ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DU
DOCTORAT EN GÉNIE
Ph.D.

PAR
Jean-François FRANCHE

TRANSCODAGE RAPIDE DE H.264 À HEVC BASÉ SUR LA PROPAGATION DU MOUVEMENT ET UNE TRAVERSÉE POSTFIXE DES UNITÉS DE CODAGE ARBORESCENT

MONTRÉAL, LE 19 DÉCEMBRE 2016

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE:

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

Mme Catherine Laporte, présidente du jury
Département de génie électrique à l'École de technologie supérieure

M. André Zaccarin, examinateur externe
Département de génie électrique et génie informatique à l'Université Laval

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

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

LE 26 OCTOBRE 2016

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à remercier chaleureusement mon directeur de recherche, Stéphane Coulombe, pour son immense support durant la réalisation de ce travail. Il a grandement contribué à l'aboutissement de ce projet par son soutien moral et technique, ainsi que par sa disponibilité. Je remercie également mes confrères du laboratoire, en particulier Luc Trudeau et Nick Desjardins.

J'exprime également mes remerciements aux membres du jury, Catherine Laporte, André Zaccarin et Carlos Vazquez, qui ont accepté de consacrer une partie de leur précieux temps à l'évaluation de ma thèse.

Merci à Vantrix, entreprise québécoise spécialisée dans l'optimisation de services vidéos mobiles et dans la livraison de solutions en matière de vidéo en continu, de navigation web et de messagerie, pour sa contribution financière, technique et matérielle. Merci aussi au conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) pour son apport financier.

Je remercie également ma conjointe, que j'ai eu la chance de rencontrer durant mon doctorat, et qui m'a appuyé de manière soutenue durant mes études. Enfin, j'aimerais remercier mes parents, mes frères, ma sœur et mes amis pour leur soutien moral tout au long de ce travail.

TRANSCODAGE RAPIDE DE H.264 À HEVC BASÉ SUR LA PROPAGATION DU MOUVEMENT ET UNE TRAVERSÉE POSTFIXE DES UNITÉS DE CODAGE ARBORESCENT

Jean-François FRANCHE

RÉSUMÉ

En 2013, l'ITU-T et l'ISO ont publié conjointement le plus récent standard de compression vidéo, appelé HEVC. Comparé à son prédécesseur, H.264, ce nouveau standard réduit le débit d'environ 50% pour une qualité vidéo similaire. Pour bénéficier de cette plus grande efficacité de codage, et pour assurer l'interopérabilité entre les systèmes, plusieurs séquences vidéos H.264 doivent être transcodées (converties) en séquences HEVC. La manière la plus simple de réaliser cette opération consiste à décoder entièrement la séquence H.264 source, puis à la réencoder entièrement à l'aide d'un encodeur HEVC. Cette approche, appelée transcodage en cascade dans le domaine des pixels (TCDP), produit un codage efficace et offre un maximum de flexibilité, notamment en ce qui a trait à la configuration de la séquence de sortie. Cependant, elle est très complexe en calculs. Pour réduire cette complexité, plusieurs approches réutilisent de l'information de codage (vecteurs de mouvement, modes de codage, données résiduelles, etc.) extraite de la séquence H.264, afin d'accélérer certaines étapes de l'encodage HEVC. La majorité de ces approches préserve l'efficacité de codage, mais obtient cependant des accélérations limitées (habituellement, entre 2 et 4x, selon l'approche).

Dans cette thèse, nous proposons une approche de transcodage H.264 à HEVC plus rapide que celles présentées dans la littérature. Notre solution est composée d'un algorithme de propagation du mouvement et d'une méthode pour réduire le nombre de modes HEVC à tester. L'algorithme de propagation de mouvement crée une liste des vecteurs de mouvement candidats au niveau des unités de codage arborescent (CTU) et, par la suite, sélectionne le meilleur candidat au niveau des unités de prédiction. Cette méthode élimine la redondance des calculs en précalculant l'erreur de prédiction de chaque candidat au niveau des CTUs, et réutilise cette information pour différentes tailles de partitionnement. Pour sa part, l'algorithme de réduction des modes est basé sur un parcours postfixe de la CTU traitée. Cet algorithme permet notamment d'arrêter prématurément le traitement d'un mode jugé non prometteur.

Par rapport à une approche de transcodage TCDP, nos résultats expérimentaux montrent que la solution proposée est en moyenne 7.81 fois plus rapide, pour une augmentation moyenne du BD-Rate de 2.05%. Nous expériences montrent également que les résultats obtenus sont significativement supérieurs à ceux de l'état de l'art.

Mots clés: H.264/AVC, H.265/HEVC, transcodage vidéo, estimation du mouvement, sélection du mode à encoder

FAST H.264-TO-HEVC TRANSCODING BASED ON MOTION PROPAGATION AND POST-ORDER TRAVERSAL OF CODING TREE UNITS

Jean-François FRANCHE

ABSTRACT

In 2013, the Joint Collaborative Team on Video Coding completed HEVC, the most recent video compression standard. Compared to its predecessor, H.264, the new standard HEVC can save approximately 50% of the bitrate for similar video quality. To take advantage of HEVC coding efficiency and to ensure systems interoperability, several H.264 sequences must be transcoded to HEVC. The simplest video transcoding approach, called cascade pixel-domain transcoding (CPDT), decodes the input sequence entirely and re-encodes the pixel data in the output format. This approach achieves high coding efficiency and offers great flexibility on video encoding parameters. However, it is very complex computationally. To reduce this complexity, several approaches reuse extracted information – such as coding modes, motion information and encoded residuals – from the incoming bitstream to speed up the HEVC encoding process. For inter frames, these approaches typically focus on fast mode decision and fast motion estimation. Most of these approaches preserve the coding efficiency, but achieve limited speedup (usually between 2 and 4x).

In this thesis, we propose a faster H.264 to HEVC transcoder that also preserves the coding efficiency. Our solution is composed of a motion propagation algorithm and a fast mode decision framework. The motion propagation algorithm creates a motion vector candidate list at the coding tree unit (CTU) level and, thereafter, selects the best candidate at the prediction unit level. This method eliminates computational redundancy by pre-computing the prediction error of each candidate at the CTU level and reusing the information for various partition sizes. The fast mode decision framework is based on a post-order traversal of the CTU, and includes several mode reduction techniques. In particular, the framework permits the early termination of the rate-distortion cost computation, a highly complex task, when a mode is unpromising. Moreover, a method is presented to recursively determine, with the help of H.264 modes and information created by the motion propagation algorithm, whether or not each coding unit must be split. This allows the pruning of unpromising sub-partitions.

Compared to a CPDT approach, the experimental results show that the proposed solution is on average 7.81 times faster, for an average BD-Rate of 2.05%. Our experiment shows that these results exceed those of state-of-the-art methods.

Keywords: H.264/AVC, H.265/HEVC, video transcoding, fast mode decision, motion estimation

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

Afin de faciliter la lecture, les abréviations, signes et acronymes utilisés dans cette thèse correspondent à ceux habituellement utilisés dans le domaine de la compression vidéo, indépendamment de leur langue d'origine.

AMVP	Prédiction avancée du vecteur de mouvement *Advanced Motion Vector Prediction*
BD	*Bjøntegaard delta*
CABAC	Codage arithmétique binaire adaptable au contexte *Context-Adaptive Binary Arithmetic Coding*
CAVLC	Codage à longueur variable adaptable au contexte *Context-Adaptive Variable-Length Coding*
CM	Compensation du mouvement
CTU	Unité de codage arborescent *Coding Tree Unit*
CU	Unité de codage *Coding Unit*
DAM	Décision accélérée du mode
DCT	Transformée en cosinus discrète *Discrete Cosinus Transform*
DST	Transformée en sinus discrète *Discrete Sinus Transform*
EM	Estimation du mouvement
EPZS	*Enhanced Predictive Zonal Search*
EQM	Erreur quadratique moyenne
ISO	Organisation internationale de normalisation *International Organization for Standardization*

ITU	Union internationale des télécommunications
	International Telecommunication Union
JCT-VC	*Joint Collaborative Team on Video Coding*
LAN	Réseau local
	Local Area Network
MPEG	*Moving Picture Experts Group*
PQM	Partitionnement quaternaire basé le mouvement
PQS	Partitionnement quaternaire structurel
PML	Propagation du mouvement local
PSNR	Rapport signal à bruit de crête
	Peak Signal-to-Noise Ratio
PU	Unité de prédiction
	Prediction Unit
QP	Paramètre de quantification
	Quantization Parameter
SAD	Somme des différences absolues
	Sum of Absolute Differences
SAO	Décalage adaptatif des échantillons
	Sample Adaptative Offset
SATD	Somme des différences absolues transformées
	Sum of Absolute Transformed Differences
SIMD	Instruction unique sur des données multiples
	Single instruction multiple data
SSE	Somme des erreurs quadratiques
	Sum of Squared Errors
TCDP	Transcodage en cascade dans le domaine des pixels
TU	Unité de transformée
	Transform Unit

UMA	Accès multimédia universel *Universal Multimedia Access*
VCEG	*Video Coding Experts Group*

