans la frontière de l'image.

Ordonnancement des coefficients situés dans le core de l'image.

Chaque partie d'ordonnancement se compose en trois parties :

Ordonner les coefficients Luma.

Ordonner les coefficients chroma U.

Ordonner les coefficients chroma V.

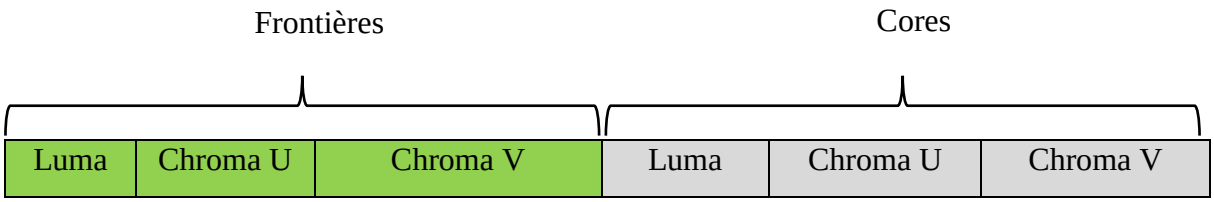


Figure III. 13. *Structure d'ordonnancement générale*

Les algorithmes d'ordonnancements des trois composantes sont identiques dans le cas de frontières. De même ces algorithmes sont identiques dans le cas de cores.

L'algorithme d'ordonnancement des frontières est constitué de quatre cas (voir Table 8 dans l'annexe 8) :

Horizontal haut.

Horizontal bas.

Vertical gauche.

Vertical droite.

L'algorithme proposé s'effectue sur un bloc 4×4 d'une image de format CIF pour une première version de la manière suivante (Figure 14) :

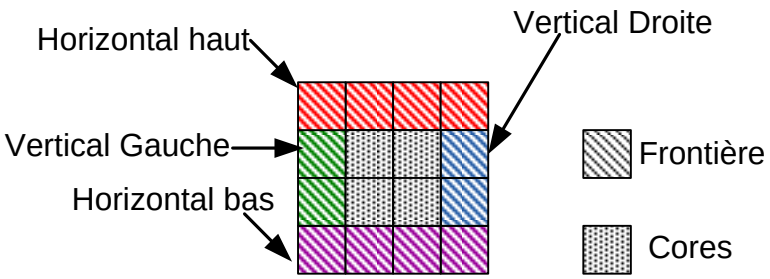


Figure III. 14. *Partitionnement du bloc 4×4 en quatre cas.*

L'algorithme d'ordonnancement se fait sur une matrice de taille 352×288 pour faciliter l'implémentation aux programmateurs.

Puisque je travaille sur le format CIF 352×288 et l'ordonnancement se fait par bloc 4×4, cette image se compose en 88 blocs horizontalement et en 72 blocs verticalement.

L'algorithme d'ordonnancement des coefficients des frontières se fait en quatre étapes (voir Figure 15) :

Je fixe deux compteurs qui correspondent aux coordonnées x et y associés aux lignes et colonnes de l'image pour se déplacer et indiquer la position courante dans l'image.

On commence par transférer les 12 premiers pixels des quatre cas de la source à la destination selon l'ordre mentionné dans la Figure 14.

Cet enchaînement est répété jusqu'à la fin de 88 blocs de la ligne 0. Puis j'incrémente la ligne 1 et je fais transférer de la même manière les 12 pixels jusqu'à terminer les pixels de 88 blocs.

Je stocke les résultats à la destination de la même façon jusqu'à la fin de toutes les lignes c'est-à-dire jusqu'à ce que j'atteins la ligne 287.

L'algorithme d'ordonnancement des coefficients des cores se fait d'une autre façon. Je réalise un balayage ligne par ligne et je stocke les coefficients de la source à la destination moyennant un transfert de quatre pixels bloc par bloc et ligne par ligne jusqu'à la fin de tous les coefficients de l'image (voir Figure 16).

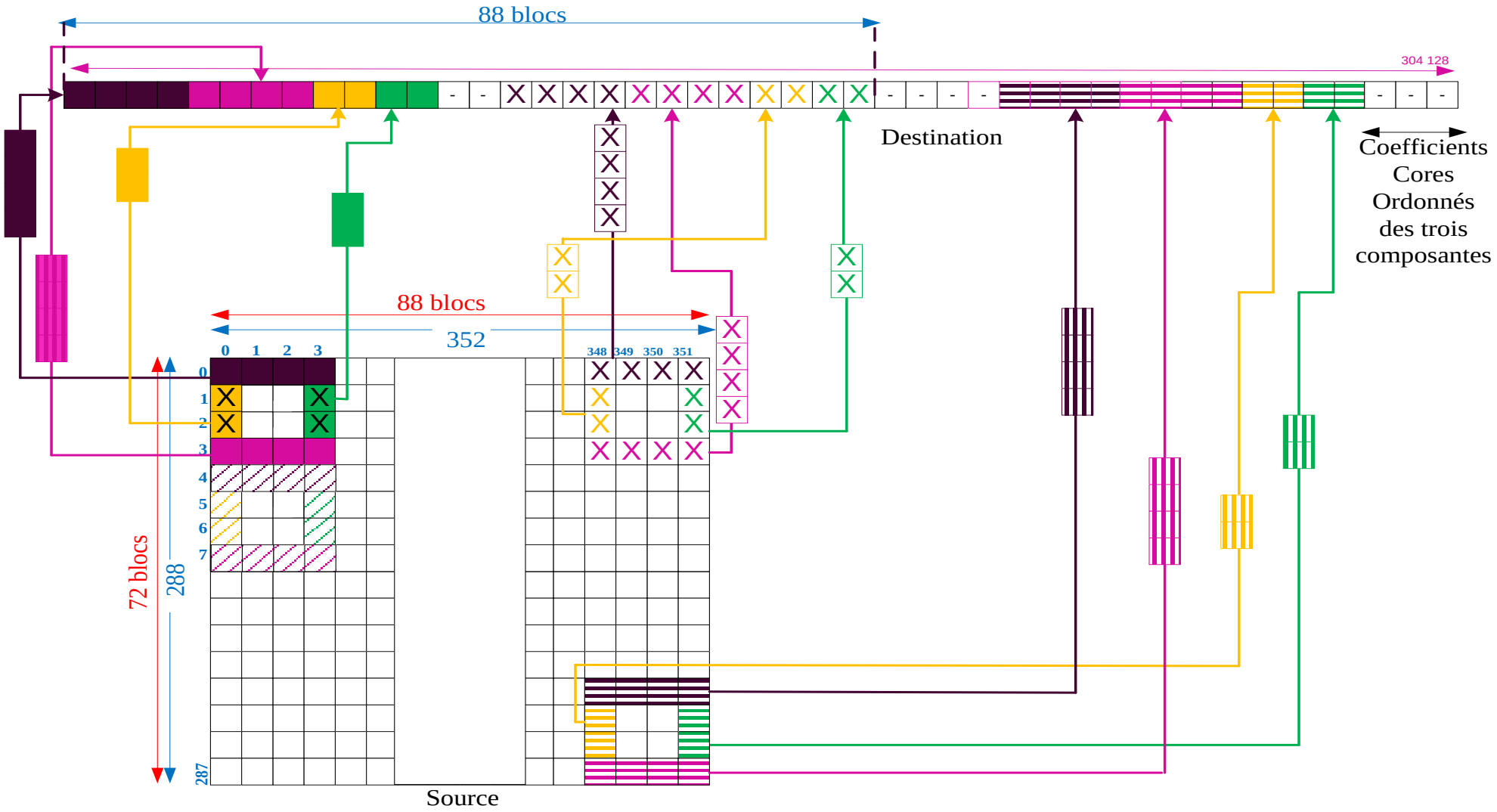


Figure III. 15. *Déroulement d'algorithme d'ordonnancement des coefficients sur les frontières*

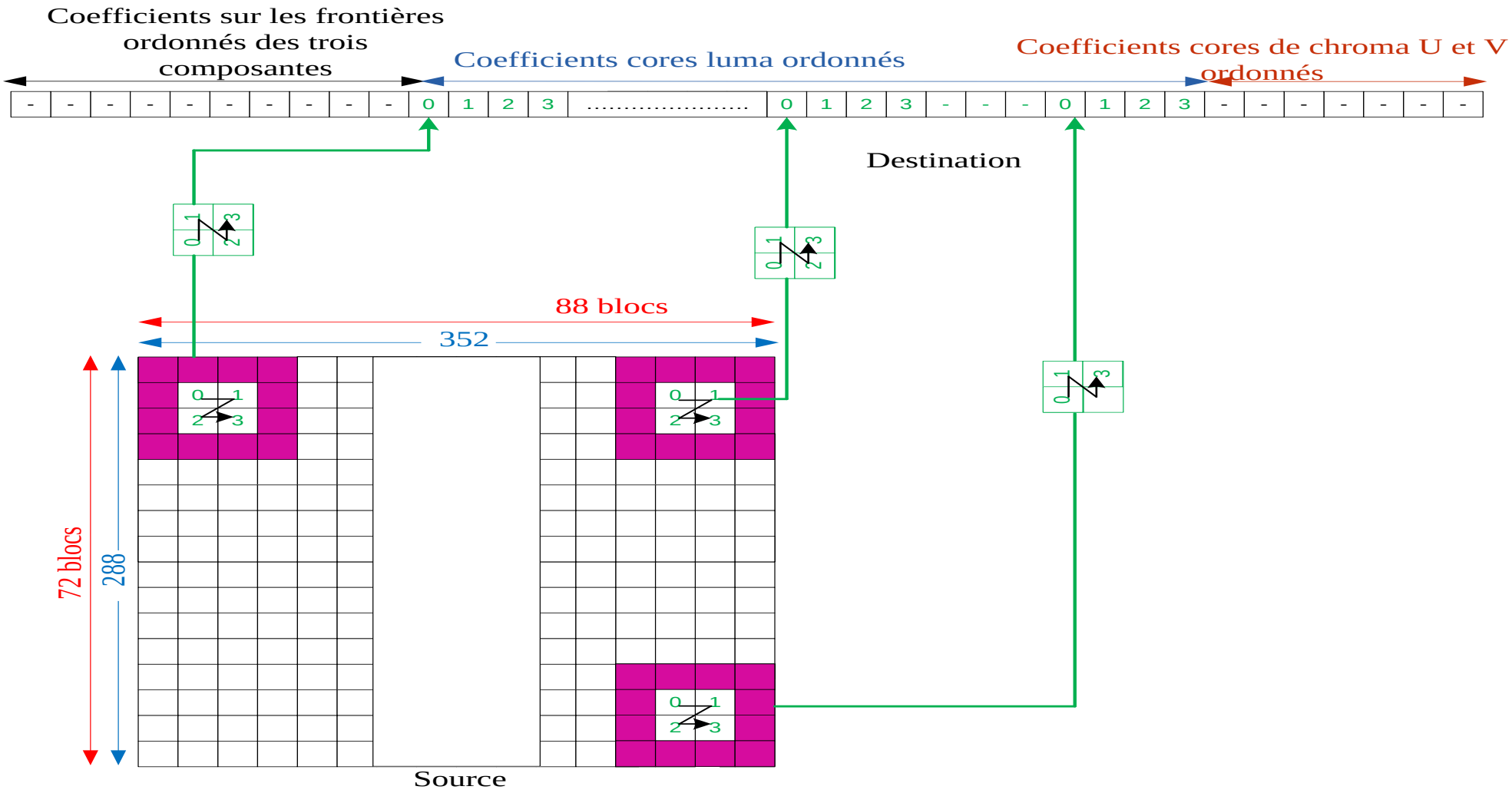


Figure III. 16. *Déroulement d'algorithme d'ordonnancement des coefficientsde cores*

Enfin, je fais une concaténation entre les coefficients d'ordonnancement des frontières et les coefficients d'ordonnancement des cores pour former une image de différence ordonnée. Cette image va être appliquée à l'entrée du codeur entropique CAVLC du code LETI. Il est prévu que cet ordonnancement améliore la performance de ce codeur [23-24].

Avant d'effectuer les tests sur le codeur entropique CAVLC, j'ai implémenté le codage de Huffman qui sert à réaliser les codes binaires des symboles obtenus et par la suite générer le bitstream de la couche d'amélioration (EL).

Principe du codage Huffman :

L'algorithme de codage de Huffman [25] porte le nom de son inventeur, David Huffman, qui a développé cet algorithme en 1950. L'algorithme d'Huffman est une solution au problème de l'élaboration des codes. Cette méthode permet une analyse statistique des symboles composant le fichier source afin d'obtenir une compression sans perte des données. Le principe de la méthode d'Huffman consiste à associer aux symboles les plus probables les mots binaires les plus courts.

Les codes obtenus par l'algorithme d'Huffman sont des codes à longueurs variables mais préfixés. En effet un code est dit préfixé s'il n'est le début d'aucun autre. Cette propriété permet de décoder sans ambiguïté toute séquence, ce qui évite d'inclure des séparateurs entre les mots.

Exemple :

Considérons l'exemple d'un message **« 0 3 -3 1 -8 -1 -1 6 -3 -5 -4 -1 -1 -4 5 8 -1 3 -2 3 -1 -1 -2 3 4 5 7…….** ». La répartition des fréquences des symboles est la suivante : f(-)=123637 ; f(1)=103693 ; f(0)=62778 ; f(2)=59781 ; f(3)=32444 ; f(4)=18926 ; f(5)=12016 ; f(6)=7357 ; f(7)=4707 ; f(8)=2932 ; f(9)=1766. On construit alors l'arbre binaire d'Huffman suivant les 3 étapes suivantes :

1ére étape : Classer les symboles dans l'ordre décroissant de leurs fréquences d'apparition. Le tableau des fréquences est donné ci-dessous.

Caractères	Fréquences d'apparition
-	123637
1	103693
0	62778
2	59781
3	32444
4	18926
5	12016
6	7357
7	4707
8	2932
9	1766

Tableau III. 3. *Fréquences d'apparition*

2éme étape :

Regrouper séquentiellement les paires de symboles de plus faible probabilité. Le déroulement est résumé ci-dessous dans la Figure 17. On obtient une arborescence dite : « arborescence de Huffman ».

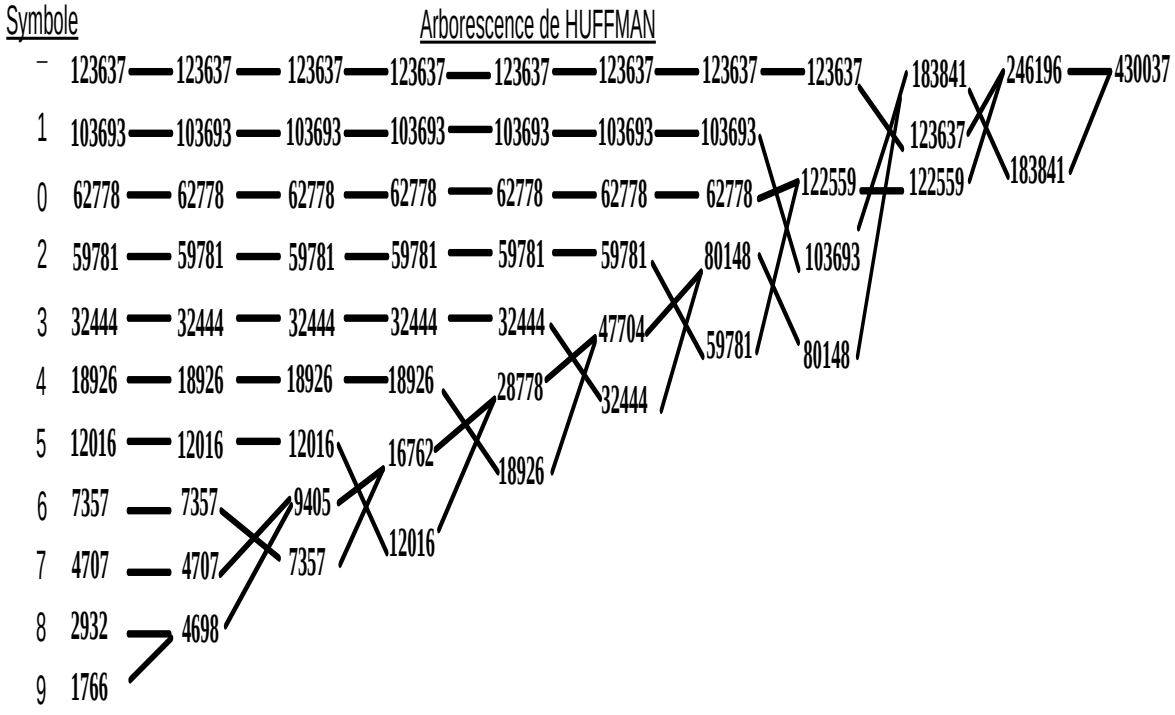


Figure III.17. *Arborescence de Huffman*

3éme étape :

Coder avec retour arrière depuis le dernier groupe, et en ajoutant un 0 ou un 1 pour différencier les symboles préalablement regroupés.

