ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION
M.Ing.

PAR
Louis POUDRIER-RACETTE

CODAGE VIDÉO BASSE COMPLEXITÉ SANS PERTE DE QUALITÉ VISUELLE SUPPORTANT LES RÉSOLUTIONS HD SUR LE DSP ASYNCHRONE OPUS D'OCTASIC

MONTRÉAL, LE 18 FÉVRIER 2014

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Stéphane Coulombe, directeur de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Claude Thibeault, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Chamseddine Talhi, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 23 JANVIER 2014

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Ce mémoire est le fruit de quatre ans d'études et de recherche réalisées en parallèle avec un emploi à temps plein chez Octasic. Je tiens à remercier les gens et l'entreprise qui l'ont rendu possible.

Premièrement, je remercie Stéphane Coulombe, professeur au département de génie logiciel et des TI à l'École de technologie supérieure, pour toute son assistance lors de la recherche et de la rédaction de ce mémoire.

Ensuite, je remercie la compagnie Octasic, pour laquelle l'implémentation de cette recherche a été faite et qui m'a soutenu dans ce projet.

Je veux également remercier les membres du jury, Claude Thibeault et Chamseddine Talhi, qui ont accepté de consacrer leur temps précieux à évaluer ce mémoire.

Enfin, je désire remercier ma femme Audrey Morin pour son soutien et sa patience inépuisable.

CODAGE VIDÉO BASSE COMPLEXITÉ SANS PERTE DE QUALITÉ VISUELLE SUPPORTANT LES RÉSOLUTIONS HD SUR LE DSP ASYNCHRONE OPUS D'OCTASIC

Louis POUDRIER-RACETTE

RÉSUMÉ

Ce mémoire de recherche présente une nouvelle méthode de codage vidéo permettant l'optimisation des transferts de données vidéo à l'intérieur du système *Vocallo MGW* d'Octasic. Cette méthode est présentée comme une solution aux problèmes posés par l'échange de quantités importantes de données générées dans un système dans lequel un contenu vidéo haute-définition se trouve à l'état non compressé. Plus spécifiquement, ce travail se penche sur le problème du transfert de données vidéo sur un lien Gigabit Ethernet entre les cœurs d'une puce d'Octasic.

Le *Vocallo MGW* est une application du processeur de traitement de signal asynchrone produit par Octasic. Cette technologie impose son lot de contraintes, tant au niveau des instructions à privilégier que de la gestion de la mémoire. La méthode de compression développée dans ce projet de recherche vise à satisfaire trois contraintes. La première est le taux de compression requis pour permettre une communication bidirectionnelle sur un lien Gigabit Ethernet. La seconde est le maintien de la qualité visuelle. La troisième est le temps d'exécution puisque la solution développée doit opérer en temps réel sur le système de traitement *Vocallo MGW* d'Octasic.

La méthode de compression proposée est basée sur l'algorithme HACP-SBT tiré de la littérature mais contient plusieurs améliorations significatives pour rendre ce dernier fonctionnel dans le contexte du *Vocallo MGW*. Les contributions de ce travail incluent la réduction de la complexité de l'algorithme de prédiction, la définition d'une syntaxe et d'une technique flexible pour la mise en paquet ainsi que la création de trois modes de codage visant à respecter les contraintes fixées. Ces trois modes sont constitués d'un mode *sans perte* supplémenté de deux modes avec perte : le mode *avec perte calculée* favorisant la qualité visuelle et le mode *avec perte rapide* favorisant le temps d'exécution.

L'analyse de ces trois modes de codage montre qu'ils permettent tous de respecter la contrainte du taux de compression et de la qualité visuelle. Cependant, seulement le traitement de petites résolutions, dont le QCIF, est assez rapide pour se conformer à la contrainte du temps réel.

Mot-clés : compression vidéo, basse complexité, sans perte de qualité visuelle, compression avec perte

LOW COMPLEXITY VIDEO CODING FOR VISUALLY LOSSLESS COMPRESSION SUPPORTING HD RESOLUTIONS ON OCTASIC'S OPUS ASYNCHRONOUS DSP

Louis POUDRIER-RACETTE

ABSTRACT

This research thesis presents a new method for video coding, which optimizes the transfer of video data within the Octasic *Vocallo MGW* system. This method is presented as a solution to the problems posed by the exchange of large amounts of data generated by a system in which a high-definition video content is present in an uncompressed form. More specifically, this work addresses the problem of transferring video data over a Gigabit Ethernet link between the cores of an Octasic silicon chip.

The *Vocallo MGW* application runs on Octasic's asynchronous DSP. This technology imposes several constraints related to the selection of instructions and memory management. The compression method developed in this research project aims to satisfy three constraints. The first one is the compression ratio required to enable bidirectional communication over a Gigabit Ethernet link. The second one is to preserve the visual quality. The third one is the execution time since the solution must operate in real-time on Octasic's *Vocallo MGW* processing system.

The proposed compression method is based on the HACP-SBT algorithm available from the literature but contains several key improvements to make it functional in the context of the *Vocallo MGW*. This work's contributions include the simplification of the prediction algorithm, the adaptation of the coding syntax to allow packetisation and the definition of a flexible packet header, and the creation of three coding modes to satisfy the constraints. These three modes include a *lossless* coding mode supplemented by two lossy compression modes : the *calculated lossy* mode promoting visual quality and the *rapid lossy* mode favoring lower execution times.

The analysis of these three coding modes shows that all the modes meet the constraint of compression ratio and visual quality. However, only the processing of small resolutions, such as QCIF, is fast enough to meet the real-time constraint.

Keywords: video compression, low complexity, visually lossless, lossy compression

TABLE DES MATIÈRES

Page

INTRODUCTION 1

CHAPITRE 1 CONCEPTS DE BASE 3
1.1 Compression de l'information 3
1.1.1 Théorie de l'information 3
1.1.2 Types de compression 4
1.2 Outil de compression d'images et de vidéos 4
1.2.1 Prédiction 4
1.2.1.1 Prédiction spatiale 5
1.2.1.2 Prédiction temporelle 6
1.2.2 Transformation 7
1.2.3 Codage entropique 8
1.2.4 Quantification 8
1.3 Résumé des outils de codage 8
1.3.1 Outils de compression sans perte d'information 8
1.3.2 Outils de compression avec perte d'information 9

CHAPITRE 2 SYSTÈME - CONTEXTE D'IMPLÉMENTATION 11
2.1 Contraintes de l'unité de calcul 11
2.2 Contraintes de mémoire 12
2.3 Résumé des contraintes 14

CHAPITRE 3 ÉTAT DE L'ART 17
3.1 Compression sans perte d'information 17
3.1.1 La prédiction par différence de pixel adjacent (DAP) 18
3.1.2 Prédiction par détection de contour médian (MED) 19
3.1.3 Prédiction ajustée en fonction du gradient (GAP) 20
3.1.4 Prédiction hiérarchique (HACP) 20
3.1.5 Évaluation des prédicteurs 21
3.1.5.1 Opérations par pixel 21
3.1.5.2 Profondeur de prédiction 22
3.1.5.3 Ratio entropique 22
3.1.5.4 Résultats 23
3.2 Codage avec perte d'information sans-perte de qualité visuelle 23
3.2.1 Perception visuelle humaine des images 23
3.2.2 Particularités de la vision humaine utiles à la compression 24
3.2.3 Application des particularités de la vision humaine à la compression 24
3.3 Description du HACP-SBT 25
3.3.1 Prédiction 25

3.3.1.1 Calculs des prédicteurs 26
3.3.2 Codage 26
3.3.3 Avantages du HACP-SBT 29
3.3.3.1 Faible profondeur de dépendance de données 29
3.3.3.2 Calcul de prédicteur simple avec des instructions SIMD 29
3.3.3.3 Efficacité du prédicteur 30
3.3.3.4 Accès aléatoire 30
3.3.4 Désavantages du HACP-SBT 31
3.3.4.1 Dépendances de données 31
3.3.4.2 Algorithme de transfert mémoire 31
3.3.4.3 Taux de compression variable 32
3.4 Résumé 32

CHAPITRE 4 AMÉLIORATIONS APPORTÉES AU PRÉDICTEUR HACP ET AU CODAGE PAR SBT 33
4.1 Simplification de la hiérarchie de prédiction du HACP 33
4.1.1 Élimination d'un niveau de dépendance 34
4.1.2 Calcul du prédicteur du premier niveau 34
4.1.3 Regroupement de pixels pour l'unification 36
4.1.4 Comparaison des méthodes proposées 37
4.2 Adaptation de l'HACP-SBT pour la communication 38
4.2.1 Division de l'image 38
4.2.2 Syntaxe de codage 42
4.2.2.1 Entête d'unité 45
4.3 Extension du codage par SBT pour le codage avec perte 46
4.3.1 Troncature des bits les moins significatifs 46
4.3.2 Conservation de l'information du résultat de prédiction sur huit bits 48
4.3.2.1 Erreur de quantification de l'erreur de prédiction 48
4.3.2.2 Problème de signe 50
4.3.3 Sélection des groupes à tronquer 51
4.3.3.1 Chrominance 52
4.3.3.2 Haute fréquence spatiale 52
4.3.4 Sélection du nombre de bits à tronquer 52
4.3.4.1 Détermination de la taille codée par itération de codage 53
4.3.4.2 Estimation de la taille codée d'un bloc pour une décision a priori 54
4.3.4.3 Séparation des niveaux de codage 54
4.3.5 Modes de codage 55
4.3.5.1 Sans perte (0) 55
4.3.5.2 Avec perte rapide (1) 56
4.3.5.3 Avec perte calculée (2) 56

4.4 Résumé 56

CHAPITRE 5 CONSIDÉRATIONS D'IMPLÉMENTATION 59
5.1 Étapes d'encodage 59
5.1.1 Prédiction 60
5.1.2 Codage des niveaux 1, 2 et 3 60
5.1.3 Analyse et codage du niveau 4 61
5.2 Étape de décodage 62
5.3 Gestion de la mémoire 62
5.3.1 Gestion de la mémoire à l'encodage 62
5.3.2 Gestion de la mémoire au décodage 64
5.4 Résumé 64

CHAPITRE 6 RÉSULTATS DE L'ÉVALUATION DE L'ALGORITHME PROPOSÉ 65
6.1 Procédure de test pour l'analyse de l'algorithme proposé 65
6.2 Influence des modes de codage sur la performance 66
6.2.1 Taux de compression 66
6.2.2 Qualité visuelle 67
6.2.3 Temps d'exécution 68
6.3 Analyse des facteurs affectant le taux de compression 70
6.3.1 Taux de compression après codage H.264 70
6.3.2 Taux de compression par mode par résolution 70
6.3.3 Taux de compression par mode par vecteur de test 71
6.4 Analyse des facteurs affectant la qualité d'image 72
6.4.1 SSIM par mode par résolution 72
6.4.2 SSIM par mode par vecteur de test 73
6.5 Analyse du temps d'exécution par section de l'algorithme 73
6.5.1 Contraintes du système 73
6.5.1.1 Dépendance des données et instructions SIMD 75
6.5.1.2 Gestion de la mémoire 76
6.6 Résumé 76

CONCLUSION 77

ANNEXE I CONFIGURATION DE TEST 79

ANNEXE II LES VECTEURS DE TEST 81

ANNEXE III PROCÉDURE DE TEST POUR L'ANALYSE DES PRÉDICTEURS DE LA LITTÉRATURE 85

ANNEXE IV ANALYSE DES MÉTHODES DE SIMPLIFICATION DE LA HIÉRARCHIE DE PRÉDICTION DU HACP 89

ANNEXE V RÉSULTATS COMPLET DES TESTS D'ANALYSE DE L'ALGORITHME PRÉSENTÉ 91

BIBLIOGRAPHIE 108

LISTE DES TABLEAUX

Page

Tableau 2.1 Comparaison des temps d'exécution d'une boucle pour une unité de calcul Opus1 12

Tableau 3.1 Comparaison des prédicteurs de la littérature 23

Tableau 3.2 Codage par SBT 28

Tableau 3.3 Valeur de BLH pour chaque intervalle d'erreur 28

Tableau 4.1 Avantage des regroupements de pixels 36

Tableau 4.2 Performances des méthodes d'unification 37

Tableau 4.3 Huit premières valeurs de N_{MB} 42

Tableau 4.4 Description des champs de l'entête d'unité 47

Tableau 6.1 Temps d'exécution de l'encodeur en millisecondes par image 69

Tableau 6.2 Temps d'exécution du décodeur en millisecondes par image 69

Tableau 6.3 Taux de compression par mode en fonction de la présence de compression 70

Tableau 6.4 Taux de compression par mode en fonction de la résolution 71

Tableau 6.5 Point exclu de la figure 6.4 72

Tableau 6.6 Qualité d'image (SSIM) moyenne en fonction du mode de compression 72

Tableau 6.7 Qualité d'image (SSIM) par mode en fonction de la résolution 73

Tableau 6.8 Temps d'exécution de l'encodage pour chaque section de l'algorithme pour une unité de codage, en microsecondes 74

Tableau 6.9 Temps d'exécution du décodage pour chaque section de l'algorithme pour une unité de codage, en microsecondes 74

Tableau 6.10 Résumé de l'atteinte des objectifs pour chaque mode de codage 78

Tableau V.1 Résultats expérimentaux 107

LISTE DES FIGURES

Page

Figure 2.1 Schéma bloc de la puce OCT1010 13

Figure 2.2 Schéma bloc de la puce OCT2224M..... 14

Figure 2.3 Temps total de transfert d'une image YUV 4:2:0 720p à partir de la mémoire externe vers la mémoire interne (cache) en fonction de la taille de chaque transfert 15

Figure 3.1 Pixels voisins du pixel **p** utilisés pour calculer une prédiction spatiale 18

Figure 3.2 Illustration de la prédiction par DAP pour un bloc 8×8 19

Figure 3.3 Pixels nécessaires pour le calcul du prédicteur GAP..... 20

Figure 3.4 Prédicteurs spatiaux des groupes de SBT dans un bloc de pixels 27

Figure 3.5 Profondeur de prédiction dans un bloc de pixel 27

Figure 3.6 Groupe SBT 29

Figure 3.7 Description des paramètres de l'instruction SIMD *pavgub* du processeur Opus. 30

Figure 4.1 Simplification de la hiérarchie de prédiction du HACP 34

Figure 4.2 Pixels résultant de l'unification des niveaux hiérarchiques 35

Figure 4.3 Un macrobloc dans le domaine de couleur YUV 420..... 39

Figure 4.4 Syntaxe de codage d'une unité encapsulé dans le protocole RTP 39

Figure 4.5 Représentation de l'unité d'accès aléatoire 43

Figure 4.6 Un groupe de SBT avec les tailles présentées en bits 43

Figure 4.7 Sérialisation des blocs de pixels d'une unité d'accès aléatoire 45

Figure 4.8 Entête d'unité 45

Figure 4.9 Influence de l'arrondissement sur l'erreur de quantification..... 49

Figure 4.10 Influence de l'arrondissement sur l'erreur de quantification..... 50

Figure 4.11 Anneau $\mathbb{Z}/2^8\mathbb{Z}$ avec erreur de prédiction quantifiée illustrant problème de signe 51

Figure 4.12 Relation entre le nombre de bit tronqué et l'erreur 53

Figure 4.13 Moyenne du nombre de bits par pixel nécessaire pour coder chaque groupe SBT sans perte d'information 55

Figure 4.14 Algorithme de sélection des valeurs **LQbits** et **CQbits** 57

Figure 6.1 Distribution de probabilité cumulative du taux de compression par mode de codage 67

Figure 6.2 Distribution de probabilité cumulative de la qualité visuelle par mode de codage 68

Figure 6.3 Distribution de probabilité cumulative de la qualité visuelle par mode de codage pour les séquences 1080p 69

Figure 6.4 Taux de compression moyen par clip 71

Figure 6.5 Qualité moyenne (SSIM) par clip 74

LISTE DES ALGORITHMES

Page

Algorithme 3.1 Calcul du prédicteur par GAP 21

Algorithme 5.1 Étapes d'encodage de la solution implémentée 59

Algorithme 5.2 Étapes de décodage de la solution implémentée 63

LISTE DES EXTRAITS DE CODE

Page

Extrait 6.1 Calcul de quatre prédictions PMH pour le HACP, avec l'instruction *pavgub* 75

Extrait 6.2 Calcul de quatre prédictions PMH pour le HACP, sans l'instruction *pavgub* 75

Extrait II.2 Conversion de résolution avec FFmpeg 82

Extrait II.1 Information de version de FFmpeg 83

Extrait II.3 Encodage et décodage H.264 avec FFmpeg 83

Extrait III.1 Code C pour le calcul de l'entropie 85

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

1080p	Résolution d'une image ayant 1080 lignes et 1920 colonnes
1080p30	Flux vidéo de résolution 1080p de 30 images par secondes
720p	Résolution d'une image ayant 720 lignes et 1280 colonnes
ALU	*Arithmetic and Logic Unit* (unité arithmétique et logique)
BLH	*Bit Length Header* (pour calcul pour le codage SBT)
BPP	*Bit Per Pixel* (pour calcul pour le codage SBT)
CALIC	*Context Based Adaptive Lossless Image Codec* (algorithme de compression proposé par (Wu et Memon, 1997))
CIF	*Common Intermediate Format* (format d'image ayant une résolution de 288 lignes et 352 colonnes)
CODEC	COdeur - DÉCodeur
DAP	*Differential of Adjacent Pixel* (prédiction par différence de pixel adjacent)
DMA	*Direct Memory Access* (accès direct à la mémoire)
DSP	*Digital Signal Processor* (processeur de signal numérique)
GAP	*Gradient-Adjusted Prediction* (prédiction ajustée en fonction du gradient)
HACP	*Hierarchical Average Copy Prediction* (prédiction par moyenne et copie hiérarchique)
HD	Haute définition, un qualitatif donné à une images ou une résolution pour indiquer qu'elle est plus grande que la résolution standard
IP	*Internet Protocol* (protocole internet)
JPEG	*Joint Photographic Expert Group* (standard de compression d'image)

JPEG-LS	JPEG Lossless (version sans perte du standard de compression JPEG (ISO-14495-1/ITU-T.87))
LOCO-I	*LOw COmplexity LOssless COmpression for Images* (algorithme de compression proposé par (Weinberger *et al.*, 2000))
MED	*Median Edge Detector* (prédiction par détection de contour médian)
NTSC	*National Television System Committee* (dit des images avec résolution de 480 lignes et 720 colonnes)
OCT1010	Processeur de signal numérique de première génération de la compagnie Octasic
OCT2224M	Processeur de signal numérique de deuxième génération de la compagnie Octasic
Opus	Unité arithmétique et logique asynchrone contenue dans les processeur de signal numérique de la compagnie Octasic
PCH	Prédiction par Copie Horizontale (calcul pour le prédicteur HACP)
PCV	Prédiction par Copie Verticale (calcul pour le prédicteur HACP)
PM	Prédiction par Moyenne (calcul pour le prédicteur HACP)
PMH	Prédiction par Moyenne Horizontale (calcul pour le prédicteur HACP)
PMV	Prédiction par Moyenne Verticale (calcul pour le prédicteur HACP)
PNG	*Portable Network Graphic* (standard de compression d'image sans perte)
PSNR	*Peak Signal to Noise Ratio* (rapport signal à bruit crête)
QCIF	*Quarter Common Intermediate Format* (format d'image ayant une résolution de 144 lignes et 176 colonnes)
RLE	*Run-Length-Encoding* (codage par plage)

RTP	*Real-Time Transport Protocol* (protocole de transport temps réel)
SBT	textitSignificant Bit Truncation (codage par troncature de bit)
SIMD	*Single Instruction Multiple Data* (une instruction de processeur qui s'exécute sur plusieurs donnée simultanément)
SSIM	*Structural SIMilarity* (mesure de similarité)
SVH	Système Visuel Humain
UDP	*User Datagram Protocol* (protocole de datagramme utilisateur)
YUV	Méthode pour représenter des couleurs, le Y représente la luminance tandis que U et V représentent la chrominance

INTRODUCTION

Problématique

Dans plusieurs systèmes de traitement vidéo, à un certain moment, le contenu vidéo se trouve à l'état non compressé. Avec la popularisation de vidéo haute définition, la bande passante utilisée pour les données vidéo non compressées est considérable. En effet, un contenu vidéo avec une résolution de 1920×1080 et un débit de 30 images par secondes (1080p30) requiert une bande passante de 1.5 Go/s pour un système de communication bidirectionnel lorsque ce dernier est représenté dans le domaine de couleurs YUV. Cette quantité de données devient importante, voire déshabilitante, lorsqu'elles doivent être transférées sur des liens Ethernet. La vidéo doit donc être compressée efficacement lors de ces transferts. Toutefois, les méthodes de compression sont très variées et offrent différents compromis entre le taux de compression, la qualité visuelle, et la complexité en calcul. Ainsi, pour un contexte applicatif précis, déterminer une méthode adéquate de compression peut s'avérer un défi imposant. C'est le cas pour le système multicœur de traitement vidéo d'Octasic.

Objectif

L'objectif de ce travail de recherche est de développer et d'implémenter une méthode de compression pour le transport de contenu vidéo respectant plusieurs contraintes importantes liées au système multicœur de traitement vidéo d'Octasic dénommé la passerelle de média *Vocallo MGW* d'Octasic. Le transport de données vidéo dont il est question dans ce travail est celui effectué entre les cœurs d'une puce d'Octasic.

Contraintes de la passerelle de média Vocallo MGW

La méthode de compression développée doit satisfaire trois contraintes.

La première contrainte est le taux de compression. L'algorithme de compression proposé doit permettre d'atteindre un taux de compression supérieur à 1.5. Ceci est nécessaire pour permettre une communication bidirectionnelle 1080p30 sur un lien Gigabit Ethernet.

La seconde contrainte est le maintien de la qualité visuelle. Pour qu'un algorithme de compression puisse être utilisé pour remplacer un transfert de données vidéo non compressées, il ne doit pas affecter la qualité visuelle des images.

La troisième contrainte est le temps d'exécution. Pour que les algorithmes puissent être utilisés pour une communication à 30 images par seconde, le temps d'exécution doit être moins de 33.33 ms pour encoder ou décoder une image, et ce sur une puce d'Octasic.

Organisation du mémoire

Le chapitre 1 présente les concepts de base reliés à la compression d'information, plus précisément la compression des images et des vidéos. Le chapitre 2 présente les contraintes et les défis d'implémentation pour une puce d'Octasic. Le chapitre 3 explore la littérature et fait une revue des algorithmes de compression de basse complexité proposés par différents chercheurs. Le chapitre 4 présente les principales contributions de ce travail, sous la forme de plusieurs améliorations à un algorithme tiré de la littérature en regard des contraintes liées aux puces d'Octasic. Le chapitre 5 décrit les techniques utilisées pour implémenter l'algorithme. Le chapitre 6 analyse le comportement et les performances de l'algorithme présenté par rapport à l'atteinte de l'objectif. Une conclusion résume les aspects importants de ce travail.

CHAPITRE 1

CONCEPTS DE BASE

Ce chapitre présente les principes de base de la compression d'images et de séquences vidéos. Ces concepts de base sont détaillés dans le livre (Sayood, 2005).

1.1 Compression de l'information

1.1.1 Théorie de l'information

Avant d'explorer les principes de compression, il est important de comprendre les bases de la théorie de l'information. Celle-ci ont été établies par Claude E. Shannon en 1948 (Shannon, 1948). Il propose une définition mathématique de l'information contenue dans une communication.

Une communication est l'échange d'un message entre un émetteur et un récepteur. L'entropie est une mesure de l'information contenue dans un message issu d'une source d'information. Ce message est défini par un alphabet, soit une liste de symboles. Ainsi, l'équation (1.1) définit l'entropie $H(x)$ d'une source d'information avec alphabet $x = \{x_1, x_2, ..., x_n\}$ de n symboles, où $p(x_i)$ est la probabilité que le symbole x_i soit transmis.

$$H(x) = -\sum_{i=1}^{n} p(x_i) \log_b \left(p(x_i)\right) \tag{1.1}$$

Dans l'équation (1.1), b définit l'unité de mesure de l'information. Ainsi, pour appliquer cette équation dans le contexte actuel, $b = 2$, car l'unité de mesure est le bit. De cette façon, l'équation (1.2) peut être définie pour un message composé d'images avec des pixels de 8 bits, soit un alphabet de 256 symboles.

$$H_p(x) = -\sum_{i=0}^{255} p(x_i) \log_2 \left(p(x_i)\right) \tag{1.2}$$

La mesure de $H_p(x)$ permet de mesurer quantité de redondance dans une image ou une section d'image x. La plus grande valeur de $H_p(x)$ est atteinte lorsque la distribution des probabilités est uniforme. Dans ce cas, nous avons très peu d'information a priori sur le contenu du message. À l'inverse, une petite valeur de $H_p(x)$ indique que certaines valeurs de pixel sont beaucoup plus probables que d'autres, donc qu'il y a plus de redondance statistique dans le message. Concrètement, $H_p(x)$ avec $b = 2$ est une mesure du nombre minimal de bits par pixel moyen requis pour représenter une image ou une vidéo sans perte d'information.

1.1.2 Types de compression

Il existe deux types de compression, soit la compression sans perte d'information et la compression avec perte d'information. En utilisant des techniques de compression avec perte d'information, il est possible d'atteindre des taux de compression dépassant la limite proposée par la théorie de l'information de Shannon. Cependant, pour pouvoir utiliser un tel mode de compression, les détériorations du message doivent être bien contrôlées pour assurer que le message demeure compréhensible.

1.2 Outil de compression d'images et de vidéos

La compression vidéo se distingue de la compression d'image par l'exploitation des redondances supplémentaires, soit les redondances temporelles.

1.2.1 Prédiction

Un des principes de base de la compression d'images et de vidéos est la prédiction. Celle-ci vise à réduire l'information à transmettre dans une section d'image en prédisant son contenu. Elle peut être appliquée à un bloc de pixel ou à des pixels individuellement, le concept est le même. Grâce à un prédicteur $\hat{p}$, un pixel p peut être transmis par un émetteur sous la forme

de p' selon l'équation (1.3). Le récepteur, connaissant $\hat{p}$, peut reconstruire le pixel p sans perte d'information selon l'équation (1.4).

$$p - \hat{p} = p' \tag{1.3}$$

$$p' + \hat{p} = p \tag{1.4}$$

Un bon prédicteur permettra de diminuer significativement l'amplitude du signal à transmettre et réduira ainsi le nombre de bits. En effet, c'est un principe bien connu en compression que le nombre de bits requis pour transmettre un signal augmente avec la variance de ce dernier. Un prédicteur doit donc réduire les redondances statistiques d'un message.

Il y a deux formes principales de prédiction utilisées en compression vidéo, soit la prédiction temporelle et la prédiction spatiale. Pour la compression d'image, seulement la prédiction spatiale peut être utilisée, car aucune information temporelle n'est disponible. C'est ce qui différencie ces deux types de compression.

1.2.1.1 Prédiction spatiale

La prédiction spatiale consiste à utiliser le ou les pixels voisins d'un pixel p en tant que prédicteur $\hat{p}$. L'efficacité de ce mode de prédiction se base sur le fait que les images naturelles présentent une redondance spatiale considérable. C'est-à-dire que dans une image, les pixels autour d'un pixel lui ressemblent.

$$\hat{p}_{(i,j)} = p_{(i-a,j-b)} \tag{1.5}$$

Une forme générale du prédicteur $\hat{p}$ est montrée dans l'équation (1.5) où i et j sont, respectivement, la position horizontale et verticale du pixel et a et b sont la distance horizontale et verticale entre le prédicteur et la donnée prédite. Des prédicteurs spatiaux peuvent être constitués de sommes pondérées de pixels voisins. Ces prédicteurs sont souvent plus efficaces mais demandent plus de calculs.

1.2.1.2 Prédiction temporelle

Lorsque l'information d'un pixel est disponible dans le temps, celle-ci présente un autre aspect redondant, surtout si les deux échantillons de temps sont rapprochés. Ainsi ayant les images au temps t et $t-1$, l'information de l'image précédente $(t-1)$ peut être utilisée pour prédire l'image t. La méthode la plus simple ignore le mouvement entre deux images consécutives pour prédire un pixel à l'aide du pixel aux mêmes coordonnées spatiales dans l'image précédente. Ceci et présenté par l'équation (1.6) où $p_{(i,j)}(t)$ est un pixel au temps t à la position (x, y).

$$\hat{p}_{(i,j)}(t) = p_{(i,j)}(t-1) \tag{1.6}$$

Cependant les séquences vidéo naturelles présentent habituellement du mouvement. Donc, pour augmenter l'efficacité de la prédiction temporelle, il faut estimer le mouvement et compenser les différences causées par celui-ci. Ainsi, lorsqu'un émetteur transmet un message utilisant de la prédiction temporelle avec de la compensation de mouvement, il doit aussi transmettre des descripteurs de mouvement. Ceux-ci prennent habituellement la forme de vecteur de mouvement (a, b) indiquant à quel endroit dans l'image précédente le prédicteur du pixel peut être trouvé, comme le montre l'équation (1.7). Ainsi nous pouvons parler de prédiction spatio-temporelle puisque ce prédicteur est en fait la combinaison de ces deux techniques.

$$\hat{x}_{(i,j)}(t) = x_{(i-a,j-b)(t-1)} \tag{1.7}$$

Souvent, la prédiction temporelle est appliquée sur un groupe ou un bloc de pixels. Ceci simplifie la recherche de mouvement étant donné qu'elle ne doit pas être faite sur chaque pixel. D'ailleurs, quelques algorithmes plus complexes augmentent l'efficacité de la prédiction temporelle en utilisant plusieurs images de référence dans le passé et le futur, et ce, en prenant de l'information d'une ou de plusieurs de celles-ci. Bien sûr, l'utilisation d'information dans le futur demande de changer l'ordre de transmission du message. Cela revient au récepteur de remettre le message dans l'ordre approprié pour qu'il puisse être compris.

1.2.2 Transformation

Une transformation permet de changer le domaine des données d'un message. Par exemple, avec une transformée de Fourier discrète ou une transformée en cosinus discrète, on peut passer du domaine spatial au domaine fréquentiel. L'objectif est de compacter l'information contenue dans un groupe de pixels sur un nombre restreint de coefficients (ex. des coefficients fréquentiels) permettant de reconstruire une bonne approximation des pixels originaux avec un nombre réduit de coefficients (en ignorant les coefficients où l'information est peu concentrée). Les processus de transformation et de transformation inverse sont définis respectivement comme :

$$H = F(X) \tag{1.8}$$

$$\hat{X} = F'(H_s) \tag{1.9}$$

où F est la transformée, F' la transformée inverse, $H = [h_0, h_1 \dots h_{N-1}]$ un vecteur de coefficients transformés, $X = [x_0, x_1 \dots x_{N-1}]$ un vecteur de pixels originaux, N la taille du vecteur (bloc) de pixels, H_s un vecteur coefficients à inverser approximant H (ex. H dont une ou plusieurs composantes ont été mises à zéro), et $\hat{X}$ une approximation de X. On a $\hat{X} = X$ ssi $H_s = H$.

Ainsi, une transformation compacte efficacement un message si les pixels originaux peuvent être reconstruits avec une bonne fidélité à l'aide d'une transformation inverse sur un nombre réduit de coefficients obtenus lors de la transformation. Dans ce cas, une transformation est une autre approche utilisée pour réduire les redondances dans un message. L'efficacité de cette méthode est accrue lorsqu'elle est combinée avec de la quantification (voir la section 1.2.4) pour éliminer les coefficients les moins significatifs ou réduire leur nombre de bits.