LISTE DES SYMBOLES ET UNITÉS DE MESURE

B_n^f	Bloc original n de la trame f
P_n^f	Bloc prédit n de la trame f
R_n^f	Bloc résiduel n de la trame f
$C_{i,j}$	Représente la j-ième CU située au niveau de profondeur i
∇_{AMVP}	Vecteurs de mouvement prédits pour une partition AMVP
$\mathbf{v}_{\text{p}}$	Vecteur de mouvement prédit pour une partition AMVP et sélectionné
$\mathbf{v}_{\text{b}}$	Meilleur vecteur de mouvement trouvé pour une partition AMVP
∇_{merge}	Vecteurs de mouvement prédits par le mode merge
$\mathbf{v}_{\text{m}}$	Vecteur de mouvement merge sélectionné
$\mathbf{v}_{\text{zero}}$	Vecteur de mouvement nul
$\nabla_{\text{H.264}}$	Vecteurs de mouvement H.264 extraits par la méthode de propagation du mouvement proposée
∇_{HEVC}	Vecteurs de mouvement HEVC extraits par la méthode de propagation du mouvement proposée
∇_{liste}	Les K vecteurs de mouvement appartenant à la liste des vecteurs de mouvement candidats
α	Largeur de la zone d'extraction H.264 étendue de l'algorithme de propagation du mouvement
r	Plage de recherche utilisée par l'estimation du mouvement
J_{rd}	Fonction cout débit-distorsion
J_{motion}	Fonction cout du mouvement
J_{pm}	Fonction cout de faible complexité basée sur la SATD
$\mathbf{E}^{4\times4}$	Matrice des erreurs de prédiction précalculées qui sont associées à la CTU courante et à la liste des vecteurs de mouvement candidats.

$e^{4\times4}_{x,y,k}$	Élément de la matrice $\mathbf{E}^{4\times4}$, contient la SATD du résiduel d'un bloc 4×4 situé à la position $(4x, 4y)$, par rapport au coin gauche supérieur de la CTU, et associé au k-ième vecteur de la liste des vecteurs de mouvement candidats
$e^{4N\times4M}_{x,y,k}$	SATD d'une partition de $4N\times4M$ pixels située à la position $(4x, 4y)$, par rapport au coin supérieur gauche de la CTU, et associé au k-ième vecteur de la liste des vecteurs de mouvement candidats
T_r	Temps d'encodage HEVC obtenu par un transcodage en cascade
T_m	Temps d'encodage HEVC obtenu par une des approches de transcodage proposée
A	Accélération obtenue par une approche de transcodage proposée par rapport à un transcodage en cascade
$D_{\text{bloc}}(\mathbf{b_1}, \mathbf{b_2})$	Distance de Tchebychev entre les blocs $\mathbf{b_1}$ et $\mathbf{b_2}$ (mesurée en blocs de 4×4 pixels)
$D_{\text{v}}(\mathbf{v}_1, \mathbf{v}_2)$	Distance de Tchebychev entre les vecteurs de mouvement $\mathbf{v}_1$ et $\mathbf{v}_2$
τ	Distance de Tchebychev maximale entre deux vecteurs de mouvement pour qu'ils soient considérés comme similaires

INTRODUCTION

Depuis plus d'une vingtaine d'années, le *Video Coding Experts Group* (VCEG) de l'union internationale des télécommunications (*International Telecommunication Union*, ITU) et le *Moving Picture Experts Group* (MPEG) de l'organisation internationale de normalisation (*International Organization for Standardization*, ISO) occupent une place prépondérante dans le domaine de la compression vidéo. En vue d'assurer l'interopérabilité des systèmes, ces groupes développent et publient des standards de compression vidéo. Le premier a notamment publié le standard H.261 (ITU-T (1990)), qui a été popularisé par la console de jeu PlayStation de Sony. Le second groupe a pour sa part publié le très populaire standard MPEG-2 (ISO/IEC (1994)). Un standard qui a largement contribué à l'essor de la télévision numérique en plus d'être utilisé pour l'encodage des DVDs vidéos.

Plus récemment, les deux groupes ont publié conjointement le standard H.264/MPEG-4 Advanced Video Coding (AVC) (ITU-T (2003)). Ce standard a obtenu rapidement un fort succès pour différents types d'applications vidéos, dont le stockage sur disques Blu-ray, la télévision numérique et la diffusion de vidéos sur Internet. Comme le soutient Ozer (2015), H.264 est de nos jours encore très populaire. Par exemple, c'est le format utilisé par défaut pour l'enregistrement vidéo sur les appareils iPad et iPhone d'Apple. De plus, il est encore couramment employé pour diffuser des vidéos sur Internet, notamment par Netflix et YouTube, souvent pour des raisons de compatibilité.

Enfin, l'ITU et l'ISO ont publié un nouveau standard appelé H.265/HEVC (ITU-T (2013)). Par rapport à H.264, ce standard réduit le débit de moitié pour une qualité vidéo similaire (il obtient donc une meilleure efficacité de codage). Pour les diffuseurs de contenu vidéo, cette réduction du débit représente une diminution des couts associés à l'utilisation de la bande passante et à l'espace de stockage. Pour les opérateurs mobiles, elle permet de servir plus de clients sur les infrastructures existantes. Alternativement, HEVC peut être utilisé pour améliorer l'expérience utilisateur en offrant une plus grande qualité vidéo que H.264 pour un débit comparable. Enfin, HEVC a été conçu spécialement pour supporter des séquences vidéos de très hautes résolutions, allant jusqu'à 8K (7680×4320), même s'il supporte de plus petites résolutions.

Par exemple, la Blu-ray Disc Association (BDA) a choisi ce format pour l'encodage vidéo 4K (3820×2160) des disques Ultra HD Blu-ray, comme le souligne Hermann (2015). Amazon et Netflix emploient aussi HEVC pour encoder et diffuser des séquences vidéos 4K ou de plus petites résolutions.

Certaines applications vidéos récentes supportent uniquement le format HEVC, c'est notamment le cas des disques Ultra HD Blu-ray. Cependant, plusieurs autres applications autorisent d'autres formats vidéos, en plus du format HEVC. Par exemple, Netflix supporte les formats vidéos VC-1, H.264 et HEVC, comme l'expliquent Aaron et Ronca (2015). Pour sa part, le lecteur vidéo Fire TV d'Amazon est en mesure de décoder uniquement les formats vidéos H.264 et HEVC (Amazon (2016)). De manière générale, un système va supporter plusieurs formats vidéos afin d'assurer l'interopérabilité avec différents appareils. En effet, plusieurs appareils mobiles et lecteurs vidéos supportent uniquement le format H.264. Pour ces appareils, le support d'un format plus récent, comme le format HEVC, est souvent impossible pour différentes raisons (processeur trop lent, cout des brevets, absence d'un décodeur supportant le format vidéo désiré, etc.).

Lorsqu'une application supporte plusieurs formats vidéos, il est souvent nécessaire, ou utile, de convertir une séquence existante vers un nouveau format. Par exemple, plusieurs applications de diffusion vidéos contiennent une imposante base de séquences vidéos encodées selon le format H.264. Pour bénéficier d'une meilleure efficacité de codage, ces séquences peuvent être converties vers le format HEVC. On parle alors de transcodage (conversion) vidéo H.264 à HEVC. Ce type de transcodage s'applique aussi à d'autres contextes. Par exemple, un utilisateur enregistre une séquence vidéo H.264 et la transmet à un appareil compatible avec HEVC. Durant la transmission, un serveur de transcodage peut agir d'intermédiaire entre les deux appareils afin de convertir la séquence H.264 en séquence HEVC. En somme, le transcodage d'une séquence H.264 vers une séquence HEVC est une opération très courante de nos jours étant donné la popularité de ces deux formats et de leur cohabitation fréquente sur un même système.

La manière la plus simple de transcoder une séquence H.264 vers une séquence HEVC consiste à décoder la séquence d'origine au complet, puis à réencoder entièrement la séquence décodée. Cette approche, appelée transcodage en cascade dans le domaine des pixels (Björk et Christopoulos (1998)), est simple d'implémentation et offre un maximum de flexibilité, notamment en ce qui a trait à la configuration de la séquence de sortie. Cependant, elle est très complexe en calculs. Pour un système de diffusion de contenus vidéos en ligne, cette complexité a pour effet de limiter le nombre de canaux disponibles pour effectuer du transcodage. Pour compenser cette complexité, il est donc nécessaire d'augmenter la capacité calculatoire du système, par exemple, en ajoutant davantage de processeurs, ce qui peut occasionner des couts économiques et écologiques importants. Pour un système de transcodage agissant entre deux appareils, cette complexité rend, dans certains cas, le traitement en temps réel impossible, en particulier pour des séquences à haute-résolution, qui nécessitent normalement plus de calculs.