Pour obtenir les codes, il suffit de parcourir l'arbre depuis le motif étudié jusqu'à la fin du chemin en empilant les symboles binaires rencontrés sur le chemin. Les résultats de recherche sont donnés dans la figure suivante :

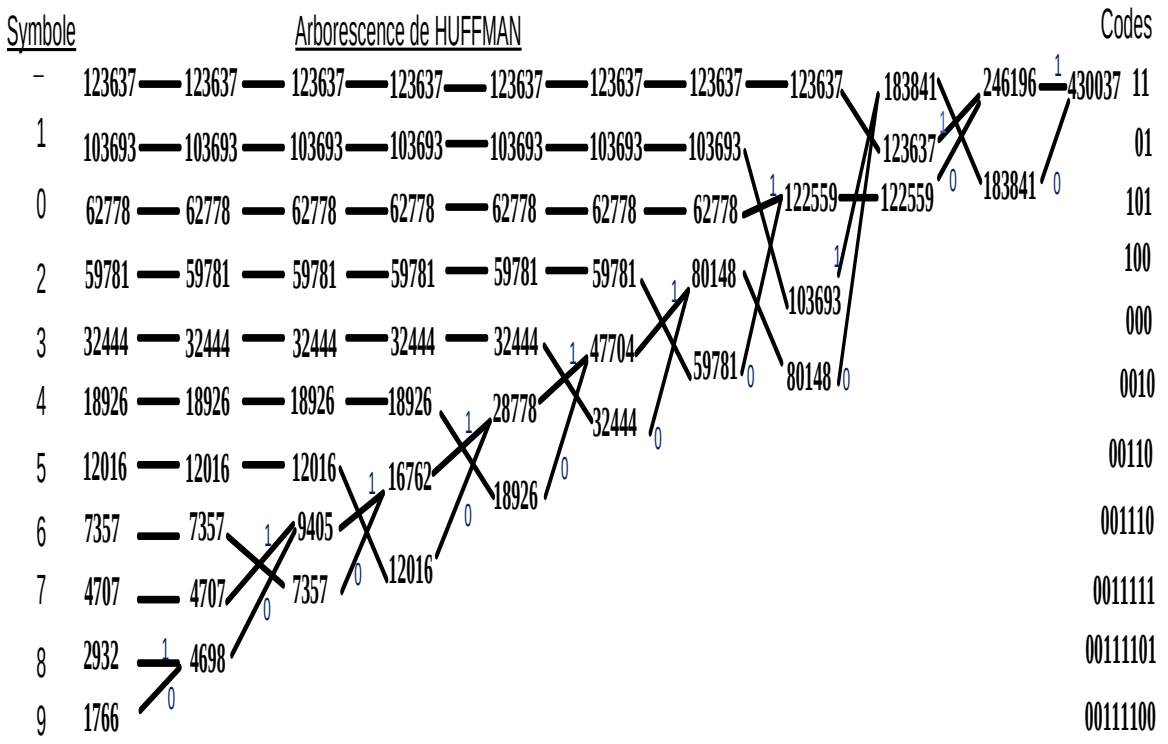


Figure III.18. *Arborescence de Huffman avec codes*

On remarque que le fichier compacté comporte 1170128 bits (32444×3+18926×4+12016×5+7357×6+1766×8+2932×8+4707×7+103693×2+59781×3+627 78×3+123637×2=1170128) contre 3440296 bits pour le fichier original.

Afin d'obtenir ce bitstream, j'ai créé un projet qui est constitué de trois étapes relatives au codeur de Huffman :

Etape 1: Calcul des probabilités d'apparition de chaque caractère dans l'image ordonnée. Dans l'image de différence ordonnée, j'ai obtenu onze symboles (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -). Pour un symbole donné, on associe la valeur correspondant de la table ASCII (American Standard Code for Information Interchange).Par exemple, si nous comptons les occurrences du nombre 7 dans ce fichier, alors on aura : compteur [7]= compteur [55].

Symboles	Fréquence d'apparition
3	**32444**
4	**18926**
5	**12016**
6	**7357**
9	**1766**
8	**2932**
7	**4707**
1	**103693**
2	**59781**
0	**62778**
-	**123637**

Tableau III. 4. *Les probabilités d'apparition des symboles*

L'établissement du tableau de fréquences se manifeste dans l'organigramme suivant :

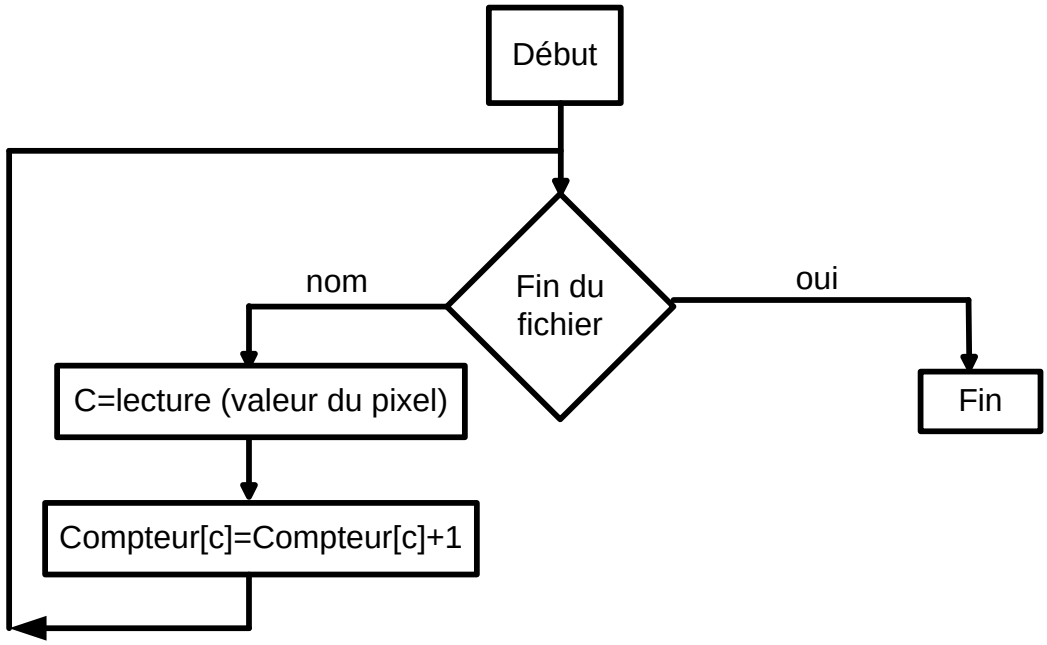


Figure III.19. *Organigramme de la fonction de comptage des symboles*

Etape 2:Génération des codes binaires selon ces fréquences utilisant le codage d'Huffman (voir Tableau 5).

Symboles	**Code binaire correspondant**
3	**000**
4	**0010**
5	**00110**
6	**001110**
9	**00111100**
8	**00111101**
7	**0011111**
1	**01**
2	**100**
0	**101**
-	**11**

Tableau III. 5. *Les codes binaires des symboles*

Etape 3 : Construction du bitstream.

Une partie de déroulement d'algorithme de construction du bitstream est résumé parle Table 9 dans l'annexe 9.

Après l'implémentation de ce projet, j'ai fait des tests pour comparer nos résultats avec les travaux de l'état de l'art. Le tableau suivant résume les résultats trouvés. Notre méthode fournit des résultats meilleurs que celle de H.264/AVC_LS _DPCM. Nous trouvons ces résultats car la structure à deux couches maintient les deux blocs de transformation et quantification pour profiter de leur taux de compression et corrige les erreurs effectuées de la couche de base par un second bitstream. Par contre, H.264/AVC_LS _DPCM transmit les valeurs du bloc de prédiction directement aux codages entropiques et ces valeurs sont très grandes et fortement corrélées (voir Tableau 6).

Sequences (50 frames)	H.264/AVC_LS_DPCM (Kilooctets)	Méthode de Wei Da Chien (Kilooctets)	Notre méthode (Kilooctets)
Foreman	7955	7707	4608
Mobile	11611	10573	7216
Akiyo	6432	5944	3764
FootBall	7929	7515	5247
WaterFall	10213	9453	6881

Tableau III. 6. *Comparaison de la structure à deux couches avec H.264/AVC_LS _DPCM [26]*

Pendant la première étape c'est-à-dire le codage de nombre de coefficients non nuls, le CAVLC se base sur des tableaux qu'il les choisit selon un paramètre de sélection N. ce paramètre dépend des blocs voisins en haut et à gauche du bloc courant (Nu et $N_{L)}$.Par contre, le codage Huffman n'utilise pas les blocs voisins, il traite seulement les coefficients courants c'est-à-dire les 11 symboles dans notre cas (0-9 et le signe (-). En se basant sur le fait que les codes binaires dans les tableaux VLC sont de longueurs variables, on peut prédire donc que la taille de bitstream codé par CAVLC est plus importante que celui obtenu par le codage huffman. Donc notre méthode atteint des résultats meilleurs que celle de Wei Da Chien puisque les codes binaires d'huffman sont de taille inférieure par rapport à CAVLC (voir Tableau 5).

7. Présentation et comparaison des méthodes de la norme HEVC

7.1. *Méthodes basées sur la prédiction*

7.1.1. *La prédiction angulaire intra par blocs dans la norme HEVC avec perte [27]*

Dans la norme HEVC [27], une prédiction angulaire intra par blocs est définie pour exploiter la redondance spatiale des échantillons dans les CU (Coding Unit) intra-codées. Comme le montre la Figure 20, 33 angles sont définis pour la prédiction angulaire, qui peut être classés en deux catégories : les prédictions angulaires verticales et horizontales.

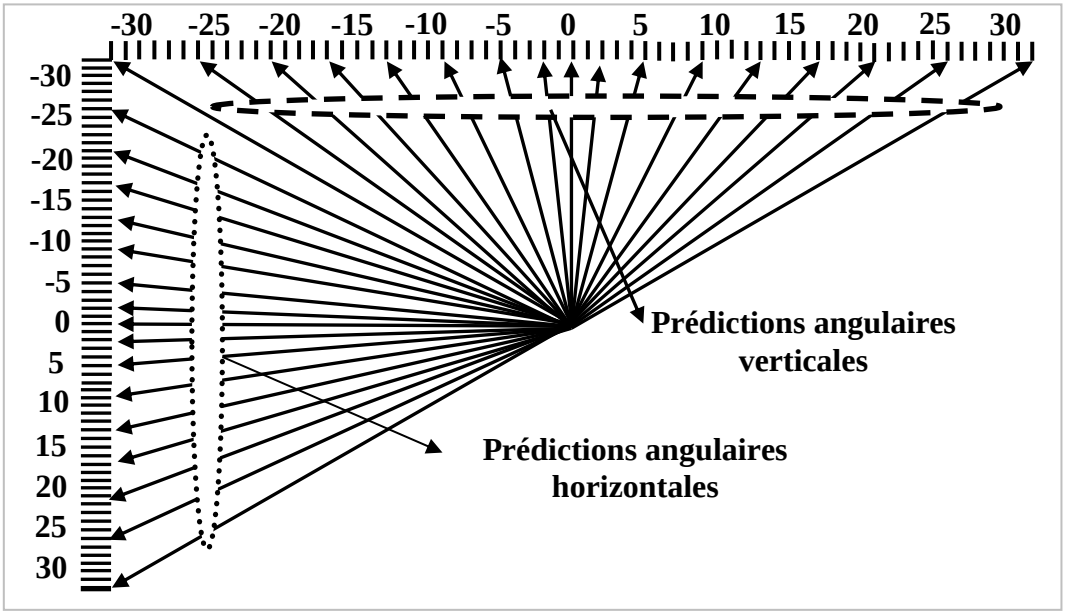


Figure III.20. *Définition de l'angle de prédiction intra angulaire en HEVC*

Comme la montre la Figure 21, pour une unité de prédiction (PU) N × N, la prédiction intra angulaire à base de bloc implique un total de 4N + 1 échantillons de référence provenant des PU voisines et tous les échantillons à l'intérieur du PU partagent le même angle de prédiction. Les échantillons de prédiction sont générés en utilisant uniquement les échantillons de référence des PU voisins gauches et supérieurs.

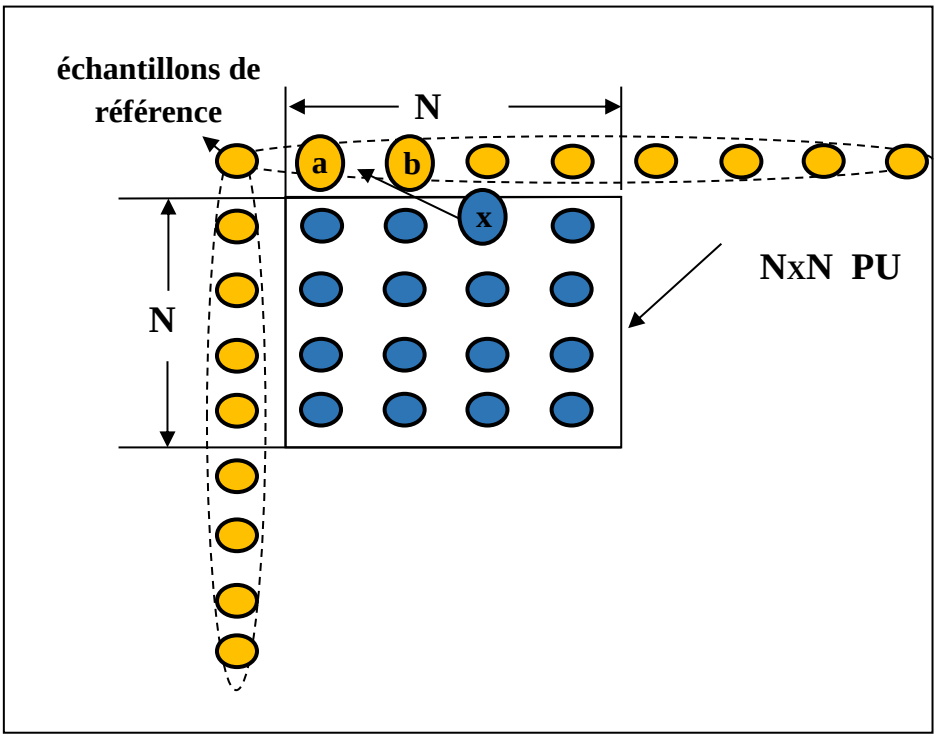


Figure III.21. *Prédiction intra angulaire à base de bloc en HEVC*

7.1.2. *Prédiction intra angulaire par échantillon SAP (Sample Angular Prediction)*

Comme illustré dans la Figure 22, le codage sans perte peut être réalisé en supprimant l'étape de transformation et de quantification du processus de codage. Bien que cela ne nécessite que quelques modifications de la chaine de codage avec perte originale, la seule modification requise est dans l'étape de de prédiction. Le processus de prédiction peut être modifié afin d'avoir une meilleure précision et des valeurs résiduelles plus petites.