1.2.3 Codage entropique

Les techniques de codage entropique utilisent les redondances d'un message. Elles n'utilisent pas les propriétés dérivées de ce que le message représente, elles se basent seulement sur les statistiques des symboles d'un message. Ainsi, elles peuvent s'appliquer sur n'importe quel message pour lequel il est possible de calculer la distribution statistique des symboles qui le composent.

Il existe plusieurs méthodes de codage entropique bien documentées, telles que le RLE, Huffman, arithmétiques, Golomb et Lempel-Ziv.

1.2.4 Quantification

La quantification est un outil utilisé pour la compression avec perte d'information. Celle-ci permet de représenter un grand nombre, possiblement infini, de valeurs avec un plus petit ensemble fini de symboles.

La conversion d'un nombre réel à un nombre entier contenu dans un certain intervalle est un exemple de quantification. Une fois convertie en entier, l'information fractionnaire du nombre est perdue.

C'est souvent ce type de perte d'information que l'on attribue aux algorithmes de compression avec perte d'information. Cette perte d'information est aussi appelée l'erreur de quantification.

1.3 Résumé des outils de codage

Voici un résumé des outils de compression principaux utilisés dans la compression d'image et de vidéo et présentés dans ce chapitre.

1.3.1 Outils de compression sans perte d'information

- La prédiction spatio-temporelle pour éliminer les redondances propres au contenu compressé.

- Transformation pour compacter l'information dans un nombre restreint de coefficients.
- Codage entropique pour éliminer les redondances d'information plus générique.

1.3.2 Outils de compression avec perte d'information

La quantification permet d'éliminer de l'information non redondante, mais peu significative.

CHAPITRE 2

SYSTÈME - CONTEXTE D'IMPLÉMENTATION

Ce chapitre présente les défis d'implémentation dus à la technologie d'unité de calcul asynchrone Opus développé par Octasic. Ces contraintes concernent principalement l'unité de calcul et la mémoire. Celles-ci sont expliquées dans les sections 2.1 et 2.2 respectivement.

2.1 Contraintes de l'unité de calcul

Chaque unité de calcul Opus est composée de 16 unités arithmétiques et logiques asynchrones (ALU). Celles-ci peuvent exécuter des instructions de façon quasi concurrente. Les changements d'état ne sont pas limités à des incréments d'horloge. Dès que toutes les ressources nécessaires à l'exécution d'une instruction sont disponibles, elle est exécutée. Ensuite, dès que l'exécution est terminée, le résultat est disponible pour les prochaines instructions. Ces unités de calcul sont parfois désignées sous le nom de cœur.

La nature asynchrone des coeurs est un avantage de cette technologie, mais aussi une contrainte, car les dépendances de donnée entre des instructions séquentielles peuvent limiter leurs exécutions à un enchaînement quasi séquentiel. En d'autres mots, deux instructions d'addition placées de façon séquentielle dans le code peuvent s'exécuter presque en même temps, seulement si le résultat de la première n'est pas nécessaire pour la deuxième. Ainsi les dépendances de données peuvent multiplier le temps d'exécution par un facteur significatif comparativement à un algorithme ayant le même nombre d'opérations sans dépendances.

Le tableau 2.1 démontre l'importance d'utiliser les instructions spécialisées disponibles, particulièrement les instructions de type SIMD, pour satisfaire les contraintes de temps réel des vidéos à haute définition. Par exemple, à 60 images par seconde, chaque image doit être traitée en 16 ms. Le tableau 2.1 montre qu'une seule opération de soustraction par pixel ne peut pas être réalisée en temps réel si des boucles imbriquées sont utilisées pour traiter une image HD. On voit aussi que les opérations SIMD vont permettre d'effectuer le maximum d'opérations

Tableau 2.1 Comparaison des temps d'exécution d'une boucle exécutant une soustraction par pixel d'une image 1080p, pour différentes techniques d'optimisation implémentées pour une unité de calcul Opus1

	Boucle imbriquée de soustraction d'octets	Boucle optimisée en assembleur Opus de soustraction d'octets	Boucle optimisée en assembleur Opus de soustraction d'octets avec des instructions SIMD. L'instruction `psubbb` s'exécute avec des registres de 32 bits et effectue quatre soustractions d'octets par exécution.

Extrait de code	`for(x=0;x<192;x++)`
	`  for(y=0;y=1080;y++)`
	`  p1 = p2-p3;`
		`_asm mov cnt, 1920*1080`
		`LOOP:`
		`_asm sub p11, p21, p31`
		`_asm sub p12, p22, p32`
		`_asm sub p13, p23, p33`
		`_asm sub p14, p24, p34`
		`_asm sub p15, p25, p35`
		`_asm sub p16, p26, p36`
		`_asm sub p17, p27, p37`
		`_asm sub p18, p28, p38`
		`_asm d8jnsrp cnt, LOOP`
			`_asm mov cnt, 1920*1080`
			`LOOP:`
			`_asm pssubb p11, p21, p31`
			`_asm psubbb p12, p22, p32`
			`_asm psubbb p13, p23, p33`
			`_asm psubbb p14, p24, p34`
			`_asm psubbb p15, p25, p35`
			`_asm psubbb p16, p26, p36`
			`_asm psubbb p17, p27, p37`
			`_asm psubbb p18, p28, p38`
			`_asm d32jnsrp cnt, LOOP`

Temps d'exécution	24.7 ms	4.2 ms	1 ms

par pixels, mais que le nombre d'opérations est tout de même très limité (l'équivalent de 16 soustractions par pixel pour une image HD à 60 images par seconde).

2.2 Contraintes de mémoire

Octasic produit actuellement deux puces avec la technologie Opus. Soit la première génération le OCT1010 (figure 2.1) et son successeur le OCT2224M (figure 2.2). La première génération comporte seize unités de calcul Opus, chacun avec 96 kilo-octets de mémoire cache. La deuxième génération est composée de vingt-quatre unités de calcul Opus, avec chacun 144 kilo-octets de mémoire. Les deux produits utilisent une architecture de mémoire de type Von Neumann (von Neumann, 1993), partageant cette mémoire entre les données et le code.

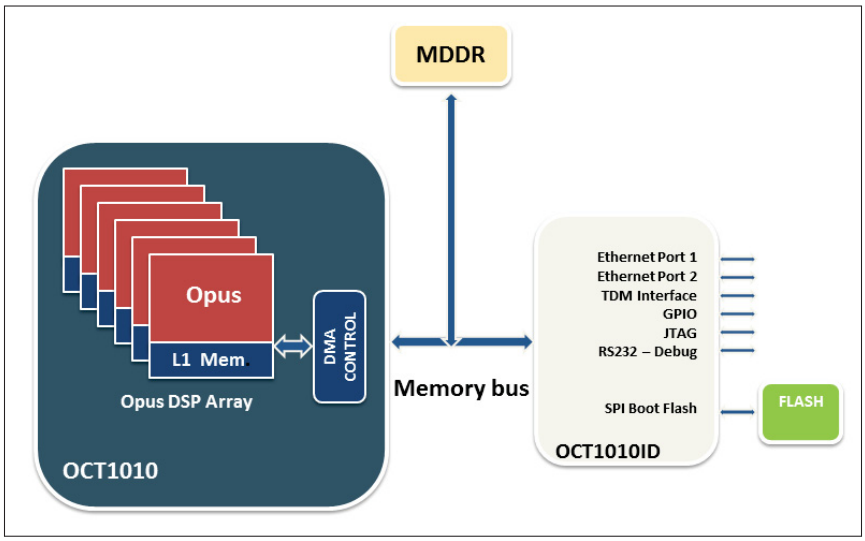


Figure 2.1 Schéma bloc de la puce OCT1010,
tiré de http://www.octasic.com/products/oct1010

Pour permettre à une application d'être exécutée sur les deux types de puces, le code et les données applicatives doivent respecter les limites de taille du dispositif de première génération.

De plus, il n'y a pas d'engin matériel pour gérer les entrées dans la mémoire cache. Cette gestion est effectuée par un logiciel bas-niveau. Ce code bas niveau doit être résident en mémoire cache. Ainsi l'espace disponible pour l'application est d'environ 64 kilo-octets.

Les transferts de code et de données entre la mémoire externe et interne (cache) sont gérés par l'engin d'accès direct à la mémoire ou *Direct Memory Access* (DMA). Il n'y a qu'un seul engin DMA pour l'ensemble des coeurs Opus. De plus, l'architecture mémoire empêche le coeur Opus de lire de nouvelles instructions pendant un transfert DMA et son temps de configuration. Le temps total de transfert, incluant le temps de configuration, doit être considéré lorsque le temps d'exécution d'un algorithme est évalué.

Cette gestion de la mémoire devient un enjeu important pour des applications de traitement vidéo haute définition. Étant donné que 64 kilo-octets ne sont pas suffisants pour représenter une image HD, elle doit être transférée et traitée par section. Le choix de la taille de ces sections d'image est cruciale. La figure 2.3 montre que le temps fixe de configuration par transfert a un impact significatif sur le temps de transfert total si de petits transferts sont utilisés. De plus, la

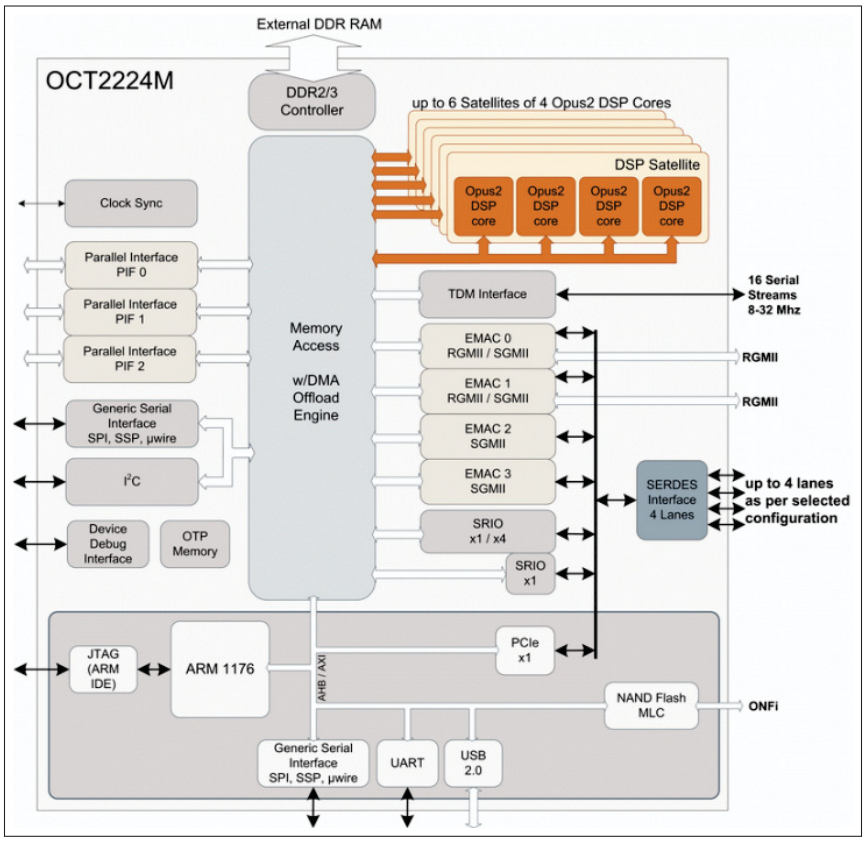


Figure 2.2 Schéma bloc de la puce OCT2224M,
tiré de http://www.octasic.com/products/oct2224m

gestion de ces sections d'image est un compromis entre le temps total de transfert d'une image et l'espace utilisé en mémoire cache. Ces contraintes montrent que le temps de lecture d'une image HD, même en augmentant la taille des transferts, représente une portion significative de la tranche de temps disponible pour demeurer temps réel.

2.3 Résumé des contraintes

Voici un résumé des contraintes et caractéristiques de la technologie Opus d'Octasic.

- Les calculs sont efficaces s'ils sont parallèles (SIMD et sans dépendance).
- Peu d'opérations par pixel sont permis pour garder le temps d'exécution sous la barrière du temps réel (davantage important pour le HD).

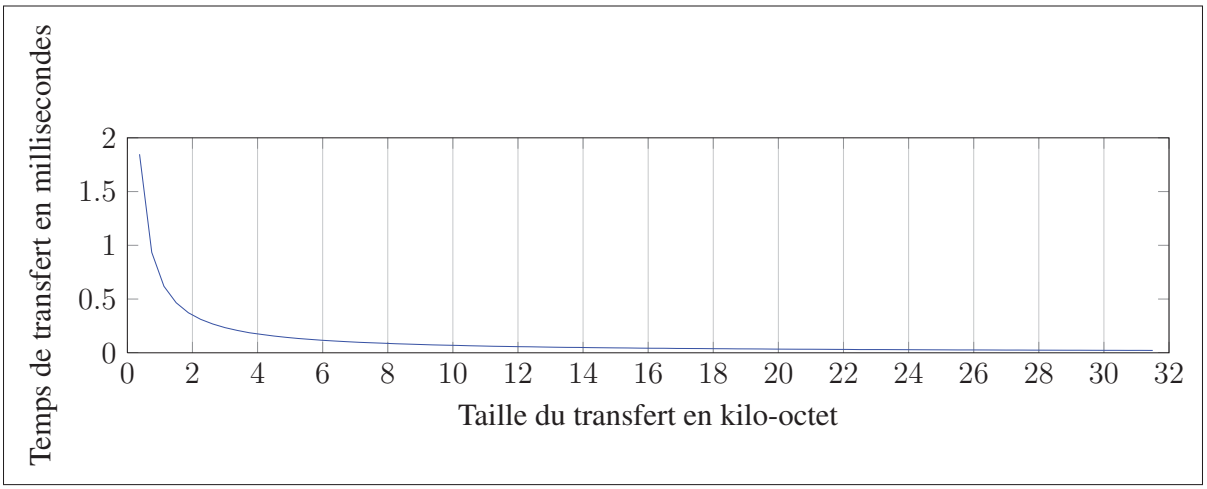


Figure 2.3 Temps total de transfert d'une image YUV 4:2:0 720p à partir de la mémoire externe vers la mémoire interne (cache) en fonction de la taille de chaque transfert

- Puisque la quantité de mémoire cache est très limitée, la gestion de la mémoire est un enjeu important.

CHAPITRE 3

ÉTAT DE L'ART

Le problème de compression auquel nous nous intéressons dans ce travail a été étudié sous deux fronts. Premièrement, nous nous sommes attelés à la recherche d'algorithmes de compression sans perte d'information. Deuxièmement, les principes du système visuel humain ont été étudiés. Plus précisément, ceux qui peuvent être utilisés pour rendre les artefacts de compression avec pertes d'information moins visibles. Ce second point est nécessaire pour garantir un taux de compression permettant d'atteindre l'objectif de ce travail. Toutefois, puisque le taux de compression nécessaire est approximativement atteint par les algorithmes de compression sans perte d'information, l'effort de recherche a donc été concentré sur cette partie présentée dans la section 3.1. La compression avec perte d'information a été traitée comme un cas d'exception présenté dans la section 3.2.

3.1 Compression sans perte d'information

Les conclusions du chapitre précédent entraînent que la prédiction temporelle ne peut pas être utilisée. Ainsi, la compression vidéo proprement dite n'est pas possible. Les efforts de recherche ont été concentrés dans le domaine de la compression d'images, puisqu'une séquence vidéo peut être traitée comme une série d'images séquentielles.

Selon (Martin et De, 2005), la majorité des techniques de compression d'images modernes ont une approche similaire, soit une prédiction suivie d'une modélisation de contexte et d'un codage entropique basé sur le contexte. Ce contexte de compression permet d'adapter les codeurs entropiques aux différents contenus possibles d'une image. Les sous-sections 3.1.1 à 3.1.5 explorent divers algorithmes de prédiction tirés de la littérature. Les sous-sections **??** examinent les contextes de codage et le codage entropique.

3.1.1 La prédiction par différence de pixel adjacent (DAP)

Une des méthodes de prédiction spatiale les plus simples est de prédire un pixel **p** à partir de son voisin. Cette technique est utilisée dans le standard de compression d'image Portable Network Graphic (PNG) (Duce, 2003). Ce standard définit cinq modes de prédiction spatiale. De ces modes, deux utilisent un pixel voisin pour prédire un pixel **p**. On utilise soit le pixel de gauche ou celui en haut du pixel **p** à coder. Ces pixels sont respectivement les pixels a et b de la figure 3.1.

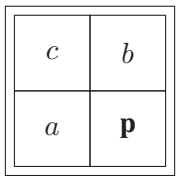


Figure 3.1 Pixels voisins du pixel **p** utilisés pour calculer une prédiction spatiale

La prédiction par différence de pixel adjacent (DAP) de l'anglais *Differential of Adjacent Pixel* (Song et Shimamoto, 2007) utilise aussi les pixels voisins pour la prédiction spatiale. L'aspect intéressant est l'implémentation de cette méthode. L'ordre des opérations fait que la dépendance de données permet l'exécution en parallèle de l'algorithme. Ceci est montré à la figure 3.2.

Cette figure représente un bloc 8×8, dans lequel les flèches pointent vers le pixel utilisé comme prédicteur pour chaque pixel. Dans ce bloc, le pixel 0 n'est pas prédit. Ensuite, le pixel 1 est prédit par le pixel 0, et ainsi de suite pour les pixels de 1 jusqu'à 7. Par la suite, tous les pixels de la colonne de pixels 8 peuvent être prédits en parallèle. Ce même concept s'applique pour les autres pixels, jusqu'à la colonne de pixels 14. Ceci illustre bien le concept de profondeur de prédiction, c.-à-d. la séquence d'opérations dépendante qui limite la reconstruction en parallèle de pixels. Dans le cas de DAP, après la phase initiale où un pixel est reconstruit à la fois, toutes les données sont disponibles pour permettre de reconstruire huit pixels en même temps. Ceci est détaillé dans la section 3.1.5.2.

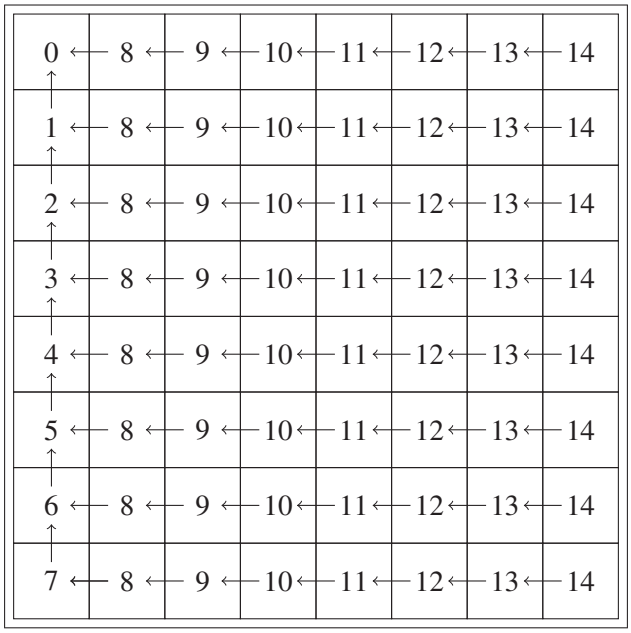


Figure 3.2 Illustration de la prédiction par DAP pour un bloc 8×8

3.1.2 Prédiction par détection de contour médian (MED)

Un inconvénient de l'utilisation d'un voisin fixe pour la prédiction apparait lorsque le pixel se trouve sur une bordure ou un contour. Ce problème est abordé par la prédiction par détection de contour médian (MED) de l'anglais *Median Edge Detector* utilisée dans l'algorithme LOCO-I (Weinberger *et al.*, 2000) qui est la base du standard JPEG-LS (ISO-14495-1/ITU-T.87), la version sans perte du standard de compression JPEG. La détection de contour est effectuée en comparant les pixels voisins. Ce calcul est détaillé avec l'équation (3.1) qui réfère aux pixels définis dans la figure 3.1 et où $\hat{p}$ représente la valeur de **p** prédite (aussi appelée, dans ce mémoire, la prédiction ou le préditeur de **p**).

$$\hat{p} = \begin{cases} \min(a,b) & \text{si } c \geq \max(a,b) \\ \max(a,b) & \text{si } c \leq \min(a,b) \\ a+b-c & \text{autrement} \end{cases} \tag{3.1}$$

3.1.3 Prédiction ajustée en fonction du gradient (GAP)

Un autre algorithme qui a été considéré pour la standardisation de JPEG-LS est le CALIC (Wu et Memon, 1997). Celui-ci utilise la prédiction ajustée en fonction du gradient (GAP) de l'anglais *Gradient-Adjusted Prediction*. Pour détecter la présence de contour, cet algorithme utilise les pixels identifiés à la figure 3.3. Ces pixels sont utilisés pour calculer un gradient horizontal (équation (3.2)) et un gradient vertical (équation (3.3)).

		p_{nn}	p_{nne}
	p_{no}	p_n	p_{ne}
p_{oo}	p_o	**p**	

Figure 3.3 Pixels nécessaires pour le calcul du prédicteur GAP

$$d_h = |p_o - p_{oo}| + |p_n - p_{no}| + |p_n - p_{ne}| \tag{3.2}$$
$$d_v = |p_o - p_{no}| + |p_n - p_{nn}| + |p_{ne} - p_{nne}| \tag{3.3}$$

Ensuite, ces gradients sont comparés pour déterminer la direction et l'intensité de la bordure. Cette procédure est décrite dans l'algorithme 3.1 permettant ainsi de déterminer comment sera calculé le prédicteur $\hat{p}$.

3.1.4 Prédiction hiérarchique (HACP)

Contrairement aux autres méthodes expliquées précédemment, dans (Kim et Kyung, 2010), les auteurs proposent une méthode de prédiction hiérarchique. Cette méthode ne définit pas une formule pour calculer le prédicteur de chaque pixel, mais une hiérarchie de prédiction. Celle-ci est définie par une séquence d'opérations sur un bloc 8×8 de pixels. Cette séquence minimise les dépendances de données et permet une architecture de traitement parallèle permettant de diminuer la profondeur de prédiction. Cet algorithme est décrit en détail dans la section 3.3.

Entrées : d_v et d_h calculés avec les équations (3.3) et (3.2), respectivement
si $d_v - d_h > 80$ **alors**
    fort contour horizontal;
    $\hat{p} = p_o$;
sinon si $d_v - d_h < -80$ **alors**
    fort contour vertical;
    $\hat{p} = p_n$;
sinon
    $\hat{p} = \frac{p_o+p_n}{2} + \frac{p_{ne}+p_{nw}}{4}$;
    si $d_v - d_h > 32$ **alors**
        contour horizontal;
        $\hat{p} = \frac{\hat{p}+p_o}{2}$;
    sinon si $d_v - d_h > 8$ **alors**
        faible contour horizontal;
        $\hat{p} = \frac{3\hat{p}+p_o}{4}$;
    sinon si $d_v - d_h < -32$ **alors**
        contour vertical;
        $\hat{p} = \frac{\hat{p}+p_n}{2}$;
    sinon si $d_v - d_h < -8$ **alors**
        faible contour vertical;
        $\hat{p} = \frac{3\hat{p}+p_n}{4}$;
fin

Algorithme 3.1 : Calcul du prédicteur par GAP

3.1.5 Évaluation des prédicteurs

Le prédicteur recherché doit être simple et efficace pour permettre une prédiction en temps réel et un bon taux de compression. Trois facteurs sont considérés pour comparer ces prédicteurs. Le tableau 3.1 montre les résultats de cette comparaison.

3.1.5.1 Opérations par pixel

Le premier facteur est le nombre d'opérations par pixels. Celui-ci est une bonne indication de la complexité des calculs impliqués, et ainsi du temps d'exécution de l'implémentation de l'algorithme.

3.1.5.2 Profondeur de prédiction

La profondeur de prédiction est une métrique mesurant la dépendance des données dans un groupe de pixels. Ce groupe peut être une image ou un bloc à l'intérieur d'une image. Concrètement, elle représente le nombre maximal d'étapes successives de prédiction qui sépare un pixel du premier pixel d'un groupe de pixel. Ce concept est bien illustré par la prédiction DAP à la section 3.1.1. Une petite profondeur de prédiction est une indication que l'exécution d'un algorithme peut être accélérée lorsqu'il est implémenté dans un environnement permettant des calculs en parallèle comme les DSP utilisant la technologie Opus. Pour mesurer la profondeur de prédiction des algorithmes de prédiction, les pixels sont regroupés en blocs 8×8.

3.1.5.3 Ratio entropique

Le ratio entropique est utilisé pour estimer la qualité de la prédiction. Un prédicteur de bonne qualité diminue la quantité d'information redondante dans un message. Pour l'évaluation des prédicteurs, un bloc de 8×8 pixels est utilisé comme message.

$$R_{ent} = \frac{H_p(\mathbf{p})}{H_p(\hat{p})} \tag{3.4}$$

Le ratio entropique (R_{ent}) est défini par l'équation (3.4) et représente une approximation du taux de compression qu'il est possible d'atteindre avec une méthode de prédiction donnée. Elle utilise le rapport de l'entropie ($H_p(x)$, équation (1.2)) d'un bloc de 8×8 pixels, avant et après la prédiction. Donc, le ratio entropique est le rapport entre la quantité d'information dans le bloc original et le bloc prédit. Ainsi un grand rapport entropique indique une bonne qualité de prédiction. Ici, $\hat{p}$ et $\hat{p}$ sont des variables aléatoires représentant respectivement les pixels de l'image et leur prédiction.

3.1.5.4 Résultats

La table 3.1 montre la comparaison des métriques définies plus haut pour les prédicteurs spatiaux présentés dans cette section. Ces résultats expérimentaux ont été obtenus avec la procédure décrite à l'annexe III. Certains résultats montrent un intervalle pour le nombre d'opérations par pixels. Ceci est dû au fait que l'algorithme en question utilise un nombre d'opérations variable. L'intervalle représente le nombre minimal et maximal d'opérations par pixel.

Tableau 3.1 Comparaison des prédicteurs de la littérature

Algorithme de prédiction	Opérations par pixel	Profondeur de prédiction	Ratio entropique moyen
DAP (Song et Shimamoto, 2007)	0	14	1.21
MED (Weinberger *et al.*, 2000)	2-4	14	1.61
GAP (Wu et Memon, 1997)	18-36	21	1.20
HACP (Kim et Kyung, 2010)	0-4	5	1.28

Ces résultats montrent que la prédiction par HACP est supérieure au DAP et GAP en ce qui concerne le ratio entropique moyen. Il se démarque aussi par sa petite profondeur de prédiction. Quant au nombre d'opérations par pixel, il se rapproche du DAP, qui ne nécessite aucun calcul pour déterminer le prédicteur. Ceci justifie le choix de l'algorithme de HACP comme base pour ce travail.

3.2 Codage avec perte d'information sans-perte de qualité visuelle

3.2.1 Perception visuelle humaine des images

La vision humaine est sensible à deux composantes : la luminance et la chrominance. La luminance représente l'intensité lumineuse, soit la vision en noir et blanc. Tandis que la chrominance représente l'intensité chromatique, soit la perception de couleur (Wang *et al.*, 2002). Ces caractéristiques sont perçues par la rétine, un organe important de la vision. Celle-ci est principalement composée de deux types de cellule : les cônes et les bâtonnets. Les bâtonnets sont surtout situés en périphérie et ne perçoivent pas les couleurs. Les cônes se situent plus au

centre et sont responsables de la perception des couleurs (Sayood, 2005). Il existe trois types de cônes chacun responsable de la perception de différente tonalité chromatique (Young, 1802).

3.2.2 Particularités de la vision humaine utiles à la compression

Lors de la compression avec perte d'information, certains artefacts de compression sont insérés dans les images ou les séquences vidéos. Il existe quelques particularités du système visuel humain (SVH) qui peuvent rendre ces artefacts moins visibles et ainsi permettre de conserver une bonne qualité visuelle.

Premièrement, la sensibilité du SVH aux composantes de la lumière n'est pas égale. En effet, l'oeil humain est plus sensible à la luminance qu'à la chrominance (Pritchard, 1977). Ainsi, les pertes d'information concernant les couleurs sont moins visibles.

Il faut aussi considérer la perception de contraste. Celle-ci est plus aiguë lorsque les fréquences spatiales et temporelles sont petites (Robson, 1966). Ceci indique qu'une perte d'information dans une région d'image ayant un contenu fréquentiel élevé est moins apparente. Une telle section d'image peut représenter une région avec beaucoup de détail ou beaucoup de mouvement.

3.2.3 Application des particularités de la vision humaine à la compression

Plusieurs standards de compression d'image (JPEG (International Telecommunications Union, 2004)) et vidéo (MPEG-2, H.264, H.265 (ISO/IEC, 2009; Richardson, 2003; International Telecommunications Union, 2013b)) utilisent une représentation de couleur et de luminosité pour laquelle les valeurs de chrominance sont sous-échantillonnées. En effet, avec la représentation YUV420, les valeurs de chrominance U et V sont sous-échantillonnées d'un facteur de quatre en sous-échantillonnant par un facteur de deux horizontalement et verticalement. Ainsi, une valeur de U et une valeur de V sont utilisées pour quatre valeurs de luminance (Y), et ceci en conservant une bonne qualité visuelle.

La diminution de sensibilité aux hautes fréquences spatiales est utilisée dans un algorithme de compression proposé dans (Võ *et al.*, 2011b,a). Dans cet algorithme, les blocs sont classifiés en

trois classes : lisses, avec détails, aléatoires. Un facteur de quantification différent est appliqué pour chaque classe de bloc pour minimiser les distorsions, surtout pour les blocs lisses, qui représentent des régions d'image de basse fréquence.

3.3 Description du HACP-SBT

L'algorithme de HACP-SBT (Kim et Kyung, 2010) aborde plusieurs concepts intéressants. Il est présenté en deux parties, soit un prédicteur spatial le HACP (de l'anglais *Hierarchical Average Copy Prediction*) et un codage entropique simple à longueur fixe utilisant le SBT (de l'anglais *Significant Bit Truncation*).

L'algorithme de compression présenté est destiné à optimiser les transferts mémoires d'un codec H.264 dans un système embarqué. Dans certains systèmes, ceci est nécessaire pour respecter les limitations de bande passante imposées par les modules d'accès direct à la mémoire. Ceci est accompli en intégrant un module de compression d'image à ce module matériel. Pour ne pas augmenter le temps d'encodage, l'algorithme doit être de basse complexité.

L'autre aspect très intéressant est l'option de l'accès aléatoire aux données de l'image. Pour pouvoir accéder à certaines parties de l'image sans décoder l'image au complet, les dépendances de données doivent être restreintes à l'unité d'accès aléatoire. Dans ce cas, cette unité est un macrobloc 16×16 formé de quatre blocs 8×8. Ainsi, aucune prédiction temporelle n'est permise et toute prédiction spatiale doit être effectuée à l'intérieur du macrobloc. C'est pourquoi un prédicteur hiérarchique est utilisé puisque ce dernier ne présente aucune dépendance de données à l'extérieur du bloc 8×8 prédit.

3.3.1 Prédiction

La prédiction spatiale par HACP favorise les prédictions horizontales, car l'auteur explique que les images naturelles ont une plus grande corrélation horizontale que verticale.