Pour réduire cette complexité, il est possible de simplifier certaines étapes de l'encodage HEVC en réutilisant de l'information extraite durant le décodage de la séquence H.264 source. Il faut savoir que la majorité des formats de codage vidéos, dont H.264 et HEVC, utilisent des modèles de prédiction pour prédire les prochaines données à encoder. À titre d'exemple, un modèle de prédiction temporelle peut décrire le mouvement entre deux trames consécutives à l'aide de vecteurs de mouvement. Ces modèles supportent des structures de partitionnement flexibles qui permettent de diviser une région traitée en blocs de différentes tailles. Typiquement, les régions où le mouvement est simple sont représentées par des gros blocs et, inversement, les régions où le mouvement est complexe sont représentées par des plus petits blocs afin de mieux suivre le mouvement. Déterminer les paramètres de prédiction à encoder constitue le principal défi d'un encodeur vidéo. Ce dernier doit choisir la meilleure combinaison de paramètres de prédiction parmi plusieurs combinaisons. Cette combinaison de paramètres est par la suite encodée avec les erreurs de prédiction qui lui sont associées. C'est cette même information qui est réutilisée dans un transcodeur pour simplifier l'encodage de la séquence HEVC cible.

Le rôle de ce type de transcodeur vidéo consiste donc à adapter l'information de prédiction d'une séquence H.264 source vers une séquence HEVC cible. Cette adaptation doit être faite en tenant compte des différences entre les modèles de prédiction supportés par les deux formats vidéos. Comme nous le verrons plus loin, HEVC supporte des structures de partitionnement et des méthodes de prédiction qui sont généralement beaucoup plus flexibles et efficaces que celles définies par H.264. Nous verrons notamment que HEVC supporte des partitions couvrant des régions aussi grandes que 64×64 pixels, contre 16×16 pixels pour H.264. Cette caractéristique lui permet de représenter à moindre cout les régions où le mouvement est simple ou inexistant, ce qui est particulièrement avantageux pour les séquences vidéos de hautes-résolutions. En contrepartie, H.264 supporte des partitions aussi petites que 4×4 pixels pour représenter le mouvement, contre des partitions de 8×4 et 4×8 pour HEVC. En pratique, cette différence a peu d'incidence sur l'efficacité de codage HEVC.

En plus des différences entre les formats vidéos H.264 et HEVC, ce type de transcodeur doit aussi considérer les différences entre la séquence vidéo originale, la séquence vidéo H.264 et la séquence vidéo HEVC, puisque le codage et le transcodage vidéo ont généralement pour effet de réduire la qualité visuelle de la séquence traitée. Enfin, d'autres facteurs sont à considérer comme les paramètres de codage et les implémentations H.264 et HEVC utilisées. Idéalement, ce type de transcodage ne doit pas réduire significativement l'efficacité de codage comparativement à un transcodage en cascade dans le domaine des pixels. Ainsi, l'objectif premier d'un tel type de transcodeur vidéo est d'offrir un bon compromis entre l'efficacité de codage et la complexité (Ahmad *et al.* (2005)) en réutilisant l'information de codage H.264.

En pratique, l'adaptation de l'information de prédiction H.264 vers un format HEVC constitue un problème complexe qui est à l'origine de plusieurs travaux de recherche qui ont pour objectif premier d'accélérer le transcodage. Dans le cadre de cette thèse, on s'intéresse plus spécifiquement à deux sous-problèmes couramment traités par ces travaux, soient l'accélération de l'estimation de mouvement et la réduction du nombre de modes à tester. Par rapport à ces travaux, la motivation première de notre recherche est d'accélérer davantage le transcodage, tout en obtenant une efficacité de codage élevée. Les prochains paragraphes

présentent sommairement ces sous-problèmes ainsi que les solutions proposées dans la littérature. Nous aborderons plus en détail ces sous-problèmes au chapitre 2, une fois les notions fondamentales sur le codage et le transcodage vidéos présentées.

Accélération de l'estimation de mouvement

L'estimation de mouvement est le module d'un encodeur et d'un transcodeur vidéo qui a pour rôle de déterminer le vecteur de mouvement à encoder pour une partition (région) donnée. Selon Bossen *et al.* (2012), ce module peut représenter jusqu'à 40 % du temps d'exécution de l'encodeur HEVC de référence (JCT-VC (2014)). La réduction de sa complexité constitue donc un problème important, et cela, autant pour l'encodage vidéo que pour le transcodage vidéo. Dans un contexte d'encodage vidéo, il est commun d'employer un algorithme d'estimation de mouvement qui évalue plusieurs vecteurs de mouvement situés à l'intérieur d'une fenêtre de recherche afin de déterminer le vecteur à encoder. Puisque le mouvement est à priori inconnu, ce type de méthode emploie généralement une grande fenêtre de recherche afin de détecter les mouvements de grandes amplitudes. Évidemment, plus la fenêtre est grande et plus la complexité est élevée.

Dans un contexte de transcodage vidéo H.264 à HEVC, la situation est cependant différente, puisque les vecteurs de mouvement extraits de la séquence source fournissent de l'information utile sur le mouvement. Les approches proposées dans la littérature supposent que cette information constitue un bon de point de départ pour l'estimation de mouvement, mais que cette information doit être raffinée, notamment pour tenir compte des différences entre H.264 et HEVC (Peixoto *et al.* (2014a); Shen *et al.* (2013); Fang *et al.* (2014); Zong-Yi *et al.* (2013)). Ces approches raffinent donc le mouvement extrait en employant une fenêtre de recherche habituellement beaucoup plus petite que celle utilisée dans un contexte d'encodage vidéo. Elles se distinguent les unes des autres sur deux principaux aspects : d'abord, sur la méthode utilisée pour déterminer le vecteur de mouvement initial (le point central de la fenêtre de recherche), ensuite, sur la taille de fenêtre de raffinement utilisée. Nous aborderons ces deux aspects à la section 2.2.2.

Côté performance, la plupart de ces approches réduisent significativement la complexité de l'estimation de mouvement en affectant peu l'efficacité de codage. Cependant, elles effectuent plusieurs calculs redondants, puisqu'elles sont appliquées sur des partitions qui se superposent comme nous l'expliquerons plus loin. De plus, il faut savoir que l'estimation de mouvement se divise généralement en deux étapes : une étape appliquée sur des pixels situés à des positions entières, et une étape de raffinement appliquée sur des pixels situés à des positions fractionnaires, c'est-à-dire, sur des pixels interpolés. Ces approches de raffinement réduisent seulement la complexité de la première étape, alors que la seconde étape représente une proportion non négligeable du temps d'exécution d'un encodage HEVC, plus spécifiquement, jusqu'à 20 % selon Bossen *et al.* (2012).

Réduction des modes à tester

Les encodeurs et les transcodeurs vidéos sont typiquement constitués d'un module qui teste différents modes de codage (différentes combinaisons de structures de partitionnement et de paramètres de prédiction) sur la région traitée. La sélection du meilleur mode s'effectue généralement à l'aide d'une fonction cout débit-distorsion, c'est-à-dire, en tenant compte du nombre de bits nécessaires (débit) à l'encodage d'un mode et à la perte de qualité (distorsion) qui lui est associé. Afin de réduire la complexité de cette étape, il est possible d'éliminer des modes sans les avoir testés, par exemple, lorsque le meilleur mode actuel a déjà un cout débit-distorsion bas. On parle alors de réduction des modes à tester. Cette réduction constitue un problème très important, autant pour l'encodage que le transcodage vidéo, puisque la complexité d'un encodeur HEVC est fortement reliée au nombre de modes testés. Cette réduction doit être réalisée prudemment, sans quoi elle peut affecter significativement l'efficacité de codage.

Plusieurs méthodes ont été proposées dans la littérature pour réduire le nombre de modes à tester dans un contexte de transcodage H.264 vers HEVC (Fang *et al.* (2014); Jiang *et al.* (2013); Peixoto *et al.* (2014a); Zhang *et al.* (2012); Jiang et Chen (2013); Zong-Yi *et al.* (2013); Luo *et al.* (2014); Shen *et al.* (2013); Chen *et al.* (2014); Xing *et al.* (2013)). La majorité de ces

méthodes réutilisent uniquement l'information extraite de la séquence H.264 pour déterminer les modes HEVC à tester. Ces dernières préservent généralement l'efficacité de codage, mais réduisent rarement le temps d'exécution de l'encodage HEVC de plus de 50 %.

Pour réduire davantage ce temps d'exécution, certaines approches exploitent l'information HEVC générée durant l'encodage HEVC, en plus de l'information extraite de la séquence H.264. Parmi ces méthodes, seule la méthode de Peixoto *et al.* (2014b) se démarque par ses performances élevées. Cette dernière utilise l'information H.264 pour définir une liste de modes HEVC à tester qui seront testés les uns à la suite des autres. Lorsque le mode HEVC testé obtient un cout satisfaisant, le traitement des prochains modes est arrêté prématurément. Cette méthode teste d'abord des modes simples (constitués de quelques grandes partitions) et efficaces pour représenter des régions où le mouvement est inexistant ou continu. Par la suite, elle teste des modes de plus en plus complexes (constitués de plusieurs petites partitions) et généralement plus efficaces pour représenter des régions où le mouvement est complexe. Ainsi, l'approche proposée par ces auteurs obtient des transcodages particulièrement rapides pour des séquences simples, qui sont à la base déjà plus rapides à encoder que des séquences complexes. En contrepartie, l'approche est moins bien adaptée aux séquences complexes qui nécessitent généralement l'utilisation de modes complexes (finement partitionnés). Enfin, cette approche affecte peu l'efficacité de codage et réduit généralement le temps d'exécution de l'encodage HEVC d'au moins 75 %.