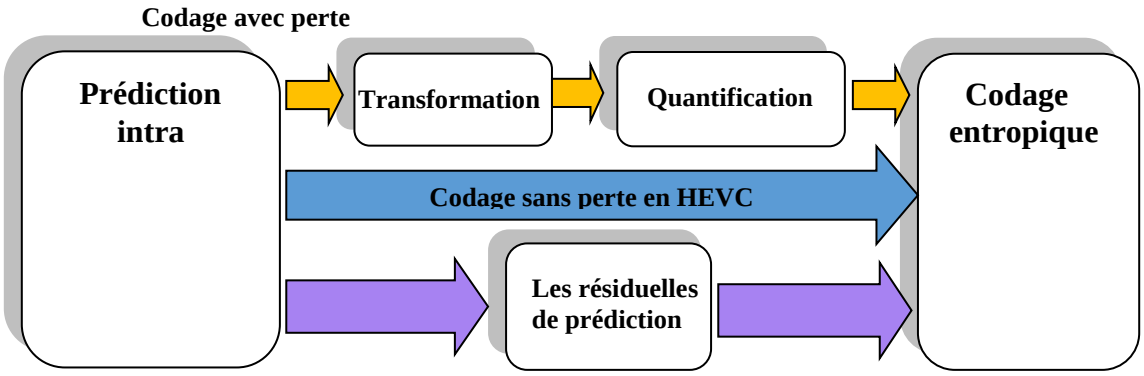


Figure III.22. *Codage sans perte dans HEVC*

Le mode de codage sans perte HEVC actuel peut être encore amélioré en introduisant des outils plus avancés. Un domaine qui peut être étudié pour l'amélioration de l'efficacité du codage sans perte est la prédiction intra. La prédiction intra devient plus importante dans le codage sans perte, car les statistiques montrent qu'il existe un pourcentage significatif de CU intra codées même lorsque des tranches P et / ou B sont utilisées [34].

Le SAP (Sample Angular Prediction) est conçu pour mieux exploiter la redondance spatiale dans le mode de codage sans perte en générant des échantillons intra-prédiction à partir de voisins adjacents. Le principe de conception ici est très similaire au DPCM basé sur l'échantillon présenté dans [28] pour le codage sans perte de la norme H.264 / MPEG-4 AVC [29], mais la méthode SAP est entièrement harmonisée avec la prédiction intra angulaire à base de bloc de la norme HEVC, et peut être appliquée à tous les modes de prédiction intra angulaire spécifiés dans HEVC.

Comme indiqué dans les Figures 23 et 24, dans la méthode SAP [30], [31], [32], la prédiction angulaire est effectuée échantillon par échantillon. Les échantillons voisins adjacents a, b de l'échantillon actuel x dans le courant PU sont utilisés pour la prédiction.

Pour les angles de prédiction verticaux (Figure 23), le SAP est traité ligne par ligne dans une unité de prédiction, tandis que pour les angles de prédiction horizontaux (Figure 24), le SAP est traité colonne par colonne dans une unité de prédiction.

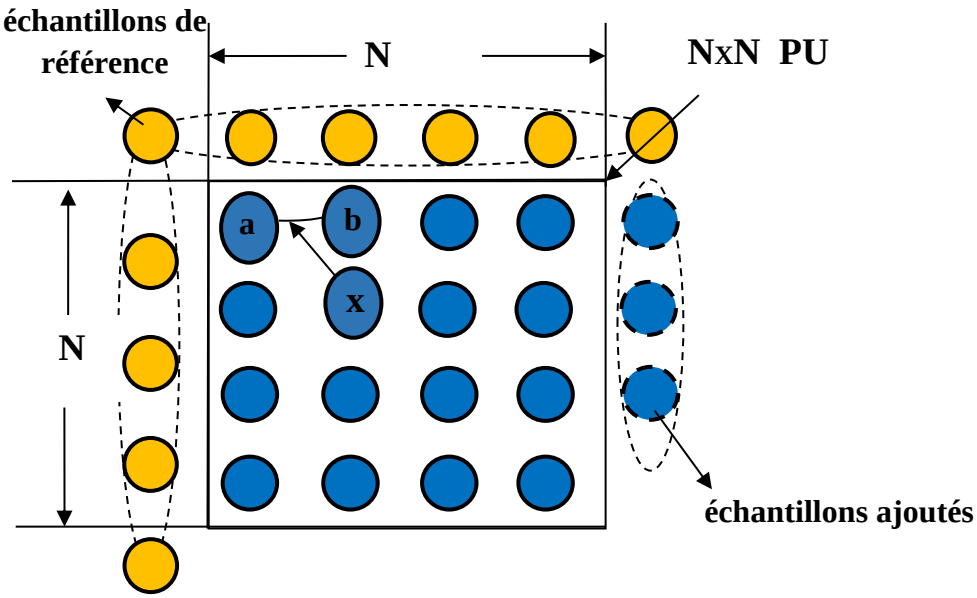


Figure III.23. *Sélection de l'échantillon de référence de SAP pour les angles de prédiction verticaux*

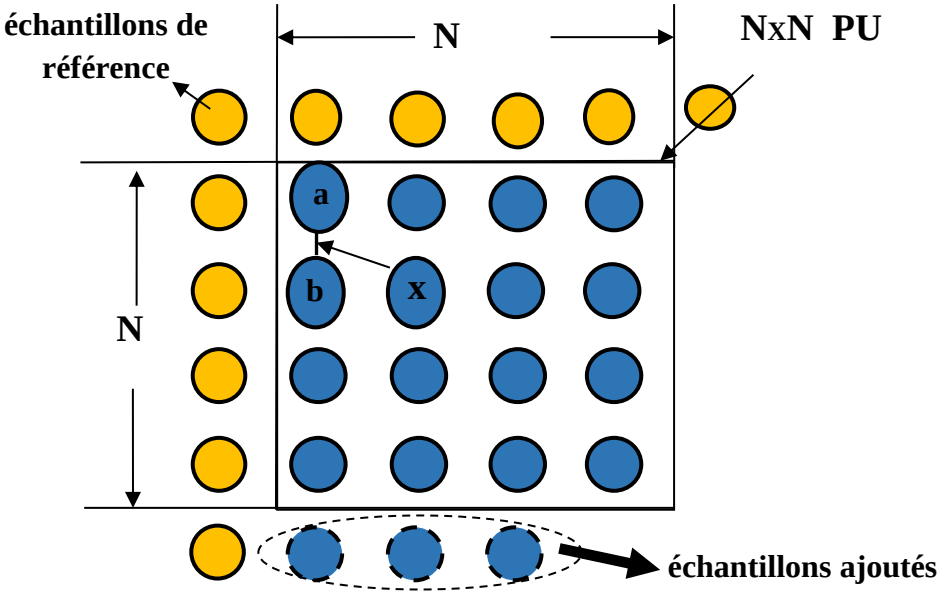


Figure III.24. *Sélection d'échantillon de référence de SAP pour les angles de prédiction horizontaux*

Le Tableau 8 montre les résultats de la comparaison de la performance de deux méthodes SAP et la méthode de codage sans perte réalisée dans la référence software HM 6.0 [33]. Le

Tableau 7 fournit les valeurs des taux de compression de la méthode de codage sans perte utilisant HM 6.0.

Tableau III. 7. *Taux de compression du codage sans perte HM6.0 [33]*

Classes des séquences	Taux de compression							
	AI-Main	RA-Main	LB-Main	LP-Main	AI-HE10	RA-HE10	LB-HE10	LP-HE10
A	2.05	2.40	2.41	2.35	1.21	1.40	1.41	1.39
B	2.03	2.35	2.36	2.29	1.21	1.39	1.39	1.37
C	1.92	2.51	2.52	2.43	1.13	1.45	1.46	1.44
D	1.79	2.63	2.64	2.55	1.06	1.50	1.50	1.49
E	2.65	3.23	3.19	3.11	1.47	1.80	1.81	1.81
Moyenne	2.09	2.62	2.62	2.55	1.22	1.51	1.51	1.50

Tableau III. 8. *Réduction du débit binaire (%) du SAP par rapport à la codage sans perte HM6.0 [34]*

Classes des séquences	Différence de débit binaire (%)							
	AI-Main	RA-Main	LB-Main	LP-Main	AI-HE10	RA-HE10	LB-HE10	LP-HE10
Classe A	−8.8	−2.8	−2.3	−3.0	−12.3	−4.1	−3.7	−4.6
Classe B	−5.1	−1.0	−0.7	−1.1	−8.5	−1.9	−1.4	−1.9
Classe C	−6.9	−1.8	−1.4	−1.5	−10.4	−3.0	−2.3	−2.5
Classe D	−8.4	−2.0	−1.5	−1.6	−12.0	−2.9	−1.9	−2.0
Classe E	−10.6	−3.1	−3.3	−4.3	−15.7	−4.8	−4.2	−4.2
Moyenne	−8.0	−2.1	−1.8	−2.3	−11.8	−3.3	−2.7	−4.2

Le SAP utilise les mêmes modes de prédiction et la même méthode d'interpolation d'échantillon que la prédiction angulaire à base de bloc de la norme HEVC, mais utilise des voisins adjacents pour une meilleure précision de prédiction intra et effectue la prédiction échantillon par échantillon.

Il existe deux approches principales pour améliorer la prédiction intra à base de bloc dans le codage intra sans perte. Dans le premier groupe d'approches, la prédiction intra à base de bloc est d'abord effectuée, puis le bloc d'erreur de prédiction est traité de nouveau avec une seconde étape de prédiction pixel par pixel [28, 35, 36, 37, 38] (par exemple les méthodes de prédiction RDPCM et CRDPCM de la référence 35. Dans le second groupe d'approches, l'approche de prédiction à base de bloc est directement remplacée par une approche de prédiction pixel par pixel [34, 39, 40, 41, 42] (par exemple les méthodes de prédiction PWDC, SAP et AD-SAP de la référence 39, 34 et 41. Le tableau suivant résume tous les méthodes basées sur la prédiction et compare leurs efficacités de codage.

Tableau III. 9. *Réduction en pourcentage du débit moyen de plusieurs méthodes de la littérature et de la méthode proposée de « 3-tap filtering » pour le codage sans perte de la norme HEVC [42]*

Classes des séquences	RDPCM [35]	CRDPCM [35]	PWDC [39]	SAP [34]	AD-SAP [41]	3-tap filtering [42]
Classe A	7.19	11.3	10.06	8.7	11.11	15.67
Classe B	3.54	3.91	5.29	5.11	5.45	7.81
Classe C	4.46	4.76	4.29	6.99	6.47	8.88
Classe D	6.3	6.91	4.89	8.65	8.85	11.1
Classe E	8.32	9.72	6.74	10.54	10.84	14.38
Classe F	9.82	10.11	4.77	12.44	13.75	11.81
Moyenne	6.12	8.43	5.95	8.51	9.41	11.34

Le Tableau 10 donne les temps d'exécution de codage et décodage de deux méthodes de prédiction SAP et 3-tap filtering

Tableau III. 10. *Temps de codage et décodage*

	Temps de codage	Temps de décodage
Méthode HEVC	100.0%	100.0%
Méthode SAP [34]	101.8%	94.4%
3-tap filtering [42]	109.7%	87.3%

7.2. *Méthodes basées sur le codage entropique modifié*

En codage avec perte, les coefficients de transformation des résiduelles de la prédiction sont codés par codage entropique, tandis que dans le codage sans perte, les résiduelles de prédiction sont directement codés par codage entropique. Considérant la différence des caractéristiques statistiques entre les coefficients transformés quantifiés et des coefficients des résiduelles de la prédiction, plusieurs modifications du codage entropique ont été proposées pour le codage sans perte dans la norme HEVC [43, 44, 45, 46, 47, 48].

Dans les travaux de Choi et al. [44], l'ordre de balayage est inversé et la binarisation utilisée dans le codeur entropique est modifiée. Un ordre de balayage dépendant du mode intra est proposé dans [45]. Le codage du dernier indicateur de position « flag » est modifié dans [46, 47] et le codage des niveaux des coefficients résiduelles est modifié dans [48]. Des résultats expérimentaux sont présentés dans le Tableau 11. On peut voir que la méthode proposée donne une efficacité de compression supplémentaire d'environ 0,72% de gain en

bits et de 2,10% de gain en bits au maximum par rapport au mode sans perte HEVC. À partir du Tableau 11, cette méthode fournit la meilleure performance de codage, comparée au mode sans perte de la norme HEVC [44].

Tableau III. 11. *Comparaison de gain de bits pour le mode sans perte HEVC et la méthode proposée [44]*

Sequences	**HEVC Mode sans perte (bytes)**	**Méthode de Choi et al. [46]**		**Gain en bits de la Méthode I (%)**	**Gain en bits de la Méthode II (%)**
		Méthode I (bytes)	**Méthode II (bytes)**		
Traffic	292043389	289021760	288978917	-1.03	-1.05
PeopleOnStreet	285283752	279453835	279453835	-2.04	-2.10
Kimono	135252302	135252302	134976664	-0.20	-0.21
ParkScene	162834309	162834309	161872833	-0.59	-0.60
Cactus	165415216	165415216	164741620	-0.41	-0.42
BasketballDrive	142954441	142954441	142590363	-0.25	-0.26
BQTerrace	159248773	159248773	158651984	-0.37	-0.43
BasketballDrill	29015828	29015828	28759282	-0.88	-0.90
BQMall	31047873	31047873	30892776	-0.50	-0.52
PartyScene	38443400	38443400	38237571	-0.54	-0.54
RaceHorses	31042618	31042618	30855443	-0.60	-0.59
BasketballPass	6938810	6938810	6853810	-1.22	-1.24
BQSquare	8940364	8940364	8893462	-0.52	-0.83
BlowingBubbles	9009542	9009542	8975937	-0.37	-0.37
RaceHorses	8309686	8309686	8249459	-0.72	-0.79
Moyenne				**-0.69**	**-0.72**

Tableau III. 12. *Changement de temps de codage (%)*

Sequences	**Méthode de Choi et al. [46]**	
	Méthode I	**Méthode II**
Traffic	+0.10	+0.05
PeopleOnStreet	-0.10	-0.78
Kimono	-0.31	-0.79
ParkScene	+0.65	-0.01
Cactus	-0.20	-0.01
BasketballDrive	-0.30	-0.06
BQTerrace	+0.51	-0.22
BasketballDrill	+0.33	-0.16
BQMall	+0.35	+0.18
PartyScene	-0.08	+0.23
RaceHorses	-0.05	-0.05
BasketballPass	-1.96	-1.16

BQSquare	-0.44	-0.39
BlowingBubbles	+0.18	-0.05
RaceHorses	-0.17	+0.50

Il est montré que toutes les augmentations du temps de codage sont inférieures à 0,65%. Dans certains cas, le temps de codage est plutôt diminué. La quantité de temps de codage diminué est de 1,96% au maximum, comparé au mode sans perte HEVC (voir Tableau 12) [44]. Les Tableaux 13 et 14 donnent les résultats de comparaisons de la méthode proposée dans [48] par rapport aux diverses méthodes existant dans la littérature.