3.3.1.1 Calculs des prédicteurs

Les calculs de prédicteur de chaque pixel dépendent de son emplacement à l'intérieur du bloc 8×8. Ceux-ci sont définis par les équations (3.5) à (3.9) à l'aide de la figure 3.4. Dans ces dernières, les positions de pixels sont indiquées par un indice horizontale (i) et un indice vertical (j). Ceux-ci ont comme origine le coin supérieur gauche et on les incrémente respectivement vers la droite et vers le bas. Ainsi, $p(0,0)$ représente la valeur du pixel situé dans le coin supérieur gauche, alors que $p(4,0)$ représente celle du cinquième pixel de la première ligne de la figure 3.4. Les divisions sont implémentées par des décalages vers la droite (un décalage d'une position pour une division par 2 et de deux positions pour une division par 4).

Cet algorithme présente cinq niveaux de profondeur de prédiction, illustrés dans la figure 3.5. Ces niveaux découlent directement de l'ordre des calculs des prédicteurs qu'il est très important de respecter. Cet ordre est (PM), (PMV_2 et PCV_2), (PMH_2 et PCH_2), (PMV_1 et PCV_1), (PMH_1 et PCH_1) pour un total de cinq niveaux. Ainsi, le niveau 1 doit être reconstruit avant que les prédicteurs du niveau 2 puissent être calculés, et ainsi de suite.

3.3.2 Codage

Le codage entropique par SBT est basé sur le fait que lorsqu'une prédiction est bien appliquée, les erreurs sont petites et centrées autour de zéro. Le tableau 3.2 montre que très peu d'information est contenue dans les bits les plus significatifs de petits nombres représentés sur huit bits. En fait, la seule information est le signe. Ces bits de signe présentent des redondances pour tous les nombres entre $[-64, 63]$ représentés sur huit bits. La méthode de codage par SBT prescrit simplement de tronquer ces bits de signe redondants.

Le processus de décodage doit connaitre le nombre de bits tronqués pour permettre la reconstruction des erreurs. Pour ce faire, les erreurs sont regroupées et un entête de quatre bits précède les erreurs codées du groupe. L'auteur nomme cet entête le BLH de l'anglais *Bit Length Header*. Celui-ci contient une valeur de zéro à huit qui indique le nombre de bits utilisés pour repré-

Figure 3.4 Prédicteurs spatiaux des groupes de SBT dans un bloc de pixels

$$\text{PM} = \frac{p(0,0) + p(0,4) + p(4,0) + p(4,4)}{4} \tag{3.5}$$

$$\text{PMV}_d(i,j) = \frac{p(i,j-d) + p(i,j+d)}{2}, \quad \textit{Prédiction par Moyenne Verticale} \tag{3.6}$$

$$\text{PCV}_d(i,j) = p(i,j-d), \quad \textit{Prédiction par Copie Verticale} \tag{3.7}$$

$$\text{PMH}_d(i,j) = \frac{p(i-d,j) + p(i+d,j)}{2}, \quad \textit{Prédiction par Moyenne Horizontale} \tag{3.8}$$

$$\text{PCH}_d(i,j) = p(i-d,j), \quad \textit{Prédiction par Copie Horizontale} \tag{3.9}$$

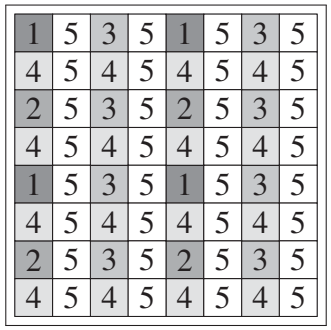


Figure 3.5 Profondeur de prédiction dans un bloc de pixel

Tableau 3.2 Codage par SBT

Erreur de prédiction en décimale signée	Erreur de prédiction en binaire (complément à 2)	Erreur de prédiction codée par SBT
-3	1111**1**101	**1**101
-2	1111**1**110	**1**110
-1	1111**1**111	**1**111
0	0000**0**000	**0**000
4	0000**0**100	**0**100
0	0000**0**000	**0**000
-5	1111**1**011	**1**011
2	0000**0**010	**0**010
	total : 64 bits	total : 36 bits

senter chaque erreur. Le tableau 3.3 indique l'intervalle d'erreurs représentables pour chaque valeur de BLH.

De plus, l'auteur montre que, statistiquement, les groupes de huit pixels offrent une meilleure compression. La figure 3.6 montre les neuf groupes de pixels pour un bloc 8×8 définis par l'auteur. Ces groupes sont nommés *groupe SBT* dans le reste de ce travail. Il est à noter que nous avons 9 groupes et que certains de ces groupes ne comptent que 4 pixels puisque des pixels évalués à des niveaux différents ne peuvent pas faire partie du même groupe.

Tableau 3.3 Valeur de BLH pour chaque intervalle d'erreur

Borne inférieure		Borne supérieure		BLH
0	00000000	0	00000000	0 (0000)
-1	11111111	0	00000000	1 (0001)
-2	11111110	1	00000001	2 (0010)
-4	11111100	3	00000011	3 (0011)
-8	11111000	7	00000111	4 (0100)
-16	11110000	15	00001111	5 (0101)
-32	11100000	31	00011111	6 (0110)
-64	11000000	63	00111111	7 (0111)
-128	10000000	127	01111111	8 (1000)

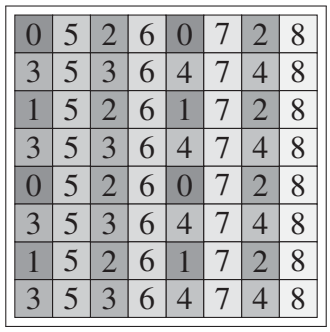


Figure 3.6 Groupe SBT

3.3.3 Avantages du HACP-SBT

Ce travail se base sur l'algorithme de HACP-SBT, car celui-ci présente plusieurs avantages que nous présentons dans les sous-sections qui suivent.

3.3.3.1 Faible profondeur de dépendance de données

La profondeur de prédiction, de seulement cinq niveaux pour un bloc 8×8, permet aux processeurs utilisant la technologie Opus d'exécuter plusieurs instructions en parallèle. Ceci a un effet bénéfique sur la vitesse d'exécution.

3.3.3.2 Calcul de prédicteur simple avec des instructions SIMD

Le calcul des prédicteurs nécessite très peu d'instructions par pixel. Les prédictions par PMV et PMH (éq. (3.6) et (3.8)) sont les plus complexes. Cependant, les processeurs Opus implémentent une instruction SIMD qui exécute la majorité de ces calculs. L'instruction *pavgub* (Octasic, 2011) s'exécute sur les octets de deux registres de 32 bits, calculant la moyenne pour chaque position d'octet. L'invocation de cette instruction utilise la syntaxe suivante `pavgub d, s1, s0` pour exécuter les équations (3.10) à (3.13) en un seul appel. La figure 3.7 décrit les opérandes de ces équations. Ainsi, les prédicteurs pour 4 pixels sont calculés

avec une seule instruction.

$$uD_3 = (uS0_3 + uS1_3 + 1) \gg 1 \tag{3.10}$$

$$uD_2 = (uS0_2 + uS1_2 + 1) \gg 1 \tag{3.11}$$

$$uD_1 = (uS0_1 + uS1_1 + 1) \gg 1 \tag{3.12}$$

$$uD_0 = (uS0_0 + uS1_0 + 1) \gg 1 \tag{3.13}$$

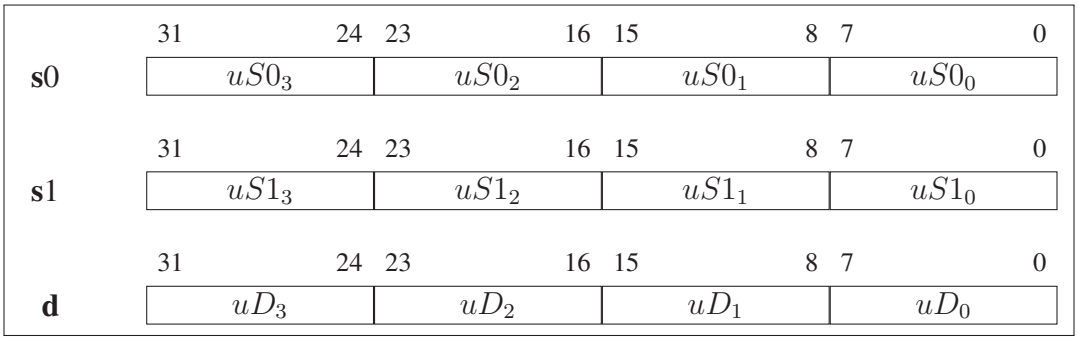


Figure 3.7 Description des paramètres de l'instruction SIMD *pavgub* du processeur Opus. Les trois paramètres (s0, s1 et d) sont des registres de 32 bits.
Adaptée de (Octasic, 2011)

3.3.3.3 Efficacité du prédicteur

Les résultats expérimentaux de la section 3.1.5.4 montrent que cet algorithme permet d'atteindre un taux de compression (ratio entropique) comparable et dans certains cas meilleur que les autres prédicteurs analysés.

3.3.3.4 Accès aléatoire

Le problème que tente de résoudre l'algorithme nécessite l'accès aléatoire aux données d'une image. Cela est principalement requis pour isoler les transferts mémoires. Ainsi, le concept

d'indépendance d'une section de données est une des bases de l'approche et un avantage indéniable.

3.3.4 Désavantages du HACP-SBT

Cet algorithme comporte quelques inconvénients. Ceux-ci sont résumés ci-dessous et seront résolus dans le chapitre 4.

3.3.4.1 Dépendances de données

L'algorithme présente un très faible nombre de niveaux de dépendance de données. En effet, on en compte seulement cinq (voir figure 3.5). Cependant, les deux premiers niveaux présentent deux inconvénients.

Premièrement, ces deux niveaux comptent seulement quatre pixels. Lors du codage par SBT, les groupes de 8 pixels sont plus efficaces en ce qui concerne le taux de compression.

Deuxièmement, la nature asynchrone du processeur Opus permet l'exécution en parallèle de seize instructions indépendantes. Cependant, puisque les niveaux 1 et 2 n'ont que quatre pixels, moins de seize instructions sont nécessaires pour effectuer la reconstruction et cela représente une perte d'efficacité. Ainsi, la dépendance de reconstruction entre les niveaux 1, 2 et 3 peut présenter un goulot d'étranglement pour l'exécution d'instructions en parallèle.

3.3.4.2 Algorithme de transfert mémoire

Le HACP-SBT, comme présenté dans la littérature, n'est pas un algorithme de compression utilisable pour la communication ou le stockage, et ce principalement parce que l'accès aléatoire aux données est basé sur les positions absolues des blocs dans un espace mémoire réservé pour une image. Ceci est dû au fait que l'objectif de l'algorithme original est d'optimiser le débit de transfert mémoire pour les unités d'accès aléatoires. Dans un contexte de communication et de stockage, l'objectif est différent. La simplicité de l'accès aléatoire doit laisser sa place à une

réelle compression et sérialisation des données. De plus, des entêtes doivent être définies pour permettre le transport de segments d'image.

3.3.4.3 Taux de compression variable

Le mode de prédiction et de codage présenté n'offre aucune limite minimale concernant le taux de compression. Le cas où le taux de compression pour un macrobloc est inférieur à un est traité comme un cas d'exception pour lequel le macrobloc non compressé est transmis en mémoire externe. Cette exception doit être signalée. Ainsi le taux de compression réel peut être inférieur à un. Ceci n'est pas acceptable pour ce projet. La méthode de compression proposée par ce travail doit permettre de garantir un taux de compression minimal.

3.4 Résumé

Dans ce chapitre, des techniques de compression sans et avec perte d'information ont été étudiées. L'exploration de la prédiction sans perte s'est faite par la comparaison de quatre prédicteurs spatiaux, dont l'algorithme HACP qui a été sélectionné comme le plus avantageux. L'analyse de la compression avec perte s'est concentrée sur la perception visuelle humaine des artefacts de compression.

CHAPITRE 4

AMÉLIORATIONS APPORTÉES AU PRÉDICTEUR HACP ET AU CODAGE PAR SBT

Ce chapitre présente les contributions de ce travail. Celles-ci prennent la forme d'améliorations à l'algorithme HACP-SBT pour qu'il puisse satisfaire les objectifs de ce projet.

L'objectif principal de cette recherche est de permettre le transfert de données vidéo haute-définition (HD) sur un lien Ethernet Gigabit en temps réel. Ce type de lien permet un débit théorique de 1Gb/s. La bande passante nécessaire pour un canal bidirectionnel de vidéo HD 1080p YUV 420, soit 1920 par 1080 pixels, à 30 images par secondes est d'environ 1.5Gb/s. Tout transfert sur un réseau par paquet nécessite des entêtes de paquets. En considérant les caractéristiques de la vidéo et le surdébit causé par le groupage des données par paquets, le taux de compression nécessaire reste similaire, soit d'environ 1.58. Le calcul de ce taux est détaillé dans la section 4.3.

Les améliorations algorithmiques présentées dans ce chapitre concernent trois aspects : la simplification de la hiérarchie de prédiction (section 4.1), l'adaptation de l'algorithme pour permettre la communication et le stockage (section 4.2), et l'adaptation de l'algorithme pour permettre une compression à taux fixe sans perte de qualité visuelle (section 4.3).

4.1 Simplification de la hiérarchie de prédiction du HACP

La hiérarchie de prédiction spatiale de pixels est basée sur un seul prédicteur. Ce dernier permet de reconstruire le premier niveau de prédiction hiérarchique. Chaque niveau subséquent utilise le niveau précédent pour calculer les nouveaux prédicteurs. Ainsi, tous les niveaux doivent être reconstruits dans l'ordre. Ceci provoque une dépendance de reconstruction entre chaque niveau aussi appelé profondeur du prédicteur.

4.1.1 Élimination d'un niveau de dépendance

La simplification de la hiérarchie de prédiction présentée dans cette sous-section vise à éliminer un de ces niveaux de dépendance. Ainsi, la méthode proposée sera constituée de quatre (4) niveaux de dépendance. La figure 4.1 montre que l'élimination de cette étape de dépendance est accomplie par l'unification des niveaux 1 et 2 de l'algorithme original. La section 3.3.4.1 explique pourquoi ces deux niveaux sont sélectionnés.

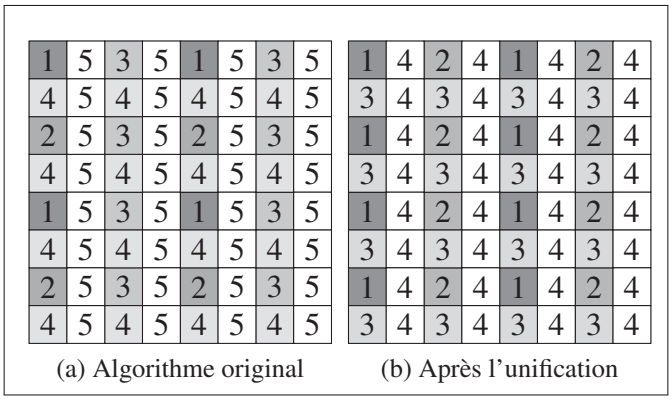


Figure 4.1 Simplification de la hiérarchie de prédiction du HACP par l'unification des niveaux de dépendance de donnée 1 et 2, diminuant la profondeur de prédiction. La hiérarchie originale est présentée dans (a) et la simplification est montrée dans (b).

4.1.2 Calcul du prédicteur du premier niveau

Cette unification nécessite un nouvel ensemble de prédicteurs pour le niveau 2, puisque le niveau 1 ne joue plus ce rôle.

Le prédicteur original du niveau 1 est la moyenne où la division est réalisée par décalage. Ce prédicteur, dénoté PM (éq. (4.1)), permet de répartir l'énergie de l'erreur sur les quatre pixels du niveau. Dans la majorité des cas, cette répartition permet de coder les pixels de ce niveau avec moins de bits, puisqu'elle permet d'obtenir le *BLH* (éq. (4.3)) minimal.

$$PM = \frac{A + B + C + D}{4} \tag{4.1}$$

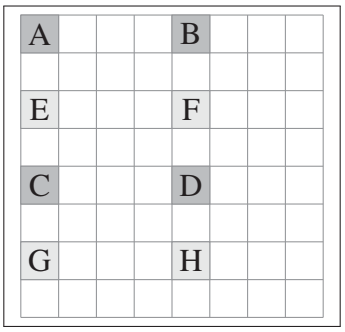


Figure 4.2 Les huit (8) pixels résultant de l'unification des niveaux de profondeur de prédiction 1 (foncé) et 2 (pâle)

Cependant, dans certains cas, répartir l'énergie de l'erreur de façon équivalente entre les pixels ne donne pas le *BPP* minimal. Ces cas sont principalement des cas limites, où l'erreur de prédiction du prédicteur *PM* est sur la limite des bornes de codage du SBT (tableau 3.3).

$$BPP(x) = \begin{cases} \lceil \log_2(|x|) \rceil + 1, & \text{si } x < 0 \\ \lceil \log_2(x+1) \rceil + 1, & \text{sinon.} \end{cases} \tag{4.2}$$

$$BLH(w, x, y, z, p) = \max\left(BPP(x-p)\right), \text{ pour } x \in \{w, x, y, z\} \tag{4.3}$$

Par exemple, pour le cas où les valeurs de (A, B, C, D) sont $(129, 161, 192, 160)$, le prédicteur *PM* sera de 160. L'erreur de prédiction est ainsi $(-31, 1, 32, 0)$. Par conséquent, les équations (4.2) et (4.3) montre que sept bits sont nécessaires pour coder la plus grande erreur de ce groupe de pixel puisque le $BPP(32)$ est 7. Cependant, si le prédicteur avait été 161, les erreurs de prédiction auraient été $(-32, 0, 31, -1)$, ce qui aurait permis de coder les erreurs avec six bits puisque le $BPP(-32)$ est 6. Ainsi, le prédicteur optimal minimisant le *BLH* est présenté à l'équation (4.4).

$$P_{optimal} = \arg\min_{p} \left[BLH(A, B, C, D, p)\right], \text{ pour } p \in \{0...255\} \tag{4.4}$$

Cependant, faute de temps, les 256 prédicteurs possibles ne peuvent pas être testés. Par conséquent, un sous-ensemble composé des pixels eux-mêmes et de leur moyenne est utilisé pour tenter de minimiser l'erreur. Ce prédicteur est nommé prédicteur minimisant l'erreur (*PME*) dans le reste de ce document (éq. (4.5)).

$$\textit{PME} = \arg\min_{p} \left[BLH(A, B, C, D, p)\right], \text{ pour } p \in \{A, B, C, D, \textit{PM}\} \tag{4.5}$$

4.1.3 Regroupement de pixels pour l'unification

La figure 4.2 montre les huit pixels des niveaux 1 et 2 de l'algorithme original. Les groupes de pixels originaux sont (A, B, C, D) et (E, F, G, H). L'algorithme original favorise la prédiction horizontale puisque les images naturelles ont une corrélation horizontale plus grande que verticale. Par ailleurs, la corrélation est plus grande entre les pixels ayant une faible distance spatiale (Wang *et al.*, 2002). Ainsi, les distances entre les pixels peuvent être diminuées en regroupant les pixels verticalement pour créer les groupes (A, E, C, G) et (B, F, D, H).

Cependant, ces deux méthodes nécessitent deux prédicteurs. En utilisant un groupe contenant plus de pixels, l'efficacité de prédiction est réduite en échange d'une diminution des données superflues, c.-à-d. le prédicteur du deuxième niveau. Les différentes possibilités de regroupements de pixels proposées et leurs avantages sont présentés au tableau 4.1.

Tableau 4.1 Avantage des regroupements de pixels

Regroupement de pixels	Groupe	Avantage
Horizontal	(A, B, C, D) et (E, F, G, H)	Plus grande corrélation horizontale. (regroupement original)
Vertical	(A, E, C, G) et (B, F, D, H)	Les pixels sont spatialement plus près.
Groupe	(A, E, C, G, B, F, D, H)	Seulement un prédicteur est nécessaire.

4.1.4 Comparaison des méthodes proposées

Dans cette sous-section, nous comparons les performances de l'algorithme original avec celles des prédicteurs *PM* et *PME* utilisant les divers regroupements de pixels proposés au tableau 4.1. Nous assignons un nom d'algorithme à chaque combinaison de prédicteur et de regroupement de pixels.

Le tableau 4.2 montre les résultats expérimentaux comparant les méthodes d'unification proposées pour chaque algorithme. On y retrouve, pour chaque algorithme, le taux de compression moyen et le nombre d'opérations. La procédure de test est décrite dans l'annexe IV.

Tableau 4.2 Performances des méthodes d'unification

Nom de l'algorithme	Calcul du prédicteur	Regroupement de pixels	Taux de compression moyen	Nombre d'opérations
MG	*PM*	Groupe	1.7262	8
ORIGINAL	*PM* + *PMV*	Horizontal	1.7176	6
MV	*PM*	Vertical	1.6886	8
MH	*PM*	Horizontal	1.6849	8
EG	*PME*	Groupe	1.7268	224 - 296
EV	*PME*	Vertical	1.6929	224 - 296
EH	*PME*	Horizontal	1.6895	224 - 296

Ces résultats démontrent que le cout supplémentaire d'un deuxième prédicteur rend les algorithmes *moyenne verticale* (dénoté MV), *moyenne horizontale* (dénoté MH), *erreur verticale* (dénoté EV), et *erreur horizontale* (dénoté EH) légèrement moins performants pour la compression que les algorithmes groupés, soit *moyenne groupée* (dénoté MG) et *erreur groupée* (dénoté EG). Ainsi, le nombre d'opérations nécessaires pour chaque algorithme de prédiction devient le facteur décisif. Le calcul du prédicteur avec le *PME* nécessite environ 30 fois plus d'opérations que *PM*. Il est ainsi possible de conclure que la méthode d'unification par MG est supérieure aux autres méthodes essayées.

Une autre approche possible aurait été de signaler le mode de prédiction au décodeur pour tirer profit des avantages de chaque mode. Cependant, ceci aurait nécessité une recherche pour

trouver le prédicteur à utiliser. Ces calculs supplémentaires ajoutent trop de complexité à l'algorithme. C'est pourquoi un seul mode de prédiction est adopté.

4.2 Adaptation de l'HACP-SBT pour la communication

Le concept d'accès aléatoire demeure possible, malgré le fait que, comme discuté au chapitre précédent, la méthode originale ne s'applique pas à la transmission de données. En effet, il s'agit d'utiliser une syntaxe qui définit une unité d'accès aléatoire et sa transmission.

La stratégie de division de l'image en unités d'accès aléatoire est définie dans la sous-section 4.2.1. Ensuite, la définition d'une syntaxe de codage permettant la compression pour le transfert par paquets est présenté dans la sous-section 4.2.2.

4.2.1 Division de l'image

Le principe d'accès aléatoire est très intéressant puisqu'il permet la transmission et la reconstruction désordonnée des unités d'accès aléatoire. Ceci facilite la division du travail d'encodage et de décodage sur plusieurs unités de travail d'un processeur multicœur, puisqu'aucune sérialisation des données n'est nécessaire à l'extérieur de l'unité d'accès aléatoire. De plus, l'indépendance des données augmente la résilience aux erreurs en empêchant la désynchronisation du flux de données compressé lors de la perte d'information et ainsi rend possible la dissimulation des erreurs.

Chacune de ces unités représente une section d'image. Plusieurs algorithmes de compression transmettent séquentiellement des macroblocs, soit une région d'image de 16×16 pixels. Ainsi, il est avantageux de suivre cette approche pour faciliter le passage d'information entre un encodeur et un décodeur utilisant des syntaxes différentes. Un macrobloc, dans le domaine de couleur sous-échantillonné qu'est YUV 420, est composé d'un bloc de 16×16 de luminance et de deux blocs 8×8 de chrominance. Ceci est illustré à la figure 4.3.

Une petite unité d'accès aléatoire est nécessaire pour l'application originale. Cependant, dans le contexte de transport de données, chaque unité doit être sérialisée et identifiée par un entête.

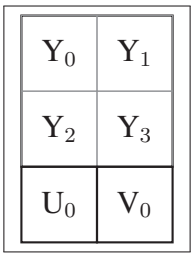


Figure 4.3 Un macrobloc dans le domaine de couleur YUV 420

Ainsi des unités plus larges sont avantageuses pour augmenter le ratio entre la taille de la charge utile et la taille totale transportée. De plus, puisque l'algorithme de prédiction est seulement défini pour des blocs 8×8, une unité doit être composée d'un nombre entier de macroblocs pour prendre en considération les blocs de chrominance.

Sur un réseau Ethernet(IEEE 802.3 working group, 2012), l'unité de base est le paquet. Il est ainsi souhaitable de définir le paquet comme l'unité d'accès aléatoire. Pour ce faire, le nombre de macroblocs dans un paquet doit être constant. De plus, pour permettre la transmission, une syntaxe définissant les entêtes et la sérialisation des données est nécessaire. Il faut aussi définir une stratégie d'encapsulation s'adaptant à la couche de transport utilisée pour la transmission.

Le protocole RTP (Schulzrinne *et al.*, 2003) est communément utilisé pour transporter des données audiovisuelles sur un lien Ethernet en utilisant le protocole UDP (Postel, 1980) encapsulé dans le protocole Internet (IP) (Postel, 1981). Ainsi, pour transmettre les unités sur un réseau Ethernet, elles doivent être encapsulées dans la partie de charge utile d'un paquet RTP. Comme montré à la figure 4.4, la charge utile d'un paquet RTP transmis par Ethernet est de 1460 octets (Hornig, 1984).

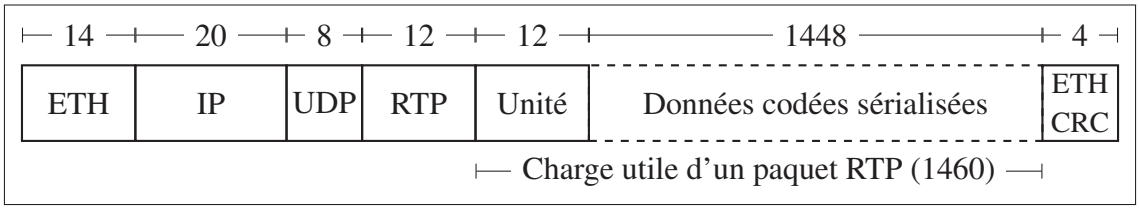


Figure 4.4 Séquence d'entêtes précédant une unité codée, lorsqu'elle est encapsulée dans le protocole RTP sur un lien Ethernet, les tailles sont présentées en octets

Ce protocole définit un entête dans lequel toutes les informations temporelles nécessaires sont présentes. Cette notion de temps est importante pour identifier à quelle image appartient un paquet. Cependant, cette information n'est pas suffisante pour permettre une transmission désordonnée des unités à l'intérieur d'une image. En effet, la charge utile du paquet RTP doit débuter par un entête supplémentaire, soit l'entête d'unité qui précisera les paramètres de compression, les informations de recadrage de l'image et autres champs pertinents. Les champs et les détails de cet entête seront présentés dans la section 4.2.2.1. Notons seulement pour l'instant que la taille de cet entête est de 12 octets (ou 96 bits).

Considérant ces hypothèses, les équations suivantes résument les contraintes du système à concevoir au niveau de la mise en paquets de l'information :

$$R_{bps} = N_C \times F \times S\left(T_{\mathrm{U}} + H_{\mathrm{ERTP}}\right) = N_C \times F \times S\left(\frac{T_{\mathrm{I}}}{\alpha S} + H_{\mathrm{ERTP}}\right) \leq R_{\mathrm{max}} \tag{4.6}$$

$$\frac{T_{\mathrm{I}}}{\alpha S} + H_{\mathrm{RTP}} \leq T_{\mathrm{IPmax}} \tag{4.7}$$

$$N_{\mathrm{MB}} = \frac{T_{\mathrm{I}}}{S \times T_{\mathrm{MB}}} \in \mathcal{Z} \tag{4.8}$$

où N_C est le nombre de canaux, F est le taux de trames, T_{U} la taille de la charge utile d'un paquet RTP (taille d'unité) qui est considérée constante (voir figure 4.4), H_{U} l'entête d'unité, T_{I} la taille de l'image non compressée, α le taux de compression, S le nombre de segments (c.-à-d. de paquets ou d'unités) utilisé pour transmettre l'image, H_{ERTP} la taille l'entête RTP à Ethernet (c.-à-d. l'entête complète jusqu'à la charge utile), H_{RTP} la taille l'entête RTP (c.-à-d. l'entête de la charge utile jusqu'au niveau IP inclusivement à la figure 4.4), R_{max} le débit maximum du lien de communication, R_{bps} le débit effectif de communication, T_{IPmax} la taille maximale d'un paquet IP, T_{MB} la taille d'un macrobloc, et N_{MB} le nombre de macroblocs par segment.

Dans notre cadre applicatif, nous avons $H_{\text{ERTP}} = 58$ octets, $H_{\text{RTP}} = 40$ octets, $H_{\text{U}} = 12$ octets, $R_{\text{max}} = 1$ Gbps, $T_{\text{IPmax}} = 1500$ octets, $T_{\text{MB}} = 384$ octets (car $16{\times}16{+}8{\times}8{+}8{\times}8 = 384$). Pour une séquence HD 1080p30 transmise de manière bidirectionnelle, nous avons $N_C = 2$, $F = 30$ fps, l'image est composée $120{\times}68$ macroblocs et ainsi $T_{\text{I}} = 3\,133\,440$ octets ($120{\times}16{\times}68{\times}16{\times}1.5$, car chaque pixel d'une image YUV420 utilise 1.5 octets).

L'équation (4.6) exprime la contrainte sur le débit Ethernet de 1Gbps (Frazier, 1998; IEEE 802.3 Working Group, 1999). En effet, nous transmettons des segments utiles de taille $\frac{T_{\text{I}}}{S}$ compressés par un facteur α qui auront chacun un entête de taille H_{ERTP}. Nous transmettons S de ces segments par trame et F trames par seconde pour chacune des N_C directions.

L'équation (4.7) exprime la contrainte de la taille maximale d'un paquet IP. En effet, chaque paquet IP avec contenu compressé sera de taille $\frac{T_{\text{I}}}{\alpha S}$ avec un entête de taille H_{RTP} et le tout devra être inférieur à $T_{\text{IPmax}} = 1500$ octets. Si le nombre de segments est fixé, le taux de compression α minimum nécessaire pour respecter cette contrainte, dénoté α_{min}, est donné par l'équation (4.9).

$$\alpha \geq \frac{T_{\text{I}}}{S \times (T_{\text{IPmax}} - H_{\text{RTP}})} \Rightarrow \alpha_{\text{min}} \geq \frac{T_{\text{I}}}{S \times (T_{\text{IPmax}} - H_{\text{RTP}})} \tag{4.9}$$

Finalement, l'équation (4.8) contraint le transfert à un nombre entier de macroblocs pour chaque segment ou unité. Ceci est nécessaire pour pouvoir déterminer l'emplacement de ce dernier dans une image qu'avec son indice. Le tableau 4.3 montre qu'en fixant N_{MB} les valeurs de α_{min} et R_{bps} associées peuvent être calculées.

Plus le nombre de segments S sera faible, et plus cela imposera un haut taux de compression. Donc, on cherche le plus grand nombre de segments sans toutefois que le surdébit devienne tellement important que l'on dépasse la contrainte du 1 Gbps. Nous pouvons voir qu'utiliser 1360 segments rencontre toutes les contraintes.