But de la thèse et transcodeur proposé

Le but de nos travaux était de développer une nouvelle approche rapide et efficace de transcodage H.264 à HEVC. Nous avons atteint ce but en proposant une nouvelle solution pour accélérer l'estimation de mouvement ainsi qu'une nouvelle solution pour réduire le nombre de modes à tester. Le transcodeur que nous proposons atteint des accélérations supérieures à celles obtenues par les autres approches proposées dans la littérature, tout en préservant une haute efficacité de codage.

La méthode que nous avons développée pour accélérer l'estimation de mouvement suppose que le champ de vecteurs de mouvement extrait de la séquence H.264 est suffisamment précis pour être réutilisé durant l'encodage HEVC sans devoir y appliquer une méthode de raffinement. En contrepartie, elle tient compte du fait que les vecteurs de mouvement H.264 se propagent de manière différente dans la séquence HEVC, c'est-à-dire, que le champ de vecteurs de mouvement HEVC peut être différent même s'il contient des vecteurs de mouvement déjà présents dans la trame H.264. Ainsi, nous verrons que le meilleur vecteur de mouvement pour la région traitée dans HEVC est généralement localisé dans la région H.264 correspondante, comme le supposent les autres approches proposées dans la littérature. Cependant, nous montrons aussi qu'un meilleur vecteur de mouvement peut parfois être situé à l'extérieur de cette région, par exemple, dans le voisinage de la région H.264 correspondante, ou encore, dans des régions HEVC qui ont été précédemment encodées. Contrairement aux méthodes de raffinement du mouvement proposées dans la littérature, notre méthode de propagation de mouvement élimine les calculs redondants. De plus, elle génère des données pouvant être réutilisées par le module d'évaluation des modes HEVC afin de réduire le nombre de modes testés, comme nous le montrons au chapitre 4.

La méthode que nous proposons pour réduire le nombre de modes HEVC à tester exploite à la fois de l'information extraite de la séquence H.264 et de l'information générée durant l'encodage HEVC, tout comme c'est le cas pour la méthode proposée par Peixoto *et al.* (2014b). Contrairement à cette dernière, notre approche teste d'abord les modes les plus complexes, en éliminant d'abord certains modes complexes jugés peu prometteurs selon l'information H.264 disponible et l'information générée par notre algorithme de propagation du mouvement. Par la suite, l'approche proposée teste des modes de moins en moins complexes. Elle utilise l'information HEVC disponible pour arrêter prématurément le traitement des modes jugés peu prometteurs. Selon nos résultats expérimentaux, notre approche préserve l'efficacité de codage, en plus d'obtenir de meilleures accélérations que celles obtenues par les autres approches proposées dans la littérature. Le gain de vitesse est notamment obtenu en arrêtant plus rapidement le traitements de modes peu prometteurs.

Organisation de la thèse

Afin de faciliter la lecture de ce manuscrit, nous avons organisé cette thèse en quatre chapitres.

Le premier chapitre a pour objectif de présenter les fondements du codage vidéo qui seront nécessaires à la compréhension des chapitres suivants. Il débute par une présentation des notions fondamentales du codage vidéo. Par la suite, il aborde les spécificités des standards H.264 et HEVC en portant une attention particulière sur leurs distinctions. Enfin, il se termine en décrivant sommairement l'encodeur HEVC, dont son algorithme d'estimation de mouvement rapide et sa méthode pour sélectionner le mode HEVC à encoder. Le lecteur familiarisé avec ces notions est invité à feuilleter rapidement ce chapitre en portant attention aux différentes notations et équations qui y sont présentées.

Le second chapitre porte sur le transcodage vidéo. Il présente d'abord le domaine du transcodage vidéo en décrivant les opérations de transcodage courantes, les principales architectures de transcodage ainsi que quelques techniques de transcodage. Dans un second temps, il présente plus en détail l'état de l'art sur le transcodage vidéo H.264 à HEVC. Cette revue de la littérature porte plus spécifiquement sur les deux sous-problèmes que nous traitons dans cette thèse, soit l'accélération de l'estimation de mouvement et la réduction des modes HEVC à tester.

Le troisième chapitre présente notre approche de propagation du mouvement. Il fait d'abord un retour sur les méthodes de raffinement du mouvement en décrivant les problèmes de ce type de méthodes dans un contexte de transcodage H.264 à HEVC. Par la suite, il présente l'approche de propagation du mouvement proposée. Puis, il analyse de quelle manière les vecteurs de mouvement existants se propagent dans la séquence HEVC lorsque notre approche est employée. Enfin, le chapitre se termine en décrivant les expériences qui ont été menées pour évaluer les performances de notre approche.

Enfin, le dernier chapitre présente notre approche pour réduire le nombre de modes HEVC à évaluer durant le transcodage. D'abord, il décrit les différents problèmes que nous avons

identifiés dans les approches proposées dans la littérature. Par la suite, ce chapitre décrit l'architecture de l'approche que nous proposons et décrit ses principales méthodes. Le chapitre se conclut sur la présentation des résultats expérimentaux et une comparaison avec l'état de l'art.

Le chapitre de conclusion récapitule nos principales contributions et présente différentes voies à explorer afin d'améliorer l'approche de transcodage proposée, ou l'adapter à d'autres contextes.

CHAPITRE 1

INTRODUCTION AU CODAGE VIDÉO

Ce premier chapitre a pour objectif de présenter les fondements du codage vidéo nécessaires à la compréhension des prochains chapitres. Nous y présentons d'abord les notions fondamentales du codage vidéo. Ces notions sont communes à plusieurs formats et encodeurs vidéos. Par la suite, nous abordons les spécificités des standards H.264 et HEVC en portant une attention particulière sur les aspects qui les distinguent. Enfin, nous terminons ce chapitre en présentant l'encodeur HEVC de référence.

Nous conseillons au lecteur n'ayant aucune base sur la compression vidéo de lire d'abord l'ouvrage de Li *et al.* (2014), puis, de lire par la suite des ouvrages d'introduction sur les standards H.264 et HEVC (ITU-T (2003, 2013)). Nous conseillons notamment le livre de Richardson (2011), qui couvre le standard H.264, et les articles de Sullivan *et al.* (2012) et Ohm *et al.* (2012), qui couvrent le standard HEVC.

1.1 Principes du codage vidéo

Le codage vidéo a pour objectif premier de réduire le nombre de bits nécessaires à la représentation d'une séquence vidéo afin de faciliter son stockage ou sa transmission. Ce codage est dit sans perte lorsque le décodage de la séquence encodée produit une séquence vidéo identique à la séquence originale. Ce type de codage a pour avantage de préserver la qualité vidéo, cependant il atteint des taux de compression relativement faibles. Par exemple, une séquence encodée avec un encodeur HEVC configuré en mode sans perte sera seulement environ deux fois plus petite que la séquence originale selon Choi et Ho (2013). Malgré sa faible efficacité de codage (compression), ce type de codage est tout de même employé par certains types d'applications où la qualité vidéo est cruciale, par exemple, pour des applications médicales.

Inversement, le codage est dit avec perte lorsqu'il entraine une perte de qualité. C'est ce type de codage qui est normalement utilisé lorsqu'une perte de qualité (perceptible ou pas) est acceptable, par exemple, pour la diffusion vidéo ciblant le grand public (DVDs, Blu-Rays, télévision numérique, etc.). Par ailleurs, c'est le type de codage que nous ciblons dans nos travaux de recherche. La perte de qualité entrainée par ce type de codage est le résultat d'une élimination d'information qui a pour effet de rendre le codage plus efficace. De manière générale, plus la perte de qualité est grande et plus le fichier compressé sera petit. Par exemple, il est possible d'atteindre un facteur de compression aussi élevé que 50 pour une perte de qualité négligeable. Ce facteur peut augmenter jusqu'à 200, voire davantage, pour une perte de qualité plus importante. Il dépend de plusieurs facteurs, dont l'encodeur vidéo employé, les paramètres de codage ainsi que les caractéristiques de la séquence à encoder.

Pour atteindre des taux de compression aussi élevés, le codage vidéo repose sur plusieurs principes fondamentaux dont les deux plus importants sont : l'emploi de modèles pour prédire les données à encoder et l'utilisation de méthodes avec perte pour encoder la différence entre les données à encoder et celles prédites. Les modèles de prédiction ont pour objectif de réduire la redondance d'information, alors que les méthodes avec perte ont pour but d'éliminer de l'information.

Les modèles de prédiction prédisent les données à encoder à l'aide de données précédemment encodées appelées données de prédiction (ou données de référence). On parle de prédiction temporelle lorsque ces données proviennent de trames (images) antérieures. Ces dernières sont appelées images de référence puisque la prédiction temporelle peut s'y référer. On parle plutôt de prédiction spatiale lorsque les données proviennent directement de la trame traitée. La prédiction temporelle est aussi appelée prédiction inter puisqu'elle représente une prédiction effectuée « entre » deux trames. Inversement, la prédiction spatiale est appelée prédiction intra puisqu'elle fait référence à des données situées « à l'intérieur de » la trame traitée.