Tableau III. 13. *Comparaison du taux de compression de diverses méthodes de codage sans perte [48]*

Classe	Sequences	HEVC	H.264 /AVC	JPEG 2000	JPEG-LS	SAP [4,6]	SAP+ méthode de choi et al.[24]
A	Traffic	2.16	2.28	2.45	**2.56**	2.43	2.50
	PeopleOnStreet	2.21	2.29	2.50	**2.70**	2.45	2.53
B	Kimono	2.32	2.35	2.58	**2.59**	2.50	2.55
	ParkScene	1.96	1.93	2.17	**2.20**	2.08	2.10
	Cactus	1.94	1.85	2.03	**2.06**	2.02	2.02
	BQTerrace	1.91	1.77	1.98	2.00	2.00	**2.02**
	BasketballDrive	2.19	2.20	2.26	2.29	2.36	**2.38**
C	RaceHorses	2.00	1.82	2.20	**2.27**	2.16	2.21
	BQMall	2.08	2.05	2.16	**2.26**	2.21	2.24
	PartyScene	1.65	1.39	1.72	1.79	1.74	**2.90**
	BasketballDrill	2.10	1.91	2.06	2.18	2.27	**2.31**
D	RaceHorses	1.86	1.67	2.07	**2.14**	2.05	2.09
	BQSquare	1.69	1.42	1.72	**1.83**	1.78	1.81
	BlowingBubbles	1.59	1.33	1.67	1.74	1.74	**1.75**
	BasketballPass	2.15	2.18	2.33	**2.53**	2.42	2.48
E	FourPeople	2.55	2.64	2.82	**3.00**	2.87	2.93
	Johnny	2.78	2.85	2.99	3.12	3.09	**3.15**
	KristenAndSara	2.79	2.86	3.00	**3.19**	3.12	3.18
F	BasketballDrillText	2.14	1.95	2.07	2.21	2.33	**2.38**
	ChinaSpeed	3.00	3.23	2.85	3.31	3.59	**4.16**
	SlideEditing	3.73	3.80	2.95	4.03	4.14	**5.21**
	SlideShow	9.25	9.93	8.79	10.45	10.15	**11.32**
Moyenne		2.55	2.53	2.61	2.84	2.80	**3.01**

Tableau III. 14. *Gain de temps de décodage de la méthode de Choi et al. par rapport à la méthode SAP [48]*

Classe	Sequences	SAP [4,6]	SAP+ méthode de choi et al.[24]
A	Traffic	−4.86	−4.60
	PeopleOnStreet	−4.00	−3.98
B	Kimono	−2.79	−2.97
	ParkScene	−0.36	−0.35
	Cactus	0.02	−0.30
	BQTerrace	−0.76	−1.24
	BasketballDrive	−1.40	−2.38
C	RaceHorses	−2.80	−3.65
	BQMall	−1.62	−2.01
	PartyScene	−0.30	−0.56
	BasketballDrill	−3.21	−3.60
D	RaceHorses	−2.76	−3.28
	BQSquare	0.00	−0.70
	BlowingBubbles	−2.41	−2.92
	BasketballPass	−3.06	−3.23
E	FourPeople	−2.92	−3.55
	Johnny	−1.49	−1.67
	KristenAndSara	−2.26	−2.70
F	BasketballDrillText	−3.17	−3.65
	ChinaSpeed	−6.53	−8.78
	SlideEditing	−2.11	−5.34
	SlideShow	−3.31	−4.97
Moyenne		−2.37	−3.02

7.3. *Méthodes basées sur des architectures à deux couches*

7.3.1. *Introduction*

La Figure 25 illustre le schéma de l'encodeur de codage sans perte HEVC dans lequel la transformation, la quantification et les filtres sont supprimés. La prédiction intra, la prédiction inter et le codage entropique sont utilisés tels quels dans le mode de codage sans perte pour exploiter la redondance spatiale, temporelle et statistique [49].

Dans la norme de codage HEVC, le codage sans perte est réalisé en codant les résiduelles de prédiction intra / inter directement dans le domaine spatial. En codage sans perte, l'encodeur HEVC commence par la division de l'image d'entrée en unités de codage UC (Coding Unit) / unités de prédiction PU (Prediction Unit) puis la prédiction intra / inter, et enfin les résidus de prédiction sont obtenus. Pour coder ces résidus, les étapes de transformation et de quantification sont ignorées. Les signaux résiduels sont codés directement en utilisant le codage entropique CABAC (Context Adaptive Binary Arithmetic Coder). Le schéma

synoptique de la Figure 25 illustre cette procédure. Un avantage de la stratégie de codage sans perte de la norme HEVC est la résolution du problème de la complexité de calcul, lorsque le calcul intensif de la transformation DCT (Discret Cosine Transform) est surmonté.

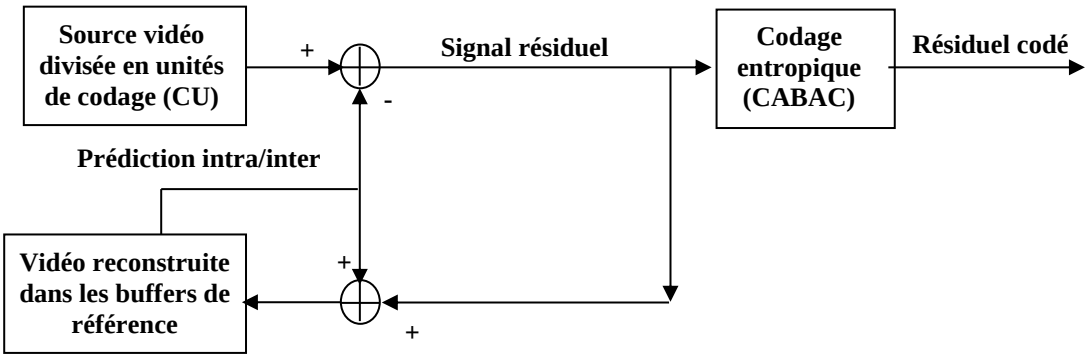


Figure III.25. *Schéma de la norme HEVC avec le mode de codage sans perte en générale [49]*

7.3.2. *Méthode de Xun Caiet al.*

La Méhode de Xun Cai et al sépare un bloc de signal résiduel en deux parties. La première partie est constituée des coefficients DCT quantifiés et la deuxième partie est formée par les coefficients de l'erreur de quantification. Pour être précis, tout d'abord, les coefficients DCT des résiduelles de prédiction sont obtenus. Puis ces coefficients DCT sont quantifiés et codés en tant que bloc DCT. D'autre part, les coefficients quantifiés transformés sont utilisés pour reconstruire un bloc avec perte et il est soustrait du bloc résiduel afin d'obtenir l'erreur de quantification. Cette erreur de quantification est codée en tant que bloc spatial. En choisissant un paramètre de quantification approprié pour chaque bloc, les deux parties peuvent être codées efficacement en utilisant CABAC comme codeur entropique de la norme HEVC. Le schéma synoptique de la Figure 26 illustre cette idée [49].

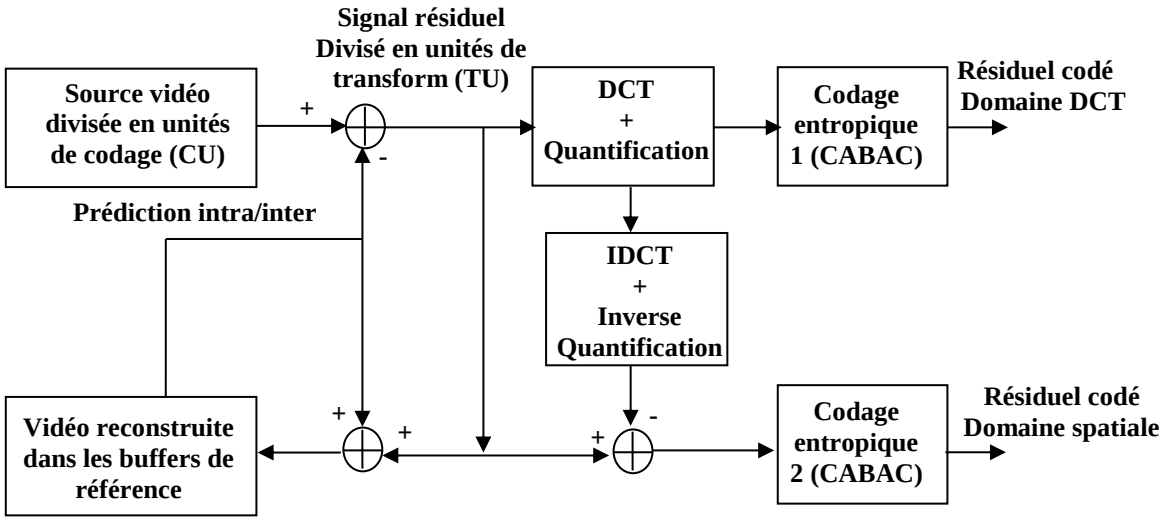


Figure III.26. *Codage sans perte en deux étapes [49]*

L'evaluation de la méthode de *Xun Cai* et al. est résumée dans le tableau suivant :

Tableau III. 15. *Performance du codage en deux étapes [49]*

Paramètres	Gain de codage
720 Intra	12.14%
720 Intra+Inter	5.69%
480 Intra	4.48%
480 Intra+Inter	1.64%

7.3.3. *Méthode de Andreas Heindel et al.*

7.3.3.1. Première version

Premièrement, la vidéo peut être compressée par un codeur vidéo avec perte et transmise au décodeur. Ces images compressées avec perte sont mentionnées comme la couche de base (BL). Dans la deuxième étape, les informations perdues par la procédure de compression avec perte sont envoyées. Ceci est dénoté comme la couche d'amélioration (EL). Finalement, la reconstruction sans perte peut être réalisée en combinant les deux couches [50].

Un système de codage vidéo sans perte "scalable" est proposé, qui étend la couche de base (BL) de la norme HEVC par une couche d'amélioration (EL). Une vue d'ensemble de ce système est présentée dans la Figure 27. La vidéo reconstruite avec perte du codeur BL est utilisée pour le calcul de l'erreur de reconstruction, qui est par la suite codée dans la couche d'amélioration (EL). Le codeur EL est interprété comme un nouvel outil, qui permet un codage sans perte efficace s'il est activé.

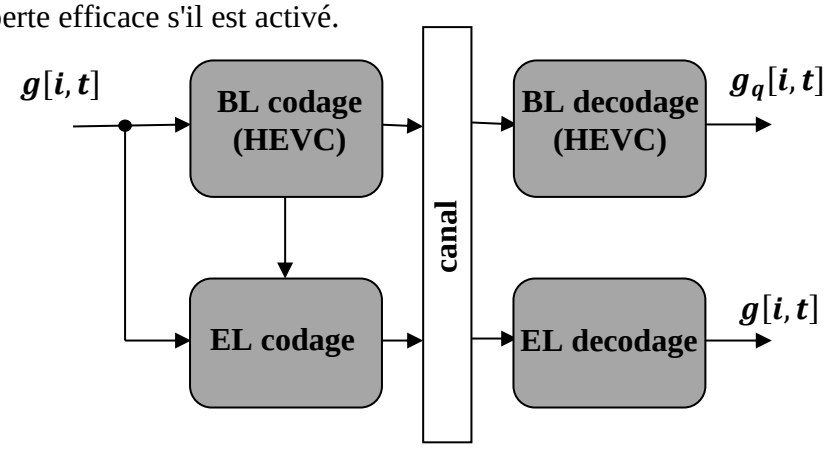


Figure III.27. *Structure générale du système proposé permettant un codage vidéo "scalable" avec perte et sans perte [50]*

7.3.3.2. Deuxième version

Pour une compression efficace de l'EL, un codeur dédié est proposée. L'algorithme de compression suggéré est basé sur l'algorithme de prédiction intra SWP (Sample-based Weighted Prediction) ([54], [55], [56]) et donc appelé SELC (Sample-based weighted prediction for Enhancement Layer Coding). Après l'étape de prédiction SWP, l'erreur de prédiction obtenue est codée par le codage entropique CABAC (Context-based Adaptive Binary Arithmetic Coding) [57] (voir Figure 28) [51].

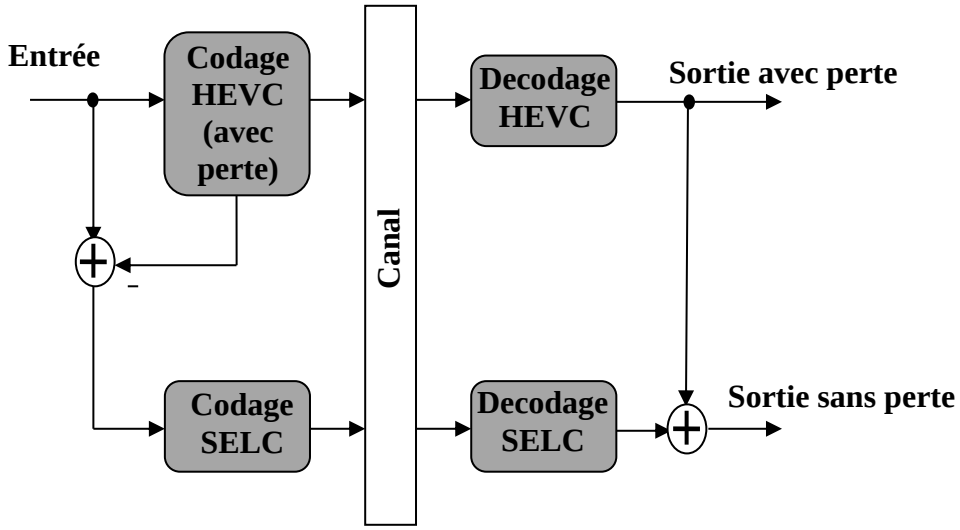


Figure III.28. *Amélioration de la structure hiérarchique utilisant le concept SELC [51]*

Le système proposé permettant un codage vidéo "scalable" avec perte et sans perte est introduit. La Figure 29 donne une vue d'ensemble détaillée de ce système. Les blocs individuels dénotent les composants comme suit:

_ transformation et quantification (T / Q)

_ quantification inverse et transformation inverse (IQ / IT)

_ prédiction intra (Pred Intra.)

_ filtre (LF)

_ buffer d'image décodé (DPB)

_ prédiction à compensation de mouvement (MC Pred.)

_codage entropique (EC) et décodage entropique (ED) (CABAC a été utilisé comme codage entropique).

Les deux parties supérieures de la Figure 29 représentent le codeur et le décodeur pour le BL. Du fait de la conformité de ces codeurs avec la norme HEVC, une implémentation matérielle peut être utilisée dans la pratique. Dans le cadre d'application software de ce travail, le logiciel de référence de la norme HEVC « HM-11.0 » est utilisé pour ce but. Les parties inférieures de la Figure 29représentent le codeur et le décodeur pour l'EL.

La vidéo d'entrée d'origine est appelée g [i; t], où i est un index bidimensionnel dénotant la position dans l'image et t représente la dépendance au temps. Deux signaux de sortie peuvent être obtenus de l'encodeur BL. Une sortie est le "bitstream" codé, qui est immédiatement envoyé au récepteur via le canal de communication à être utiliser. De plus, les données vidéo reconstruites avec pertes gq [i; t] est accessible depuis la boucle de décodage dans l'encodeur. gq[i; t] peut être utilisé pour calculer la différences des images r[i; t], contenant les informations perdues lorsque l'utilisateur demande la reconstruction sans perte de certaines images, les résiduelles de reconstruction obtenues r [i; t] doit être compressé et transmis dans le EL.

Le schéma de codage proposé SELC pour l'EL nécessite deux signaux d'entrée, à savoir la vidéo d'entrée g [i, t] et la vidéo gq [i, t], qui est créée par le codeur BL.