En résumé, pour satisfaire nos contraintes, on choisira $S = 1360$. Cela entrainera que chaque unité contiendra $N_{\text{MB}} = 6$ macroblocs. Par conséquent, de (4.9), on aura $\alpha \geq 1.58$ et le débit

Tableau 4.3 Huit premières valeurs de N_{MB}

N_{MB}	S (éq. (4.8))	α_{min} (éq. (4.9))	R_{bps} (éq. (4.6))	R_{Gbps}
1	8160.000	0.263	5945702400	5.946
2	4080.000	0.526	2972851200	2.973
3	2720.000	0.789	1981900800	1.982
4	2040.000	1.052	1486425600	1.486
5	1632.000	1.315	1189140480	1.189
6	1360.000	1.578	990950400	0.991
7	1165.714	1.841	849385849	0.849
8	1020.000	2.104	743212800	0.743

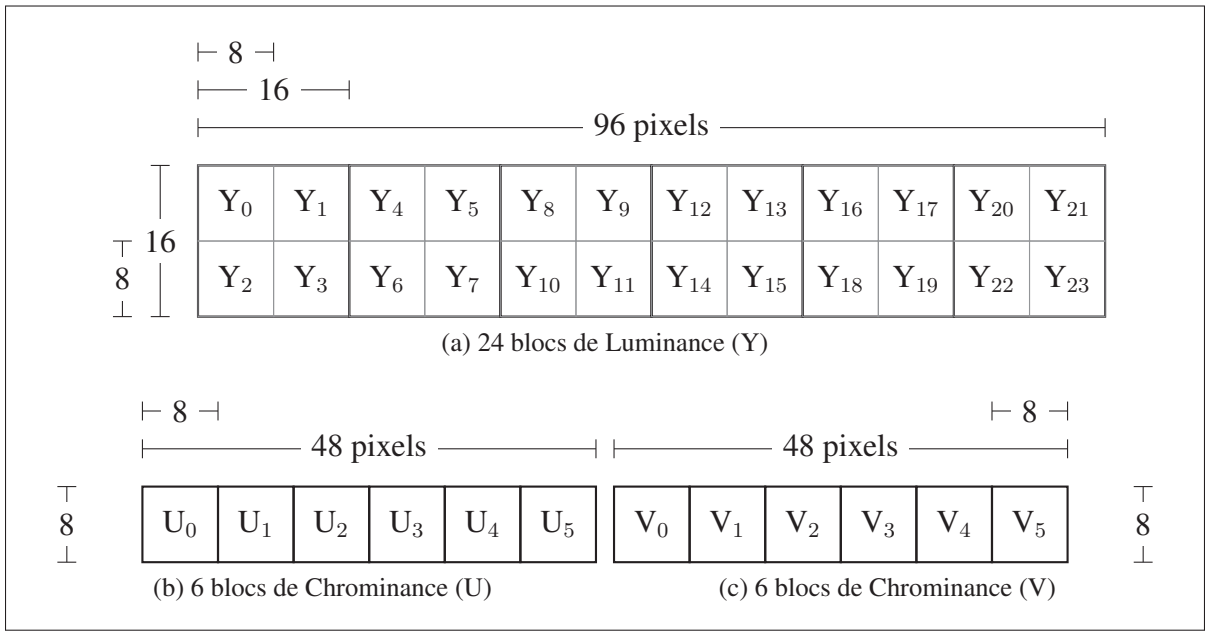
sera de $R_{\text{bps}} = 990\,950\,400$ bps ($R_{\text{Gbps}} = 0.991$ Gbps), nous laissant une marge de manœuvre pour compenser pour le fait que le 1 Gbps est plutôt théorique et qu'en réalité le débit réel est légèrement inférieur.

Ainsi, comme présentée dans la figure 4.5, l'unité d'accès aléatoire est composée d'une section d'image de 6 macroblocs pour former un bloc de 96 par 16 pixels. Dorénavant, le terme *unité* est utilisé pour identifier ce regroupement de pixels.

4.2.2 Syntaxe de codage

Après l'entête d'unité de la figure 4.4, qui sera présenté dans la prochaine sous-section, la charge utile du paquet RTP est remplie de l'information pour les blocs de pixels 8×8 codés. Chacun de ces blocs de pixels est composé d'un prédicteur *PM* de huit bits suivi des groupes de SBT. Syntaxiquement, un groupe de SBT est composé de 4 bits pour le *BLH* et de l'erreur de prédiction de huit pixels, ceci est illustré à la figure 4.6. Le nombre de bits avec lesquels les erreurs de prédiction sont codées est défini par le *BLH*.

En bref, une unité est composée de son entête suivi de six macroblocs, chacun composé de six blocs. Les équations (4.10) et (4.11) expriment la taille totale d'une unité.



(a) 24 blocs de Luminance (Y)

(b) 6 blocs de Chrominance (U)

(c) 6 blocs de Chrominance (V)

Figure 4.5 Section de l'image représentée par une unité d'accès aléatoire, dans (a) les pixels dans le plan de la luminance, dans (b) et (c), les pixels dans le plan de la chrominance. Les tailles sont présentées en octets

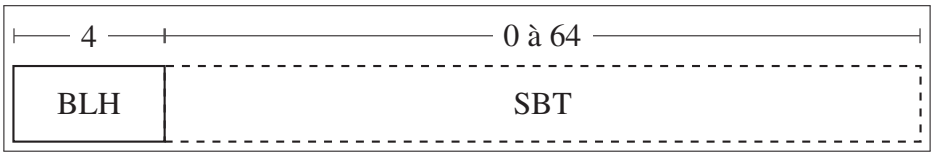


Figure 4.6 Un groupe de SBT avec les tailles présentées en bits

$$\text{Taille total d'une unité} = H_{\text{U}} + \sum_{\text{MB}=0}^{N_{\text{MB}}-1} \left(\sum_{\text{bloc}=0}^{N_{\text{B}}-1} \left(\sum_{\text{groupe}=0}^{N_{\text{G}}-1} (BPP_{\text{groupe}} \times N_{\text{G}} + H_{\text{BLH}}) + N_p \right) \right) \tag{4.10}$$

où H_{U} représente l'entête d'unité, H_{BLH} la taille de l'entête *BLH*, N_{MB} le nombre de macroblocs dans une unité de transport, N_{B} le nombre de blocs dans un macrobloc, N_{G} le nombre de pixels dans un groupe, et N_p la taille du prédicteur *PM*. Dans notre cadre applicatif, nous avons :

$$\text{Taille total d'une unité}_{\text{(en bits)}} = 96 + \sum_{\text{MB}=0}^{5} \left(\sum_{\text{bloc}=0}^{5} \left(\sum_{\text{groupe}=0}^{7} (BPP_{\text{groupe}} \times 8 + 4) + 8 \right) \right) \tag{4.11}$$

Puisque les erreurs de prédiction peuvent être codées avec de zéro à huit bits, l'équation (4.11) montre que la taille totale d'une unité peut varier entre 192 et 2496 octets (entre 1536 et $19\,968$ bits). La taille maximale est plus grande que la charge utile maximale, donc il est important de définir une syntaxe permettant de représenter ces cas limites. Le champ **Type** de l'entête d'unité, que l'on définira à la prochaine sous-section, permettra au cas limite d'être codé sur plus d'un paquet.

La stratégie permettant de coder une unité sur deux paquets est basée sur la profondeur du prédicteur. Cette dernière permet de garder la plus grande indépendance des données. Tel qu'illustré à la figure 4.1b, les colonnes impaires du bloc 8×8 forment le dernier niveau de profondeur. Ce dernier niveau a la propriété particulière qu'aucun autre niveau ne dépend du résultat de sa reconstruction.

Cette particularité est utilisée pour la compression avec perte définie dans la section 4.3. Dans ce cas, si les erreurs de prédiction de ce niveau ne sont pas codées ou codées partiellement, les erreurs de reconstruction ne se propageront pas aux autres pixels.

Donc, la charge utile maximale est respectée, pour le pire cas de compression, en codant les colonnes paires dans un premier paquet ayant une taille maximale de 1272 octets et les colonnes impaires dans un deuxième paquet. Ce deuxième paquet a une taille maximale de 1236 octets. Ainsi, les erreurs de prédiction sont sérialisées pour permettre cette séparation. La figure 4.7 montre que les colonnes paires sont sérialisées en premier, suivies ensuite par les colonnes impaires de tous les blocs de l'unité.

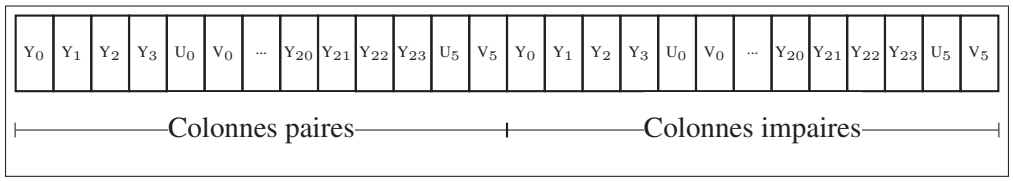


Figure 4.7 Sérialisation des blocs de pixels d'une unité d'accès aléatoire

4.2.2.1 Entête d'unité

Cette sous-section décrit les champs nécessaires pour la reconstruction désordonnée d'une unité. Ces champs sont identifiés dans la figure 4.8 et le tableau 4.4 explique leurs significations. De plus, les paragraphes 4.2.2.1.1 à 4.2.2.1.3 décrivent les concepts importants de l'entête.

Certains champ de l'entête sont reliés au codage avec perte. Ce principe est expliqué dans la section 4.3. Ces champ sont identifiés avec une police italique dans le tableau 4.4.

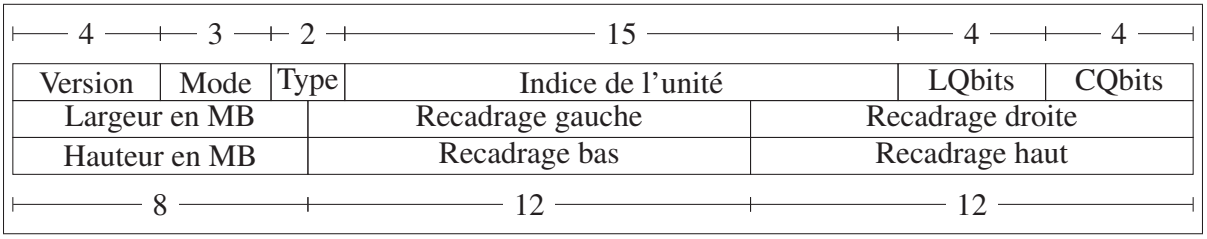


Figure 4.8 Entête d'unité de 96 bits (12 octets) avec les tailles présentées en bits

4.2.2.1.1 Information spatiale

Pour permettre sa reconstruction désordonnée, l'information spatio-temporelle de chaque unité doit être connue. L'entête RTP contient l'information temporelle qui permet d'identifier l'image à laquelle l'unité appartient. Étant donné que le nombre de macroblocs dans une unité est constant, la seule information spatiale nécessaire pour identifier l'endroit dans l'image que l'unité représente est l'indice du premier macrobloc de l'unité. Cette information est dans le champ **Indice** de 15 bits, ce qui permet de représenter des images contenant jusqu'à 32768 macroblocs. Ceci représente des résolutions jusqu'à 4K (Acharya et Petrin, 2012).

4.2.2.1.2 Paramètre de quantification

Le champ **Mode** définit les stratégies de codage qui permettent à une unité de respecter la taille d'un paquet. Certaines valeurs de ce champ nécessitent des paramètres supplémentaires pour permettre un codage avec perte. Cette information est contenue dans les champs **LQbits** et **CQbits**. La sous-section 4.3 explique leur utilisation.

4.2.2.1.3 Taille et recadrage

La taille ou résolution de l'image est définie par la largeur et la hauteur du plan de luminance. Cependant, certains standards de compression, comme le populaire standard H.264, définissent la résolution en nombre de macroblocs. Ceci est logique puisque le macrobloc est l'unité de base de division de l'image. Toutefois, il existe quelques résolutions standards qui ne sont pas des multiples entiers de macroblocs, comme la haute définition 1080p. Ainsi, des paramètres de recadrage sont nécessaires pour indiquer quelle partie des macroblocs transmis correspond à l'image. Pour simplifier la traduction d'un flux de données au standard H.264 (International Telecommunications Union, 2013a), le nombre de bits utilisés pour représenter chaque paramètre est le même que dans l'entête de ce standard.

4.3 Extension du codage par SBT pour le codage avec perte

Jusqu'à présent, l'algorithme a été étendu pour l'adapter à une syntaxe de transmission dans la section 4.2 et pour diminuer la profondeur du prédicteur dans la section 4.1. Malgré ces améliorations, la taille maximale théorique dépasse toujours l'objectif concernant le taux de compression. Cette section explique comment l'ajout de la troncature des bits les moins significatifs permet d'atteindre un taux de compression garanti de 1.58.

4.3.1 Troncature des bits les moins significatifs

Le codage par SBT définit le nombre de bits les plus significatifs pouvant être tronqués et reconstruits par extension de signe sans perte de données. Théoriquement, cette technique ne

Tableau 4.4 Description des champs de l'entête d'unité

Nom du champ	Taille	Description
Version	4 bits	Version de l'entête. Pour l'instant, seulement la version 0 est définie.
Mode	3 bits	*Mode de codage :* 0 : Sans perte d'information. 1 : Avec perte rapide (meilleure vitesse). 2 : Avec perte calculée (meilleure qualité).
Type	2 bits	Indique si le paquet est le premier (1) ou le deuxième (2) si l'unité est codée avec deux paquets. Si l'unité est codée avec un seul paquet, ce champ a la valeur zéro. 0 : Un seul paquet. 1 : Premier paquet de deux. 2 : Deuxième paquet de deux.
Indice de l'unité	15 bits	Indice du premier macrobloc de l'unité dans l'image, débutant par l'indice 0.
LQbits	4 bits	*Facteur de quantification. Représente le nombre de bits moins significatifs des erreurs de prédiction qui ne sont pas transmises pour les pixels de luminance.*
CQbits	4 bits	*Facteur de quantification. Représente le nombre de bits moins significatifs des erreurs de prédiction qui ne sont pas transmises pour les pixels de chrominance.*
Largeur en MB	8 bits	Indique la largeur de l'image en macroblocs.
Recadrage gauche	12 bits	Indique le nombre pixel à partir de la gauche de l'image qui sont rejeté lors du recadrage.
Recadrage droite	12 bits	Indique le nombre de pixels à partir de la droite de l'image qui sont rejetés lors du recadrage.
Hauteur en MB	8 bits	Indique la hauteur de l'image en macroblocs.
Recadrage bas	12 bits	Indique le nombre de pixels à partir du bas de l'image qui sont rejetés lors du recadrage.
Recadrage haut	12 bits	Indique le nombre de pixels à partir du haut de l'image qui sont rejetés lors du recadrage.

permet pas d'établir un taux de compression fixe puisque jusqu'à huit bits peuvent être nécessaires pour représenter l'erreur. Pour pouvoir garantir un taux de compression, cette technique de codage doit être étendue pour permettre un codage avec perte de données.

L'algorithme original tire profit du fait qu'aucune information n'est contenue dans les bits de signes redondants. Le bit suivant les bits de signe contient le plus d'information, par la suite, les bits contiennent de moins en moins d'information jusqu'au bit le moins significatif. Lorsque

la compression due aux bits de signes redondants n'est pas suffisante, la troncature des bits les moins significatifs permet d'augmenter la compression. Ces bits sont tronqués en ordre de signification croissante. Ainsi, la perte d'information affecte moins la qualité visuelle. Cette méthode est une façon de décrire une quantification des données.

4.3.2 Conservation de l'information du résultat de prédiction sur huit bits

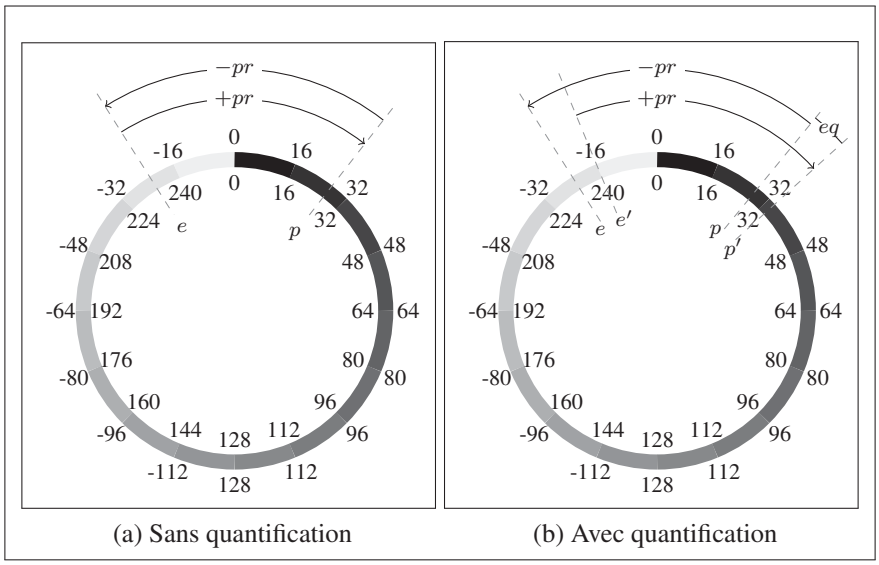
Puisque le pixel et le prédicteur sont des entiers entre 0 et 255, ils peuvent être représentés avec huit bits. Cependant, l'erreur de prédiction peut prendre des valeurs de -255 à 255, donc neuf bits sont nécessaires pour représenter sa pleine plage. Cependant, Kim et Kyung (2010) mentionnent qu'il est possible de représenter l'erreur de prédiction avec huit bits. Les opérations d'addition et de soustraction dans le domaine modulo huit équivalents à une simple transition sur l'anneau $\mathbb{Z}/2^8\mathbb{Z}$. La figure 4.9(a) utilise les équations (4.12) et (4.13) pour illustrer ce concept avec une valeur de pixel (p) de 27 et un prédicteur (pr) de 50, ce qui crée une erreur de prédiction (e) de -23. Toutefois, le e transmis sur 8 bits devient 233. Ensuite $p = e + pr = 233 + 50 = 283$ devient 27 sur 8 bits. Cela montre le principe de réduction du nombre de bits en effectuant les opérations dans le domaine du modulo.

$$e = p - pr \tag{4.12}$$

$$p = e + pr \tag{4.13}$$

4.3.2.1 Erreur de quantification de l'erreur de prédiction

Nous venons de voir que la relation $p = p'$ est conservée en représentant l'erreur de prédiction (e) sur seulement huit bits. Cette relation n'est cependant pas conservée lorsque l'erreur est quantifiée (e'). Ceci est dû à l'erreur de reconstruction inhérente à la quantification eq. La figure 4.9(b) montre que le pixel reconstruit (p') et différent du pixel original (p) avec les équations



(a) Sans quantification (b) Avec quantification

Figure 4.9 Illustration de la prédiction et reconstruction d'un pixel sur l'anneau $\mathbb{Z}/2^8\mathbb{Z}$ sans (a) et avec (b) quantification de l'erreur de prédiction

(4.14) et (4.15) (où $\lfloor x \rfloor$ signifie la valeur plancher de x).

$$e' = \lfloor (e/2^q) \rfloor \times 2^q \tag{4.14}$$

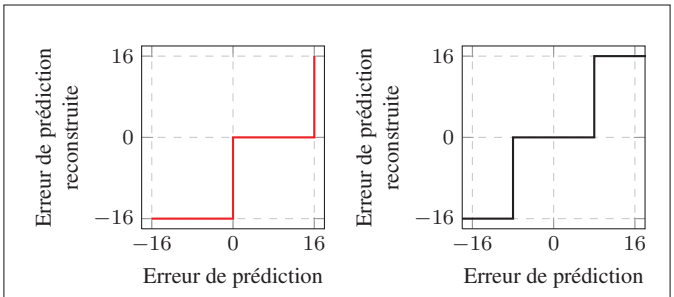
$$p' = e' + pr \tag{4.15}$$

Une technique pour diminuer l'erreur de quantification est d'arrondir le résultat de la division avant de garder seulement la partie entière en utilisant l'équation (4.16) au lieu de l'équation (4.14).

$$e' = \lfloor ((e + 2^{q-1})/2^q) \rfloor \times 2^q \tag{4.16}$$

Ceci recentre le pixel reconstruit dans la plage de quantification. Les graphiques de la figure 4.10 illustrent ceci.

Grâce à l'arrondissement, la plus grande erreur de quantification passe de 2^q à $2^q/2$. Ce qui fait que l'erreur absolue maximale est de 3.125% avec un facteur de quantification de 16 ($q = 4$).



(a) Quantification sans arrondissement

(b) Quantification avec arrondissement

Figure 4.10 La reconstruction d'une erreur de prédiction suite à la quantification avec un facteur de quantification de 16 ($q = 4$), sans arrondissement (a) et avec arrondissement (b)

4.3.2.2 Problème de signe

Toutefois, puisque la quantification est appliquée à une erreur de prédiction et non au pixel directement, il existe certains cas où l'erreur peut dépasser $2^q/2$ et atteindre presque 2^8.

Lorsqu'un nombre quantifié fait partie d'une équation mathématique, le résultat n'est pas limité à des multiples de l'erreur de quantification. Seulement l'erreur est limitée à des multiples de l'erreur de quantification. Aucune explication ou solution à ce problème n'a été trouvée dans la littérature.

Concrètement, dans la figure 4.11, l'erreur de prédiction reconstruite (e') est limitée à des multiples de 2^q mais le pixel reconstruit p' ne l'est pas. L'effet de la quantification sur pixel p, soit p' est : $p - (2^q/2) \leq p' \leq p + (2^q/2)$. Ceci pose un problème lorsque $|p| < (2^q/2)$ car l'erreur de quantification peut faire changer le signe de p'.

Comme montrent les nombres à l'extérieur de l'anneau $\mathbb{Z}/2^8\mathbb{Z}$ dans la figure 4.11, un petit nombre négatif dans ce domaine est à une distance de l'erreur de quantification maximale $(2^q/2)$ d'un petit nombre positif. Cependant, le pixel reconstruit ne prend pas de valeur négative comme le montrent les valeurs de pixels tout juste au-dessus de l'anneau $\mathbb{Z}/2^8\mathbb{Z}$. Dans ce cas-ci, la valeur du pixel (p) est de 253 soit -2 en représentation signée et la valeur du pixel reconstruit est de $+2$. Mathématiquement, leur distance euclidienne est de quatre. Cependant, le pixel

original est blanc (253) alors que le pixel reconstruit est noir (2). Ceci a un effet catastrophique que la qualité de la reconstruction des images.

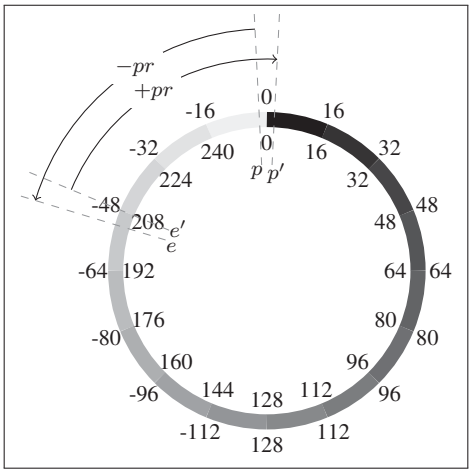


Figure 4.11 Anneau $\mathbb{Z}/2^8\mathbb{Z}$ avec erreur de prédiction quantifiée illustrant problème de signe

Pour éviter ce problème, les valeurs de pixels pouvant créer un problème sont remplacées par les pixels les plus près qui ne causent aucun problème de signe. En traitant le pixel comme une valeur de huit bits signés, le remplacement de valeur a lieu si la valeur absolue du pixel est inférieure à l'erreur de quantification. Dans ce cas, la valeur du pixel est remplacée par la valeur de l'erreur de quantification avec le signe original du pixel.

4.3.3 Sélection des groupes à tronquer

L'objectif secondaire de compression est que les pertes d'information affectent le moins possible la qualité de l'image. Pour ce faire, les groupes de SBT dans lesquels les erreurs de prédiction sont quantifiées doivent être judicieusement choisis. Deux facteurs sont utilisés pour choisir les groupes où la perte d'information sera moins perceptible. Ces deux facteurs sont basés sur le système visuel humain décrit dans la section 3.2. Ils sont la sensibilité aux couleurs et aux hautes fréquences spatiales de l'oeil humain.

4.3.3.1 Chrominance

Pour limiter l'impact de la reconstruction imparfaite, les groupes de SBT dans les blocs de chrominance sont quantifiés en premier puisque l'œil humain est moins sensible à ce type d'information.

4.3.3.2 Haute fréquence spatiale

L'oeil humain est aussi moins sensible aux artefacts de compression dans des zones d'image avec un contenu élevé de hautes fréquences spatiales. La détection des groupes de SBT avec cette caractéristique est implicite au type de prédiction utilisé. La prédiction spatiale de chaque pixel est basée sur les voisins de chacun d'eux, donc une grande erreur de prédiction est une bonne indication d'une zone d'image contenant plusieurs hautes fréquences spatiales.

4.3.4 Sélection du nombre de bits à tronquer

La figure 4.12 montre une évaluation théorique de la proportion de l'erreur de quantification maximale. Celle-ci est basée sur la quantité d'information perdue avec un nombre de bits tronqués q par rapport à la pleine plage des valeurs possibles de pixel, soit $\frac{2^q/2}{2^8}$. Étant donné le comportement exponentiel de cette fonction, chaque bit peu significatif tronqué double l'erreur maximale. Un compromis intéressant est atteint avec la troncature d'un nombre de bits inférieur ou égal à quatre. Ceci limite l'erreur maximale à 3.125%.

Ainsi, si le nombre de bits utilisés pour coder l'erreur varie entre zéro et huit, l'équation (4.11) indique que la taille des unités varie entre 192 et 1344 octets, ce qui satisfait le critère de la taille maximale de la charge utile d'un paquet RTP.

Cependant, comme mentionné dans la section 3.3.1, chaque niveau de profondeur de la prédiction hiérarchique utilise les profondeurs plus hautes comme prédicteur. Ainsi, chaque prédicteur doit être calculé avec les pixels reconstruits pour éviter les erreurs de divergence entre l'encodeur et le décodeur. Avec une compression sans perte, ceci n'a aucun impact puisque les pixels originaux et reconstruits sont identiques.

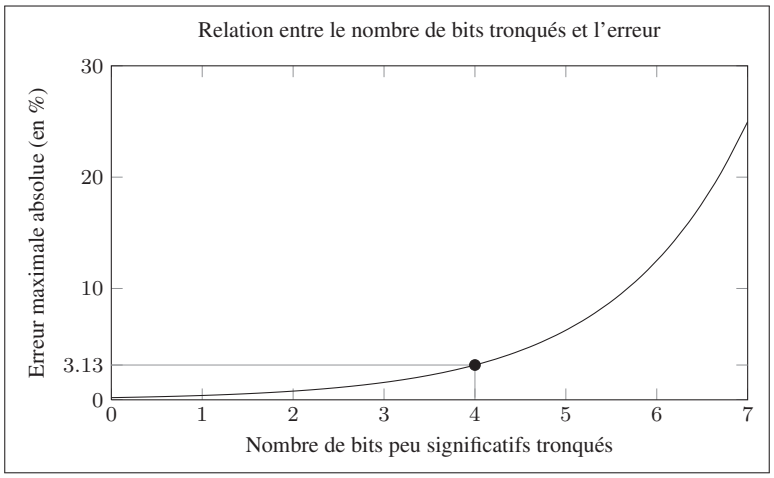


Figure 4.12 Relation entre le nombre de bit tronqué et l'erreur

Cependant, si les erreurs de prédiction sont quantifiées, la reconstruction sera imparfaite. Ainsi le traitement complet de chaque niveau de profondeur, incluant l'étape de codage et reconstruction, doit être complété avant de traiter les prochains niveaux de profondeur. Ceci ajoute des goulots d'étranglement dans l'algorithme qui peuvent nuire aux performances au niveau de la vitesse d'exécution.

De plus, le nombre de bits tronqués pour chaque groupe de SBT doit être choisi avant de connaitre la taille totale de l'unité codée. Ceci pose un problème. Pour maximiser la qualité globale des images reconstruites, seulement les unités ne pouvant pas être codées avec un taux de compression de 1.58 doivent être codées avec perte.

4.3.4.1 Détermination de la taille codée par itération de codage

Une solution possible est une approche itérative. C'est-à-dire une approche où les valeurs de q sont itérativement augmentées, à partir de la valeur initiale de zéro, jusqu'à ce que la taille de l'unité codée satisfasse les contraintes. Cette technique peut augmenter significativement le temps d'exécution si plusieurs itérations sont nécessaires.

4.3.4.2 Estimation de la taille codée d'un bloc pour une décision a priori

Une autre solution qui évite les itérations de codage est une approche prédictive. Il s'agit de prédire si la taille d'une unité compressée sera plus grande que la charge utile maximale d'un paquet RTP et ensuite d'estimer le nombre de bits q à tronquer pour satisfaire les contraintes. Pour ce faire, une certaine mesure quantitative doit être appliquée sur les blocs de l'unité. Ensuite, un seuil est établi statistiquement pour indiquer si une compression avec perte est nécessaire. Le problème avec cette approche, comme toute technique de classification, est que les résultats ne sont pas fiable à 100%. En effet, si l'algorithme se fie uniquement à un seuil et non à un résultat de prédiction, certains blocs seront codés avec perte alors qu'ils auraient satisfait les exigences pour un codage sans perte, et vise versa.

4.3.4.3 Séparation des niveaux de codage

Nous proposons, dans cette sous-section, une solution permettant d'éviter le goulot d'étranglement et le choix hâtif du nombre de bits tronqués pour chaque groupe de SBT. Pour ce faire, seulement le dernier des quatre niveaux de profondeur de prédiction de chaque bloc conservera la possibilité d'être codé avec perte, par la quantification de l'erreur de prédiction. Ainsi, la prédiction hiérarchique complète peut être faite et la taille du bloc connu avant de prendre la décision de quantifier ou pas. En effet, par définition de la méthode de prédiction hiérarchique, aucun calcul de prédicteur n'est effectué sur des pixels du dernier niveau de profondeur et les autres niveaux sont toujours codés sans perte. Rappelons que les groupes SBT 4 à 7 correspondent au dernier niveau (voir figure 3.6).

De plus les groupes 4, 5 et 6 du dernier niveau bénéficient du prédicteur le plus efficace, soit la prédiction par moyenne horizontale avec une distance d'un. Cette efficacité est démontrée par la figure 4.13 qui montre les moyennes du nombre de bits nécessaires pour coder l'erreur de prédiction pour chaque groupe SBT d'un bloc. Les groupes 4, 5 et 6 montrent un nombre de bits moyen par pixel plus petit donc une meilleure prédiction que les autres groupes. Ceci implique qu'en moyenne moins d'information est transmise pour le dernier niveau.

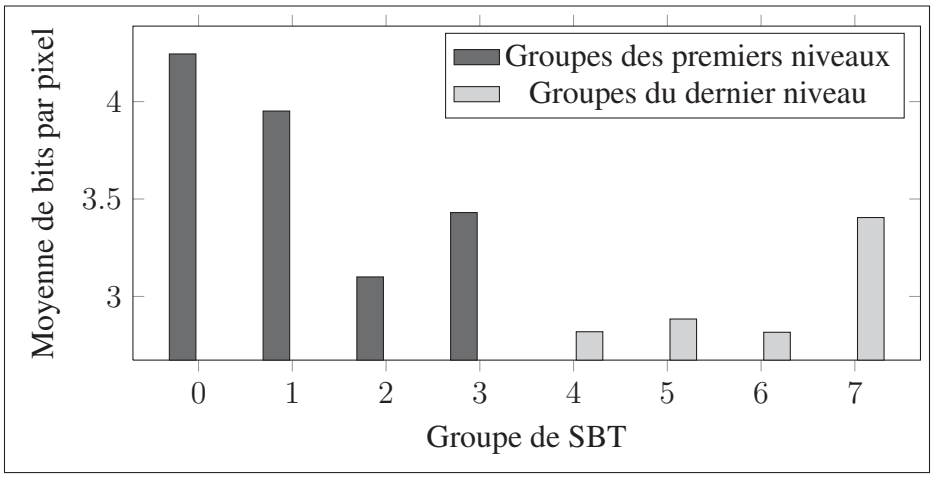


Figure 4.13 Moyenne du nombre de bits par pixel nécessaire pour coder chaque groupe SBT sans perte d'information

Ce dernier niveau contient la moitié des pixels d'un bloc, soit 32. Ainsi, 32 pixels sont codés sans perte. Si le dernier niveau de prédiction n'est simplement pas codé, la taille des unités codées varie entre 192 et 1344 octets, ce qui laisse entre zéro et huit bits par erreur de prédiction du dernier niveau pour un total de 1344 octets.