Les modèles de prédiction utilisés en vidéo ne permettent pas toujours d'effectuer une prédiction parfaite des données à encoder, notamment parce que certaines parties d'une

séquence vidéo contiennent de l'information nouvelle. Les encodeurs vidéos doivent donc encoder les erreurs de prédiction, aussi appelées les données résiduelles ou, plus simplement, le résiduel. Ce résiduel est encodé à l'aide d'une série d'opérations qui a pour effet d'éliminer de l'information (représenter moins fidèlement l'information et ainsi causer une perte de qualité). Cette élimination d'information est habituellement effectuée en tenant compte des caractéristiques du système visuel humain de manière à éliminer d'abord l'information peu ou pas perceptible par l'humain. L'information qui n'a pas été éliminée est par la suite encodée à l'aide d'un encodeur entropique.

1.1.1 Schéma de codage vidéo hybride

Depuis l'apparition du standard H.261 en 1988, le schéma de codage vidéo hybride constitue l'architecture de codage de base employée par les standards développés par l'union internationale des télécommunications (*International Telecommunication Union*, ITU) et l'organisation internationale de normalisation (*International Organization for Standardization*, ISO), dont les standards H.264 et HEVC. Il est aussi à la base de l'architecture d'autres formats de codage vidéos populaires, tels que VP8 et VP9 de Google (WebM (2016a,b)). Le codage est dit hybride parce que l'architecture combine des méthodes de codage différentiel (autrement dit, des méthodes de prédiction) et des méthodes de codage par transformée.

La figure 1.1 illustre la structure générale d'un encodeur vidéo hybride. Dans ce schéma, l'image originale est d'abord partitionnée en blocs appelés superblocs (par exemple, en blocs de 16×16 pixels). Ces blocs sont notés B_n^f, où f représente le numéro de l'image et n, le numéro du bloc. Afin de réduire la redondance d'information, le contenu de chaque superbloc est prédit à l'aide d'une méthode de prédiction intra (spatiale) ou inter (temporelle). Le bloc prédit, noté P_n^f, est par la suite soustrait du bloc original pour former un bloc représentant le résiduel (l'erreur de prédiction), noté R_n^f. Une transformée T est par la suite appliquée sur le résiduel afin d'obtenir une représentation des données dans le domaine fréquentiel. Comme nous le verrons plus loin, ce type de représentation concentre l'énergie du signal dans les coefficients de basses fréquences pour rendre la compression plus efficace. Afin d'éliminer

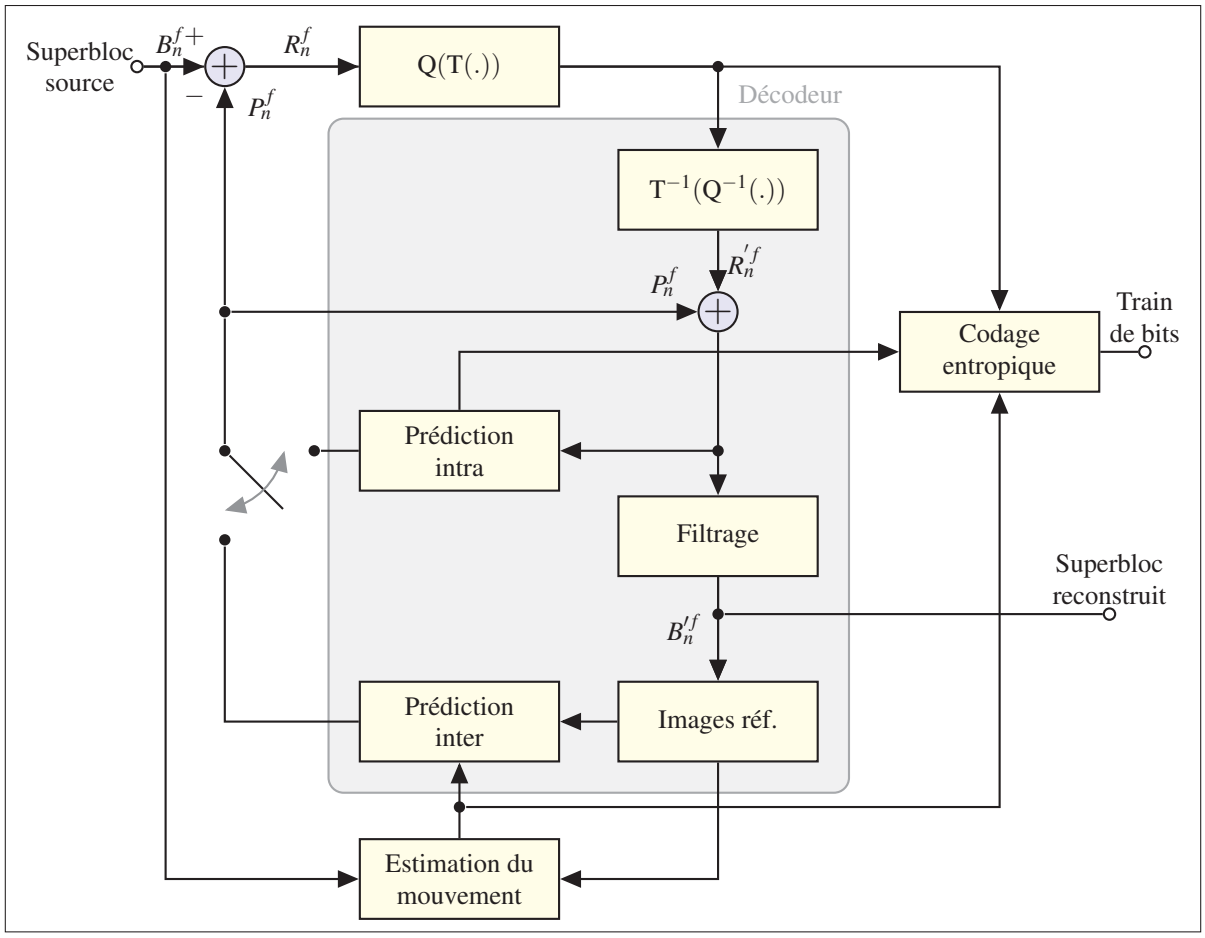


Figure 1.1 Schéma d'un encodeur vidéo hybride
Adaptée de Wien (2015, p. 38)

de l'information, les coefficients retournés par la transformée sont ensuite quantifiés par une quantification Q. Cette quantification a pour effet d'améliorer l'efficacité du codage au prix d'une perte de qualité visuelle. Finalement, le codage entropique est appliqué sur les données quantifiées et sur les paramètres de prédiction pour former le signal compressé (le train de bits) associé au bloc traité.

Pour demeurer parfaitement synchronisé avec le décodeur, l'encodeur reconstruit ce bloc de la même manière que le ferait le décodeur, soit en appliquant les opérations inverses (sauf pour le codage entropique puisqu'il est sans perte). Ainsi, au décodeur, le train de bits sera d'abord décodé pour récupérer les paramètres de prédiction et les données quantifiées. Par la

suite, les données résiduelles sont reconstruites en appliquant une quantification inverse suivie d'une transformée inverse [1]. Ces données sont notées $R_n^{'f}$ et diffèrent généralement des données résiduelles originales R_n^f à cause de la perte de qualité causée par la quantification. Ensuite, le décodeur reconstruit les données prédites en appliquant les paramètres de prédiction. Enfin, les données prédites sont additionnées aux données résiduelles pour former les blocs reconstruits. Ces derniers peuvent par la suite être filtrés afin d'atténuer certains artéfacts (distorsions visuelles) causés par la quantification durant le codage. On obtient alors les blocs reconstruits et filtrés, notés $B_n'^f$.

Il est important de remarquer qu'afin d'assurer la cohésion entre les données prédites par l'encodeur et celles prédites par le décodeur, les modèles de prédiction doivent être appliqués sur les mêmes données de prédiction. Sachant que le codage vidéo entraine une perte de qualité, les données de la séquence originale ne peuvent pas être utilisées à cette fin, puisque le décodeur n'y a pas accès (il a seulement accès aux blocs $B_n'^f$). Pour contourner ce problème, l'encodeur vidéo comprend un module de décodage comme présenté à la figure 1.1. Ce module reproduit les mêmes données que les données reconstruites par le décodeur (les blocs $B_n'^f$). Ces données seront par la suite utilisées par les modules de prédiction.

1.1.1.1 Décomposition d'une séquence vidéo

Un codeur vidéo décompose la séquence vidéo traitée en une série d'images fixes appelées trames. Une trame est de type intra lorsqu'elle ne dépend d'aucune autre trame. Et inversement, elle est de type inter lorsqu'elle dépend d'au moins une autre trame. La prédiction intra est supportée par ces deux types de trames. Cependant, la prédiction inter est seulement supportée par la trame inter. Grâce à ce mécanisme de prédiction supplémentaire, la trame inter obtient généralement une efficacité de codage nettement supérieure à celle de la trame intra. Pour sa part, la trame intra a l'avantage de pouvoir être utilisée comme point d'accès aléatoire (un point où le décodage d'une séquence peut débuter). C'est ce type de point qui permet de naviguer

1. Les données reconstruites sont généralement différentes des données originales puisque la quantification n'est pas réversible.