Ainsi, r [i, t], résulte de la soustraction de gq [i, t] du signal original g [i, t] :

$$r[\mathbf{i}, t] = g[\mathbf{i}, t] - gq[\mathbf{i}, t] \quad (4)$$

Ensuite une prédiction SWP est appliquée. Le concept général de SWP est basé sur l'idée de prédire un pixel r [i] en faisant la moyenne de plusieurs pixels voisins, provenant d'un voisinage causal. Pour le calcul de l'erreur de prédiction e [i, t], la différence entre le résidu de reconstruction r [i, t] et la prédiction p [i, t] est calculée comme suit

$$e[\mathbf{i},t] = r[\mathbf{i},t] - p[\mathbf{i},t] \quad (5)$$

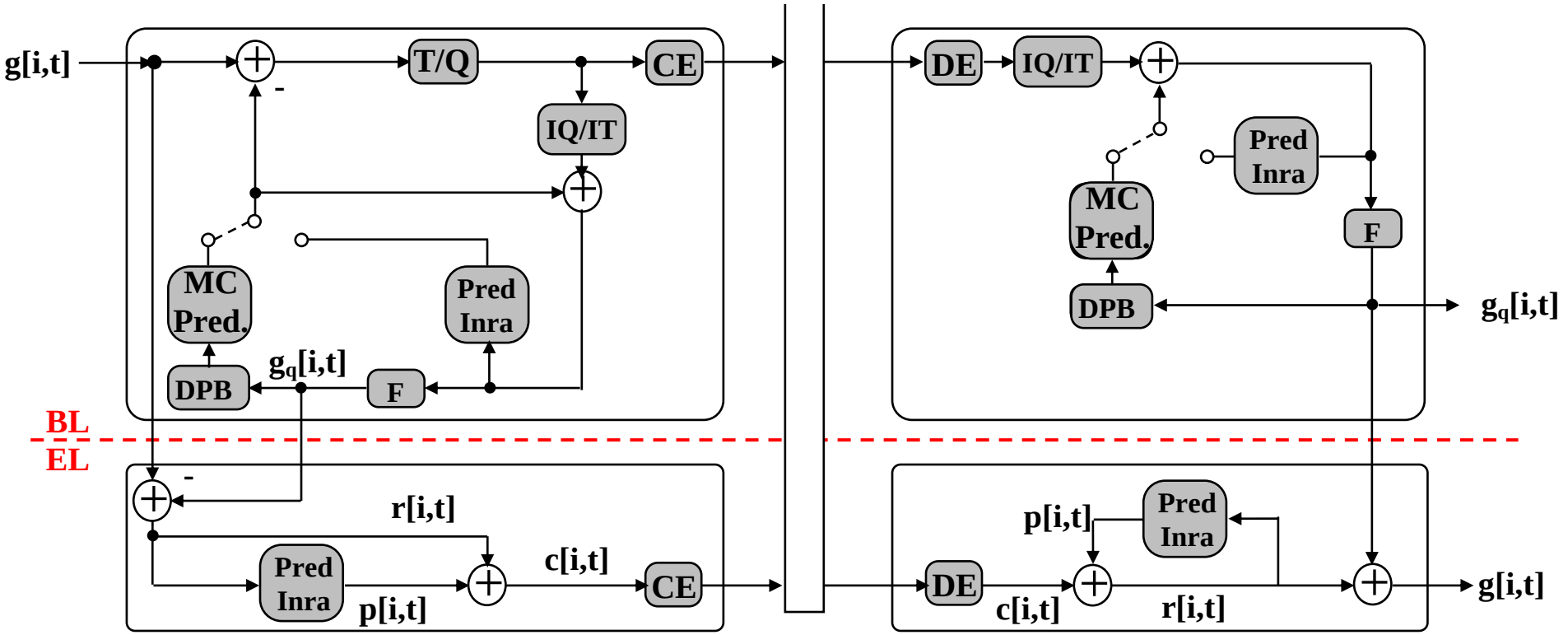


Figure III.29. *Structure détaillée du système proposé [51,52]*

Cette section donne aussi des résultats de comparaison des méthodes proposées dans la littérature pour la structure hiérarchique (voir Tableau 16, 17 et 18) [51, 53].

Tableau III. 16. *La différence de débit relative pour le codage EL avec hm-11,0 et SELC [51]*

Sequences	HM-11.0				SELC			
	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37
PeopleOnStreet	1.0%	1.0%	1.4%	2.9%	-5.6%	-8.7%	-10.1%	-9.9%
Kimono	0.6%	5.5%	-0.9%	-0.5%	-3.5%	-6.6%	-9.1%	-9.9%
ParkScene	0.8%	1.7%	0.2%	0.1%	-1.7%	-3.1%	-5.9%	-7.7%
ElFuente	8.7%	10.0%	11.8%	17.0%	102.0%	92.1%	83.9%	86.7%
BQMall	6.2%	0.6%	0.2%	0.8%	-0.6%	-1.8%	-4.2%	-5.3%
PartyScene	0.7%	0.3%	0.0%	0.4%	-0.4%	-1.0%	-2.0%	-3.7%
RaceHorses	1.1%	1.0%	1.1%	1.8%	-2.7%	-5.2%	-7.0%	-8.1%
BasketballPass	0.5%	-0.1%	0.7%	2.3%	-2.1%	-4.6%	-5.7%	-5.4%
MobileCalendar	0.8%	0.7%	0.5%	0.3%	-0.5%	-1.0%	-1.8%	-3.6%
City	0.8%	-0.0%	-0.9%	-0.6%	-3.4%	-6.3%	-9.2%	-10.5%
Akiyo	0.9%	0.2%	0.1%	1.6%	-2.7%	-6.8%	-8.3%	-7.7%
HallMonitor	0.8%	0.3%	-0.1%	0.2%	-6.3%	-8.2%	-9.3%	-9.4%
Silent	0.8%	0.5%	0.6%	0.8%	-2.2%	-2.8%	-4.8%	-6.7%
Moyenne	1.8%	1.7%	1.1%	2.1%	5.4%	2.8%	0.5%	-0.1%
Moyenne par rapport à ElFuente	1.2%	1.0%	0.3%	0.8%	-2.6%	-4.7%	-6.5%	-7.3%

Tableau III. 17. *Temps de codage supplémentaires pour l'EL par rapport à codage du BL [51]*

Sequence	SHM-2.1				HM-11.0				SELC			
	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37
PeopleOnStreet	19.7%	23.9%	28.1%	30.9%	14.3%	17.4%	20.4%	22.8%	0.7%	0.3%	0.7%	0.5%
Kimono	25.6%	29.6%	34.0%	36.6%	17.9%	21.7%	24.6%	27.1%	0.8%	0.1%	0.3%	0.3%
ParkScene	26.7%	32.3%	37.1%	40.0%	19.5%	23.8%	27.2%	29.5%	0.2%	0.2%	0.4%	0.3%
ElFuente	22.8%	23.0%	23.1%	23.0%	19.7%	19.7%	19.8%	19.7%	1.0%	1.2%	1.2%	0.6%
BQMall	26.2%	31.0%	35.4%	39.0%	19.0%	22.4%	25.5%	28.0%	0.9%	0.7%	0.5%	0.7%
PartyScene	21.0%	26.7%	32.1%	36.7%	15.3%	19.4%	23.3%	26.6%	0.1%	0.4%	0.9%	1.1%
RaceHorses	17.9%	21.3%	25.7%	29.0%	12.5%	15.2%	18.4%	21.6%	0.6%	0.7%	0.7%	0.2%
BasketballPass	20.4%	24.1%	28.6%	33.1%	14.8%	17.6%	20.8%	23.8%	0.5%	0.6%	0.6%	1.0%
MobileCalendar	26.5%	34.9%	39.8%	41.3%	18.6%	25.4%	28.3%	30.1%	0.5%	0.4%	0.8%	0.8%
City	24.6%	33.0%	37.6%	40.2%	17.8%	24.1%	27.3%	29.5%	0.2%	0.7%	0.5%	0.4%
Akiyo	41.0%	44.4%	47.2%	48.9%	29.3%	32.1%	34.1%	35.5%	1.3%	1.4%	1.6%	1.7%
HallMonitor	27.3%	38.6%	45.8%	48.4%	19.9%	28.3%	33.0%	35.1%	0.6%	0.8%	1.7%	1.6%
Silent	29.8%	34.8%	38.7%	42.9%	21.6%	25.3%	28.2%	30.9%	0.7%	1.4%	1.4%	1.6%
Average	25.3%	30.6%	34.9%	37.7%	18.5%	22.5%	25.5%	27.7%	0.6%	0.7%	0.9%	0.8%

Tableau III. 18. *Temps de décodage supplémentaires pour l'EL par rapport au décodage du BL [51]*

Sequence	SHM-2.1				HM-11.0				SELC			
	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37
PeopleOnStreet	208.3%	304.2%	414.5%	501.8%	216.8%	317.4%	416.4%	495.5%	176.5%	228.5%	273.7%	315.1%
Kimono	268.1%	356.1%	469.5%	558.1%	286.0%	418.5%	476.1%	547.2%	217.1%	295.9%	333.8%	365.3%
ParkScene	251.6%	333.4%	438.5%	559.5%	267.7%	366.6%	459.9%	564.7%	193.9%	301.8%	354.6%	396.9%
ElFuente	130.8%	136.2%	135.8%	137.0%	161.3%	164.9%	166.5%	169.8%	205.5%	260.8%	251.8%	250.2%
BQMall	260.3%	340.3%	469.5%	580.2%	291.9%	367.8%	478.1%	571.0%	200.7%	284.6%	357.3%	401.7%
PartyScene	210.0%	294.1%	401.0%	528.7%	224.7%	314.5%	415.6%	529.2%	154.0%	225.7%	301.6%	393.9%
RaceHorses	178.8%	254.9%	344.8%	446.3%	182.3%	256.5%	346.7%	445.2%	143.7%	194.1%	249.0%	293.2%
BasketballPass	205.8%	296.4%	422.9%	492.6%	223.1%	307.8%	408.0%	476.7%	164.3%	224.2%	277.4%	306.3%
MobileCalendar	260.4%	382.1%	463.7%	538.3%	272.2%	398.5%	481.4%	558.8%	192.0%	295.3%	382.6%	437.3%
City	252.4%	381.0%	495.8%	623.9%	269.1%	420.9%	528.6%	629.7%	207.2%	323.0%	373.9%	418.1%
Akiyo	365.9%	476.5%	608.9%	675.7%	379.8%	493.3%	588.4%	653.3%	324.0%	361.3%	393.8%	417.8%
HallMonitor	276.5%	446.0%	593.3%	701.8%	291.0%	466.2%	590.8%	674.8%	237.5%	348.9%	411.8%	449.5%
Silent	308.1%	404.7%	508.2%	626.4%	315.0%	411.3%	511.2%	623.1%	219.7%	290.0%	385.1%	427.3%
Average	244.4%	338.9%	443.6%	536.2%	260.1%	361.9%	451.4%	533.8%	202.8%	279.6%	334.3%	374.8%

8. Conclusion

Une nouvelle technique d'amélioration de la méthode de compression vidéo sans perte à deux couches a été traité dans ce chapitre. En effet, cette amélioration consiste à ajouter trois blocs dans la structure hiérarchique de Wei Da Chien. Le premier bloc est consacré pour faire un code de différence entre l'image reconstruite et l'image source. Le deuxième bloc sert à implémenter un code d'ordonnancement des coefficients des frontières de la composante de la luminance suit du code d'ordonnancement des coefficients des frontières des deux composantes de la chrominance. Un code d'ordonnancement pour le reste des coefficients "cores" de la luminance et un code d'ordonnancement pour le reste des coefficients "cores" de la chrominance ont été élaborés. Le troisième bloc consiste à faire intégrer le code du codage entropique CAVLC du code LETI dans le logiciel JM. Pour corriger l'erreur du bitstream de la première couche, je fais une concaténation entre les coefficients du bitstream avec perte et les coefficients du bitstream généré de la seconde couche.

La compression sans perte joue un rôle important dans le domaine du cinéma numérique et les applications des hautes définitions. La norme HEVC est destinée pour les applications vidéo de grandes résolutions. Il y a une variété de méthodes qui modifient la norme HEVC pour la rendre conservative.

Conclusion Générale

Conclusion générale et perspectives

J'ai montré par une recherche bibliographique l'existence de trois grandes familles de méthodes de compression vidéo sans perte. La première famille englobe des méthodes qui demeurent secrètes sur leurs algorithmes dans un but commercial. La seconde famille comprend une variété de méthodes innovantes, appelée aussi méthodes non conventionnelles. La troisième famille comporte un ensemble de méthodes basées sur des algorithmes et des normes relatifs à la compression sans perte des images fixes tel que JPEG-LS et JPEG2000 ; et des séquences vidéo tel que la norme H264/AVC. J'ai présenté quelques méthodes des trois familles dans le chapitre 1.

Le deuxième chapitre est composé par deux sections principales, la première section introduit une étude comparative entre deux codeurs entropiques CABAC et CAVLC de la norme H.264/AVC. Les résultats prévisibles de cette étude comparative montrent la performance du codeur CABAC par des taux de compression plus élevés et la rapidité du codeur CAVLC par des temps d'exécution plus bas. La deuxième section consiste à discuter les différentes méthodes de compression vidéo sans perte qui modifient la norme H264/AVC pour le rendre conservatif, à étudier et à analyser le logiciel JM, à modifier les paramètres de codage de l'approche de compression vidéo sans perte à deux couches (celle de Wei-Da Chien et AL). Il est important de souligner ici que notre méthode a amélioré les résultats publiés par Wei-Da Chien et al. par une augmentation des taux de compression.

Notre contribution à la mise au point d'une technique de compression vidéo sans perte basée sur une modification de la norme H.264 "Advanced Video Coding" (AVC) est traité dans le chapitre 3. Le but de notre travail est d'améliorer la structure hiérarchique de Wei Da Chien. En effet j'ai implémenté un code d'ordonnancement des coefficients des frontières et des coefficients des cores d'une image de format CIF. La conception d'une nouvelle technique de réarrangement des coefficients résiduels à l'entrée du codeur entropique a pour but d'améliorer le taux de compression. Dans ce cadre, je suis en train d'implémenter le codeur entropique CAVLC pour faire une étude comparative. Je suis en cours d'implémenter un code pour le codage RLE dans le but de faire une évaluation des performances et de l'efficacité de différentes méthodes.

Dans le cadre de la compression sans perte, beaucoup de recherches se dirigent vers la nouvelle norme HEVC. En effet, plusieurs méthodes ont été élaborées pour répondre aux besoins des utilisateurs.

Il y a des fichiers militaires et des rapports de recherches médicales qui ne tolèrent aucune distorsion et ils sont au même temps secrets aux propriétaires. Afin de conserver l'information source et garantir la sécurité des données, la solution est la compression vidéo sans perte avec un cryptage.

Bibliographies

Chapitre I:

[1] C.E. Shannon : A mathematical theory of communication. Bell System Technical Journal, 27:379–423, 623–656, 1948. 2.3.1.6

[2] K.H. Tzou., “Post-filtering of transform-coded images”, In Proc. SPIE Applications of Digital Image Processing XI, 974:121-126, 1988.

[3] Y. Yang, N.P. Galatsanos, and A.K. Katsaggelos., “Projection-based spatially-adaptative reconstruction of block-transform compressed images”, IEEE Trans. Image Proc. 4(7):896-908, Juillet 1995.

[4] R.C. Gonzalez and R.E. Woods., “Digital Image Processing”, Reading, MA: Addison-Wesley, 1992.

[5] M. Kaneko, Y. Hatori, and A. Koike., “Improvements of transform coding algorithm for motion-compensated inter frame prediction errors DCT/SQ coding”, IEEE J. Sel. Areas Commun, SAC 5(7):1068-1078, Aout 1987.

[6] Weinlich, A., Rehm, J., Amon, P., Hutter, A., and Kaup, A.," Massively parallel lossless compression of medical images using least-squares prediction and arithmetic coding ". In 20th IEEE International Conference on Image Processing (ICIP), 2013, pp.1680-1684.