4.3.5 Modes de codage

Des discussion précédentes, nous en arrivons à proposer un algorithme comportant trois modes de codage communiqués par le champ **Mode** défini dans la section 4.2.2.1. Ce champ peut prendre les valeurs décrites dans les sous-section suivantes.

4.3.5.1 Sans perte (0)

Avec ce mode, aucune quantification n'est appliquée. Si la taille d'une unité dépasse la taille de la charge utile d'un paquet RTP, le champ **Type** indique que le dernier niveau de profondeur de prédiction est transmis dans un deuxième paquet.

4.3.5.2 Avec perte rapide (1)

Quand ce mode est sélectionné, une unité ne peut pas être codée sur plus qu'un paquet RTP. Ainsi, lorsque le taux de compression ne permet pas ceci, le dernier niveau de profondeur n'est simplement pas transmis.

4.3.5.3 Avec perte calculée (2)

Lorsque ce mode est sélectionné, les champs **LQbits** et **CQbits** de l'entête d'unité sont considérés. Ces champs représentent le nombre de bits moins significatifs tronqués pour les blocs de luminance et de chrominance, respectivement. Donc les facteurs de quantification sont respectivement $2^{\mathbf{LQbits}}$ et $2^{\mathbf{CQbits}}$.

Ce mode de codage vise à déterminer le facteur de quantification minimal pour que l'unité puisse être codée dans un seul paquet. Ainsi, ce mode de codage avec perte maximise la qualité visuelle. Pour minimiser la perte de qualité visuelle, le facteur de quantification maximal utilisé est 2^4. Si la quantification maximale ne suffit pas pour que l'unité respecte la limite de la charge utile d'un paquet, l'unité est envoyée en deux paquets. L'algorithme de sélection des valeurs **LQbits** et **CQbits** est illustré dans la figure 4.14 et est décrit en détail au chapitre suivant.

4.4 Résumé

Ce chapitre explique les contributions de ce travail. Elles prennent la forme de trois améliorations à l'algorithme HACP. Celles-ci sont la simplification de la hiérarchie de prédiction pour éviter un goulot d'étranglement dans le traitement, l'adaptation pour la communication pour permettre la mise en paquets du flux encodé et l'extension au codage par SBT pour permettre une compression avec perte d'information pour garantir un taux de compression minimal.

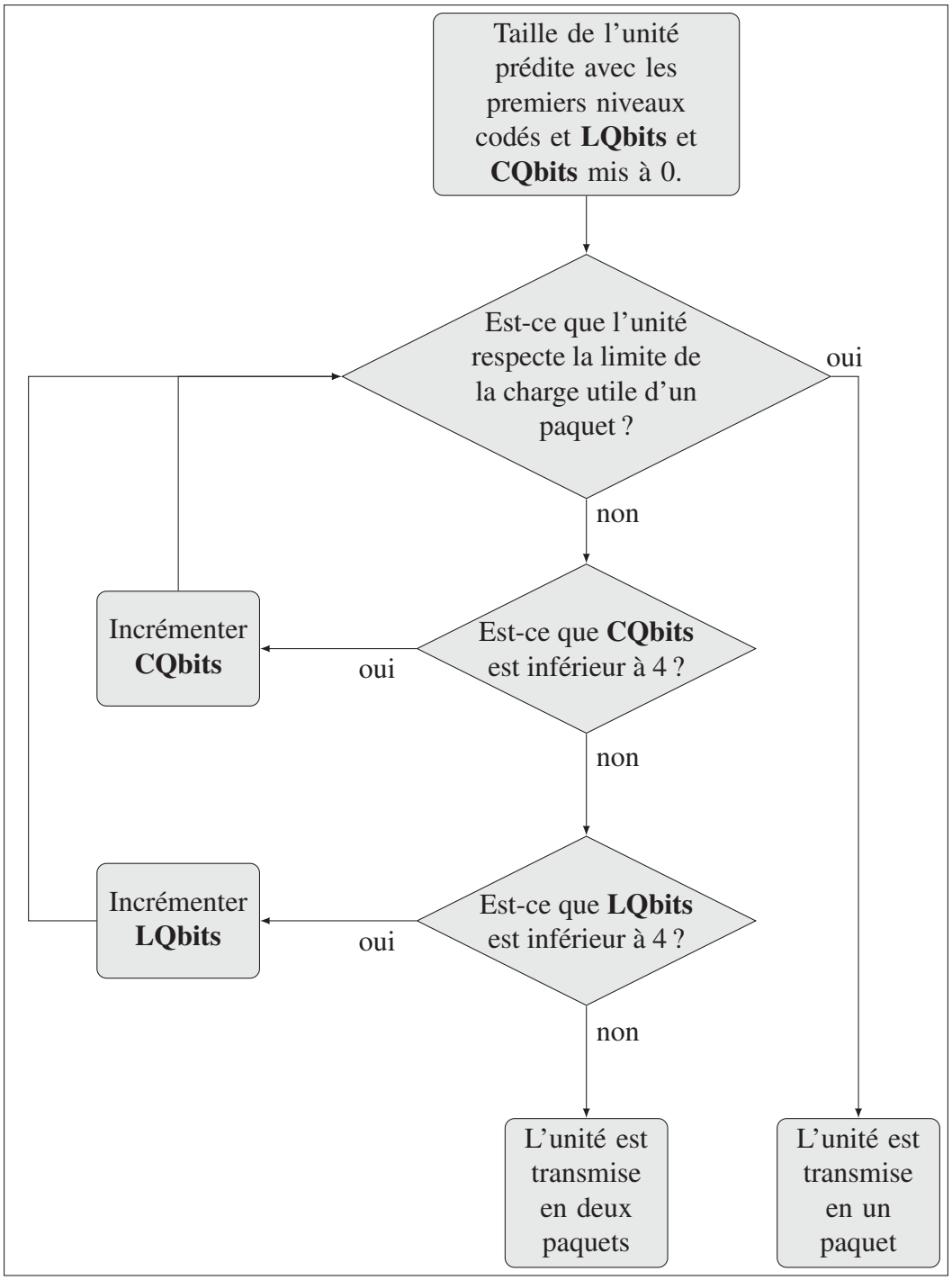


Figure 4.14 Algorithme de sélection des valeurs **LQbits** et **CQbits**

CHAPITRE 5

CONSIDÉRATIONS D'IMPLÉMENTATION

Ce chapitre présente les considérations et détails d'implémentation de l'algorithme de codage et de mise en paquet présenté dans le chapitre 4 pour le système présenté dans le chapitre 2. Les sections 5.1 et 5.2 expliquent respectivement les algorithmes d'encodage et de décodage implémentés. Puisque les limitations les plus contraignantes du système sont reliées à la mémoire, la technique utilisée pour gérer la mémoire est décrite dans la section 5.3.

5.1 Étapes d'encodage

Cette section explique les étapes de codage d'une unité tel que spécifié dans la section 4.2. Le processus de codage est spécifié par unité pour maintenir leur indépendance. Ainsi, une implémentation de codage parallèle est possible.

L'algorithme 5.1 énumère l'ordre des étapes de codage. Chacune de celles-ci est expliquée dans les sous-sections qui suivent.

```
pour tous les unités de l'image ; faire
    pour tous les macroblocs de l'unité ; faire
        pour tous les blocs 8 × 8 du macrobloc ; faire
            5.1.1 Prédiction;
            5.1.2 Codage des niveaux 1, 2 et 3;
        fin
    fin
    pour tous les macroblocs de l'unité ; faire
        pour tous les blocs 8 × 8 du macrobloc ; faire
            5.1.3 Analyse et codage du niveau 4;
        fin
    fin
fin
```

Algorithme 5.1 : Étapes d'encodage de la solution implémentée

5.1.1 Prédiction

L'objectif de l'étape de prédiction est de réduire la quantité d'information à coder en appliquant des formules mathématiques sur les pixels du bloc. Ces formules sont définies par les équations (3.5) à (3.9). Les regroupements de pixels utilisés dans ces formules sont décrits dans la section 4.1.

La prédiction nécessite beaucoup de calculs. C'est ainsi une section de l'algorithme qui bénéficie d'une implémentation sur processeur de signaux numériques (DSP). En effet, l'application des formules de prédiction est implémentée avec les instructions SIMD du processeur Opus.

Concrètement, deux blocs de mémoire sont utilisés pour cette étape. Les pixels originaux sont conservés dans un bloc et le résiduel de prédiction est écrit dans l'autre. Ceci permet d'éviter le codage et la reconstruction des premiers niveaux avant de prédire les niveaux suivants. Ainsi, la reconstruction, soit l'application de l'inverse des formules de prédiction, n'est pas nécessaire lors de l'encodage.

C'est aussi lors de l'étape de prédiction que la correction pour le problème de signe, expliqué dans la section 4.3.2.2, est appliquée. Seulement le **mode** de codage *2 - avec perte calculée*, défini dans la section 4.3.5.3, peut souffrir de ce problème. Ainsi, si ce mode est sélectionné les résiduels de prédiction sont modifiés. En pratique, les valeurs de résiduel de prédiction du niveau 4 qui ont une valeur signée entre 0 et 7, inclusivement, sont remplacés par 8 et ceux entre -7 et -1, inclusivement, sont remplacés par -8.

5.1.2 Codage des niveaux 1, 2 et 3

Une fois la prédiction terminée, le bloc de mémoire contenant les résiduels de prédiction est codé avec des groupes de SBT de huit pixels avec la procédure expliquée dans la section 3.3.2 et le résultat est sérialisé pour la mise en paquet.

Les niveaux de codage sont des groupes de pixels qui sont codés séquentiellement. Ceux-ci sont définis dans la figure 4.1b. Les niveaux 1, 2 et 3 sont codés complètement, jusqu'à l'étape

de mise en paquet. Cependant, seulement le BLH est calculé pour les groupes SBT du niveau 4. Ce dernier est nécessaire pour l'étape suivante d'analyse.

5.1.3 Analyse et codage du niveau 4

Pour compléter le codage de l'unité, deux caractéristiques doivent être établies, soit le nombre de paquets utilisés pour coder l'unité et la quantification du niveau 4. Ce dernier point ne s'applique pas pour le **mode** de codage 0 (section 4.3.5.1) puisqu'il indique qu'aucune quantification n'est utilisée.

Premièrement, la taille totale de l'unité codée sans perte est calculée. Pour ce faire, les résultats de codage des niveaux 1, 2 et 3 ainsi que les BLH du niveau 4 sont utilisés. Si l'unité peut être transmise avec un seul paquet, le niveau 4 est codé comme les niveaux précédents et l'entête d'unité indique le **type** 0, signifiant que l'unité est codée avec un seul paquet.

Le cas d'exception est celui pour lequel l'unité ne peut être codée avec un seul paquet. Dans ce cas le comportement est défini par le **mode** de codage.

- Avec le **mode** *0 - sans perte* (section 4.3.5.1), les erreurs de prédiction des niveaux 1, 2 et 3 sont incluses dans un premier paquet avec un entête d'unité indiquant le **type** 1, pour premier paquet de deux. Les erreurs de prédiction du niveau 4 sont aussi codées avec la procédure de la section 3.3.2 et l'entête d'unité indique le **type** 2, pour deuxième paquet de deux.
- Avec le **mode** *1 - avec perte rapide* (section 4.3.5.2), les erreurs de prédiction du niveau 4 ne sont pas transmises. L'entête d'unité indique le **type** 0, pour un seul paquet.
- Avec le **mode** *2 - avec perte calculée* (section 4.3.5.3) la procédure est plus complexe. Le BLH du niveau 4 est utilisé pour déterminer les plus petits facteurs de quantification qui permettent à l'unité d'être codée avec un seul paquet. Cette procédure est expliquée dans la section 4.3. Pour empêcher que les pertes d'information dégradent la qualité visuelle en dessous d'un seuil acceptable, il est possible que les facteurs de quantification ne

permettent pas de coder l'unité avec un seul paquet. Ainsi les trois **types** peuvent être présents avec ce **mode**.

5.2 Étape de décodage

Le processus de décodage est relativement simple. Il est composé du décodage des codes SBT et de la reconstruction des pixels avec les formules de prédiction inverse. L'implémentation de ce processus gère les données reconstruites de façon à ce que les paquets puissent être décodés dans le désordre. Les détails de cette implémentation sont expliqués avec l'algorithme 5.2.

5.3 Gestion de la mémoire

Le système dans lequel l'algorithme a été implémenté présente des limitations concernant la mémoire. Celles-ci ont été expliquées dans la section 2.2. L'algorithme est contraint par la mémoire de ce système puisqu'il nécessite une grande quantité de mémoire pour représenter les données non compressées. Ainsi, une technique est essentielle pour gérer le contenu de la mémoire cache puisque celle-ci est trop petite pour contenir toute l'information.

5.3.1 Gestion de la mémoire à l'encodage

Dans le système, les données doivent être explicitement chargées dans la mémoire cache pour que celles-ci puissent être utilisées dans les calculs d'encodage. Pour diminuer les désavantages de l'engin de DMA, utilisé pour charger ces données, deux techniques ont été mises en place.

Premièrement, pour diminuer l'impact du temps de configuration de transfert, le nombre de pixels chargés en mémoire cache est supérieur au nombre nécessaire pour un encoder une unité. La quantité de pixels transférés est un multiple du nombre nécessaire pour une unité. Ce dernier est fixé au nombre de pixels pour six unités. Cet ensemble de pixels est nommé collection de pixels.

Deuxièmement, pour diminuer l'impact du temps de transfert, les transferts sont programmés de façon asynchrone. L'encodeur doit attendre la première collection de pixels qui est transfé-

```
pour tous les paquets d'une image ; faire
    Lecture des données et l'entête d'unité du paquet;
    Calcul de la position de l'unité avec l'information de l'entête;
    si type est 2 (deuxième paquet de deux) alors
        Chargement des données pour les niveaux 1, 2 et 3 préalablement décodés de
        l'unité;
    sinon si type est 1 (premier paquet de deux) alors
        Chargement des données pour le niveau 4 préalablement décodé de l'unité;
    fin
    si type n'est pas 2 (deuxième paquet de deux) alors
        pour tous les macroblocs de l'unité ; faire
            pour tous les blocs 8 × 8 du macrobloc ; faire
                pour tous les groupes de SBT des niveaux 1, 2 et 3 du bloc 8 × 8 ; faire
                    Décodage des résiduels de prédiction;
                    Reconstruction des pixels originaux (prédiction inverse);
                fin
            fin
        fin
    fin
    si type n'est pas 1 (premier paquet de deux) alors
        pour tous les macroblocs de l'unité ; faire
            pour tous les blocs 8 × 8 du macrobloc ; faire
                pour tous les groupes de SBT des niveaux 4 du bloc 8 × 8 ; faire
                    Décodage des résiduels de prédiction;
                    Reconstruction des pixels originaux (prédiction inverse);
                fin
            fin
        fin
    fin
    Sauvegarde des données décodées ;
fin
```

Algorithme 5.2 : Étapes de décodage de la solution implémentée

rée de façon synchrone. Avant de commencer le traitement des pixels de la première collection, un deuxième transfert est programmé dans un autre espace mémoire pour la collection suivante. Ainsi, les données sont présentes lorsque l'encodeur commence à traiter la deuxième collection. Donc les transferts sont programmés de façon à ce que les pixels transférés soient ceux de la prochaine itération de collection de pixels.

5.3.2 Gestion de la mémoire au décodage

Des contraintes similaires sont présentes au décodage. Cependant, les optimisations utilisées pour l'encodeur ne peuvent pas être appliquées. Pour que celles-ci soient applicables, les données doivent être ordonnées. Ce n'est pas le cas au décodage puisque l'ordre d'arrivée des paquets ne peut ne pas être prédit. Une étape supplémentaire de réordonnancement serait nécessaire pour utiliser les mêmes techniques qu'à l'encodage. Cependant, pour effectuer ceci les paquets seraient lus et réécrits dans l'ordre. Donc pour optimiser les transferts mémoire, il faudrait faire des transfert supplémentaires.

Ainsi, les données reconstruites sont lues et écrites pour chaque paquet. De plus, ce choix de conception rend possible un décodage multicoeur. Le fait que chaque paquet est traité de façon indépendante laisse seulement un cas d'exception où les données doivent être sérialisées pour permettre une reconstruction sans erreur. Ce cas se produit lorsqu'une unité est codée sur plus d'un paquet. Cette sérialisation n'est pas un obstacle majeur puisque l'objectif de l'algorithme est d'éviter cette situation le plus souvent possible.

5.4 Résumé

Ce chapitre explique les décisions d'implémentation en lien avec les contraintes du système dans lequel l'algorithme a été implémenté. De plus, les étapes d'encodage et décodage sont présentées.

CHAPITRE 6

RÉSULTATS DE L'ÉVALUATION DE L'ALGORITHME PROPOSÉ

L'algorithme proposé a été implémenté et évalué sous trois aspects : le taux de compression, la qualité visuelle, et le temps d'exécution. La première section présente la méthodologie de test utilisée. La deuxième section résume l'atteinte des objectifs. Les sections suivantes analysent les facteurs qui influencent les trois aspects évalués.

6.1 Procédure de test pour l'analyse de l'algorithme proposé

L'objectif des tests est d'analyser et de comparer les performances des modes de compression de l'algorithme proposé. Ces tests ont été exécutés avec la configuration détaillée dans l'annexe I. Plus particulièrement, les tests ont été exécutés sur un ordinateur utilisant *Windows 7 Professional 64-bit (6.1, Build 7601)* comme système d'exploitation, 6 Go de mémoire vive et un processor *Intel(R) Core(TM) i7 CPU 950 @ 30.7 GHz (8CPU), 2.8GHz.*

Les tests consistent à utiliser les 880 vecteurs de test définis dans l'annexe II, les encoder avec l'encodeur et les décoder avec le décodeur de l'algorithme présenté. Après l'encodage la première métrique, le taux de compression, est mesurée. Ensuite, après le décodage, les séquences initiales sont comparées aux séquences reconstruites et la qualité visuelle est évaluée. Deux métriques de qualité ont été évaluées, soit le PSNR et le SSIM. Ceux-ci ont été évalués avec la bibliothèque de code source libre *Image Quality Assessment (IQA)* (Tjdistler, 2011). La troisième mesure effectuée n'a pas été faite avec l'ordinateur de test identifié décrit ci-dessus, mais avec le matériel d'Octasic, soit une carte d'évaluation T51 ayant une puce OCT2224M. Notons que les vecteurs de test définis dans l'annexe II contiennent plusieurs résolutions pour lesquelles la compression des données n'est pas requise pour la transmission bidirectionnelle sur un lien Ethernet. Mais, ils servent à démontrer la consistance des performances de compression et de la qualité visuelle pour diverses résolutions et montrer comment la complexité des calculs évolue avec elles. Ces résultats peuvent servir à la conception de systèmes dont les contraintes de débit sont différentes de celles considérées dans ce travail.

Les résultats complets, sans analyse, sont présentés à l'annexe V.

6.2 Influence des modes de codage sur la performance

Cette section montre l'influence des modes de codage de l'algorithme proposé sur l'atteinte des objectifs. En résumé, les trois modes sont le mode *sans perte*, le mode *avec perte calculée* et le mode *avec perte rapide*. Le premier des trois modes représente l'état de l'art puisqu'il est l'adaptation d'un algorithme connu à un système de compression et communication par paquet. Les deux autres modes ont pour objectif d'atteindre un taux de compression déterminé. Le mode avec perte calculée a pour objectif premier de limiter la détérioration de la qualité visuelle tandis que le mode avec perte rapide a pour objectif de limiter le temps d'exécution.

Pour illustrer les performances, les distributions cumulatives de probabilité des taux de compression et de mesure de qualité visuelle sont présentées. Celles-ci permettent de bien visualiser les seuils identifiés comme objectifs pour chacun des aspects.

6.2.1 Taux de compression

Le taux de compression nécessaire pour qu'une unité puisse être contenue dans un seul paquet est de 1.58. Ce taux est l'objectif de compression des algorithmes présentés. La figure 6.1 montre la distribution cumulative des probabilités pour les échantillons testés. Cette figure indique que seul le mode *sans perte* n'atteint pas ce seuil en tout temps. Cependant, nous voyons que pour la majorité des cas de tests, le mode *sans perte* atteint le seuil requis. En effet, moins de 20% de ceux-ci y sont inférieurs. Il est à noter que le taux de compression minimal atteint par les deux algorithmes avec perte est de 1.68 pour **crowd run** en QCIF avec le mode *avec perte calculée*. Ce résultat est disponible à l'annexe V. Seulement le mode *avec perte rapide* garantit un taux de compression de 1.58 (si l'information ne rentre pas dans un paquet, elle n'est pas envoyée). Le mode *avec perte calculée*, qui se rapproche plus de la limite de 1.58, ne garantit pas de taux de compression, c'est du *best effort* jusqu'à une quantification de 4 bits de l'erreur. Si une quantification plus grande avait été nécessaire pour atteindre un taux

de 1.58, les données auraient été envoyées dans un paquet séparé pour s'assurer de garder une qualité visuelle acceptable.

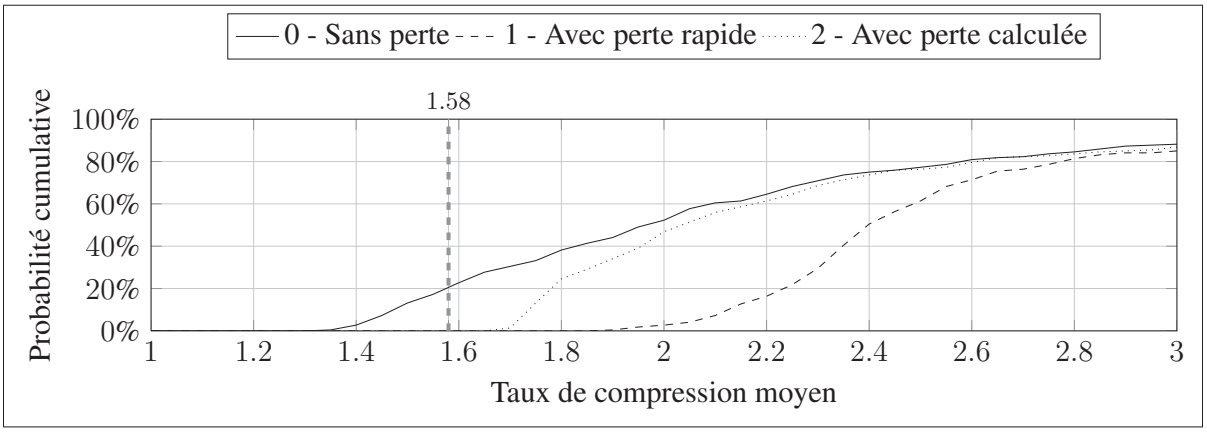


Figure 6.1 Distribution de probabilité cumulative du taux de compression par mode de codage

De plus si nous considérons seulement les séquences 1080p, qui représentent le réel objectif de ce travail, moins de 5% des séquences n'atteignent pas la limite de 1.58.

6.2.2 Qualité visuelle

Pour évaluer la qualité visuelle de l'algorithme, une métrique ayant une valeur bornée pour comparaison avec le mode sans perte est nécessaire. C'est pourquoi la métrique du SSIM a été choisie aux dépens du PSNR. Ce dernier est pourtant bien répandu et moins complexe à calculer. Le SSIM a une valeur définie à 1 lorsque l'algorithme ne présente aucune perte d'information.

Aucun critère de qualité n'a été fixé pour le mode *avec perte rapide*. Seulement les modes *sans perte* et *avec perte calculée* ont pour objectif de ne pas avoir d'impact sur la qualité des images perçues par l'oeil humain.

La figure 6.2 montre que l'objectif est atteint, puisque le mode *avec perte calculée* ne présente aucun échantillon avec un SSIM plus bas que 0.990 et seulement de 25% en bas de 0.999. Ceci

indique que les distorsions créées par ce mode de compression ne sont pas perceptibles pour la majorité des cas.

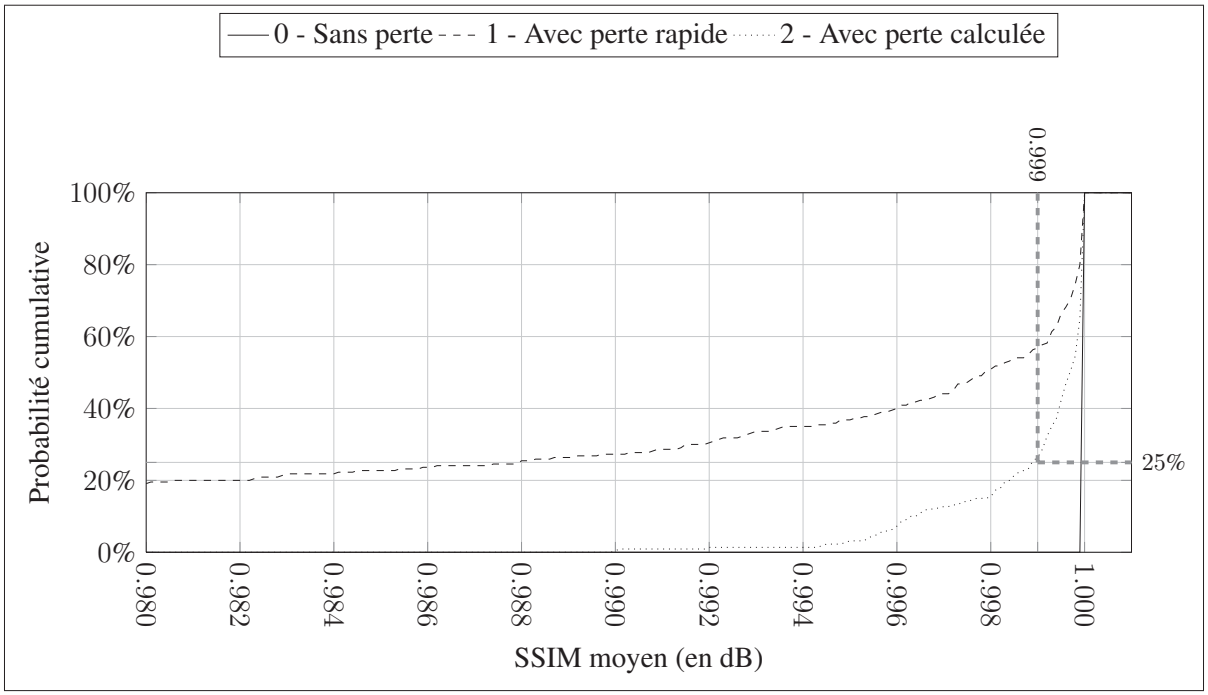


Figure 6.2 Distribution de probabilité cumulative de la qualité visuelle par mode de codage

Dans la figure 6.3, nous considérons seulement les séquences 1080p. Ainsi, il n'y a qu'une seule la séquence présentant des distorsions notables, avec un SSIM inférieur à 0.997, la séquence **zoneplate**. La faible qualité visuelle de ce cas de test s'explique par la composition du vecteur de test. C'est un motif de Fresnel elliptique créé artificiellement. Ce type de contenu n'est pas bien prédit par le prédicteur présenté.

6.2.3 Temps d'exécution

Le critère du temps d'exécution est nécessaire, puisque ces algorithmes de compression sont conçus pour des systèmes temps réel. Ceci impose que l'exécution respecte les contraintes du temps réel. En considérant 30 images par seconde, cette contrainte est de 33.3 millisecondes par image pour le processus d'encodage et 33.3 millisecondes par image pour le processus de

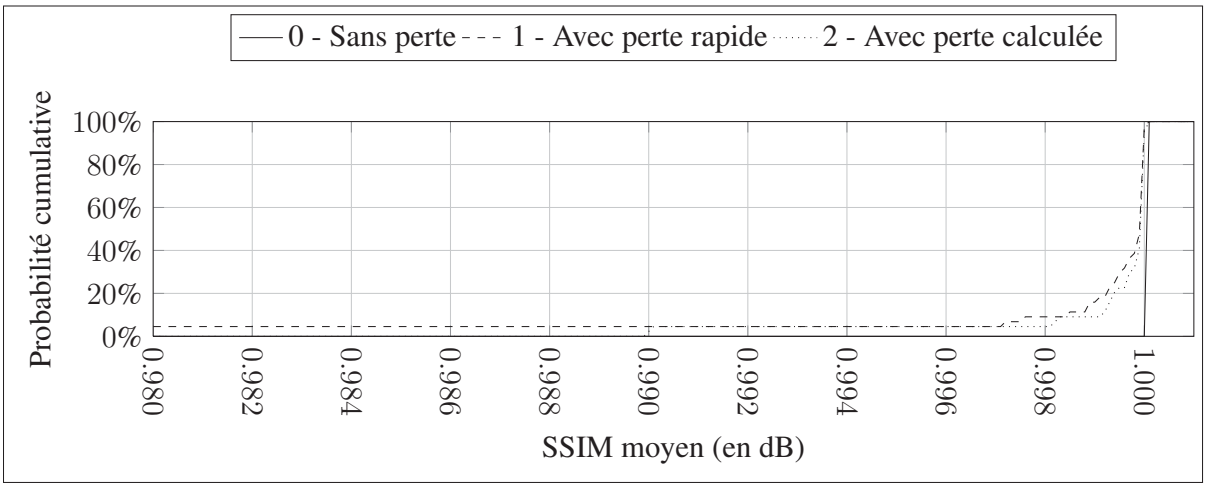


Figure 6.3 Distribution de probabilité cumulative de la qualité visuelle par mode de codage pour les séquences 1080p

décodage. Les temps d'exécution calculés pour le processus d'encodage et de décodage sont présentés dans les tableaux 6.1 et 6.2.

Tableau 6.1 Temps d'exécution de l'encodeur en millisecondes par image

Mode	1080p	720p	NTSC	CIF	QCIF
Sans perte	1342.14	553.31	207.46	60.83	15.22
Avec perte calculée	1968.73	840.22	315.04	92.42	23.11
Avec perte rapide	1309.32	553.18	207.40	60.83	15.21

Tableau 6.2 Temps d'exécution du décodeur en millisecondes par image

Mode	1080p	720p	NTSC	CIF	QCIF
Sans perte	959.38	316.67	118.45	34.61	8.64
Avec perte calculée	1942.13	802.52	300.81	88.21	22.05
Avec perte rapide	807.59	316.48	118.38	34.62	8.64

Ces résultats montrent que l'objectif de 30 images par seconde est seulement atteint pour la résolution QCIF. L'implémentation des algorithmes doit être optimisée davantage pour atteindre l'objectif de temps d'exécution. Une analyse plus approfondie est présentée dans la section 6.5.