à travers une séquence compressée. C'est pour cette raison que plusieurs applications vidéos encodent une trame intra périodiquement, par exemple, à chaque seconde.

Figure 1.2 Partitionnement d'une image en superblocs de 16×16 pixels

Comme mentionné précédemment, chaque trame est partitionnée en blocs élémentaires appelés superblocs. Dans H.264 et les standards antérieurs, ces superblocs sont appelés macroblocs et couvrent une région de 16×16 échantillons comme l'illustre la figure 1.2. Dans HEVC, ils sont plutôt appelés unités de codage arborescent (*Coding Tree Unit*, CTUs) et peuvent couvrir une région aussi grande que 64×64 échantillons. Quel que soit le standard, ces superblocs sont toujours traités (encodés et décodés) selon un balayage par ligne – c'est-à-dire, de la gauche vers la droite et de haut en bas – comme le montre le tracé rouge sur la figure 1.2.

Typiquement, les pixels d'un superbloc sont représentés selon un espace de couleur YCbCr [2] et suivent un format d'échantillonnage 4:2:0 [3], où chaque composante est représentée avec une précision de 8 bits. Ainsi, un superbloc de 16×16 pixels sera constitué d'un bloc 16×16 pour

2. Dans un espace de couleur YCbCr, la composante Y représente l'information de luminance, tandis que les composantes Cb et Cr représentent l'information de la chrominance.

3. Le format d'échantillonnage 4:2:0 conserve l'ensemble des composantes Y et sous-échantillonne, horizontalement et verticalement, les composantes Cb et Cr de manière à produire et à conserver une valeur Cb et une valeur Cr pour quatre valeurs Y.

les composantes Y et de deux blocs de 8×8 pour représenter respectivement les composantes Cb et Cr.

1.1.1.2 Prédiction inter

Une séquence vidéo représente généralement une scène constituée d'un arrière-plan et de différents objets. Tout comme la caméra, ces objets peuvent être fixes ou en mouvement. D'une image à l'autre, on peut donc observer de grandes similitudes et quelques différences causées essentiellement par le mouvement des objets et de la caméra. Les encodeurs vidéos exploitent cette redondance temporelle en prédisant le bloc à encoder à l'aide d'algorithmes d'estimation de mouvement.

Pour des raisons de complexité et d'efficacité de codage, les modèles de prédiction du mouvement utilisés en compression vidéo sont simples et supposent que le mouvement est planaire et translationnel. Ces modèles sont donc peu adaptés à d'autres types de mouvement plus complexes, comme des transformées affines, des rotations, etc. De plus, ces modèles sont peu efficaces pour prédire le mouvement d'objets complexes comme le feu, l'eau, la fumée. Et même si le mouvement est bien prédit, ces modèles peuvent obtenir une erreur de prédiction élevée, par exemple, à cause d'un changement d'éclairage à l'intérieur de la région traitée. Enfin, la prédiction inter est très peu efficace pour représenter un changement de scène. Malgré ces différents défauts, les modèles de prédiction inter utilisés en compression vidéo sont très efficaces, surtout parce qu'à des taux de trame élevés, les mouvements complexes peuvent être approximés par des mouvements planaires translationnels.

Le modèle de prédiction inter le plus simple consiste à indiquer le déplacement qu'il faut effectuer par rapport au bloc traité afin de pointer vers le bloc prédit, celui lui ressemblant le plus, selon un critère objectif[4], situé dans l'image de référence (l'image précédente, par exemple). Le vecteur de mouvement représentant ce déplacement constitue le seul paramètre de ce modèle. La recherche de ce vecteur de mouvement s'effectue durant l'encodage vidéo à

4. Nous présentons plus loin quelques critères couramment utilisés. Il est important de noter que les standards de compression vidéo ne définissent pas les critères à utiliser.

l'aide d'un algorithme d'estimation de mouvement. Il est important de noter que les standards de compression vidéo ne définissent ni l'algorithme de recherche à employer ni le critère de sélection du meilleur vecteur de mouvement. En contrepartie, ils définissent la syntaxe à utiliser pour signaler le vecteur de mouvement ainsi que ses contraintes, par exemple, son amplitude maximale. En n'imposant ni algorithme ni critère de sélection, les standards de compression vidéo permettent aux différentes implémentations de se concurrencer, par exemple, en choisissant des algorithmes offrant un bon compromis entre l'efficacité de codage et la complexité de la recherche.

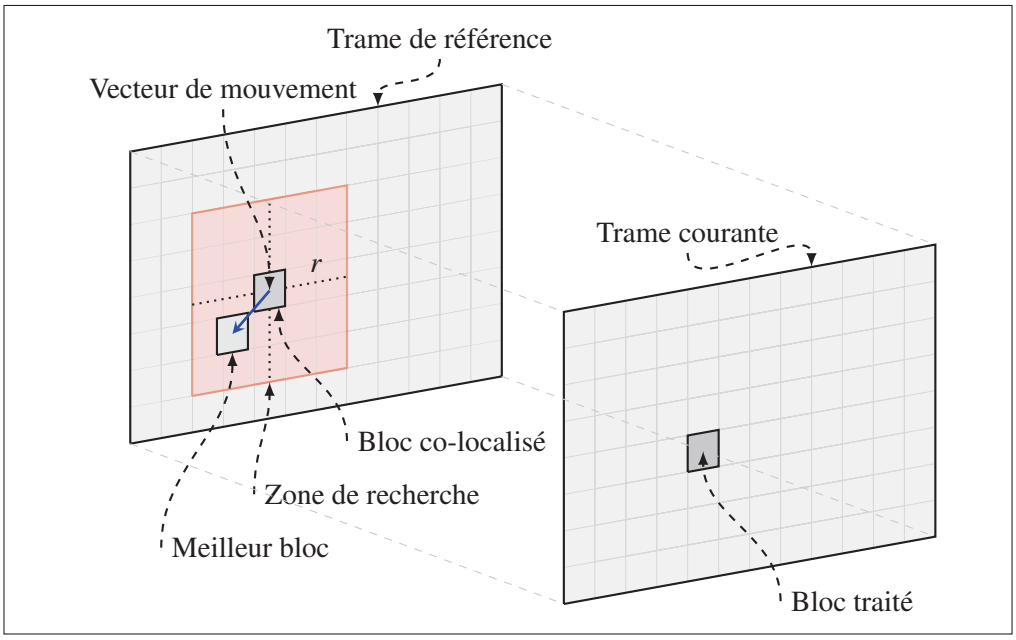


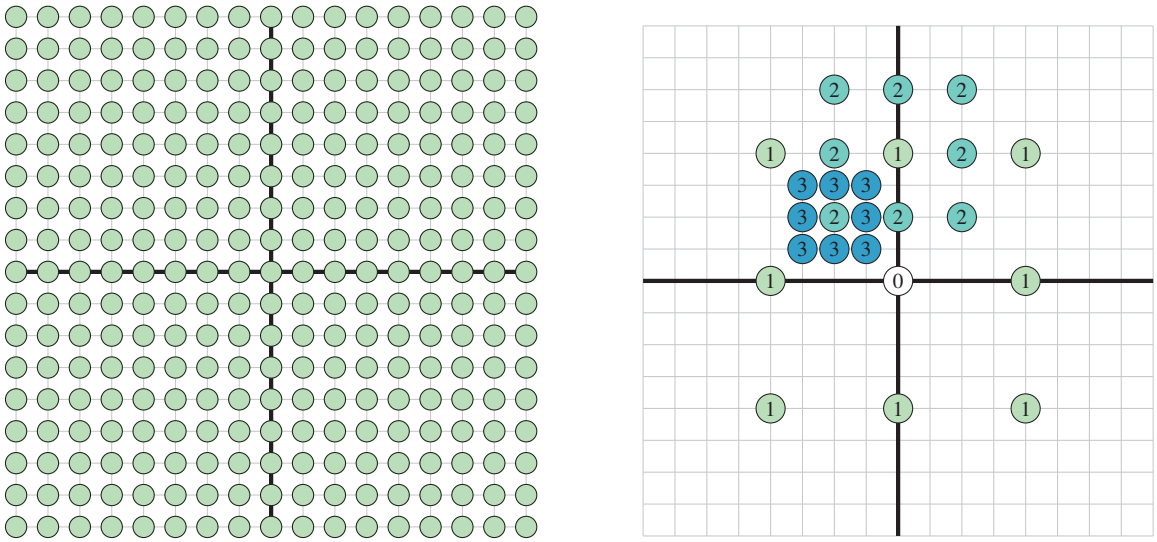
Figure 1.3 Estimation du mouvement contraint par une zone de recherche

En pratique, plusieurs encodeurs vidéos, dont l'encodeur HEVC de référence, définissent une zone de recherche carrée pour limiter l'étendue de la recherche comme l'illustre la figure 1.3. Cette zone de recherche est habituellement configurable grâce à un paramètre appelé plage de recherche et noté r. Cela veut dire que chacune des composantes du vecteur de mouvement est limitée à un déplacement de $\pm r$ par rapport au point central de la zone de recherche. Une grande plage de recherche permet de mieux détecter les mouvements rapides, mais augmente

la complexité en calculs de la recherche par rapport à une petite plage de recherche. Il est très commun d'utiliser une plage de recherche de 32 ou 64 pixels.