[7] Sanchez, V., and Bartrina-Rapesta, J., " Lossless compression of medical images based on HEVC intra coding" In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6622-6626.

[8] Taquet, J., and Labit, C., "Hierarchical oriented predictions for resolution scalable lossless and near-lossless compression of CT and MRI biomedical images" IEEE Transactions on image processing, 2012, 21(5), 2641-2652.

[9] Xu, Z., Bartrina-Rapesta, J., and Serra-Sagristà, J., "Diagnostically lossless compression strategies for x-ray angiography images", 2015.

[10] Taquet, J., and Labit, C., " Compression sans perte et presque sans perte d’images médicales à l’aide d’un prédicteur hiérarchique orienté et adaptatif", une, 4, 3π, 2010.

[11] D. Vatolin, I. Seleznev, and M. Smirnov, “Lossless Video Codecs Comparison’2007,” Technical Report of the Graphics & Media Lab (Video Group) of Moscow State University (MSU).Available:http://www.compression.ru/video/codec_comparison/lossless_codecs_2007_en.html

[12] http://compression.ru/video/ls-codec/index_en.html

[13] http://lags.leetcode.net/codec.html

[14] C. Keimel, C. Pangerl, and K. Diepold, “Comparison of Lossless Video Codecs for Crowd-based Quality Assessment on Tablets,” in Seventh International Workshop on Video Processing and Quality Metrics for Consumer Electronics – VPQM 2013, pp. 37-41, Jan., 2013.

[15] YUVsoft Corp., 2016, “Lossless Video Codec”http://web.archive.org/web/20161128130123/http://www.yuvsoft.com/2d-technologies/lossless-video-codec/

[16] Jeyakumar, S., and Sundaravadivelu, S., “A Wavelet based Lossless Video Compression using Adaptive Prediction and Motion Approximation,” A publication of The International Congress for global Science and Technology (ICGST), International Journal on Graphics, Vision and Image Processing (GVIP), August 2009, Vol9, Issue 4, pp 15-21.

[17] Bang, D., Tang, H., and Kamata, S. I., “Linear Predictor using 3-D Projection for video lossless compression,” IEEE International Symposium on Industria l Electronics (ISlE 2009), Seoul Olympic Parktel, Seoul, Korea, 5-8 July 2009, pp. 1914-1918.

[18] Andriani, S., Calvagno, G., and Mian, G. A., “Lossless video compression using a spatio temporal optimal predictor,” In Proc. EURASIP European Signal Processing Conference (EUSIPCO 2005), Antalya, Turkey, September 2005, pp. 1-4.

[19] B. Meyer and P. Tischer," GLICBAWLS – Grey Level Image Compression By Adaptive Weighted Least Squares", in Proc. Data Compression Conference (DCC 2001), Snowbird, Utah, USA, Mar. 2001, p. 503.

[20] D. Brunello, G. Calvagno, M. Durigon, and R. Rinaldo, “Optimal Weighted 3D Prediction for Lossless Video Coding”, in Proc. Non-linear Signal and Image Processing - NSIP 2003, Grado, Italy, June 2003, pp. 89.93.

[21] S. Andriani, “Lossless Compression and Interpolation for High Quality Still Images and Video Sequences“, Laurea degree thesis, Dept.of Information Engineering, University of Padova, Italy, Décembre 2006.

[22] Flécher, E., Amir, S., Babel, M., and Déforges, O., “Lar video: Lossless video coding with semantic scalability,” In: International Conference on Signals and Electronic Systems (ICSES), September 2006, p. 227-230.

[23] Park, S. G., Delp, E. J., and Yu, H., “Adaptive Lossless Video Compression Using an Integer Wavelet Transform,” in Proc., IEEE International Conference on Image Processing, ICIP’04, Singapore, October 2004, Vol. 4, pp. 2251-2254.

[24] Sahng-Gyu Park.Adaptive Lossless video Compression. Ph.D. thesis, Purdue University. School of Electrical and Computer Engineering, December 2003.

[25] J. Jiang, J. Xia, and G. Xiao, "MPEG-2 based lossless video compression," IEE Proceedings Vision, Image and Signal Processing, vol. 153, no. 2, pp. 244–252, 2006.

[26] Wang, L. L., and Siu, W. C., "Improved lossless coding algorithm in H. 264/AVC based on hierarchical intra prediction and coding-mode selection," Journal of Electronic Imaging, 2011, 20 (4), p. 043001-043001.

[27] Song, L., Luo, Z., and Xiong, C., "Improving lossless intra coding of H. 264/AVC by pixel-wise spatial interleave prediction," IEEE Transactions on Circuits and Systems for Video Technology, 2011, 21 (12), 1924-1928.

[28] Chien, W. D., Liao, K. Y., and Yang, J. F., "H.264-based Hierarchical Lossless Coding System with New Intra Prediction Method," IEEE International Conference on Intelligent Computation and Bio-Medical Instrumentation (ICBMI), 2011, pp. 171-174.

[29] Cai, Q., Song, L., Li, G., and Ling, N., "Lossy and lossless intra coding performance evaluation: HEVC, H.264/AVC, JPEG 2000 and JPEG LS," Proceeding of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Hong Kong, December 2012, pp. 1–9.

[30] Choi, J. A., and Ho, Y. S., "Efficient residual data coding in CABAC for HEVC lossless video compression, " Signal, Image and Video Processing, 2015, vol. 9, no 5, p. 1055-1066.

[31] A. R. Calderbank. I. Daubechies. W. Sweldens, and B.L.Yeo. "Wavelet transforms that map integers to integers:' Applied and Computational Harmonic Analysis, vol. 5, no. 3. pp. 332-369, July 1998.

[32] S. Takamura and Y. Yashima, H.264-based Lossless Video Coding Using Adaptive Transforms, IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP'05). Vol. 2, pp. 301±304, Mar. 2005

[33] S. Takamura, "Lossless Video Coding," Lecture of the Course EE398B on Image Communication at Stanford University. Available: http://www.stanford.edu/class/ee398b/handouts/lectures/LosslessVideoC oding.pdf

[34] Y.-L. Lee and K.-H. Han, Complexity of the Proposed Lossless Intra, ISO/IEC JTC1/SC29/WG11 and ITU-T Q6/SG16, DocumentJVT-Q035r1, Oct. 2005.

[35] Y.-L. Lee, K.-H. Han, and S.-C. Lim, Lossless Intra Coding for Improved 4:4:4 Coding in H.264/MPEG-4 AVC, ISO/IECJTC1/SC29/WG11 and ITU-T Q6/SG16 Joint Video Team document JVT-P016, Jul. 2005.

[36] Y. L. Lee, K. H. Han, and G. J. Sullivan, "Improved lossless intra coding for H.264/MPEG-4 AVC," IEEE Trans. Image Process, Vol.15, pp.2610-2615, Sept.2006.

[37] S. Wei, S. Shen, B. Liu, and J. Yang, “Lossless image and video coding based on H.264/AVC intra predictions with simplified interpolations,” International Conference on Image Processing, Nov. 2009.

[38] G. Sullivan, P. Topiwala, and A. Luthra, The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions, Proc. SPIE, Aug. 2004.

[39] Marpe, D., Wiegand, T., & Gordon, S. (2005, September). H. 264/MPEG4-AVC fidelity range extensions: Tools, profiles, performance, and application areas. In Image Processing, 2005, ICIP 2005, IEEE International Conference on (Vol.1, pp.I-593).

Chapitre II:

[1] Ammous, D., Kammoun, F., and Masmoudi, N., “A Comparative Evaluation Between CABAC and CAVLC, ”Journal of Testing and Evaluation, Vol. 46, No. 3, 2018, pp. 1–11, http://dx.doi.org/10.1520/JTE20160323. ISSN 0090-3973

[2] Chaabouni, A., Gaudeau, Y., Lambert, J., Moureaux, J. M., and Gallet, P., “H.264 medical video compression for telemedicine: A performance analysis,” IRBM, Vol. 37, No 1, 2016, pp. 40-48, doi:10.1016/j.irbm.2015.09.007.

[3] Heo, J., and Ho, Y. S., “Adaptive Entropy Coder Design Based on the Statistics of Lossless Video Signal, ”Recent Advances on Video Coding, INTECH Open Access Publisher, Rijeka, Croatia, 2011, chapter 9, pp. 201–222.

[4] Kim, S. H., Heo, J., and Ho, Y. S., “Efficient entropy coding scheme for H.264/AVC lossless video coding,” Signal Process.: Image Commun, Vol. 25, No. 9, 2010, pp. 687–696, doi: 10.1016/j.image.2010.04.003.

[5] Li, L., Song, Y., Li, S., Ikenaga, T., and Goto, S., “A hardware architecture of cabac encoding and decoding with dynamic pipeline for h.264/avc,” Journal of Signal Processing Systems, 2008, 50(1), 81-95.

[6] Sze, V., and Budagavi, M., “A Comparison of CABAC Throughput for HEVC/H.265 vs. AVC/H.264,” In: IEEE Workshop on Signal Processing Systems, 2013, pp. 165-170.

[7] Heo, J., and Ho, Y. S., “Efficient differential pixel value coding in CABAC for H. 264/AVC lossless video compression,” Circuits, Systems, and Signal Processing, 2012, vol. 31, no 2, p. 813-825.

[8] Heo, J., Kim, S. H., and Ho, Y. S., “Improved CAVLC for H.264/AVC Lossless Intra coding,” IEEE Transactions on Circuits and Systems for Video Technology, 2010, 20(2), 213–222.

[9] Heo, J., Kim, S. H., and Ho, Y. S., "New CAVLC design for lossless intra coding," In 16th IEEE International Conference on Image Processing (ICIP), November 2009, pp. 637-640.

[10] Heo, J., Kim, S. H., and Ho, Y. S, "New CAVLC encoding algorithm for lossless intra coding in H.264/AVC," in Proc. IEEE Picture Coding Symposium. (PCS 2009), May 2009, pp. 67–71.

[11] Wahid, K., Dimitrov, V., and Jullien, G., "New Encoding of 8×8 DCT to make H. 264 Lossless," In IEEE Asia Pacific Conference on Circuits and Systems APCCAS 2006, December 2006, pp. 780-783.

[12] Takamura, S., and Yashima, Y., "H. 264-based Lossless Video Coding Using Adaptive Transforms," In: IEEE International Conference Acoustics, Speech and Signal processing (ICASSP '05), Philadelphia, USA, March 2005, vol. 2, pp. 301-304.

[13] Takamura, S., 2007, "Lossless Video Coding,"Lecture of the Course EE398B on Image Communication at Stanford University, https://web.archive.org/web/20170309143734/https://web.stanford.edu/class/ee398b/handouts/lectures/LosslessVideoCoding.pdf (Last accessed March 9, 2017).

[14]KAMISLI, Fatih. Lossless Image and Intra-frame Compression with Integer-to-Integer DST. IEEE Transactions on Circuits and Systems for Video Technology, 2017.

[15]KAMISLI, Fatih., "Lossless compression in HEVC with integer-to-integer transforms," In: Multimedia Signal Processing (MMSP), 2016 IEEE 18th International Workshop on. IEEE, 2016. p. 1-6.

[16] WEI, Shih-Tse, TIEN, Chia-Wei, LIU, Bin-Da, et al., "Adaptive truncation algorithm for Hadamard-transformed H. 264/AVC lossless video coding," IEEE Transactions on Circuits and Systems for Video Technology, 2011, vol. 21, no 5, p. 538-549.

[17] WEI, Shih-Tse, KUO, Pin-Chen, LIU, Bin-Da, et al. "Efficient residual coding algorithm based on Hadamard transform in lossless H. 264/AVC,"IET Image Processing, 2013, vol.8, no 4, p.191-198.

[18] M.Anto bennet et al. PERFORMANCE, ANALYSIS OF H.264/AVC LOSELESS VIDEO CODING USING HADAMARD TRANSFORM. International Journal of Computer Science & Engineering Technology (IJCSET), Mar 2013, Vol. 4, No. 03, p. 260-267, ISSN : 2229-3345.

[19] Lee, Y. L., Han, K. H., and Sullivan, G. J., "Improved lossless intra coding for H. 264/MPEG-4 AVC," IEEE Transactions on Image Processing, 2006, 15 (9), p.2610-2615.

[20] Lee, Y. L., Han, K. H., and Lim, S. C., "Lossless Intra Coding for Improved4:4:4 Coding in H.264/MPEG-4 AVC," ISO/IEC JTC1/SC29/WG11 and ITU-T Q6/SG16 Joint Video Team, document JVT-P016, Poznan, Poland, July 2005.

[21] Lee, Y. L., and Han, K. H., "Complexity of the proposed lossless intra for 4:4:4," ISO/IEC JTC1/SC29/WG11 and ITU-T Q6/SG16, Document JVT-Q035r1, October 2005.

[22] Marpe, D., Wiegand, T., and Gordon, S., "H. 264/MPEG4-AVC fidelity range extensions: Tools, profiles, performance, and application areas," In: IEEE International Conference on Image Processing (ICIP'05), September 2005, vol. 1, pp. 5936-5939.

[23] Ammous, D., Kammoun, F., and Masmoudi, N., "Improved Hierarchical lossless video coding," in: IEEE International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), December 2014, pp. 525-530.

[24] Wei, S. T., Shen, S. R., Liu, B. D., and Yang, J. F., "Lossless image and video coding based on H.264/AVC intra predictions with simplified interpolations," in Proc., IEEE International Conference on Image Processing, ICIP'2009, Cairo, Egypt, November 2009, pp. 633–636.

[25] Karsten, S., "H.264/AVC Software Coordination", Joint Model (JM) Ver. 18.3 Available: http://web.archive.org/web/20161128102709/http://iphome.hhi.de/suehring/tml/ (Accessed November 28, 2016).

[26] ISO/IEC JTC1, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec.H.264/ISO/IEC 14496-10 AVC, in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG), Document JVTG050, March 2003.

[27] Ohm, J. R., Sullivan, G. J., Schwarz, H., Tan, T. K., and 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, 2012, 22(12), 1669-1684.

[28] Bossen, F., JCT-VC-L1100: "Common test conditions and software reference configurations," In: Proceedings of the 12th JCT-VC Meeting, Geneva, 2013.

[29] Mongi BEN HAMADOU " Développement d'outils en programmation linéaire et calcul matriciel " thèse soutenue en 22 Décembre 1994 à l'université PAUL SABATIER DE TOULOUSE (SCIENCES).

[30] Chien, W. D., Liao, K. Y., and Yang, J. F., "H.264-based Hierarchical Lossless Coding System with New Intra Prediction Method," IEEE International Conference on Intelligent Computation and Bio-Medical Instrumentation (ICBMI), 2011, pp. 171-174.
[31] Cai, Q., Song, L., Li, G., and Ling, N. , "Lossy and lossless intra coding performance evaluation: HEVC, H.264/AVC, JPEG 2000 and JPEG LS," Proceeding of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Hong Kong, December 2012, pp. 1–9.
[32] Sze, V., Budagavi, M., and Sullivan, G. J., "High efficiency video coding (HEVC)," Integrated Circuit and Systems, Algorithms and Architectures, Springer, (2014), pp. 1-375.