6.3 Analyse des facteurs affectant le taux de compression

Cette section analyse certains facteurs qui ont une influence sur le taux de compression. Ces facteurs ont un impact sur le contenu des blocs codés et ainsi affectent la performance des algorithmes.

6.3.1 Taux de compression après codage H.264

Dans le mode d'opération réel d'un système utilisant les algorithmes présentés, les données d'entrée seront issues d'un décodeur. Ainsi, les données d'entrée présenteront déjà des artefacts de compression dus au codage et au décodage subi. En théorie, les codecs utilisant une quantification dans le domaine fréquentiel, tels que H.264, ont pour effet de diminuer les hautes fréquences. Le prédicteur présenté devrait bénéficier de ceci et ainsi les taux de compression devraient être plus élevés. Pour analyser ceci, les vecteurs de test ont été compressés et décompressés avec FFMPEG et le codec H.264 avec un facteur de quantification de 25.

Tableau 6.3 Taux de compression par mode en fonction de la présence de compression

Mode	Entrée non-compressée	Entrée compressée (H.264 VBR QP 25)
Sans perte	2.04	2.34
Avec perte calculée	2.17	2.45
Avec perte rapide	2.52	2.71

Le tableau 6.3 montre que le taux de compression moyen est effectivement plus élevé lorsque les séquences vidéo ont précédemment subi un codage H.264.

6.3.2 Taux de compression par mode par résolution

Un autre facteur qui a un impact sur le contenu des blocs est la résolution. Il est possible de visualiser ceci en considérant des blocs de taille fixe provenant de deux images représentant la même scène, mais de tailles différentes. En théorie, le bloc provenant de l'image ayant la plus petite taille contient plus d'information donc une entropie plus élevée. Ainsi, les taux de compression des séquences vidéo de plus petite résolution devraient être plus petits.

Tableau 6.4 Taux de compression par mode en fonction de la résolution

Mode	1080p	720p	NTSC	CIF	QCIF
Sans perte	2.60	2.37	2.21	1.98	1.79
Avec perte calculée	2.69	2.47	2.33	2.12	1.94
Avec perte rapide	2.92	2.71	2.61	2.47	2.36

Les résultats montrés dans le tableau 6.4 montrent que le taux de compression moyen est effectivement plus élevé lorsque les séquences vidéo ont une résolution plus élevée. Et ce, pour chaque mode de codage.

6.3.3 Taux de compression par mode par vecteur de test

Le dernier facteur analysé, mais non le moindre, est la séquence vidéo elle-même. La figure 6.4 montre que le mode *sans perte* permet des taux de compression variant entre environ 1.5 et 3. Ceci indique que le facteur qui influence le plus de taux de compression est le contenu des images.

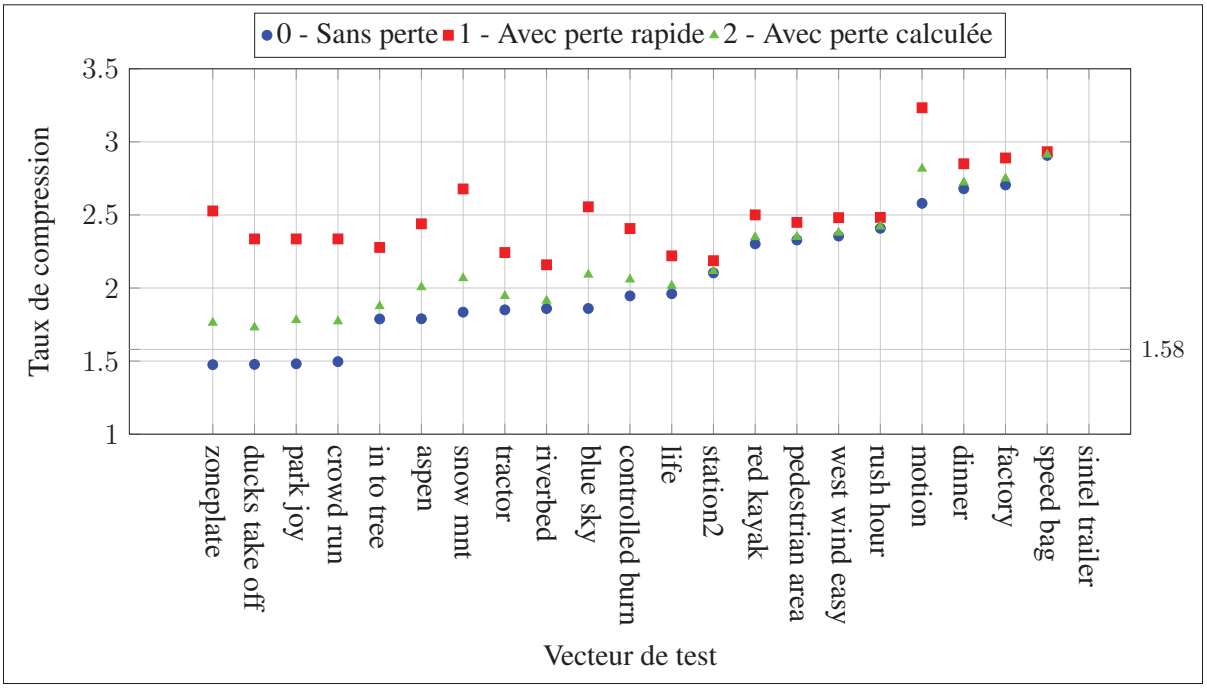


Figure 6.4 Taux de compression moyen par clip

Pour augmenter la clarté de la figure 6.4, les données pour la séquence *sintel trailer* n'y sont pas affichées. Ces données sont présentées dans le tableau 6.5. Les taux de compression atteints avec cette séquence sont bien supérieurs aux taux atteints avec les autres clips. Ceci est possible puisque cette séquence contient beaucoup d'images complètement noires. Celles-ci peuvent être prédites parfaitement puisqu'elles sont des images artificielles avec une couleur pure, sans artefact créé par un capteur d'image.

Tableau 6.5 Point exclu de la figure 6.4

Vecteur de test	Sans perte	Avec perte rapide	Avec perte calculée
sintel trailer	4.93	4.99	4.94

6.4 Analyse des facteurs affectant la qualité d'image

Les valeurs de SSIM dans le tableau 6.6 montrent que la qualité visuelle obtenue avec le mode de compression *avec perte calculée* se rapproche de celle obtenue avec le mode de compression *sans perte*. Deux facteurs ont été analysés pour comprendre leur impact sur la qualité d'image, soit la résolution et la séquence vidéo.

Tableau 6.6 Qualité d'image (SSIM) moyenne en fonction du mode de compression

Mode	SSIM
Sans perte	1.000
Avec perte calculée	0.999
Avec perte rapide	0.984

6.4.1 SSIM par mode par résolution

En analysant les valeurs de SSIM obtenues par mode de codage avec perte (tableau 6.7), il est possible de constater que la résolution a un impact sur la qualité obtenue. Cependant, cet impact n'est réellement appréciable que pour les petites résolutions (CIF et QCIF). Dans ces petites résolutions, la région de l'image que représente une unité de compression est une partie significative de l'image complète. Par exemple une unité de six macroblocs représente environ

6% de la superficie d'une image QCIF, mais moins de 1% d'une image 1080p. Ainsi, une seule unité présentant une dégradation visuelle est plus importante dans une image de plus petite résolution.

Tableau 6.7 Qualité d'image (SSIM) par mode en fonction de la résolution

Mode	1080p	720p	NTSC	CIF	QCIF
Sans perte	1.0000	1.0000	1.0000	1.0000	1.0000
Avec perte calculée	0.9993	0.9998	0.9997	0.9982	0.9979
Avec perte rapide	0.9977	0.9929	0.9937	0.9708	0.9658

6.4.2 SSIM par mode par vecteur de test

Le contenu des images a aussi un impact sur la qualité. Ceci est plus apparent pour le mode *avec perte rapide*, puisque les décisions faites dans ce mode ne considèrent pas tous les enjeux de qualité. Par opposition à cela, les données présentées dans la figure 6.5 montrent que le mode *avec perte calculée* permet d'obtenir une qualité similaire au mode *sans perte*. Le mode *avec perte rapide*, pour le vecteur de test **zoneplate**, n'est pas présenté dans la figure 6.5. La valeur de ce point est de 0.82512, il est omis pour faciliter la lecture de la figure.

6.5 Analyse du temps d'exécution par section de l'algorithme

Pour bien comprendre ce critère, le temps d'exécution a été divisé pour chaque section de l'algorithme. Ceux-ci sont présentés dans les tableaux 6.8 et 6.9. Il faut noter que ces résultats ont été obtenus avec des données aléatoires à l'entrée (de la mémoire non initialisée).

6.5.1 Contraintes du système

Le temps d'exécution est le critère mesuré où l'on perçoit l'impact des contraintes du système, décrites chapitre 2, dans lequel l'algorithme a été implémenté.

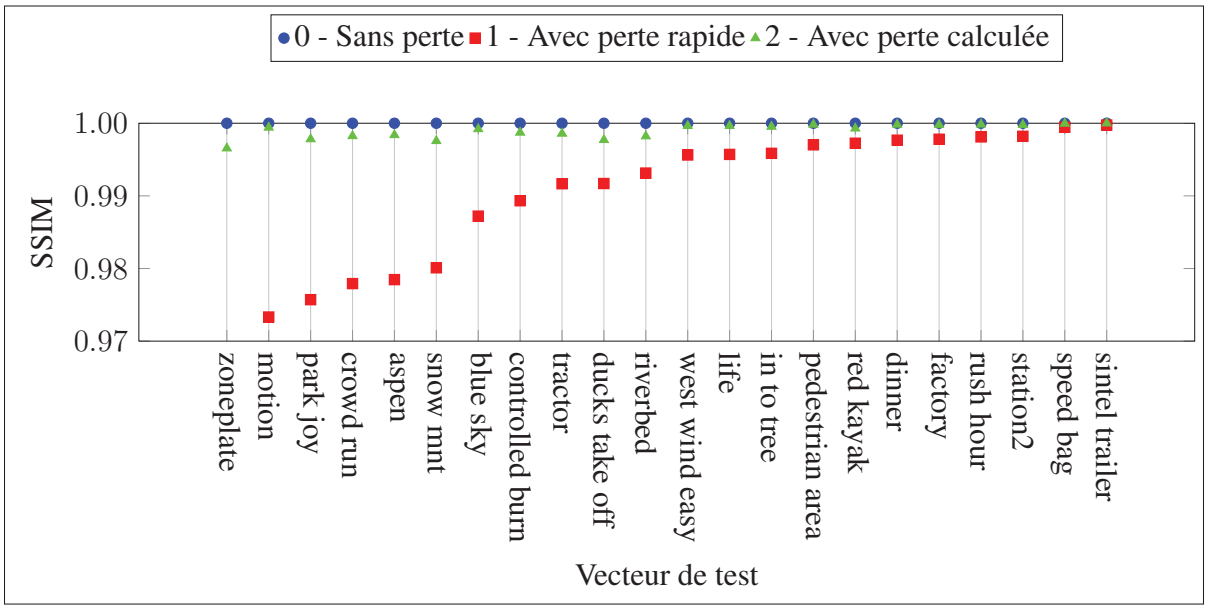


Figure 6.5 Qualité moyenne (SSIM) par clip

Tableau 6.8 Temps d'exécution de l'encodage pour chaque section de l'algorithme pour une unité de codage, en microsecondes

Section de l'algorithme	*Sans perte*	*Avec perte calculée*	*Avec perte rapide*
Prédiction et codage des niveaux sans perte	1023.00	1123.40	1023.20
Sélection du quantificateur	0.00	0.80	0.00
Codage du niveau quantifié	705.80	1019.80	710.00
Gestion de la mémoire	0.00	0.00	0.00
Total	1728.80	2144.00	1733.20

Tableau 6.9 Temps d'exécution du décodage pour chaque section de l'algorithme pour une unité de codage, en microsecondes

Section de l'algorithme	*Sans perte*	*Avec perte calculée*	*Avec perte rapide*
Décodage et prédiction inverse des niveaux sans perte	470.80	460.00	461.40
Décodage et prédiction inverse des niveaux quantifiés	182.80	432.80	172.40
Gestion de la mémoire	2.80	2.60	2.60
Total	653.60	892.80	633.80

6.5.1.1 Dépendance des données et instructions SIMD

La première contrainte est la dépendance des données et l'utilité des instructions de type SIMD. L'étape de la prédiction et de la prédiction inverse au décodeur est la section de l'algorithme qui nécessite le plus de calculs. Ainsi, c'est la section qui aurait dû bénéficier de l'utilisation d'instructions SIMD sans dépendance. Un des avantages du prédicteur sélectionné est qu'une grande partie de ses calculs peut être remplacée par une instruction OPUS, ceci est expliqué dans la section 3.3.3.2. Cependant, la majorité des calculs pouvant bénéficier de cette instruction sont effectués horizontalement, c'est-à-dire que les octets utilisés comme entrée de l'instruction ne sont pas situés de manière séquentielle dans la mémoire. Ainsi, pour pouvoir utiliser cette instruction, les octets représentant des pixels doivent être séquencés dans des registres. Ceci ajoute une série d'instructions dépendantes avant l'appel à l'instruction SIMD, ce qui diminue significativement les avantages de l'utiliser. En effet, les extraits de code 6.1 et 6.2 ont un temps d'exécution similaire. Cela explique pourquoi les temps d'exécution pour l'étape de prédiction sont si grands.

Extrait 6.1 Calcul de quatre prédictions PMH pour le HACP, avec l'instruction *pavgub*

```
#define mPACK_BYTES(A,B,C,D) (((A)<<24)|((B)<<16)|((C)<<8)|(D))

#define mUNPACK_BYTES(PackedBytes,A,B,C,D)  \
A = (PackedBytes>>24)&0xFF; \
B = (PackedBytes>>16)&0xFF; \
C = (PackedBytes>>8)&0xFF; \
D = (PackedBytes)&0xFF;

{
    unsigned long vbyPackedBytes;
    vbyPackedBytes = _psubb( _pavgub( mPACK_BYTES(pP[0*ulStride+0], pP[2*ulStride+0], pP[4*ulStride+0], pP[6*ulStride+0]),
                                      mPACK_BYTES(pP[0*ulStride+4], pP[2*ulStride+4], pP[4*ulStride+4], pP[6*ulStride+4])
                                    ),
                             mPACK_BYTES( pP[0*ulStride+2], pP[2*ulStride+2], pP[4*ulStride+2], pP[6*ulStride+2]) );

    mUNPACK_BYTES( vbyPackedBytes, pE[0*ulStride+2], pE[2*ulStride+2], pE[4*ulStride+2], pE[6*ulStride+2] );
}
```

Extrait 6.2 Calcul de quatre prédictions PMH pour le HACP, sans l'instruction *pavgub*

```
pE[0*ulStride+2] = pP[0*ulStride+2] - (pP[0*ulStride+0]+pP[0*ulStride+4]+1)/2;
pE[2*ulStride+2] = pP[2*ulStride+2] - (pP[2*ulStride+0]+pP[2*ulStride+4]+1)/2;
pE[4*ulStride+2] = pP[4*ulStride+2] - (pP[4*ulStride+0]+pP[4*ulStride+4]+1)/2;
pE[6*ulStride+2] = pP[6*ulStride+2] - (pP[6*ulStride+0]+pP[6*ulStride+4]+1)/2;
```

6.5.1.2 Gestion de la mémoire

La deuxième contrainte est la gestion de la mémoire. La section 5.3 explique l'optimisation de la gestion de la mémoire appliquée à l'encodeur et la raison pour laquelle elle ne s'applique pas au décodeur. Les tableaux 6.8 et 6.9 montrent que, bien qu'un grand nombre de données soient traitées par les deux processus, l'encodeur prend significativement moins de temps pour gérer la mémoire utilisée. En effet, ces temps sont relativement petits comparés aux temps des autres étapes des processus. Ceci est dû au fait que la majorité des transferts mémoires sont effectués en arrière-plan. Les temps affichés dans le tableau montrent seulement le temps pour programmer les transferts.

6.6 Résumé

Ce chapitre montre l'analyse face à trois caractéristiques de l'algorithme présenté dans ce travail. Ces trois caractéristiques sont le taux de compression, la qualité d'image et le temps d'exécution.

CONCLUSION

Dans ce travail de recherche, nous nous sommes intéressés au défi posé par la transmission par paquets en temps réel de données vidéo haute définition (1080p30) sur des liens Gigabit Ethernet entre les cœurs d'une puce de traitement de signal d'Octasic. Pour tenter de résoudre ce problème, nous avons développé un algorithme de compression et de mise en paquet inspiré de la littérature mais adapté aux nombreuses contraintes des puces d'Octasic.

Travail accompli

Nous avons entamé ce travail par une analyse comparative des prédicteurs spatiaux de basse complexité tirés de la littérature. Celle-ci conclut que le prédicteur spatial HACP et le codage par SBT qui s'y rattache sont les algorithmes les plus propices pour développer une solution au problème présenté.

Ensuite, trois améliorations à l'algorithme HACP-SBT sont proposées pour prendre en considération les limitations de l'algorithme de la littérature pour le contexte des puces d'Octasic. Celles-ci prennent la forme d'une simplification de la prédiction en éliminant certains calculs, l'ajout d'un entête pour une flexibilité accrue du traitement et de la mise en paquets, et la proposition de trois modes de codage, dont deux avec pertes, pour permettre de garantir un taux de compression acceptable dans différentes situations. En effet, le mode *sans perte* a été supplémenté par deux modes avec perte : le mode *avec perte calculée* favorisant la qualité visuelle et le mode *avec perte rapide* favorisant le temps d'exécution.

Ensuite, plusieurs obstacles liés à l'implémentation de l'algorithme sur la puce ont été soulevés et résolus, tels que l'ordre des étapes de codage et la gestion de la mémoire. L'algorithme a finalement été implémenté sur une puce de traitement de signal d'Octasic.

Pour terminer, les performances des trois modes de codage proposés sont analysées par rapport aux trois contraintes liées aux puces d'Octasic et devant être rencontrées pour atteindre l'objectif principal fixé dans ce travail. Ceux-ci sont, le taux de compression permettant l'utilisation

d'un lien Gigabit Ethernet, le maintien de la qualité visuelle des images encodées et le temps d'exécution permettant d'atteindre les contraintes du temps réel.

Rencontre des objectifs

Deux des trois objectifs (contraintes établies) ont été rencontrés. Le tableau 6.10 montre que seul l'objectif du temps d'exécution n'a pas été complètement atteint. Toutefois, il serait possible de l'atteindre en segmentant le traitement sur plusieurs cœurs. Ainsi, en segmentant le traitement, il serait possible de rencontrer tous les objectifs.

Tableau 6.10 Résumé de l'atteinte des objectifs pour chaque mode de codage

Mode	Taux de compression	Qualité visuelle	Temps d'exécution
Sans perte	non garanti	atteint	Seulement QCIF
Avec perte rapide	atteint	atteint	Seulement QCIF
Avec perte calculée	atteint	atteint	Seulement QCIF

Possibilités futures

La recherche présentée dans ce travail pourrait être poursuivie avec de nouveaux projets. Les plus importants sont présentés ci-dessous.

- **Optimisation du temps d'exécution** : Les résultats montrent que l'état actuel de l'algorithme ne satisfait pas la contrainte du temps réel. Plus de recherche et d'expérimentation concernant l'implémentation seraient nécessaires pour bien comprendre les faiblesses des algorithmes et de leur implémentation sur la puce d'Octasic. Comme mentionné approche prometteuse est la segmentation du traitement sur plusieurs cœurs de traitement.
- **Analyse de prédicteurs alternatifs** : La syntaxe de mise en paquet proposée est très flexible. Elle prévoit des champs de version et trois bits pour représenter les modes. Donc huit modes de prédiction sont possibles. Ceci permettrait une adaptation de cette syntaxe à d'autres algorithmes (modes) de codage complémentaires.

ANNEXE I

CONFIGURATION DE TEST

Les tests pour les prédicteurs, méthodes et algorithmes ont été réalisés avec des applications C écrites avec Visual C++ Express 2008 de Microsoft.

Ces applications ont été exécutées sur un ordinateur utilisant *Windows 7 Professional 64-bit (6.1, Build 7601)* comme système d'exploitation, 6 GO de mémoire vive et un processor *Intel(R) Core(TM) i7 CPU 950 @ 30.7 GHz (8CPU), 2.8GHz.*

De plus, les 220 séquences vidéo décrites à l'annexe II ont été utilisées comme donnée d'entrée.

Les résultats ont été exportés avec des fichiers en format CSV de l'anglais *Comma Separated Values*, valeurs séparées par des virgules décrit formellement dans le RFC 4180 (Shafranovich, 2005). Ce format est facilement interprété par le tableur électronique Excel de Microsoft. Ce logiciel facilite l'analyse rapide des données, offrant plusieurs fonctionnalités de statistiques et de représentation graphique.

ANNEXE II

LES VECTEURS DE TEST

Tous les vecteurs de test ont été obtenus dans la résolution 1080p à partir des endroits mentionnés dans le tableau II-2 et II-3. Ces 22 séquences vidéo ont été manipulées avec l'outil FFmpeg qui englobe plusieurs logiciels libres. L'information de version de FFmpeg est présentée dans l'extrait de code II.1

La première manipulation a servi à créer des séquences dans différentes résolutions. Les résolutions utilisées sont décrites dans le tableau II-1 et le script utilisé pour la conversion est présenté dans l'extrait de code II.2.

Tableau-A II-1 Résolution utilisée

Nom	Largeur	Hauteur	Nombre de macroblocs	Nombre d'unités
1080p	1920	1080 (1088)	8160	1360
720p	1280	720	3600	600
NTSC	720	480	1350	225
CIF	352	288	396	66
QCIF	176	144	99	16.5

La deuxième manipulation a été utilisée pour encoder et décoder chaque séquence avec le codec H.264. Les séquences résultantes ont été utilisées pour juger l'impact des détériorations dues au codage sur l'algorithme présenté. Le script utilisé pour créer ces séquences est présenté dans l'extrait de code II.3.

Pour faciliter la lecture des séquences par les applications de test, celles-ci ont été converties pour pouvoir lire un macrobloc à la fois. Pour ce faire, l'ordre des pixels à l'intérieur des images a été changé pour regrouper les pixels d'un même macrobloc. De plus les résolutions n'ayant pas un nombre entier de macroblocs ont dû être agrandies jusqu'au prochain multiple de macrobloc. La seule résolution affectée par ce changement est la résolution de 1080p qui a été convertie en 1088p.

Tableau-A II-2 Provenance des vecteurs de test

Nom	Provenance
aspen	Xiph.org Video Test Media [derf's collection]
blue sky	Xiph.org Video Test Media [derf's collection]
controlled burn	Xiph.org Video Test Media [derf's collection]
crowd run	Xiph.org Video Test Media [derf's collection]
dinner	Xiph.org Video Test Media [derf's collection]
ducks take off	Xiph.org Video Test Media [derf's collection]
factory	Xiph.org Video Test Media [derf's collection]
in to tree	Xiph.org Video Test Media [derf's collection]
life	Xiph.org Video Test Media [derf's collection]
motion	Streamcrest Motion1 Test Sequence Generator
park joy	Xiph.org Video Test Media [derf's collection]
pedestrian area	Xiph.org Video Test Media [derf's collection]
red kayak	Xiph.org Video Test Media [derf's collection]
riverbed	Xiph.org Video Test Media [derf's collection]
rush hour	Xiph.org Video Test Media [derf's collection]
sintel trailer	Xiph.org Video Test Media [derf's collection]
snow mnt	Xiph.org Video Test Media [derf's collection]
speed bag	Xiph.org Video Test Media [derf's collection]
station2	Xiph.org Video Test Media [derf's collection]
tractor	Xiph.org Video Test Media [derf's collection]
west wind easy	Xiph.org Video Test Media [derf's collection]
zoneplate	Patrons de Fresnel Elliptique : *Zone Plate Test Pattern*

Tableau-A II-3 Adresse URL des sources de séquences de test

Provenance	URL
Xiph.org Video Test Media [derf's collection]	http://media.xiph.org/video/derf/
Streamcrest Motion1 Test Sequence Generator	http://www.streamcrest.com/mp4.shtml)
Zone Plate Test Pattern	http://www.realitypixels.com/turk/opensource

Extrait II.2 Script batch pour la conversion de résolution avec FFmpeg

```
for /f %%R in ('dir /b *.y4m') do (
  ffmpeg -i %%~nR.y4m -padbottom 8 "%%~nR_(1920x1088).yuv"
  ffmpeg -i %%~nR.y4m -pix_fmt yuv420p -f rawvideo -s 1280x720 -aspect 16:9 "%%~nR(1280x720).yuv"
  ffmpeg -i %%~nR.y4m -pix_fmt yuv420p -f rawvideo -s 720x480  -aspect 3:2  "%%~nR(720x480).yuv"
  ffmpeg -i %%~nR.y4m -pix_fmt yuv420p -f rawvideo -s 352x288  -aspect 11:9 "%%~nR(352x288).yuv"
  ffmpeg -i %%~nR.y4m -pix_fmt yuv420p -f rawvideo -s 176x144  -aspect 11:9 "%%~nR(176x144).yuv"
)
```

Extrait II.1 Information de version de FFmpeg

```
This is a FFmpeg Win64 Static build by Kyle Schwarz.

Zeranoe's FFmpeg Builds Home Page: <http://ffmpeg.zeranoe.com/builds/>

This build was compiled on: Nov 25 2012 12:25:21

FFmpeg version: 2012-11-25 git-26c531c
    libavutil      52.  9.100 / 52.  9.100
    libavcodec     54. 77.100 / 54. 77.100
    libavformat    54. 37.100 / 54. 37.100
    libavdevice    54.  3.100 / 54.  3.100
    libavfilter     3. 23.102 /  3. 23.102
    libswscale      2.  1.102 /  2.  1.102
    libswresample   0. 17.101 /  0. 17.101
    libpostproc    52.  2.100 / 52.  2.100

This FFmpeg was configured with:
    --enable-gpl
    --enable-version3
    --disable-pthreads
    --enable-runtime-cpudetect
    --enable-avisynth
    --enable-bzlib
    --enable-frei0r
    --enable-libass
    --enable-libopencore-amrnb
    --enable-libopencore-amrwb
    --enable-libfreetype
    --enable-libgsm
    --enable-libmp3lame
    --enable-libnut
    --enable-libopenjpeg
    --enable-libopus
    --enable-librtmp
    --enable-libschroedinger
    --enable-libspeex
    --enable-libtheora
    --enable-libutvideo
    --enable-libvo-aacenc
    --enable-libvo-amrwbenc
    --enable-libvorbis
    --enable-libvpx
    --enable-libx264
    --enable-libxavs
    --enable-libxvid
    --enable-zlib

This build was compiled with these external libraries:
    bzip2 1.0.6: <http://www.bzip.org/>
    FreeType 2.4.8: <http://www.freetype.org/>
    Frei0r 20111008-git-16cfe66: <http://www.piksel.org/frei0r/>
    LAME 3.99.5: <http://lame.sourceforge.net/>
    libass 0.10.0: <http://code.google.com/p/libass/>
    libgsm 1.0.13-3: <http://libgsm.sourcearchive.com/>
    Theora 1.1.1: <http://www.theora.org/>
    Vorbis 1.3.2: <http://www.vorbis.com/>
    libvpx 1.1.0: <http://www.webmproject.org/>
    NUT 20110326-git-5e471c2: <git://git.ffmpeg.org/nut>
    OpenCORE AMR 0.1.2: <http://sourceforge.net/projects/opencore-amr/>
    OpenJPEG 1.5.0: <http://www.openjpeg.org/>
    Opus 1.0.1: <http://www.opus-codec.org/>
    RTMPDump 20120308-git-7340f6d: <http://rtmpdump.mplayerhq.hu/>
    Schroedinger 1.0.11: <http://diracvideo.org/>
    Speex 1.2rc1: <http://www.speex.org/>
    Ut Video 10.2.4: <http://umezawa.dyndns.info/archive/utvideo/>
    VisualOn AAC 0.1.2: <http://sourceforge.net/projects/opencore-amr/>
    VisualOn AMR-WB 0.1.2: <http://sourceforge.net/projects/opencore-amr/>
    x264 git-198a7e-20120905: <http://www.videolan.org/developers/x264.html>
    XAVS svn-55: <http://xavs.sourceforge.net/>
    Xvid 1.3.2: <http://www.xvid.org/>
    zlib 1.2.6: <http://zlib.net/>

The source code for this FFmpeg build can be found at: <http://ffmpeg.zeranoe.com/builds/source/ffmpeg/>

This build was compiled on Debian (64-bit): <http://www.debian.org/>

This build was compiled with the MinGW-w64 toolchain: <http://mingw-w64.sourceforge.net/>

License files for each library can be found in the 'licenses' folder.
```

Extrait II.3 Script batch pour l'encodage et décodage H.264 avec FFmpeg

```
@echo off
Setlocal EnableDelayedExpansion

set ffmpeg="./ffmpeg/bin/ffmpeg"

for /f %%R in ('dir /b *.yuv') do (
  for /f "tokens=2 delims=()" %%A in ("%%R") do set res=%%A
  for /f "tokens=1 delims=x" %%A in ("!res!") do set width=%%A
  for /f "tokens=2 delims=x" %%A in ("!res!") do set height=%%A

  set raw=tmp\%%~nR.yuv
  set coded=tmp\%%~nR.mp4
  set decoded=tmp\%%~nR_264qp25.yuv

  echo   Encode
  %ffmpeg% -s !width!x!height! -i !raw! -c:v libx264 -profile:v baseline -qp 25 !coded! 2> nul
  echo   Decode
  %ffmpeg% -i !coded! !decoded! 2> nul
  del /f "!coded!"
)
```

ANNEXE III

PROCÉDURE DE TEST POUR L'ANALYSE DES PRÉDICTEURS DE LA LITTÉRATURE

Ces tests ont été exécutés avec la configuration détaillée dans l'annexe I.

1. Ratio entropique

L'application servant à analyser les prédicteurs utilise un modèle très abrégé de l'encodeur. Seulement la lecture des données et la prédiction sont exécutées. L'entropie est calculée sur chaque bloc 8×8 avant et après la prédiction avec la fonction de l'extrait de code III.1.

Extrait III.1 Code C pour le calcul de l'entropie

```
double _Enthropy2d(
    unsigned char * fi_pbyData,
    unsigned long   fi_ulWidth,
    unsigned long   fi_ulHeight,
    unsigned long   fi_ulStride)
{
    unsigned long   ulNbData;
    unsigned long   ulX;
    unsigned long   ulY;
    unsigned char * pbyDataItr;

    double          dEnthropy;
    double          dProbability;
    unsigned long   aulCount[256]; // 8 bpp: 256

    // calculate the probabilities
    memset(aulCount,0,sizeof(aulCount));
    ulNbData   = 0;
    pbyDataItr = fi_pbyData;
    for(ulY=0; ulY<fi_ulHeight; ulY++)
    {
        for(ulX=0; ulX<fi_ulWidth; ulX++)
        {
            aulCount[*pbyDataItr++]++;
            ulNbData++;
        }
        pbyDataItr += (fi_ulStride-fi_ulWidth);
    }

    //calculate enthropy: -sum(aulProb[i]*log2(aulProb[i]))
    dEnthropy  = 0;
    for(ulY=0; ulY<256; ulY++)
    {
        if(aulCount[ulY])
        {
            dProbability = (double)aulCount[ulY] / (double)ulNbData;
            dEnthropy += dProbability * mLOG2(dProbability);
        }
    }

    return -dEnthropy;
}
```

2. Opérations par pixel

Le nombre d'opérations par pixel ne compte pas la soustraction nécessaire pour appliquer le prédicteur. Seulement les opérations nécessaires pour déterminer le prédicteur sont considérées. Les calculs pour chaque prédicteur sont présentés ci-dessous.