1.1.1.2.1 Algorithme de recherche

Il existe plusieurs algorithmes de recherche pour trouver le meilleur bloc (vecteur de mouvement) situé à l'intérieur de la zone de recherche. L'algorithme le plus connu est appelé algorithme de recherche exhaustive (voir figure 1.4.a) et consiste à évaluer l'ensemble des positions situées à l'intérieur de la zone de recherche puis à sélectionner la position obtenant le plus petit cout (voir Section 1.1.3.2). Cet algorithme trouve la solution optimale puisque toutes les positions sont évaluées. Cependant, sa complexité est très élevée puisque $(2r+1)^2$ positions doivent être évaluées, soit 289 positions pour l'exemple de la figure 1.4.a. Cet algorithme est donc trop complexe pour être employé dans les applications en temps réel.



a) Recherche exhaustive

b) Recherche en trois étapes

Figure 1.4 Exemples d'algorithmes de recherche du mouvement pour une plage de recherche de 8 pixels : b) le cercle numéroté «0» représente le vecteur de mouvement initial ; les autres cercles, numérotés de «1» à «3», représentent les vecteurs de mouvement évalués par les trois premières itérations de l'algorithme

Afin de réduire la complexité de cette recherche, plusieurs algorithmes sous-optimaux ont été proposés dans la littérature. La figure 1.4.b montre l'exemple d'un algorithme connu sous le nom de recherche en trois étapes (three-step search) ou, encore, sous le nom de recherche en N étapes sous sa forme plus générale. C'est un algorithme itératif qui évalue 8 positions à chaque itération. La position (0,0) est d'abord évaluée à l'initialisation de l'algorithme. Puis, l'itération 1 évalue 8 positions situées à une distance de 2^{N-1} pixels selon un patron carré. La meilleure position devient le point central de l'itération 2. Cette itération évalue 8 positions situées cette fois-ci à une distance de 2^{N-2} pixels du point central de recherche. L'algorithme se poursuit ainsi jusqu'à ce que les N itérations soient complétées. Au total, l'algorithme va évaluer $(8N+1)$ positions, soit 25 positions pour l'exemple de la figure 1.4.b. Cet algorithme est donc nettement moins complexe que l'algorithme de recherche exhaustive. Cependant, il est sous-optimal puisque sa solution ne converge pas toujours vers le minimum global.

Il existe plusieurs autres algorithmes de recherche. Nous verrons notamment plus loin l'algorithme de recherche implémenté dans l'encodeur HEVC de référence. Ces algorithmes offrent différents compromis entre le gain de vitesse et l'effet sur l'efficacité de codage. Nous référons le lecteur qui désire en apprendre davantage sur ces algorithmes aux travaux de Li *et al.* (1994), Zhu et Ma (2000), et Furht *et al.* (2012).

1.1.1.2.2 Paramètres de prédiction

Jusqu'à maintenant, nous avons vu un modèle de prédiction inter simple ayant pour seul paramètre un vecteur de mouvement représentant le mouvement à effectuer dans la trame de référence pour pointer sur le bloc prédit. C'est ce modèle que nous utiliserons pour l'implémentation de notre approche de transcodage. Cependant, il faut savoir que plusieurs formats vidéo supportent des modèles de prédiction plus complexes et ayant davantage de paramètres.

Ainsi, il est possible d'utiliser plus d'une image de référence pour effectuer l'estimation de mouvement. L'encodeur doit alors signaler, en plus du vecteur de mouvement, l'index de

l'image de référence sélectionnée. Les trames supplémentaires améliorent l'efficacité de la compression, par exemple, en permettant d'accéder à de l'information qui serait cachée dans l'image précédente, mais visible dans une image antérieure. Cependant, ces trames ont pour effet d'augmenter la complexité de l'estimation de mouvement puisque la recherche doit être effectuée dans plusieurs trames. De plus, l'efficacité de compression ajouté par des trames supplémentaires dépend du contenu de la séquence.

En plus de trames de référence supplémentaires, il est aussi possible d'utiliser deux prédicteurs pour prédire le bloc traité. Chaque prédicteur est alors associé à une image de référence et un vecteur de mouvement. Par la suite, la moyenne des deux blocs pointés par ces prédicteurs est utilisée pour représenter le bloc prédit. Ce type de prédiction est appelé bi prédiction, alors que la prédiction vue précédemment est appelée uni prédiction. Une trame inter supportant la bi prédiction est appelée trame B, alors qu'une trame inter qui ne la supporte pas est appelée trame P.

1.1.1.2.3 Partitionnement des superblocs

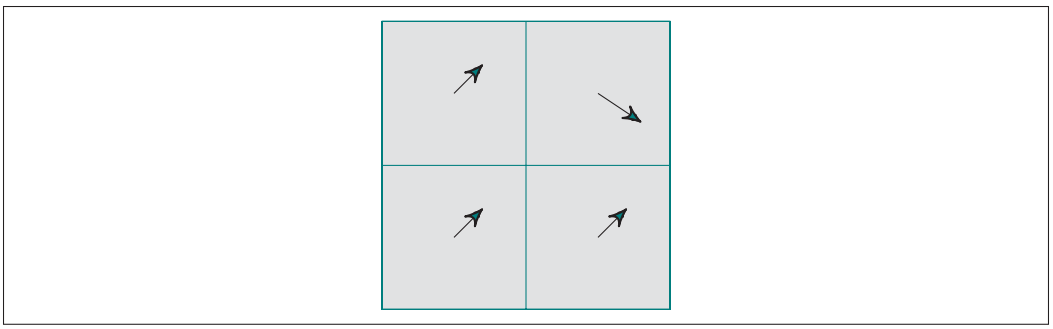


Figure 1.5 Exemple de partitionnement d'un superbloc en blocs 8×8

Le partitionnement de l'image traitée a un impact important sur l'efficacité de la prédiction inter. Par exemple, un partitionnement grossier (composé de gros blocs) aura pour effet de produire une segmentation grossière entre un objet en mouvement et son arrière-plan. Ce qui se traduit généralement par une erreur de prédiction élevée aux frontières de l'objet. Inversement, un partitionnement trop fin aura pour effet d'augmenter le nombre de bits nécessaires à la

signalisation des paramètres du modèle de prédiction (les vecteurs de mouvement). Afin de gérer le partitionnement de manière efficace, plusieurs formats vidéos supportent le partitionnement des superblocs en sous-blocs. Différents schémas de partitionnement sont généralement supportés. La figure 1.5 montre un partitionnement en blocs de 8×8, où chaque bloc est associé à un vecteur de mouvement. Plus loin,nous verrons les structures de partitionnement supportées respectivement par les standards H.264 et HEVC.

1.1.1.2.4 Précision des vecteurs de mouvement

Les premiers encodeurs vidéos supportaient une estimation de mouvement précise au pixel entier, c'est-à-dire que l'algorithme de recherche n'évaluait que des positions entières. Pour améliorer la précision de la prédiction, plusieurs standards de compression vidéo supportent des positions fractionnaires. Ces positions permettent notamment de suivre plus fidèlement les objets en déplacement. Les pixels qui correspondent à des positions fractionnaires sont obtenus par interpolation. Trois types de positions fractionnaires sont couramment utilisées : les demies, les quarts et les huitièmes de positions. On dit alors que l'estimation de mouvement est respectivement précise au demi, au quart et au huitième de pixel.

1.1.1.2.5 Prédiction du vecteur de mouvement

De manière générale, il existe une forte corrélation spatiale entre les vecteurs de mouvement. Cette corrélation est exploitée par les codeurs vidéos en prédisant le vecteur de mouvement du bloc traité à l'aide de vecteurs de mouvement situés dans le voisinage. Au lieu de signaler le vecteur de mouvement, l'encodeur signale plutôt la différence entre le vecteur de mouvement trouvé par l'algorithme d'estimation de mouvement et le vecteur de mouvement prédit à l'aide du processus défini par le format vidéo (le décodeur effectue la même prédiction afin d'assurer la cohérence avec l'encodeur). Ce vecteur transmis est appelé vecteur de mouvement différentiel.

Il est à noter que plusieurs algorithmes d'estimation du mouvement utilisent le vecteur de mouvement prédit, au lieu du vecteur nul [5] (0,0), comme point central de la zone de recherche. Cela a généralement pour effet de réduire le nombre d'itérations nécessaires à l'algorithme pour converger vers la solution.

1.1.1.3 Prédiction intra

La prédiction intra exploite la corrélation spatiale qui existe entre des blocs voisins. Elle est basée sur l'idée que la texture du bloc à encoder est généralement similaire à la texture de blocs voisins précédemment encodés. La prédiction spatiale consiste à prédire les valeurs du bloc traité à l'aide de pixels voisins. Comme la prédiction inter, elle peut être appliquée sur des blocs de différentes tailles afin d'améliorer la corrélation entre les blocs traités.