Chapitre III

[1] Richardson, I., "H.264/AVC and MPEG4 video compression", Video coding for next generation multimedia, John Wiley and Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex , Angletaire, 2003.
[2] Sahin, E., and Hamzaoglu, I., "A high performance and low power hardware architecture for H. 264 CAVLC algorithm", In: *Signal Processing Conference, 2005 13th European*, IEEE, 2005, p.1-4.
[3] Karsten, S., "H.264/AVC Software Coordination", Joint Model (JM) Ver. 18.3 Available: http://web.archive.org/web/20161128102709/http://iphome.hhi.de/suehring/tml/
(Accessed November 28, 2016).
[4] Recommandation UIT-T H.264, Codage vidéo évolué pour les services audiovisuels génériques, May 2003.
[5] DING, Jun-Ren, CHEN, Jiun-Yu, YANG, Fu-Chun, et al. Two-layer and adaptive entropy coding algorithms for H. 264-based lossless image coding. IEEE International Conference In Acoustics, Speech and Signal Processing (ICASSP 2008), March 2008, pp. 1369-1372.
[6] CHIEN, Wei-Da, LIAO, Ke-Ying, et YANG, Jar-Ferr. H. 264-based hierarchical two-layer lossless video coding method. IET Signal Processing, 2014, vol. 8, no 1, p.21-29.

[7] WANG, Li-Li et SIU, Wan-Chi, Improved lossless coding algorithm in H. 264/AVC based on hierarchical intra prediction and coding-mode selection, Journal of electronic imaging, 2011, vol.20, no4.

[8] XU, Zhongwei, BARTRINA-RAPESTA, Joan, et SERRA SAGRISTÀ, Joan. Diagnostically lossless compression strategies for x-ray angiography images, 2015.

[9] KIM, Seung-Hwan, KANG, Je-Won, et KUO, C.-C. Jay, Improved H. 264/AVC lossless intra coding with two-layered residual coding (TRC), IEEE Transactions on Circuits and Systems for Video Technology, 2011, vol. 21, no 7, p. 1005-1010.

[10] Y.-L. Lee, K.-H. Han, and G. J. Sullivan, "Improved lossless intra coding for H.264/MPEG-4 AVC," IEEE Trans. Image Process., vol. 15, no. 9, pp. 2610–2615, Sep. 2006.

[11] Q. Zhang, S.-H. Kim, Y. Dai, and C.-C. J. Kuo, "A second-order-residual(SOR) coding approach to high-bit-rate video compression," in *Proc.Conf. Vis. Inform. Process. Commun. IS&T/SPIE Electron. Imag.*, vol.7543, p. 75430F, Jan. 2010.

[12] S.-H. Kim and Y.-S. Ho, "Fine granular scalable video coding using context-based binary arithmetic coding for bit-plane coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1301–1310, Oct. 2007.

[13] Joint Video Team. *Reference Software Version 16.1* [Online]. Available: http://iphome/hhi.de/shehring/tml/download/old−jm/jm16.1.zip

[14] A. Moffat, R. Neal, and I. Witten, "Arithmetic coding revisited", Data Compression Conference., DCC '95. Proceedings, pp. 202-211, March 1995.

[15] M. Schlockermann, "Film grain coding in H.264/AVC," JVTI034d2.doc, San Diego, September, 2003.

[16] ZHANG, Qi, DAI, Yunyang, et KUO, C.-C. Jay. Lossless video compression with residual image prediction and coding (RIPC). In : Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on. IEEE, 2009. p. 617-620.

[17] Y. Dai, Q. Zhang, A. Tourapis, and C. –C. J. Kuo, "Efficient block based intra prediction for image coding with 2D geometrical manipulations", IEEE International Conference on Image Processing, San Diego, California, October 2008.

[18] Chien, W. D., Liao, K. Y., et Yang, J. F., "H.264-based hierarchical two-layer lossless video coding method," IET Signal Processing 2014, vol. 8, no 1, p. 21-29.

[19] Heindel, A., Wige, E., and Kaup, A., "Sample-based Weighted Prediction for Lossless Enhancement Layer Coding in HEVC, "Grand Challenge at Picture Coding Symposium (PCS), San José, CA, USA, December 2013.

[20] Heindel, A., Wige, E., et Kaup, A., "Sample-based Weighted Prediction for lossless enhancement layer coding in SHVC," IEEE International Conference on Image Processing (ICIP), October 2014, pp. 3656-3660.

[21] Wang, L. L., et Siu, W. C., “Improved lossless coding algorithm in H. 264/AVC based on hierarchical intra prediction and coding-mode selection, ”Journal of electronic imaging, 2011, vol. 20, no 4, p. 043001-043001-10.

[22] Heindel, A., Wige, E., and Kaup, A., “Low-Complexity Enhancement Layer Compression for Scalable Lossless Video Coding based on HEVC,” IEEE Transactions on Circuits and Systems for Video Technology, (2016).

[23] OU, Xianfeng, ZHANG, Guoyun, LONGYUAN, Guo, et al. Improved Adaptive Transform for Residue in H. 264/AVC Lossless Video Coding. automatika, 2016, vol. 57, no 4, p. 1045-1055.

[24] WEI, Shih-Tse, TIEN, Chia-Wei, LIU, Bin-Da, et al., Adaptive truncation algorithm for Hadamard-transformed H.264/AVC lossless video coding, IEEE Transactions on Circuits and Systems for Video Technology, 2011, vol.21, no5, p.538-549.

[25] Huffman D.A., “A method for the construction of minimum redundancy codes”, Proceedings of the Institute of Radio Engineers, 40 (9), Sep. 1952, pp. 1098–1101.

[26] Wei, S. T., Shen, S. R., Liu, B. D., and Yang, J. F., “Lossless image and video coding based on H.264/AVC intra predictions with simplified interpolations,” in Proc., IEEE International Conference on Image Processing, ICIP’2009, Cairo, Egypt, November 2009, pp. 633–636.

[27] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, High Efficiency Video Coding (HEVC) Text Specification Draft 8, JCT-VC document, JCTVC-J1003, Stockholm, Sweden, Jul. 2012.

[28] Y.-L. Lee, K.-H. Han, and G. J. Sullivan, “Improved lossless intra coding for H.264/MPEG-4 AVC,” IEEE Trans. Image Process., vol. 15, no. 9, pp. 2610–2615, Sep. 2006.

[29] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4) AVC: Advanced Video Coding for Generic Audiovisual Services.

[30] M. Zhou, AHG19: Method of Frame-Based Lossless Coding Mode for HEVC, JCT-VC document, JCTVC-H0083, San Jose, CA, Feb. 2012.

[31] W. Gao, M. Jiang, H. Yu, and M. Zhou, AHG19: A Lossless Coding Solution for HEVC, JCT-VC document, JCTVC-H0530, San Jose, CA, Feb. 2012.

[32] M. Zhou, AHG22: Sample-Based Angular Prediction (SAP) for HEVC Lossless Coding, JCT-VC document, JCTVC-G093, Geneva, Nov. 2011.

[33] F. Bossen, Common Test Conditions and Software Reference Configurations, JCT-VC document, JCTVC-G1100, San Jose, CA, Feb. 2012.

[34] M. Zhou, W. Gao, M. Jiang, H. Yu, HEVC lossless coding and improvements, Circuits and Systems for Video Technology, IEEE Transactions on 22 (12) (2012) 1839–1843. doi:10.1109/TCSVT.2012.2221524.

[35] S.-W. Hong, J. H. Kwak, Y.-L. Lee, Cross residual transform for lossless intra-coding for HEVC, Signal Processing: Image Communication 28 (10) (2013) 1335 – 1341.

[36] J.-H. Kwak, Y.-L. Lee, Secondary residual transform for lossless intra coding in HEVC, Journal of Broadcast Engineering 17 (5) (2012) 734–741.

[37] G. Jeon, K. Kim, J. Jeong, Improved residual DPCM for HEVC lossless coding, in: Graphics, Patterns and Images (SIBGRAPI), 2014 27th SIBGRAPI Conference on, 2014, pp. 95–102. doi:10.1109/SIBGRAPI. 2014.31.

[38] X. Cai, J. S. Lim, Adaptive residual DPCM for lossless intra coding, in: IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics, 2015, pp. 94100A–94100A.

[39] K. Kim, G. Jeon, J. Jeong, Piecewise DC prediction in HEVC, Signal Processing: Image Communication 29 (9) (2014) 945 – 950. doi:http: //dx.doi.org/10.1016/j.image.2014.07.002.

[40] E. Wige, G. Yammine, P. Amon, A. Hutter, A. Kaup, Pixel-based averaging predictor for HEVC lossless coding, in: Image Processing (ICIP), 2013 20th IEEE International Conference on, 2013, pp. 1806–1810. doi:10.1109/ICIP.2013.6738372.

[41] X.-P. Xia, E.-H. Liu, J.-J. Qin, Improved SAP based on adaptive directional prediction for fHEVCg lossless intra prediction, Journal of Visual Communication and Image Representation 33 (2015) 78 – 84. doi:http://dx.doi.org/10.1016/j.jvcir.2015.09.003.

[42] ALVAR, Saeed Ranjbar et KAMISLI, Fatih. On lossless intra coding in HEVC with 3-tap filters. Signal Processing: Image Communication, 2016, vol. 47, p. 252-262.

[43] S.-H. Kim, J. Heo, Y.-S. Ho, Efficient entropy coding scheme forH.264/AVC lossless video coding, Signal Processing: Image Communication 25 (9) (2010) 687–696.

[44] J.-A. Choi, Y.-S. Ho, Differential pixel value coding for HEVC lossless compression, INTECH Open Access Publisher, 2013.

[45] W. Gao, M. Jiang, Y. He, J. Song, H. Yu, A lossless coding solution in HEVC, JCTVC-G664, Geneva, Switzerland (2011) 21–30.

[46] S. Kim, A. Segall, Simplified CABAC for lossless compression, JCTVCH0499, San Jos´e, CA, USA (2012) 1–10.

[47] Y. Piao, J. Min, S. Lee, AHG7: Coefficient coding for lossless coding,JCTVC-L0114, Geneva, Switzerland (2013) 14–23.

[48] J.-A. Choi, Y.-S. Ho, Efficient residual data coding in CABAC for HEVC lossless video compression, Signal, Image and Video Processing 9 (5) (2015) 1055–1066. doi:10.1007/s11760-013-0545-z.

[49] X. Cai and Q. Gu, "Improved HEVC lossless compression using two-stage coding with sub-frame level optimal quantization values," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 5651–5655.

[50] HEINDEL, Andreas, WIGE, Eugen, et KAUP, André. Low-Complexity Enhancement Layer Compression for Scalable Lossless Video Coding Based on HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017, vol. 27, no 8, p. 1749-1760.

[51] Heindel, A., Wige, E., and Kaup, A., "Sample-based weighted prediction for lossless enhancement layer coding in HEVC." Grand Compression Challenge at Picture Coding Symposium (PCS) (Dec. 2013).

[52] HEINDEL, Andreas, WIGE, Eugen, et KAUP, André. Analysis of prediction algorithms for residual compression in a lossy to lossless scalable video coding system based on HEVC. In : *Applications of Digital Image Processing XXXVII*. International Society for Optics and Photonics, 2014. p. 92170M.

[53] A. Heindel, E. Wige, and A. Kaup, "Sample-based weighted prediction for lossless enhancement layer coding in SHVC," in *Proc. IEEE Int.Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 3656–3660.

[54] P. Amon, A. Hutter, E. Wige, and A. Kaup, "Intra prediction for lossless coding," document JCTVC-L0161, ITU-T VCEG and ISO/IEC MPEG (JCT-VC), Geneva, Switzerland, Jan. 2013.

[55] ——, "RCE2: Sample-based weighted intra prediction for lossless coding," document JCTVC-M0052, ITU-T VCEG and ISO/IEC MPEG (JCT-VC), Incheon, Republic of Korea, Apr. 2013.

[56] E. Wige, G. Yammine, P. Amon, A. Hutter, and A. Kaup, "Pixel based averaging predictor for HEVC lossless coding," in Proc. Of IEEE International Conference on Image Processing (ICIP), Melbourne, Australia, Sep. 2013.

[57] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 620–636, Jul.2003.

Annexes

Annexes 1

Tableau A. 1. *Une partie de l'algorithme 1*

```
/**********************calcul de la matrice
Ac****************/

if(i==0)

{

if((j>=2)&&(j<6))

                        {

           if(comp<4)

                                                {

                             for(k=0;k<36;k++)

                                                        {


A_tot[comp][k]=Ai[k];

                                                        }


                                                }

                                    comp++;

                        }


elseif((j==6))

                        {

                             printf(" hello %d %d %d %d
",A_tot[0][0],A_tot[1][0],A_tot[2][0],A_tot[3][0]);

                         printf("\n");

           for(kk=0;kk<36;kk++)
```

```
                              {

Ac[kk]=0.4096*A_tot[0][kk]+0.512*A_tot[1][kk]+0.64*A_tot[2][kk
]+0.8*A_tot[3][kk];

                              }


for(d=0;d<6;d++)

                              {

for(e=0;e<6;e++)

printf("%.8lf\t",Ac[d*6+e]);

printf("\n");

                              }

                                   /*fprintf(mat,"i=%d j=%d
\n",i,j);

                for(r=0;r<6;r++)

                     {

                          for(s=0;s<6;s++)

                           fprintf(mat,"%d ",Ac[r*6+s]);

                       fprintf(mat,"\n");

                     }




                fprintf(mat,"\n");*/



           if(comp==4)

                                                    {
```

```
                for(k=0;k<36;k++)
                                                {
                                            A_tot[0][k]=A_tot[1][k];
                                             A_tot[1][k]=A_tot[2][k];
                                             A_tot[2][k]=A_tot[3][k];
                                            A_tot[3][k]=Ai[k];
                                                }


                                                }









                    } /*fin de if((j==6))*/


elseif ((j>6))
                    {
                         printf(" hello %d %d %d %d
",A_tot[0][0],A_tot[1][0],A_tot[2][0],A_tot[3][0]);
                     printf("\n");
          for(kk=0;kk<36;kk++)
                         {

Ac[kk]=0.4096*A_tot[0][kk]+0.512*A_tot[1][kk]+0.64*A_tot[2][kk
]+0.8*A_tot[3][kk];
                         }
```

```
                        for(d=0;d<6;d++)
                        {
for(e=0;e<6;e++)
printf("%.8lf\t",Ac[d*6+e]);
printf("\n");
                        }
//printf("%d",comp);
//printf("\n");
/*   fprintf(mat,"i=%d j=%d \n",i,j);
              for(r=0;r<6;r++)
                   {
                        for(s=0;s<6;s++)
                         fprintf(mat,"%d ",Ac[r*6+s]);
                    fprintf(mat,"\n");
                   }



              fprintf(mat,"\n");*/


         if(comp==4)
                                                  {

              for(k=0;k<36;k++)
                                                  {
                    A_tot[0][k]=A_tot[1][k];
```

```
A_tot[1][k]=A_tot[2][k];

A_tot[2][k]=A_tot[3][k];
                                        A_tot[3][k]=Ai[k];
                                                    }


                                                    }
                    } /*fin de if((j>6))*/

} /*fin de if(i==0)*/
elseif(i!=0)
{
if(j==0)
                                    { comp=0;}

                                                if(comp<4)
                                                    {
                            for(k=0;k<36;k++)
                                                    {

A_tot[comp][k]=Ai[k];
                                                    }
                                                comp++;
                                                    }
if(j>=4)
                                    { printf(" hello %d %d %d %d
",A_tot[0][0],A_tot[1][0],A_tot[2][0],A_tot[3][0]);
```

```
                                printf("\n");

          for(kk=0;kk<36;kk++)

                                        {


Ac[kk]=0.4096*A_tot[0][kk]+0.512*A_tot[1][kk]+0.64*A_tot[2][kk
]+0.8*A_tot[3][kk];

                                        }


                                for(d=0;d<6;d++)

                                        {

for(e=0;e<6;e++)

printf("%.8lf\t",Ac[d*6+e]);

printf("\n");

                                        }


     /*fprintf(mat,"i=%d j=%d \n",i,j);

                for(r=0;r<6;r++)

                     {

                          for(s=0;s<6;s++)

                           fprintf(mat,"%d ",Ac[r*6+s]);

                       fprintf(mat,"\n");

                     }




                fprintf(mat,"\n");*/


          if(comp==4)
```

```
                                                    {


                for(k=0;k<36;k++)

                                                    {

                        A_tot[0][k]=A_tot[1][k];

A_tot[1][k]=A_tot[2][k];

A_tot[2][k]=A_tot[3][k];

A_tot[3][k]=Ai[k];

                                                    }


                                                    }

                                  } /*fin de if((j>=4))*/


} /*fin de if(i!=0)*/




/***********fin de calcul de Ac*********************/
```