2.1 DAP

Pour calculer le prédicteur DAP, les pixels sans modification sont utilisés comme prédicteur, ce qui ne nécessite aucun calcul.

2.2 MED

Il y a trois formules pour calculer le prédicteur MED. La sélection de la formule utilisée dépend de la relation entre le pixel c et les pixels a et b. La sélection de la formule et le calcul de la formule peut prendre entre 2 et 4 opérations par pixel. Voir le tableau III-1.

Tableau-A III-1 Nombre d'opérations pour le prédicteur MED

Opération	meilleurs cas	pire cas
Comparaisons	2	2
Arithmétiques	0	2
Total	2	4

2.3 GAP

Pour calculer le prédicteur GAP, entre 18 et 36 opérations par pixel sont nécessaires, dépendant du type de contour détecté pour le pixel, voir le tableau III-2.

2.4 HACP

Tableau-A III-2 Nombre d'opérations pour les prédicteurs du GAP

Type de contour détecté	Arithmétiques	Comparaisons	Décalages	Total
fort contour horizontal	11	7	0	18
fort contour vertical	11	8	0	19
contour horizontal	15	9	3	27
faible contour horizontal	16	10	4	30
contour vertical	17	11	5	33
faible contour vertical	18	12	6	36

Le tableau III-3 montre qu'entre zéro et quatre opérations sont nécessaires pour calculer les prédicteurs du HACP.

Tableau-A III-3 Nombre d'opérations pour les prédicteurs du HACP

Prédicteur	Arithmétiques	Décalages	Total
PM	3	1	4
PMV	1	1	2
PCV	0	0	0
PMH	1	1	2
PCH	0	0	0

3. Profondeur de prédiction

Les profondeurs de prédiction pour les prédicteurs DAP, MED et HACP sont tirées de la littérature. L'algorithme de GAP a une profondeur de 21.

ANNEXE IV

ANALYSE DES MÉTHODES DE SIMPLIFICATION DE LA HIÉRARCHIE DE PRÉDICTION DU HACP

Ces tests ont été exécutés avec la configuration détaillée dans l'annexe I.

1. Taux de compression moyen

L'application servant à analyser ces méthodes utilise le modèle complet de l'encodeur, soit la lecture des données, la prédiction et le codage. À la suite de codage, le taux de compression est calculé. Seulement l'étape de la mise en paquet n'est pas exécutée.

2. Nombre d'opérations

Deux calculs de prédicteur sont étudiés. Premièrement, la prédiction par la moyenne (PM), définie par l'équation (4.1) adaptée pour 8 pixels (PM_8). Et deuxièmement, le prédicteur minimisant l'erreur (PME) défini par l'équation (4.5) adaptée pour 8 pixels (PME_8). L'équation A IV-1 montre le nombre d'opération nécessaire pour calculer ce dernier. Le compte des opérations nécessaires est présenté dans le tableau IV-1

$$\textit{Nb opérations}(PME_8) = 9 \times (8 \times \textit{Nb opérations}(BPP)) + \textit{Nb opérations}(PM_8) \qquad \text{(A IV-1)}$$

Tableau-A IV-1 Nombre d'opérations pour les prédicteurs du HACP

Prédicteur	Arithmétiques	Comparaisons	Décalages	Logarithme	Total
PMV	1	0	1	0	2
PM_4	3	0	1	0	4
PM_8	7	0	1	0	8
BPP (pixel positif)	0	2	0	1	3
BPP (pixel negatif)	2	1	0	1	4
PME_8 (min)	7	144	1	72	224
PME_8 (max)	151	72	1	72	296

ANNEXE V

RÉSULTATS COMPLET DES TESTS D'ANALYSE DE L'ALGORITHME PRÉSENTÉ

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
crowd run	0	Sans perte	1.34	99	–	1.0000	100%
crowd run	0	Sans perte	1.38	396	–	1.0000	100%
ducks take off	0	Sans perte	1.39	99	–	1.0000	100%
park joy	0	Sans perte	1.39	99	–	1.0000	100%
park joy	0	Sans perte	1.40	396	–	1.0000	100%
park joy	0	Sans perte	1.40	1350	–	1.0000	100%
ducks take off	0	Sans perte	1.40	396	–	1.0000	100%
park joy	0	Sans perte	1.41	3600	–	1.0000	100%
park joy	0	Sans perte	1.41	8160	–	1.0000	100%
tractor	0	Sans perte	1.42	99	–	1.0000	100%
ducks take off	0	Sans perte	1.43	8160	–	1.0000	100%
crowd run	0	Sans perte	1.43	1350	–	1.0000	100%
zoneplate	25	Sans perte	1.43	99	–	1.0000	100%
crowd run	25	Sans perte	1.43	99	–	1.0000	100%
zoneplate	0	Sans perte	1.44	99	–	1.0000	100%
zoneplate	25	Sans perte	1.44	396	–	1.0000	100%
ducks take off	0	Sans perte	1.45	1350	–	1.0000	100%
riverbed	0	Sans perte	1.45	99	–	1.0000	100%
aspen	0	Sans perte	1.46	99	–	1.0000	100%
zoneplate	0	Sans perte	1.46	396	–	1.0000	100%
zoneplate	25	Sans perte	1.46	3600	–	1.0000	100%
ducks take off	0	Sans perte	1.47	3600	–	1.0000	100%
crowd run	0	Sans perte	1.47	3600	–	1.0000	100%
ducks take off	25	Sans perte	1.47	99	–	1.0000	100%
zoneplate	0	Sans perte	1.47	3600	–	1.0000	100%
ducks take off	25	Sans perte	1.48	396	–	1.0000	100%
crowd run	0	Sans perte	1.48	8160	–	1.0000	100%
zoneplate	0	Sans perte	1.49	1350	–	1.0000	100%
crowd run	25	Sans perte	1.50	396	–	1.0000	100%
blue sky	0	Sans perte	1.51	99	–	1.0000	100%
aspen	0	Sans perte	1.52	396	–	1.0000	100%
ducks take off	25	Sans perte	1.53	8160	–	1.0000	100%
park joy	25	Sans perte	1.53	99	–	1.0000	100%
zoneplate	25	Sans perte	1.54	8160	–	1.0000	100%
zoneplate	0	Sans perte	1.54	8160	–	1.0000	100%
ducks take off	25	Sans perte	1.55	1350	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
aspen	25	Sans perte	1.55	99	–	1.0000	100%
controlled burn	0	Sans perte	1.55	99	–	1.0000	100%
riverbed	0	Sans perte	1.55	396	–	1.0000	100%
tractor	25	Sans perte	1.55	99	–	1.0000	100%
park joy	25	Sans perte	1.55	396	–	1.0000	100%
tractor	0	Sans perte	1.55	396	–	1.0000	100%
park joy	25	Sans perte	1.56	1350	–	1.0000	100%
park joy	25	Sans perte	1.57	8160	–	1.0000	100%
in to tree	0	Sans perte	1.57	8160	–	1.0000	100%
blue sky	0	Sans perte	1.58	396	–	1.0000	100%
crowd run	25	Sans perte	1.58	1350	–	1.0000	100%
life	0	Sans perte	1.59	99	–	1.0000	100%
park joy	25	Sans perte	1.59	3600	–	1.0000	100%
in to tree	0	Sans perte	1.60	3600	–	1.0000	100%
in to tree	0	Sans perte	1.61	1350	–	1.0000	100%
snow mnt	0	Sans perte	1.61	99	–	1.0000	100%
ducks take off	25	Sans perte	1.62	3600	–	1.0000	100%
snow mnt	0	Sans perte	1.62	396	–	1.0000	100%
aspen	0	Sans perte	1.62	1350	–	1.0000	100%
riverbed	25	Sans perte	1.63	99	–	1.0000	100%
blue sky	25	Sans perte	1.63	99	–	1.0000	100%
controlled burn	25	Sans perte	1.64	99	–	1.0000	100%
in to tree	0	Sans perte	1.64	396	–	1.0000	100%
in to tree	0	Sans perte	1.65	99	–	1.0000	100%
aspen	25	Sans perte	1.65	396	–	1.0000	100%
snow mnt	0	Sans perte	1.66	1350	–	1.0000	100%
controlled burn	0	Sans perte	1.66	396	–	1.0000	100%
crowd run	25	Sans perte	1.67	8160	–	1.0000	100%
life	25	Sans perte	1.67	99	–	1.0000	100%
crowd run	25	Sans perte	1.68	3600	–	1.0000	100%
crowd run	0	Avec perte calculée	1.68	99	39.27	0.9960	20.34%
riverbed	0	Avec perte calculée	1.69	396	42.37	0.9954	19.68%
riverbed	0	Avec perte calculée	1.69	99	39.04	0.9920	5.88%
tractor	25	Sans perte	1.69	396	–	1.0000	100%
ducks take off	0	Avec perte calculée	1.70	99	39.24	0.9943	5.88%
west wind easy	0	Sans perte	1.70	99	–	1.0000	100%
park joy	0	Avec perte calculée	1.70	99	39.94	0.9959	27.67%
ducks take off	0	Avec perte calculée	1.70	8160	41.52	0.9993	8.78%
ducks take off	0	Avec perte calculée	1.71	396	39.77	0.9958	12.53%
riverbed	25	Avec perte calculée	1.71	99	41.85	0.9965	52.73%
ducks take off	25	Avec perte calculée	1.71	99	39.52	0.9948	18.75%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
blue sky	25	Sans perte	1.71	396	–	1.0000	100%
tractor	0	Avec perte calculée	1.71	99	40.67	0.9949	14.84%
in to tree	0	Avec perte calculée	1.71	8160	45.73	0.9996	31.39%
crowd run	0	Avec perte calculée	1.71	8160	42.68	0.9993	22.77%
ducks take off	0	Avec perte calculée	1.71	3600	41.13	0.9994	19.62%
blue sky	0	Sans perte	1.71	1350	–	1.0000	100%
park joy	0	Avec perte calculée	1.72	8160	40.92	0.9992	22.75%
park joy	0	Avec perte calculée	1.72	396	39.87	0.9954	21.35%
ducks take off	0	Avec perte calculée	1.72	1350	40.49	0.9992	20.05%
in to tree	0	Avec perte calculée	1.72	3600	45.35	0.9996	45.44%
ducks take off	25	Avec perte calculée	1.73	396	39.99	0.9961	21.87%
crowd run	0	Avec perte calculée	1.73	396	39.82	0.9961	18.47%
park joy	0	Avec perte calculée	1.73	1350	39.83	0.9991	21.38%
park joy	0	Avec perte calculée	1.73	3600	40.22	0.9992	22.02%
life	0	Avec perte calculée	1.73	99	43.27	0.9989	47.7%
crowd run	0	Avec perte calculée	1.73	3600	41.32	0.9994	20.72%
crowd run	25	Avec perte calculée	1.73	99	40.31	0.9970	44.66%
tractor	0	Avec perte calculée	1.73	396	43.28	0.9974	30.86%
riverbed	25	Sans perte	1.74	396	–	1.0000	100%
zoneplate	0	Avec perte calculée	1.74	99	36.19	0.9985	11.76%
riverbed	0	Sans perte	1.74	1350	–	1.0000	100%
crowd run	0	Avec perte calculée	1.74	1350	40.28	0.9994	18.22%
tractor	25	Avec perte calculée	1.74	99	41.15	0.9956	28.01%
rush hour	0	Sans perte	1.74	99	–	1.0000	100%
zoneplate	25	Avec perte calculée	1.75	99	36.22	0.9985	11.76%
controlled burn	0	Avec perte calculée	1.75	99	41.57	0.9973	34.17%
in to tree	0	Avec perte calculée	1.75	1350	44.34	0.9997	40.26%
zoneplate	25	Avec perte calculée	1.75	396	36.06	0.9980	9.09%
station2	0	Sans perte	1.76	99	–	1.0000	100%
ducks take off	25	Avec perte calculée	1.76	8160	41.76	0.9993	25.37%
zoneplate	0	Avec perte calculée	1.76	396	36.04	0.9980	10.61%
aspen	0	Avec perte calculée	1.76	99	40.60	0.9959	30.39%
zoneplate	0	Avec perte calculée	1.76	3600	36.39	0.9990	19.17%
zoneplate	25	Avec perte calculée	1.76	3600	36.38	0.9990	17.83%
tractor	0	Sans perte	1.76	1350	–	1.0000	100%
pedestrian area	0	Sans perte	1.76	99	–	1.0000	100%
snow mnt	0	Sans perte	1.76	3600	–	1.0000	100%
ducks take off	25	Avec perte calculée	1.76	1350	40.72	0.9992	29.61%
west wind easy	0	Avec perte calculée	1.76	99	45.33	0.9987	66.12%
zoneplate	0	Avec perte calculée	1.77	1350	36.52	0.9979	21.33%
west wind easy	25	Sans perte	1.77	99	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
in to tree	0	Avec perte calculée	1.77	99	45.23	0.9989	32.69%
riverbed	25	Avec perte calculée	1.77	396	46.55	0.9985	80.88%
riverbed	0	Avec perte calculée	1.77	1350	50.31	0.9999	77.9%
in to tree	25	Sans perte	1.78	8160	–	1.0000	100%
in to tree	0	Avec perte calculée	1.78	396	44.58	0.9984	36.82%
zoneplate	0	Avec perte calculée	1.78	8160	37.13	0.9900	29.34%
life	0	Sans perte	1.78	396	–	1.0000	100%
controlled burn	25	Sans perte	1.79	396	–	1.0000	100%
motion	25	Sans perte	1.79	99	–	1.0000	100%
life	25	Avec perte calculée	1.79	99	44.35	0.9991	58.89%
park joy	25	Avec perte calculée	1.79	99	40.51	0.9963	36.88%
west wind easy	25	Avec perte calculée	1.79	99	48.61	0.9995	87.95%
zoneplate	25	Avec perte calculée	1.79	8160	37.09	0.9900	27.87%
red kayak	0	Sans perte	1.79	99	–	1.0000	100%
controlled burn	25	Avec perte calculée	1.79	99	42.01	0.9977	45.67%
tractor	25	Avec perte calculée	1.80	396	44.88	0.9983	56.66%
station2	0	Avec perte calculée	1.80	99	48.34	0.9989	77.86%
ducks take off	25	Avec perte calculée	1.80	3600	41.67	0.9995	38.94%
station2	0	Sans perte	1.80	396	–	1.0000	100%
blue sky	0	Avec perte calculée	1.80	99	40.46	0.9983	38.36%
aspen	25	Sans perte	1.81	1350	–	1.0000	100%
snow mnt	25	Sans perte	1.81	99	–	1.0000	100%
crowd run	25	Avec perte calculée	1.81	396	40.51	0.9967	34.8%
dinner	0	Sans perte	1.81	99	–	1.0000	100%
rush hour	0	Avec perte calculée	1.81	99	46.35	0.9991	63.88%
controlled burn	0	Sans perte	1.82	1350	–	1.0000	100%
aspen	0	Avec perte calculée	1.82	396	40.72	0.9962	33.14%
aspen	25	Avec perte calculée	1.82	99	41.00	0.9964	36.47%
aspen	0	Sans perte	1.82	3600	–	1.0000	100%
aspen	0	Sans perte	1.82	3600	–	1.0000	100%
park joy	25	Avec perte calculée	1.83	396	40.12	0.9955	36.44%
station2	0	Avec perte calculée	1.84	396	50.39	0.9991	73.35%
controlled burn	0	Avec perte calculée	1.84	396	42.18	0.9965	45.43%
tractor	0	Avec perte calculée	1.84	1350	48.17	0.9998	67.3%
park joy	25	Avec perte calculée	1.85	1350	40.03	0.9991	37.35%
crowd run	25	Avec perte calculée	1.85	1350	41.06	0.9995	37.47%
motion	0	Sans perte	1.85	99	–	1.0000	100%
pedestrian area	0	Avec perte calculée	1.85	99	46.45	0.9991	60.71%
life	0	Avec perte calculée	1.85	396	46.54	0.9992	70.18%
blue sky	25	Avec perte calculée	1.86	99	40.87	0.9984	47.55%
in to tree	25	Avec perte calculée	1.86	8160	48.00	0.9998	63.77%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
pedestrian area	25	Sans perte	1.86	99	–	1.0000	100%
snow mnt	25	Sans perte	1.86	396	–	1.0000	100%
park joy	25	Avec perte calculée	1.86	3600	40.57	0.9993	40.29%
crowd run	25	Avec perte calculée	1.87	8160	43.62	0.9994	47.72%
aspen	0	Avec perte calculée	1.87	1350	41.50	0.9994	45.65%
park joy	25	Avec perte calculée	1.87	8160	41.25	0.9992	37.55%
station2	0	Sans perte	1.88	1350	–	1.0000	100%
crowd run	25	Avec perte calculée	1.88	3600	42.39	0.9995	47.33%
blue sky	0	Sans perte	1.88	3600	–	1.0000	100%
blue sky	25	Sans perte	1.88	1350	–	1.0000	100%
red kayak	0	Avec perte calculée	1.89	99	44.48	0.9976	66.42%
west wind easy	25	Avec perte rapide	1.89	99	36.56	0.9947	100%
snow mnt	0	Avec perte calculée	1.90	99	39.83	0.9955	32.76%
rush hour	25	Sans perte	1.90	99	–	1.0000	100%
life	0	Sans perte	1.90	1350	–	1.0000	100%
station2	0	Avec perte calculée	1.90	1350	54.73	0.9999	82.97%
snow mnt	25	Sans perte	1.91	1350	–	1.0000	100%
riverbed	0	Sans perte	1.91	3600	–	1.0000	100%
life	0	Sans perte	1.92	8160	–	1.0000	100%
riverbed	0	Avec perte calculée	1.92	3600	64.02	1.0000	96.5%
dinner	25	Sans perte	1.92	99	–	1.0000	100%
snow mnt	0	Sans perte	1.92	8160	–	1.0000	100%
red kayak	0	Sans perte	1.92	396	–	1.0000	100%
blue sky	0	Avec perte calculée	1.92	396	40.38	0.9980	35.16%
riverbed	25	Avec perte rapide	1.92	396	40.90	0.9956	100%
pedestrian area	25	Avec perte calculée	1.92	99	47.91	0.9994	73.62%
dinner	0	Avec perte calculée	1.92	99	46.28	0.9994	65.02%
tractor	25	Sans perte	1.92	1350	–	1.0000	100%
controlled burn	25	Avec perte calculée	1.92	396	43.00	0.9971	57.84%
west wind easy	0	Sans perte	1.92	396	–	1.0000	100%
snow mnt	0	Avec perte calculée	1.93	396	39.73	0.9944	27.96%
rush hour	25	Avec perte calculée	1.93	99	49.00	0.9995	83.72%
aspen	25	Avec perte calculée	1.93	396	41.13	0.9966	42.09%
tractor	0	Sans perte	1.93	3600	–	1.0000	100%
life	0	Avec perte calculée	1.94	1350	49.58	0.9999	82.72%
life	0	Sans perte	1.94	3600	–	1.0000	100%
riverbed	0	Avec perte rapide	1.95	3600	53.37	1.0000	100%
riverbed	0	Avec perte rapide	1.95	1350	43.41	0.9996	100%
snow mnt	0	Avec perte calculée	1.95	1350	39.82	0.9992	35.63%
riverbed	25	Sans perte	1.95	1350	–	1.0000	100%
factory	0	Sans perte	1.95	99	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
life	25	Sans perte	1.96	396	–	1.0000	100%
riverbed	25	Avec perte calculée	1.96	1350	55.32	1.0000	95.79%
tractor	25	Avec perte calculée	1.96	1350	50.19	0.9999	83.41%
controlled burn	25	Sans perte	1.96	1350	–	1.0000	100%
tractor	0	Avec perte calculée	1.96	3600	53.27	0.9999	87.91%
controlled burn	0	Avec perte calculée	1.97	1350	42.81	0.9996	59.63%
life	0	Avec perte calculée	1.97	8160	49.40	0.9999	72.66%
in to tree	25	Sans perte	1.97	99	–	1.0000	100%
west wind easy	0	Avec perte calculée	1.98	396	46.56	0.9988	76.04%
life	0	Avec perte calculée	1.98	3600	51.08	0.9999	84.8%
motion	25	Avec perte calculée	1.98	99	42.70	0.9986	53.55%
station2	0	Sans perte	1.98	3600	–	1.0000	100%
in to tree	25	Avec perte calculée	1.99	99	55.22	0.9999	94.42%
snow mnt	0	Avec perte calculée	1.99	3600	40.67	0.9995	46.6%
blue sky	25	Avec perte calculée	1.99	396	41.01	0.9983	42.94%
aspen	0	Avec perte calculée	1.99	3600	43.22	0.9996	64.46%
aspen	0	Avec perte calculée	1.99	3600	43.22	0.9996	64.46%
station2	0	Avec perte calculée	1.99	3600	60.81	1.0000	96.43%
riverbed	25	Avec perte rapide	1.99	1350	49.42	0.9999	100%
dinner	25	Avec perte calculée	1.99	99	47.19	0.9995	74.84%
life	25	Avec perte calculée	2.00	396	48.90	0.9996	84.19%
pedestrian area	0	Sans perte	2.00	396	–	1.0000	100%
station2	0	Avec perte rapide	2.00	99	40.13	0.9927	100%
red kayak	0	Avec perte calculée	2.00	396	46.07	0.9981	72.24%
west wind easy	25	Sans perte	2.00	396	–	1.0000	100%
snow mnt	25	Sans perte	2.01	3600	–	1.0000	100%
blue sky	0	Avec perte calculée	2.01	1350	41.93	0.9998	45.31%
red kayak	25	Sans perte	2.01	99	–	1.0000	100%
motion	0	Avec perte calculée	2.02	99	42.59	0.9986	57.65%
in to tree	25	Sans perte	2.02	396	–	1.0000	100%
rush hour	0	Sans perte	2.02	396	–	1.0000	100%
snow mnt	25	Avec perte calculée	2.02	99	40.37	0.9963	42.71%
aspen	25	Sans perte	2.02	3600	–	1.0000	100%
station2	0	Avec perte rapide	2.02	3600	50.63	0.9999	100%
west wind easy	25	Avec perte calculée	2.02	396	49.36	0.9996	91.76%
pedestrian area	0	Avec perte calculée	2.03	396	52.89	0.9997	88.03%
in to tree	25	Sans perte	2.03	3600	–	1.0000	100%
in to tree	25	Sans perte	2.03	1350	–	1.0000	100%
rush hour	0	Avec perte calculée	2.03	396	56.31	0.9999	91.36%
red kayak	0	Sans perte	2.04	1350	–	1.0000	100%
controlled burn	0	Sans perte	2.04	3600	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
in to tree	25	Avec perte calculée	2.04	396	52.65	0.9997	89.58%
in to tree	25	Avec perte rapide	2.04	99	42.05	0.9980	100%
factory	0	Avec perte calculée	2.04	99	47.80	0.9994	76.71%
riverbed	0	Sans perte	2.05	8160	–	1.0000	100%
riverbed	0	Avec perte calculée	2.05	8160	72.96	1.0000	99.72%
station2	25	Sans perte	2.05	99	–	1.0000	100%
riverbed	0	Avec perte rapide	2.05	8160	63.28	1.0000	100%
station2	25	Avec perte calculée	2.05	99	67.13	1.0000	99.29%
factory	25	Sans perte	2.06	99	–	1.0000	100%
station2	25	Avec perte rapide	2.06	99	56.14	0.9998	100%
snow mnt	0	Avec perte calculée	2.06	8160	42.41	0.9982	60.4%
aspen	25	Avec perte calculée	2.06	1350	41.94	0.9994	54.77%
red kayak	25	Avec perte calculée	2.06	99	46.85	0.9989	83.59%
in to tree	25	Avec perte calculée	2.07	1350	49.53	0.9999	83.81%
station2	0	Avec perte rapide	2.07	1350	42.66	0.9995	100%
in to tree	25	Avec perte calculée	2.07	3600	49.84	0.9998	83.64%
blue sky	0	Sans perte	2.07	8160	–	1.0000	100%
tractor	0	Avec perte rapide	2.07	3600	45.98	0.9997	100%
west wind easy	0	Avec perte rapide	2.08	99	33.10	0.9859	100%
blue sky	0	Avec perte calculée	2.08	3600	44.73	0.9999	58.96%
tractor	0	Sans perte	2.08	8160	–	1.0000	100%
speed bag	0	Sans perte	2.08	99	–	1.0000	100%
controlled burn	25	Avec perte calculée	2.09	1350	43.53	0.9996	67.37%
rush hour	25	Avec perte rapide	2.09	99	38.37	0.9948	100%
tractor	0	Avec perte calculée	2.09	8160	60.97	1.0000	96.37%
station2	0	Sans perte	2.09	8160	–	1.0000	100%
crowd run	25	Avec perte rapide	2.09	99	28.60	0.9536	100%
station2	0	Avec perte calculée	2.09	8160	67.04	1.0000	99.24%
aspen	0	Sans perte	2.10	8160	–	1.0000	100%
snow mnt	25	Avec perte calculée	2.10	396	40.42	0.9956	45.89%
station2	0	Avec perte rapide	2.10	396	39.91	0.9907	100%
red kayak	0	Avec perte calculée	2.10	1350	46.96	0.9998	82.99%
riverbed	25	Avec perte rapide	2.10	99	35.69	0.9841	100%
speed bag	0	Avec perte calculée	2.10	99	53.86	0.9998	95.21%
station2	0	Avec perte rapide	2.10	8160	58.22	1.0000	100%
west wind easy	25	Avec perte rapide	2.11	396	34.30	0.9933	100%
life	0	Avec perte rapide	2.11	1350	38.25	0.9993	100%
controlled burn	0	Avec perte calculée	2.11	3600	45.31	0.9998	76.59%
tractor	25	Avec perte rapide	2.12	1350	42.06	0.9994	100%
tractor	0	Avec perte rapide	2.12	1350	39.88	0.9989	100%
rush hour	0	Avec perte rapide	2.12	396	42.19	0.9979	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
tractor	0	Avec perte rapide	2.13	8160	52.60	1.0000	100%
life	0	Avec perte rapide	2.13	396	35.32	0.9919	100%
blue sky	25	Sans perte	2.13	3600	–	1.0000	100%
factory	25	Avec perte calculée	2.13	99	48.72	0.9996	81.77%
station2	25	Sans perte	2.13	396	–	1.0000	100%
station2	25	Avec perte calculée	2.13	396	59.73	0.9999	98.38%
life	0	Avec perte rapide	2.14	3600	40.16	0.9994	100%
life	25	Avec perte rapide	2.14	99	32.63	0.9874	100%
blue sky	25	Avec perte calculée	2.14	1350	42.76	0.9998	53.75%
life	25	Avec perte rapide	2.15	396	37.48	0.9955	100%
in to tree	25	Avec perte rapide	2.15	396	42.00	0.9966	100%
station2	25	Avec perte rapide	2.15	396	51.35	0.9994	100%
speed bag	0	Avec perte rapide	2.15	99	41.26	0.9975	100%
rush hour	0	Avec perte rapide	2.15	99	35.21	0.9891	100%
pedestrian area	0	Avec perte rapide	2.15	396	39.59	0.9960	100%
snow mnt	25	Avec perte calculée	2.16	1350	40.53	0.9992	48.32%
tractor	25	Sans perte	2.17	3600	–	1.0000	100%
tractor	25	Avec perte rapide	2.17	396	36.40	0.9886	100%
motion	25	Sans perte	2.17	396	–	1.0000	100%
pedestrian area	25	Sans perte	2.17	396	–	1.0000	100%
dinner	0	Sans perte	2.17	396	–	1.0000	100%
life	0	Avec perte rapide	2.17	99	31.80	0.9845	100%
tractor	25	Avec perte calculée	2.18	3600	57.61	1.0000	94.87%
pedestrian area	25	Avec perte calculée	2.18	396	57.84	0.9999	94.51%
aspen	0	Avec perte calculée	2.18	8160	46.15	0.9997	78.78%
blue sky	0	Avec perte calculée	2.19	8160	47.48	0.9998	72.16%
pedestrian area	0	Sans perte	2.19	1350	–	1.0000	100%
snow mnt	25	Sans perte	2.20	8160	–	1.0000	100%
pedestrian area	0	Avec perte calculée	2.20	1350	58.63	1.0000	96.41%
pedestrian area	25	Avec perte rapide	2.20	99	35.72	0.9909	100%
riverbed	25	Sans perte	2.20	3600	–	1.0000	100%
riverbed	25	Avec perte calculée	2.20	3600	70.48	1.0000	99.5%
red kayak	0	Avec perte rapide	2.20	99	38.19	0.9914	100%
in to tree	0	Avec perte rapide	2.20	3600	37.28	0.9980	100%
riverbed	25	Avec perte rapide	2.21	3600	60.40	1.0000	100%
life	25	Sans perte	2.21	1350	–	1.0000	100%
controlled burn	25	Sans perte	2.21	3600	–	1.0000	100%
aspen	25	Avec perte calculée	2.21	3600	43.61	0.9996	66.99%
west wind easy	0	Sans perte	2.21	1350	–	1.0000	100%
in to tree	25	Avec perte rapide	2.21	8160	39.85	0.9994	100%
life	25	Avec perte calculée	2.22	1350	53.60	1.0000	94.47%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
snow mnt	25	Avec perte calculée	2.23	3600	41.41	0.9995	54.76%
red kayak	25	Sans perte	2.23	396	–	1.0000	100%
rush hour	0	Sans perte	2.23	1350	–	1.0000	100%
park joy	0	Avec perte rapide	2.23	99	28.64	0.9449	100%
rush hour	0	Avec perte calculée	2.23	1350	72.05	1.0000	99.21%
west wind easy	0	Avec perte rapide	2.23	396	33.61	0.9880	100%
red kayak	25	Avec perte rapide	2.23	99	40.75	0.9961	100%
tractor	25	Avec perte rapide	2.24	3600	49.03	0.9998	100%
red kayak	0	Sans perte	2.24	3600	–	1.0000	100%
motion	0	Sans perte	2.24	396	–	1.0000	100%
rush hour	0	Avec perte rapide	2.24	1350	53.93	1.0000	100%
pedestrian area	0	Avec perte rapide	2.24	1350	44.27	0.9997	100%
dinner	0	Avec perte calculée	2.24	396	49.47	0.9996	85.3%
west wind easy	0	Avec perte calculée	2.25	1350	48.87	0.9999	89.72%
controlled burn	0	Sans perte	2.25	8160	–	1.0000	100%
in to tree	25	Avec perte rapide	2.25	3600	42.42	0.9994	100%
in to tree	25	Avec perte rapide	2.25	1350	41.56	0.9995	100%
rush hour	25	Sans perte	2.25	396	–	1.0000	100%
speed bag	25	Sans perte	2.25	99	–	1.0000	100%
dinner	25	Avec perte rapide	2.25	99	34.87	0.9929	100%
pedestrian area	25	Avec perte rapide	2.25	396	42.56	0.9979	100%
rush hour	25	Avec perte calculée	2.25	396	61.91	1.0000	97.94%
red kayak	25	Avec perte calculée	2.26	396	48.81	0.9991	89.19%
controlled burn	0	Avec perte calculée	2.26	8160	51.17	0.9999	93.5%
station2	25	Sans perte	2.26	1350	–	1.0000	100%
speed bag	25	Avec perte calculée	2.26	99	54.65	0.9998	96.56%
dinner	0	Avec perte rapide	2.26	99	33.47	0.9902	100%
pedestrian area	0	Avec perte rapide	2.26	99	34.01	0.9861	100%
station2	25	Avec perte calculée	2.27	1350	62.85	1.0000	99.25%
red kayak	0	Avec perte calculée	2.27	3600	49.40	0.9999	90.53%
factory	0	Sans perte	2.27	396	–	1.0000	100%
ducks take off	25	Avec perte rapide	2.27	3600	37.42	0.9987	100%
station2	25	Avec perte rapide	2.27	1350	55.41	1.0000	100%
life	0	Avec perte rapide	2.28	8160	39.25	0.9997	100%
crowd run	0	Avec perte rapide	2.28	99	27.76	0.9441	100%
controlled burn	25	Avec perte calculée	2.28	3600	46.29	0.9998	81.71%
ducks take off	0	Avec perte rapide	2.28	1350	35.28	0.9977	100%
red kayak	0	Avec perte rapide	2.28	1350	38.15	0.9988	100%
rush hour	25	Avec perte rapide	2.28	396	48.07	0.9994	100%
red kayak	0	Avec perte rapide	2.28	396	37.97	0.9914	100%
pedestrian area	0	Sans perte	2.28	3600	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
pedestrian area	0	Avec perte calculée	2.29	3600	62.90	1.0000	98.73%
factory	0	Avec perte rapide	2.29	99	36.35	0.9926	100%
ducks take off	25	Avec perte rapide	2.29	1350	36.00	0.9981	100%
life	25	Avec perte rapide	2.29	1350	41.97	0.9997	100%
ducks take off	25	Avec perte rapide	2.29	396	34.49	0.9854	100%
rush hour	0	Sans perte	2.29	3600	–	1.0000	100%
rush hour	0	Avec perte calculée	2.29	3600	92.98	1.0000	99.96%
rush hour	0	Avec perte rapide	2.29	3600	69.85	1.0000	100%
controlled burn	25	Avec perte rapide	2.30	99	31.79	0.9801	100%
pedestrian area	0	Avec perte rapide	2.30	3600	50.05	0.9999	100%
aspen	0	Avec perte rapide	2.31	3600	33.16	0.9972	100%
aspen	0	Avec perte rapide	2.31	3600	33.16	0.9972	100%
ducks take off	0	Avec perte rapide	2.31	396	33.83	0.9829	100%
speed bag	25	Avec perte rapide	2.31	99	42.86	0.9983	100%
blue sky	25	Avec perte calculée	2.31	3600	45.44	0.9999	67.39%
ducks take off	0	Avec perte rapide	2.31	3600	36.45	0.9984	100%
blue sky	0	Avec perte rapide	2.31	99	27.52	0.9662	100%
park joy	0	Avec perte rapide	2.31	8160	30.56	0.9972	100%
in to tree	0	Avec perte rapide	2.32	1350	35.95	0.9983	100%
life	25	Sans perte	2.32	3600	–	1.0000	100%
rush hour	0	Sans perte	2.32	8160	–	1.0000	100%
rush hour	0	Avec perte calculée	2.32	8160	114.05	1.0000	100%
rush hour	0	Avec perte rapide	2.32	8160	93.09	1.0000	100%
crowd run	25	Avec perte rapide	2.32	396	28.90	0.9573	100%
tractor	0	Avec perte rapide	2.32	396	34.87	0.9822	100%
dinner	25	Sans perte	2.32	396	–	1.0000	100%
aspen	25	Sans perte	2.32	8160	–	1.0000	100%
park joy	25	Avec perte rapide	2.32	99	29.57	0.9544	100%
park joy	0	Avec perte rapide	2.32	396	28.42	0.9452	100%
life	25	Avec perte calculée	2.33	3600	55.07	1.0000	96.25%
life	25	Sans perte	2.33	8160	–	1.0000	100%
blue sky	25	Avec perte rapide	2.33	99	28.34	0.9714	100%
west wind easy	25	Sans perte	2.33	1350	–	1.0000	100%
factory	25	Avec perte rapide	2.33	99	37.48	0.9946	100%
factory	0	Avec perte calculée	2.34	396	50.44	0.9995	86.32%
park joy	0	Avec perte rapide	2.34	3600	29.33	0.9921	100%
aspen	0	Avec perte rapide	2.34	1350	29.38	0.9913	100%
crowd run	25	Avec perte rapide	2.34	8160	33.24	0.9988	100%
park joy	0	Avec perte rapide	2.34	1350	28.28	0.9879	100%
controlled burn	0	Avec perte rapide	2.34	8160	43.00	0.9999	100%
aspen	0	Avec perte rapide	2.34	99	28.86	0.9433	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
controlled burn	25	Avec perte rapide	2.35	396	32.30	0.9768	100%
snow mnt	25	Avec perte calculée	2.35	8160	42.84	0.9982	66.04%
life	25	Avec perte calculée	2.35	8160	53.28	0.9999	94.42%
crowd run	25	Avec perte rapide	2.35	3600	32.08	0.9962	100%
park joy	25	Avec perte rapide	2.36	396	29.18	0.9547	100%
ducks take off	25	Avec perte rapide	2.36	8160	36.78	0.9993	100%
west wind easy	25	Avec perte calculée	2.36	1350	49.30	0.9999	92.98%
controlled burn	0	Avec perte rapide	2.36	99	31.06	0.9754	100%
pedestrian area	0	Sans perte	2.36	8160	–	1.0000	100%
ducks take off	25	Avec perte rapide	2.36	99	33.88	0.9806	100%
pedestrian area	0	Avec perte calculée	2.36	8160	65.66	1.0000	99.45%
controlled burn	0	Avec perte rapide	2.36	396	31.24	0.9703	100%
motion	25	Avec perte calculée	2.37	396	43.21	0.9988	68.34%
park joy	25	Avec perte rapide	2.37	3600	30.00	0.9935	100%
crowd run	0	Avec perte rapide	2.37	396	28.15	0.9485	100%
pedestrian area	0	Avec perte rapide	2.37	8160	55.29	1.0000	100%
park joy	25	Avec perte rapide	2.37	1350	28.93	0.9897	100%
dinner	25	Avec perte calculée	2.37	396	50.41	0.9997	89.28%
west wind easy	0	Avec perte rapide	2.37	1350	32.24	0.9977	100%
life	25	Avec perte rapide	2.38	3600	44.69	0.9998	100%
controlled burn	0	Avec perte rapide	2.38	3600	36.33	0.9989	100%
riverbed	25	Sans perte	2.38	8160	–	1.0000	100%
in to tree	0	Avec perte rapide	2.38	8160	37.44	0.9988	100%
riverbed	25	Avec perte calculée	2.38	8160	76.32	1.0000	99.88%
controlled burn	0	Avec perte rapide	2.38	1350	32.01	0.9961	100%
riverbed	25	Avec perte rapide	2.38	8160	66.52	1.0000	100%
red kayak	0	Avec perte rapide	2.38	3600	41.78	0.9996	100%
crowd run	25	Avec perte rapide	2.38	1350	30.13	0.9942	100%
crowd run	0	Avec perte rapide	2.39	8160	32.46	0.9984	100%
tractor	25	Avec perte rapide	2.39	99	33.09	0.9776	100%
speed bag	0	Sans perte	2.39	396	–	1.0000	100%
red kayak	25	Avec perte rapide	2.40	396	40.74	0.9962	100%
park joy	25	Avec perte rapide	2.40	8160	30.90	0.9976	100%
speed bag	0	Avec perte calculée	2.40	396	58.13	0.9999	97.86%
aspen	25	Avec perte rapide	2.40	99	29.19	0.9480	100%
aspen	0	Avec perte rapide	2.40	396	28.49	0.9463	100%
motion	0	Avec perte calculée	2.41	396	44.01	0.9991	74.69%
aspen	0	Avec perte rapide	2.41	8160	37.77	0.9997	100%
dinner	0	Avec perte rapide	2.41	396	38.19	0.9964	100%
crowd run	0	Avec perte rapide	2.42	3600	30.95	0.9951	100%
tractor	0	Avec perte rapide	2.42	99	32.13	0.9703	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
tractor	25	Sans perte	2.42	8160	–	1.0000	100%
crowd run	0	Avec perte rapide	2.42	1350	29.23	0.9929	100%
life	25	Avec perte rapide	2.42	8160	43.56	0.9999	100%
tractor	25	Avec perte calculée	2.43	8160	63.17	1.0000	97.48%
speed bag	0	Avec perte rapide	2.43	396	46.31	0.9993	100%
in to tree	0	Avec perte rapide	2.43	396	35.52	0.9823	100%
ducks take off	0	Avec perte rapide	2.43	99	33.12	0.9770	100%
aspen	25	Avec perte calculée	2.43	8160	46.24	0.9997	78.76%
station2	25	Sans perte	2.44	3600	–	1.0000	100%
station2	25	Avec perte calculée	2.44	3600	70.38	1.0000	99.68%
riverbed	0	Avec perte rapide	2.44	396	36.14	0.9829	100%
station2	25	Avec perte rapide	2.45	3600	60.20	1.0000	100%
red kayak	25	Sans perte	2.45	1350	–	1.0000	100%
west wind easy	25	Avec perte rapide	2.45	1350	31.48	0.9972	100%
controlled burn	25	Avec perte rapide	2.46	1350	32.83	0.9967	100%
motion	25	Avec perte rapide	2.46	99	26.08	0.9400	100%
motion	0	Avec perte rapide	2.46	99	26.14	0.9421	100%
zoneplate	0	Avec perte rapide	2.46	396	14.13	0.7306	100%
tractor	25	Avec perte rapide	2.46	8160	54.08	1.0000	100%
ducks take off	0	Avec perte rapide	2.46	8160	35.84	0.9990	100%
factory	25	Sans perte	2.47	396	–	1.0000	100%
snow mnt	0	Avec perte rapide	2.47	8160	34.50	0.9995	100%
blue sky	0	Avec perte rapide	2.49	3600	36.09	0.9991	100%
blue sky	0	Avec perte rapide	2.49	8160	40.16	0.9998	100%
aspen	25	Avec perte rapide	2.49	396	28.82	0.9513	100%
red kayak	25	Avec perte calculée	2.50	1350	48.75	0.9998	90.27%
blue sky	25	Sans perte	2.50	8160	–	1.0000	100%
zoneplate	0	Avec perte rapide	2.50	8160	16.91	0.9570	100%
dinner	25	Avec perte rapide	2.51	396	39.61	0.9975	100%
snow mnt	0	Avec perte rapide	2.51	3600	31.52	0.9968	100%
factory	25	Avec perte calculée	2.51	396	51.77	0.9997	90.51%
controlled burn	25	Avec perte rapide	2.51	3600	37.15	0.9991	100%
zoneplate	25	Avec perte rapide	2.52	8160	16.91	0.9570	100%
zoneplate	0	Avec perte rapide	2.52	1350	15.42	0.8969	100%
factory	0	Avec perte rapide	2.52	396	39.01	0.9953	100%
blue sky	0	Avec perte rapide	2.52	396	27.81	0.9678	100%
aspen	25	Avec perte rapide	2.52	1350	29.70	0.9920	100%
blue sky	0	Avec perte rapide	2.53	1350	31.38	0.9979	100%
zoneplate	25	Avec perte rapide	2.53	99	13.82	0.7088	100%
pedestrian area	25	Sans perte	2.54	1350	–	1.0000	100%
in to tree	0	Avec perte rapide	2.54	99	35.24	0.9882	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
red kayak	0	Sans perte	2.54	8160	–	1.0000	100%
pedestrian area	25	Avec perte calculée	2.54	1350	61.30	1.0000	98.73%
zoneplate	25	Avec perte rapide	2.54	3600	13.80	0.8697	100%
zoneplate	0	Avec perte rapide	2.55	3600	13.75	0.8687	100%
zoneplate	0	Avec perte rapide	2.55	99	13.80	0.7088	100%
speed bag	0	Sans perte	2.55	1350	–	1.0000	100%
speed bag	0	Avec perte calculée	2.55	1350	60.80	1.0000	98.92%
red kayak	0	Avec perte calculée	2.55	8160	54.22	0.9999	96.94%
controlled burn	25	Sans perte	2.56	8160	–	1.0000	100%
pedestrian area	25	Avec perte rapide	2.56	1350	47.32	0.9999	100%
factory	0	Sans perte	2.56	1350	–	1.0000	100%
speed bag	0	Avec perte rapide	2.57	1350	48.12	0.9999	100%
blue sky	25	Avec perte rapide	2.57	396	28.48	0.9726	100%
controlled burn	25	Avec perte calculée	2.57	8160	52.30	0.9999	96.11%
west wind easy	0	Sans perte	2.58	3600	–	1.0000	100%
dinner	0	Sans perte	2.58	1350	–	1.0000	100%
aspen	25	Avec perte rapide	2.58	3600	33.35	0.9973	100%
west wind easy	0	Avec perte calculée	2.58	3600	58.13	1.0000	98.74%
zoneplate	25	Avec perte rapide	2.58	396	14.12	0.7286	100%
snow mnt	0	Avec perte rapide	2.59	1350	28.95	0.9922	100%
speed bag	0	Sans perte	2.60	3600	–	1.0000	100%
speed bag	0	Avec perte calculée	2.60	3600	61.71	1.0000	99.38%
red kayak	0	Avec perte rapide	2.60	8160	47.85	1.0000	100%
west wind easy	0	Avec perte rapide	2.60	3600	43.86	0.9998	100%
riverbed	0	Avec perte rapide	2.60	99	33.29	0.9694	100%
factory	0	Avec perte calculée	2.61	1350	52.80	0.9999	91.34%
speed bag	0	Avec perte rapide	2.61	3600	52.00	1.0000	100%
dinner	0	Avec perte calculée	2.61	1350	53.97	1.0000	93.28%
snow mnt	0	Avec perte rapide	2.62	99	28.43	0.9506	100%
speed bag	0	Sans perte	2.62	8160	–	1.0000	100%
speed bag	0	Avec perte calculée	2.63	8160	63.22	1.0000	99.65%
blue sky	25	Avec perte calculée	2.63	8160	48.53	0.9999	77.93%
speed bag	0	Avec perte rapide	2.63	8160	55.00	1.0000	100%
controlled burn	25	Avec perte rapide	2.63	8160	44.82	0.9999	100%
red kayak	25	Avec perte rapide	2.64	1350	40.00	0.9993	100%
station2	25	Sans perte	2.64	8160	–	1.0000	100%
station2	25	Avec perte calculée	2.64	8160	73.35	1.0000	99.73%
blue sky	25	Avec perte rapide	2.65	1350	32.12	0.9982	100%
station2	25	Avec perte rapide	2.65	8160	62.12	1.0000	100%
factory	25	Avec perte rapide	2.66	396	40.36	0.9968	100%
rush hour	25	Sans perte	2.66	1350	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
rush hour	25	Avec perte calculée	2.66	1350	82.52	1.0000	99.9%
rush hour	25	Avec perte rapide	2.66	1350	62.33	1.0000	100%
blue sky	25	Avec perte rapide	2.70	3600	36.91	0.9993	100%
snow mnt	0	Avec perte rapide	2.70	396	27.98	0.9507	100%
motion	25	Sans perte	2.71	1350	–	1.0000	100%
snow mnt	25	Avec perte rapide	2.72	99	29.05	0.9585	100%
dinner	0	Avec perte rapide	2.72	1350	43.83	0.9998	100%
red kayak	25	Sans perte	2.72	3600	–	1.0000	100%
aspen	25	Avec perte rapide	2.72	8160	37.81	0.9997	100%
west wind easy	25	Sans perte	2.74	3600	–	1.0000	100%
west wind easy	25	Avec perte calculée	2.75	3600	57.75	1.0000	98.85%
red kayak	25	Avec perte calculée	2.75	3600	51.22	0.9999	94.34%
factory	0	Avec perte rapide	2.75	1350	41.72	0.9995	100%
snow mnt	25	Avec perte rapide	2.75	396	28.85	0.9624	100%
motion	0	Avec perte rapide	2.76	396	28.09	0.9683	100%
west wind easy	25	Avec perte rapide	2.77	3600	42.08	0.9997	100%
dinner	25	Sans perte	2.78	1350	–	1.0000	100%
motion	0	Sans perte	2.78	1350	–	1.0000	100%
dinner	25	Avec perte calculée	2.80	1350	56.34	1.0000	96.14%
snow mnt	25	Avec perte rapide	2.80	1350	29.53	0.9935	100%
snow mnt	25	Avec perte rapide	2.80	3600	31.93	0.9971	100%
speed bag	25	Sans perte	2.80	396	–	1.0000	100%
speed bag	25	Avec perte calculée	2.80	396	61.73	1.0000	99.08%
motion	25	Avec perte rapide	2.81	396	27.16	0.9636	100%
factory	0	Sans perte	2.81	3600	–	1.0000	100%
snow mnt	25	Avec perte rapide	2.82	8160	34.88	0.9995	100%
speed bag	25	Avec perte rapide	2.82	396	48.77	0.9996	100%
factory	0	Avec perte calculée	2.84	3600	54.83	1.0000	93.76%
motion	25	Sans perte	2.85	3600	–	1.0000	100%
red kayak	25	Avec perte rapide	2.85	3600	43.39	0.9997	100%
pedestrian area	25	Sans perte	2.85	3600	–	1.0000	100%
pedestrian area	25	Avec perte calculée	2.86	3600	65.21	1.0000	99.61%
dinner	25	Avec perte rapide	2.86	1350	45.98	0.9999	100%
pedestrian area	25	Avec perte rapide	2.86	3600	53.09	1.0000	100%
factory	25	Sans perte	2.89	1350	–	1.0000	100%
motion	0	Sans perte	2.90	3600	–	1.0000	100%
factory	25	Avec perte calculée	2.92	1350	54.86	1.0000	95.46%
dinner	0	Sans perte	2.94	3600	–	1.0000	100%
dinner	0	Avec perte calculée	2.96	3600	57.40	1.0000	96.82%
motion	25	Avec perte calculée	2.96	1350	44.19	0.9997	81.25%
factory	0	Sans perte	2.96	8160	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
factory	0	Avec perte rapide	2.97	3600	44.56	0.9997	100%
blue sky	25	Avec perte rapide	2.97	8160	40.85	0.9999	100%
factory	0	Avec perte calculée	3.00	8160	55.86	1.0000	94.42%
factory	25	Avec perte rapide	3.01	1350	43.69	0.9997	100%
west wind easy	0	Sans perte	3.02	8160	–	1.0000	100%
rush hour	25	Sans perte	3.02	3600	–	1.0000	100%
rush hour	25	Avec perte calculée	3.02	3600	113.60	1.0000	100%
rush hour	25	Avec perte rapide	3.02	3600	90.04	1.0000	100%
west wind easy	0	Avec perte calculée	3.02	8160	68.06	1.0000	99.81%
dinner	0	Avec perte rapide	3.02	3600	47.91	0.9999	100%
west wind easy	0	Avec perte rapide	3.03	8160	55.67	1.0000	100%
motion	0	Avec perte calculée	3.04	1350	44.33	0.9997	82.31%
red kayak	25	Sans perte	3.08	8160	–	1.0000	100%
motion	25	Avec perte calculée	3.09	3600	45.36	0.9998	83.96%
red kayak	25	Avec perte calculée	3.10	8160	55.56	0.9999	97.95%
factory	0	Avec perte rapide	3.12	8160	45.59	0.9999	100%
red kayak	25	Avec perte rapide	3.14	8160	49.20	1.0000	100%
motion	0	Avec perte calculée	3.15	3600	45.51	0.9999	84.79%
dinner	0	Sans perte	3.18	8160	–	1.0000	100%
dinner	0	Avec perte calculée	3.20	8160	57.68	1.0000	97.23%
dinner	25	Sans perte	3.20	3600	–	1.0000	100%
dinner	25	Avec perte calculée	3.21	3600	60.39	1.0000	98.68%
dinner	25	Avec perte rapide	3.24	3600	51.28	1.0000	100%
motion	25	Sans perte	3.26	8160	–	1.0000	100%
motion	0	Sans perte	3.26	8160	–	1.0000	100%
dinner	0	Avec perte rapide	3.26	8160	48.47	1.0000	100%
pedestrian area	25	Sans perte	3.27	8160	–	1.0000	100%
pedestrian area	25	Avec perte calculée	3.27	8160	67.67	1.0000	99.74%
west wind easy	25	Sans perte	3.27	8160	–	1.0000	100%
west wind easy	25	Avec perte calculée	3.28	8160	68.29	1.0000	99.82%
factory	25	Sans perte	3.28	3600	–	1.0000	100%
pedestrian area	25	Avec perte rapide	3.28	8160	57.35	1.0000	100%
west wind easy	25	Avec perte rapide	3.28	8160	55.61	1.0000	100%
factory	25	Avec perte calculée	3.29	3600	58.72	1.0000	97.82%
factory	25	Avec perte rapide	3.35	3600	47.92	0.9999	100%
speed bag	25	Sans perte	3.36	1350	–	1.0000	100%
speed bag	25	Avec perte calculée	3.37	1350	64.71	1.0000	99.63%
speed bag	25	Avec perte rapide	3.38	1350	51.13	1.0000	100%
motion	25	Avec perte rapide	3.41	1350	22.25	0.9644	100%
motion	0	Avec perte rapide	3.49	1350	22.27	0.9647	100%
motion	25	Avec perte rapide	3.50	3600	28.81	0.9957	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
motion	0	Avec perte rapide	3.55	3600	28.87	0.9958	100%
motion	0	Avec perte calculée	3.58	8160	46.01	0.9999	86.86%
motion	25	Avec perte calculée	3.58	8160	46.01	0.9999	86.57%
sintel trailer	0	Sans perte	3.59	99	–	1.0000	100%
sintel trailer	0	Avec perte calculée	3.61	99	58.63	1.0000	96.82%
rush hour	25	Sans perte	3.66	8160	–	1.0000	100%
rush hour	25	Avec perte calculée	3.66	8160	–	1.0000	100%
rush hour	25	Avec perte rapide	3.66	8160	–	1.0000	100%
sintel trailer	0	Avec perte rapide	3.72	99	45.78	0.9994	100%
factory	25	Sans perte	3.80	8160	–	1.0000	100%
factory	25	Avec perte calculée	3.82	8160	58.98	1.0000	97.71%
speed bag	25	Sans perte	3.83	3600	–	1.0000	100%
speed bag	25	Avec perte calculée	3.83	3600	64.41	1.0000	99.78%
speed bag	25	Avec perte rapide	3.84	3600	55.32	1.0000	100%
sintel trailer	25	Sans perte	3.89	99	–	1.0000	100%
dinner	25	Sans perte	3.90	8160	–	1.0000	100%
sintel trailer	25	Avec perte calculée	3.90	99	60.42	1.0000	98.39%
factory	25	Avec perte rapide	3.90	8160	48.65	1.0000	100%
dinner	25	Avec perte calculée	3.91	8160	60.51	1.0000	98.77%
motion	0	Avec perte rapide	3.94	8160	33.78	0.9993	100%
dinner	25	Avec perte rapide	3.95	8160	51.26	1.0000	100%
motion	25	Avec perte rapide	3.95	8160	33.75	0.9993	100%
sintel trailer	25	Avec perte rapide	3.96	99	48.69	0.9997	100%
sintel trailer	0	Sans perte	4.15	396	–	1.0000	100%
sintel trailer	0	Avec perte calculée	4.16	396	59.39	1.0000	98.57%
sintel trailer	0	Avec perte rapide	4.22	396	41.67	0.9993	100%
speed bag	25	Avec perte calculée	4.58	8160	64.60	1.0000	99.84%
speed bag	25	Sans perte	4.58	8160	–	1.0000	100%
speed bag	25	Avec perte rapide	4.59	8160	57.33	1.0000	100%
sintel trailer	25	Sans perte	4.60	396	–	1.0000	100%
sintel trailer	25	Avec perte calculée	4.61	396	60.81	1.0000	99.35%
sintel trailer	25	Avec perte rapide	4.64	396	42.04	0.9994	100%
sintel trailer	0	Sans perte	4.65	1350	–	1.0000	100%
sintel trailer	0	Avec perte calculée	4.66	1350	60.86	1.0000	99.14%
sintel trailer	0	Avec perte rapide	4.70	1350	44.05	0.9999	100%
sintel trailer	0	Sans perte	5.02	3600	–	1.0000	100%
sintel trailer	0	Avec perte calculée	5.02	3600	63.72	1.0000	99.37%
sintel trailer	0	Avec perte rapide	5.07	3600	51.24	1.0000	100%
sintel trailer	0	Sans perte	5.29	8160	–	1.0000	100%
sintel trailer	0	Avec perte calculée	5.30	8160	63.15	1.0000	99.35%
sintel trailer	25	Sans perte	5.34	1350	–	1.0000	100%