Annexes 2

Tableau A. 2. *Une partie de l'algorithme 2*

```
int MethodeGauss(double NewMat[t][2*t])

{

int inversible = 1;

int k,i,pos,colonne,colonnebis;

double var,var1;

    k=0;

while((inversible == 1)&&(k<t))

    {

if (NewMat[k][k] != 0)

{

                var = NewMat[k][k];

for (colonne=0;colonne<2*t;colonne++)

                {

                    NewMat[k][colonne] =
NewMat[k][colonne]/var;  //Normalisation de la ligne contenant
l'élément diagonal

                    }

for (i=0;i<t;i++)

                    {

if (i != k)

                        {

                            var1=NewMat[i][k];

for (colonnebis=0;colonnebis<2*t;colonnebis++)
```

```
                                {
                                    NewMat[i][colonnebis] =
NewMat[i][colonnebis] - NewMat[k][colonnebis]*var1;
                                }
                            }
                        }



            }
else// NewMat[k][k]=0
            {
                    if(k==(t-1))
                    {
                        inversible=0;
                    }
                    else
                    {

                        /********** permutation de deux
lignes ***************/
                        pos=0;
                        for (i=1;((i<t)&&(pos==0));i++)
                        {
                            if(i>k)
                              {
                                if (NewMat[i][k]!= 0 )
                                {
                                    pos=i;
```

```
                                                    for
(colonne=0;colonne<2*t;colonne++)

                                                    {

                                                     MH[colonne]=
NewMat[k][colonne];

NewMat[k][colonne]=NewMat[pos][colonne] ;//on peut ecrit pos=i

NewMat[pos][colonne]=MH[colonne];

                                                    }

                                              }

                                         }

                               }

     /*************************************************************
**/


                        /*traitement normale de gauss jordan*/

                             var = NewMat[k][k];

                             for (colonne=0;colonne<2*t;colonne++)

                             {

                                  NewMat[k][colonne] =
NewMat[k][colonne]/var;  //Normalisation de la ligne contenant
l'élément diagonal

                             }

                                  for (i=0;i<t;i++)

                                  {

                                       if (i != k)

                                       {

                                            var1=NewMat[i][k];
```

```
                                                for
(colonnebis=0;colonnebis<2*t;colonnebis++)

                                                {


     NewMat[i][colonnebis] = NewMat[i][colonnebis] -
NewMat[k][colonnebis]*var1;

                                                }

                                            }

                                        }


     /************fin de gauss jordan******************/


                            }

                        }

                         k++;


    } // fin de while

return inversible;

}
```

Annexes3

Tableau A. 3. *Codes de Tot_Coeff et T1s*

Tls	Tot_Coeff	$0 \leq n_C < 2$	$2 \leq n_C < 4$	$4 \leq n_C < 8$	$n_C \geq 8$
0	0	1	11	1111	000011
0	1	000101	001011	001111	000000
1	1	01	10	1110	000001
0	2	00000111	000111	001011	000100
1	2	000100	00111	01111	000101
2	2	001	011	1101	000110
0	3	000000111	0000111	001000	001000
1	3	00000110	001010	01100	001001
2	3	0000101	001001	01110	001010
3	3	00011	0101	1100	001011
0	4	0000000111	00000111	0001111	001100
1	4	000000110	000110	01010	001101
2	4	00000101	000101	01011	001110
3	4	000011	0100	1011	001111
0	5	00000000111	00000100	0001011	010000
1	5	0000000110	0000110	01000	010001
2	5	000000101	0000101	01001	010010
3	5	0000100	00110	1010	010011
0	6	0000000001111	000000111	0001001	010100
1	6	00000000110	00000110	001110	010101
2	6	0000000101	00000101	001101	010110
3	6	00000100	001000	1001	010111
0	7	0000000001011	00000001111	0001000	011000
1	7	0000000001110	000000110	001010	011001
2	7	00000000101	000000101	001001	011010
3	7	000000100	000100	1000	011011
0	8	0000000001000	00000001011	00001111	011100
1	8	0000000001010	00000001110	0001110	011101
2	8	0000000001101	00000001101	0001101	011110
3	8	0000000100	0000100	01101	011111
0	9	00000000001111	000000001111	00001011	100000
1	9	00000000001110	00000001010	00001110	100001
2	9	0000000001001	00000001001	0001010	100010
3	9	00000000100	000000100	00110	10011
0	10	00000000001011	000000001011	000001111	100100
1	10	000000001010	000000001110	00001010	100101
2	10	00000000001101	000000001101	00001101	10010
3	10	0000000001100	00000001100	0001100	100111
0	11	00000000000111	000000001000	000001011	101000
1	11	000000000001110	000000001010	000001110	101001

2	11	00000000001001	000000001001	00001001	101010
3	11	00000000001100	00000001000	00001100	101011
0	12	000000000001011	0000000001111	000001000	101100
1	12	000000000001010	0000000001110	000001010	101101
2	12	000000000001101	0000000001101	000001101	101110
3	12	0000000000100	000000001100	00001000	101111
0	13	0000000000001111	0000000001011	0000001101	110000
1	13	000000000000001	0000000001010	000000111	110001
2	13	000000000001001	0000000001001	000001001	110010
3	13	000000000001100	0000000001100	000001100	110011
0	14	0000000000001011	0000000000111	0000001001	110100
1	14	0000000000001110	00000000001011	0000001100	110101
2	14	0000000000001101	0000000000110	0000001011	110110
3	14	000000000001000	0000000001000	0000001010	110111
0	15	0000000000000111	00000000001001	0000000101	111000
1	15	0000000000001010	00000000001000	0000001000	111001
2	15	0000000000001100	0000000000001	0000000110	111011
3	15	0000000000001100	0000000000001	0000000110	111011
0	16	0000000000000100	00000000000111	0000000001	111100
1	16	0000000000000110	00000000000110	0000000100	111101
2	16	0000000000000101	00000000000101	0000000011	111110
3	16	0000000000001000	00000000000100	0000000010	111111

Annexes 4

Tableau A. 4. *Codes de NZ Coeff selon les tables VLC*

VLC 0		VLC 1		VLC 2		VLC 3		VLC 4		VLC 5		VLC 6	
Code	NZ Coeff	Code	NZ Coeff	Code	NZ Coeff	Code	NZ Coeff	Code	NZ Coeff	Code	NZ Coeff	Code	NZ Coeff
1	1	1x	±1	1xx	±1 à ±2	1xxx	±1 à ±4	1xxxx	±1 à ±8	1xxxxx	±1 à ±16	1xxxxxx	±1 à ±32
01	-1	01x	±2	01xx	±3 à ±4	01xxx	±5 à ±8	01xxxx	±9 à ±16	01xxxxx	±17 à ±32	01xxxxxx	±33 à ±64
001	2	001x	±3	001xx	±5 à ±6	001xxx	±9 à ±12	001xxxx	±17 à ±24	001xxxxx	±33 à ±64	…	…
0001	-2	0001x	±4	0001xx	±7 à ±8	0001xxx	±13 à ±15	0001xxxx	±25 à ±32	…	…	…	…
00001	3	00001x	±5	00001xx	±9 à ±10	00001xxx	±17 à ±19	…	…	…	…	…	…
000001	-3	000001x	±6	000001xx	±11 à ±12	…	…	…	…	…	…	…	…
0000001	4	0000001x	±7	…	…	…	…	…	…	…	…	…	…
00000001	-4	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…

Annexes 5

Tableau A. 5. *Codes de Tot_Zeros en fonction de Tot_Coeff pour les blocs 4x4*

Tot_Zeros	Tot_Coeff														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	111	0101	00011	0101	000001	000001	000001	000001	00001	0000	0000	000	00	0
1	011	110	111	111	0100	00001	00001	0001	000000	00000	0001	0001	001	01	1
2	010	101	110	0101	0011	111	101	00001	0001	001	001	01	1	1	
3	0011	100	101	0100	111	110	100	011	11	11	010	1	01		
4	0010	011	0100	110	110	101	011	11	10	10	1	001			
5	00011	0101	0011	101	101	100	11	10	001	01	011				
6	00010	0100	100	100	100	011	010	010	01	0001					
7	000011	0011	011	0011	011	010	0001	001	00001						
8	000010	0010	0010	011	0010	0001	001	000000							
9	0000011	00011	00011	0010	00001	001	000000								
10	0000010	00010	00010	00010	0001	000000									
11	00000011	000011	000001	00001	00000										
12	00000010	000010	00001	00000											
13	000000011	000001	000000												
14	000000010	000000													
15	000000001														

Annexes 6

Tableau A. 6. *Codes des Run_Before*

Run_Before	Zeros_Left						
	1	2	3	4	5	6	>6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8	-	-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	0000000001
14	-	-	-	-	-	-	00000000001

Annexes 7

Tableau A. 7. *Une partie de l'algorithme 3*

```
////////////////////////////image source/////////////////////////////////

for(i = 0 ; i < 304128 ; i++)
      {
      tab[i]= * (external_buffer_src + i);
      fprintf(Test,"%d ",tab[i]);
      ///printf("%d ",tab[i]);
/*if (i==1759)
      {fprintf(Test, "verficatio= %d ",i);}*///verification
      }

////////////////////////////image source/////////////////////////////////
//////////////////////image transformée quantifié inverse /////////////////
for(i=0;i<p_Vid->height;i++)
  {
for (j=0; j<p_Vid->width; j++)
            {
                  m[(i*(p_Vid->width)+j)]=p_Vid->enc_picture->imgY[i][j] ;
}}
for(i=0;i<p_Vid->height_cr;i++)
  {
for (j=0; j<p_Vid->width_cr; j++)
            {
                  m[101376+(i*(p_Vid->width)+j)]=p_Vid->enc_picture-
>imgUV[0][i][j];    }}
for(i=0;i<p_Vid->height_cr;i++)
  {
for (j=0; j<p_Vid->width_cr; j++)
            {
                  m[(2*101376)+(i*(p_Vid->width)+j)]=p_Vid->enc_picture-
>imgUV[1][i][j];    }}
for(i=0;i<304128;i++)
{fprintf(essai2,"%d ",m[i]);
/*if (i==1759)
      {fprintf(essai2, "verficatio= %d ",i);}*///verification
}
fclose(essai2);
//////////////////////image transformée quantifié inverse /////////////////
////////////////////////////image difference non ordonnée/////////////////
fp_residu  = fopen(im_residu,  "a+");//amina
for(ir = 0 ; ir < 304128 ; ir++)
{
residu_vect[ir] = tab[ir] - m[ir];
      fprintf(fp_residu, "%d
",residu_vect[ir]);//residu_vect[ir]:coefficient non ordonnée
      /*if (ir==1759)
      {fprintf(fp_residu, "verficatio= %d ",ir);}*///verification

}
fclose(fp_residu);

////////////////////////////image difference non ordonnée/////////////////
```

Annexes 8

Tableau A. 8. *Une partie de l'algorithme 4*

```
for (x=0; x<MBX; x++)
      {
            comp = 4*x ;//x designe le numéro du bloc courant sur la ligne
de l'image qui contient 88 bloc càd bloc 0,bloc 1,bloc
2.................bloc 87
             jr = 4*4*88*y; //y designe le numéro du bloc courant sur la
colonne de l'image qui contient 72 bloc càd bloc 0,bloc 1,bloc
2.................bloc 71
      /// horital one
            for (i=0; i<4; i++)
            {

            residu_ord[j] = residu_vect[i + comp +jr];

            fprintf(fp_residu_ord, "%d",residu_ord[j]);
               j++;

            }

      //horizontal bas

            for (i=0; i<4; i++)
            {
            residu_ord[j] = residu_vect[352*3+i + comp +jr];
            fprintf(fp_residu_ord, "%d",residu_ord[j]);
               j++;

            }

            //gauche

            for (i=1; i<3; i++)
            {
            residu_ord[j] = residu_vect[352*i + comp + jr ];
            fprintf(fp_residu_ord, "%d",residu_ord[j]);
               j++;

            }

            // droite
      for (i=1; i<3; i++)
            {
            residu_ord[j] = residu_vect[352*i + 3 + comp +jr  ];
            fprintf(fp_residu_ord, "%d",residu_ord[j]);
               j++;

            }

//    fprintf(fp_residu_ord, " fin MBX %d \n ", x);
```

Annexes 9

Tableau A. 9. *Une partie de l'algorithme 5*

```
dec_val1= (unsignedchar*)malloc(sizeof(unsignedchar) * (size1));

bin_val1= (unsignedchar*)malloc(sizeof(unsignedchar) * (size2));

fichier1 = fopen("partie2.txt","r");

                fichier2 = fopen("correction_2.txt","a+");

                fichier3 = fopen("verification2.txt","a+");

                getchar();//donia

        do

        {

                for(jj=0;jj<304128;jj++)

                                {

                 fread(&dec_val1[jj],1,(size*1.45),fichier1);

                                                if(dec_val1[jj]==49)

                                                   {    printf ("valeur=%d", dec_val1[jj]);

fprintf (fichier3, "valeur=%d", dec_val1[jj]);

                                                                for (i = 0; i < 2; ++i)

                                {

                                                        bin_val1[k] = arr1[i];

                                                            //printf("bin_val1[%d]=%d,

arr1[%d]=%d", k, bin_val1[k], i, arr1[i]);

/*printf("%d", bin_val1[k]);*///essai1

fprintf(fichier2,"%d", bin_val1[k]);//donia

                                                //  fprintf(fichier2,"bin_val1[%d]=%d", k,

bin_val1[k]);//donia

                                k++;

                                                        }
```

```
//getchar();//donia
                                        }
```

République Tunisienne
Ministère de l'Enseignement Supérieur, de
la Recherche Scientifique
et de la Technologie

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Ecole Doctorale
Sciences et Technologies

Thèse de *DOCTORAT*
Génie électrique

N° d'ordre: 2017–53

Étude comparative des codeurs CABAC et CAVLC et contribution à l'implémentation d'une technique de compression vidéo sans perte

Donia AMMOUS DAMAK

الخلاصة: تتضمن هذه الأطروحة جزئين رئيسيين.الأول هو دراسة مقارنة بين إثنين من التشفير كاباك و السي الفي أل سي والثاني هو المساهمة في تطوير تقنية ضغط الفيديو بدون خسارة إستنادا إلى تعديل التشفير264 / إي في سي

Résumé : **Cette thèse comporte principalement deux parties. La première est une étude comparative des deux codeurs entropiques "Context adaptive binary arithmetic coding" (CABAC) et "Context adaptive variable length coding" (CAVLC). La seconde est une contribution à la mise au point d'une technique de compression vidéo sans perte basée sur une modification de la norme H.264 "Advanced Video Coding" (AVC).**

Abstract : This thesis consists mainly of two parts. The first is a comparative study of the two entropy coders "Context adaptive binary arithmetic coding" (CABAC) and "Context adaptive variable length coding" (CAVLC). The second is a contribution to the development of a lossless video compression technique based on a modification of the standard H.264\AVC.

المفاتيح:264 / إي في سي، التشفير ، كاباك ، السي الفي أل سي ، ضغط الفيديو بدون خسارة , ضغط الفيديو بخسارة

Mots clés : H.264/AVC, codage entropique, CABAC, CAVLC, Compression vidéo sans perte, Compression vidéo avec perte.

Key-words : H.264/AVC, Entropy coding, CABAC, CAVLC, Lossless compression of video, Lossy compression of video.