Clip	QP	Mode	Taux de compression	Nombre de macroblocs par image	PSNR	SSIM	Pourcentage d'unités codées sans perte
sintel trailer	25	Avec perte calculée	5.35	1350	61.82	1.0000	99.65%
sintel trailer	0	Avec perte rapide	5.35	8160	52.42	1.0000	100%
sintel trailer	25	Avec perte rapide	5.37	1350	44.36	0.9999	100%
sintel trailer	25	Avec perte calculée	5.94	3600	66.24	1.0000	99.84%
sintel trailer	25	Sans perte	5.95	3600	–	1.0000	100%
sintel trailer	25	Avec perte rapide	5.96	3600	51.69	1.0000	100%
sintel trailer	25	Sans perte	6.83	8160	–	1.0000	100%
sintel trailer	25	Avec perte calculée	6.84	8160	65.60	1.0000	99.74%
sintel trailer	25	Avec perte rapide	6.87	8160	54.41	1.0000	100%

Tableau V.1 Résultats expérimentaux

BIBLIOGRAPHIE

Acharya, S. et G. Petrin. Mai 2012. « Ultra High Definition Television : Threshold of a new age - ITU Recommendations on UHDTV standards agreed ». http://www.itu.int/net/pressoffice/press_releases/2012/31.aspx. Accessed : 2013-02-11.

Duce, D. Décembre 2003. « Portable Network Graphics (PNG) Specification (Second Edition) ». <http://www.w3.org/TR/PNG>.

Frazier, H. 1998. « The 802.3z Gigabit Ethernet Standard ». *Network, IEEE*, vol. 12, n° 3, p. 6-7.

Hornig, C. Avril 1984. « A Standard for the Transmission of IP Datagrams over Ethernet Networks ». RFC 894 (Standard). <http://www.ietf.org/rfc/rfc894.txt>.

IEEE 802.3 Working Group. 1999. « IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. Supplement to Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Physical Layer Parameters and Specifications for 1000 Mb/s Operation Over 4-Pair of Category 5 Balanced Copper Cabling, Type 1000BASE-T ». *IEEE Std 802.3ab-1999*.

IEEE 802.3 working group. Décembre 2012. IEEE Std 802.3 - IEEE Standard for Ethernet.

International Telecommunications Union. Avril 2004. ITU-T Recommendation T.81 T.81 : Information technology - Digital compression and coding of continuous-tone still images . http ://www.itu.int/rec/T-REC-T.81/en.

International Telecommunications Union. Avril 2013a. ITU-T Recommendation H.264 : Advanced video coding for generic audiovisual services. http ://www.itu.int/rec/T-REC-H.264/en.

International Telecommunications Union. Avril 2013b. ITU-T Recommendation H.265 : High efficiency video coding. http ://www.itu.int/rec/T-REC-H.265/en.

ISO/IEC. Septembre 2009. ISO/IEC 14496-2 :2004 : Information technology – Coding of audio-visual objects – Part 2 : Visual.

Kim, J. et C.-M. Kyung. Juin 2010. « A Lossless Embedded Compression Using Significant Bit Truncation for HD Video Coding ». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, n° 6, p. 848–860.

Martin, E. S. G. Carotti et J. C. De. 2005. « Lossless video coding using multi-frame motion compensation ». In *EURASIP European Signal Processing Conf.*

Octasic. Décembre 2011. *Opus2 Instruction Set Reference*. Octasic Inc., 4101 Molson St., Suite 300, Montreal, Quebec, Canada, H1Y 3L1. OCTDSPRM8000 Rev. 1.02.

Postel, J. Août 1980. « User Datagram Protocol ». RFC 768 (Standard). <http://www.ietf.org/rfc/rfc768.txt>.

Postel, J. Septembre 1981. « Internet Protocol ». RFC 791 (Standard). <http://www.ietf.org/rfc/rfc791.txt>. Updated by RFCs 1349, 2474.

Pritchard, D. Novembre 1977. « US Color Television Fundamentals-a Review ». *IEEE Transactions on Consumer Electronics*, vol. CE-23, n° 4, p. 467–478.

Richardson, Iain E. G., 2003. *H.264 and MPEG-4 Video Compression - Video Coding for Next-generation Multimedia*. John Wiley and Sons, Inc.

Robson, J. G. Août 1966. « Spatial and Temporal Contrast-Sensitivity Functions of the Visual System ». *Journal of the Optical Society of America*, vol. 56, n° 8, p. 1141.

Sayood, K., Décembre 2005. *Introduction to Data Compression, Third Edition (Morgan Kaufmann Series in Multimedia Information and Systems)*. éd. 3. Morgan Kaufmann.

Schulzrinne, H., S. Casner, R. Frederick, et V. Jacobson. Juillet 2003. « RTP : A Transport Protocol for Real-Time Applications ». RFC 3550 (Standard). <http://www.ietf.org/rfc/rfc3550.txt>. Updated by RFCs 5506, 5761, 6051, 6222.

Shafranovich, Y. Octobre 2005. « Common Format and MIME Type for Comma-Separated Values (CSV) Files ». RFC 4180 (Informational). <http://www.ietf.org/rfc/rfc4180.txt>.

Shannon, C. E. 1948. « The mathematical theory of communication ». *Bell System Technical Journal*, vol. 27, n° July and October, p. 279–423,623–656.

Song, T. et T. Shimamoto. 2007. « Reference frame data compression method for H.264/AVC ». *IEICE Electronics Express*, vol. 4, n° 3, p. 121–126.

Tjdistler. 2011. « Image Quality Assessment (IQA) ». <http://sourceforge.net/projects/iqa/>.

Võ, D. T., S. Lertrattanapanich, et Y.-T. Kim. 2011a. « Visually Lossless Compression for Color Images with Low Line Memory Requirement using Non-uniform Quantizers ». In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*. p. 187–195.

Võ, D. T., S. Lertrattanapanich, et Y.-T. Kim. 2011b. « Low Line Memory Visually Lossless Compression for Color Images using Non-uniform Quantizers ». *Consumer Electronics, IEEE Transactions on*, vol. 57, p. 187–195.

von Neumann, John. Octobre 1993. « First Draft of a Report on the EDVAC ». *IEEE Ann. Hist. Comput.*, vol. 15, n° 4, p. 27–75.

Wang, Y., J. Ostermann, et Y.-Q. Zhang, 2002. *Video Processing and Communications*. Prentice Hall Signal Processing Series. Prentice Hall.

Weinberger, M. J., G. Seroussi, et G. Sapiro. Janvier 2000. « The LOCO-I lossless image compression algorithm : principles and standardization into JPEG-LS. ». *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 9, n° 8, p. 1309–24.

Wu, X. et N. Memon. Avril 1997. « Context-based, adaptive, lossless image coding ». *IEEE Transactions on Communications*, vol. 45, n° 4, p. 437–444.

Young, T. Janvier 1802. « The Bakerian Lecture : On the Theory of Light and Colours ». *Philosophical Transactions of the Royal Society of London*, vol. 92, p. 12–48.