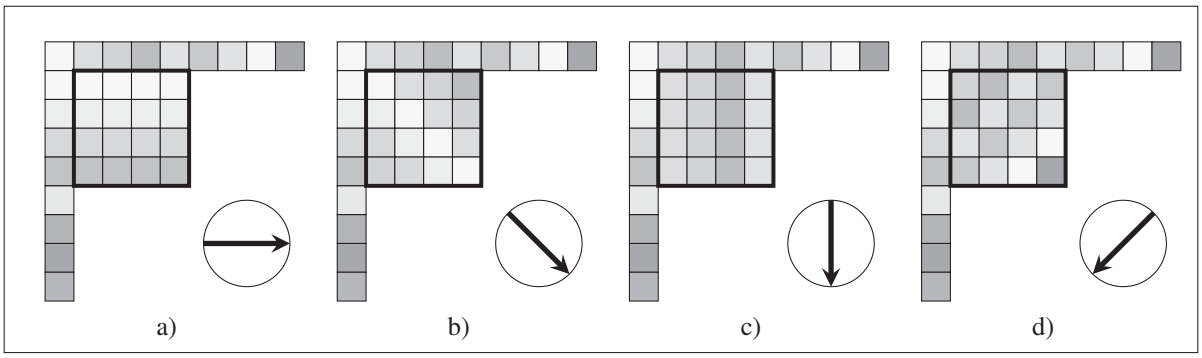


Figure 1.6 Exemples de prédicteurs intra directionnels
Adaptée de Wien (2015, p. 41)

En pratique, les encodeurs vidéos évaluent généralement plusieurs modes de prédiction (prédicteurs) et sélectionnent le meilleur mode selon une fonction cout. La figure 1.6 montre l'exemple de quatre modes de prédiction directionnels. Ces types de prédicteurs prolongent les pixels de prédiction (les voisins du bloc à prédire) pour former le bloc prédit. Il existe d'autres types de prédiction que nous aborderons plus loin, par exemple, la prédiction planaire et la prédiction DC.

5. Le vecteur nul est souvent utilisé, car il arrive fréquemment que des zones d'image demeurent immobiles.

De manière générale, la prédiction intra est moins efficace que la prédiction inter, surtout lorsque le mouvement est simple. Cependant, elle peut être plus efficace dans certains cas, par exemple, aux bordures d'un objet en mouvement ou encore dans une région plate (non texturée). C'est pourquoi la majorité des encodeurs l'utilisent aussi pour coder certains blocs de trames inter. Enfin, il faut noter que l'efficacité de la prédiction intra dépend grandement du standard de compression vidéo utilisé. Les standards récents supportent davantage de modes de prédiction et des structures de partitionnement plus flexibles et complexes.

1.1.1.4 Codage du résiduel

À ce stade, une partie de la corrélation (redondance) a été éliminée à l'aide du modèle de prédiction sélectionné. Malgré tout, les données du résiduel sont souvent spatialement corrélées. Pour réduire cette corrélation, la majorité des encodeurs appliquent une transformée sur le résiduel dans le but concentrer l'énergie du signal dans les basses fréquences. Il existe différentes transformées. Les plus populaires dans le domaine du codage vidéo sont : la transformée en cosinus discrète (*Discrete Cosinus Transform*, DCT), la DCT entière, la transformée en sinus discrète (*Discrete Sinus Transform*, DST) ainsi que la transformée de Hadamard. À titre d'exemple, la figure 1.7 montre les coefficients obtenus suite à l'application d'une DCT 4×4 sur le résiduel. Les coefficients situés en haut à gauche représentent les basses fréquences. Inversement, les coefficients situés en bas à droite représentent les hautes fréquences. On remarque que l'énergie est surtout située dans les basses fréquences comme attendu.

Une fois le résiduel transformé, les coefficients sont quantifiés, c'est-à-dire, qu'ils sont divisés et arrondis vers des valeurs entières. Cette opération a pour effet de détruire de l'information dans le but d'améliorer l'efficacité de la compression. La figure 1.7 montre le résultat d'une quantification uniforme où chaque coefficient a été divisé par 16, puis arrondi. On remarque plusieurs coefficients égaux à zéro suite à cette opération. C'est un résultat recherché, puisque les encodeurs vidéos sont généralement constitués d'une méthode de compression efficace pour encoder une série de zéros. La figure 1.7 montre aussi le résiduel reconstruit après l'application

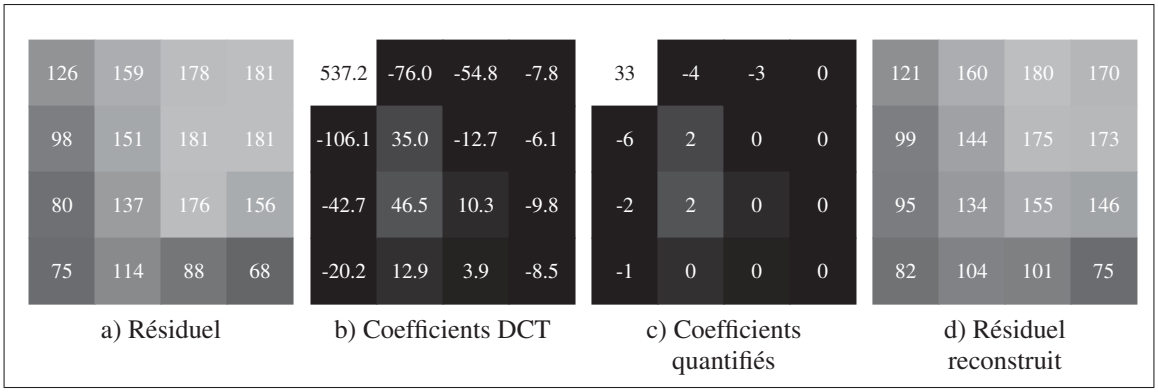


Figure 1.7 Exemple de codage et de reconstruction du résiduel

d'une quantification inverse suivie d'une DCT inverse. On remarque une petite différence entre le résiduel original et le résiduel reconstruit.

Une fois les données quantifiées calculées, l'encodeur applique un codage entropique. Ce codage a pour objectif de convertir les données quantifiées en un train de bits. Nous n'abordons cependant pas ce sujet afin de ne pas alourdir inutilement le texte. Le lecteur doit cependant savoir que la taille du train de bits est seulement connue après l'application du codage entropique. De plus, le codage entropique utilisé dans les encodeurs modernes s'adapte au contexte, c'est-à-dire que le train de bits qui sera généré pour le bloc traité dépend des données qui ont été précédemment encodées.

1.1.1.5 Filtrage des trames reconstruites

L'application de la quantification cause une perte d'information qui a pour effet de réduire la qualité visuelle de la séquence compressée. Cette perte de qualité se manifeste par la présence de différents artéfacts dans les images reconstruites. On compte deux principaux types d'artéfacts :

- L'effet de bloc (blocking) : ce phénomène se caractérise par une discontinuité à la frontière de deux blocs. Il est le résultat d'un traitement effectué de manière indépendante sur les blocs. Il est lié à l'application successive de la transformée et de la quantification ainsi qu'à l'application de méthodes de prédiction.

- Le bruit de moustique (mosquito noise) : ce phénomène se caractérise par un effet de flou et de décoloration près des bordures des objets. Ces distorsions ont l'aspect d'un essaim de moustiques tourbillonnement sur place lorsqu'on considère l'aspect temporel d'une séquence vidéo. Enfin, cet artefact est lié à l'application successive de la transformée et de la quantification, tout comme l'effet de bloc.

Afin de réduire ces artéfacts, plusieurs encodeurs et décodeurs vidéos appliquent un ou des filtres après la reconstruction d'une trame. Afin d'assurer la cohésion entre les données prédites, les mêmes filtres sont appliqués par l'encodeur et le décodeur. Pour réduire l'effet des blocs, la majorité des formats vidéos supportent une méthode de filtrage appelée filtre de déblocage (deblocking filter). Ce filtre est généralement adaptatif. Par exemple, les filtres de déblocage de H.264 et HEVC effectuent d'abord une analyse des données, puis ils appliquent une méthode de filtrage adaptée à la région traitée. L'analyse identifie les discontinuités produites par la compression et rejette les discontinuités naturelles, par exemple, les bordures et les textures, qui ne doivent pas être filtrées. L'analyse sélectionne aussi un filtre adapté au contexte. Par exemple, un bloc intra aura un filtre plus agressif, parce que ce type de bloc est reconnu pour avoir des artéfacts plus apparents. Le filtrage est par la suite appliqué sur la région traitée.

1.1.2 Profils et niveaux

Les standards de compression vidéo ont pour but premier d'assurer l'interopérabilité entre les systèmes. Cependant, les besoins applicatifs et les capacités des systèmes sont très variés. Pour tenir compte de ces différences, les standards de compression vidéo définissent différents profils et niveaux. Le profil représente l'ensemble des outils de codage qui sont mis à la disposition de l'encodeur pour effectuer le codage, alors que le niveau représente les contraintes (résolution maximale, débit maximal, etc.) que l'encodeur doit appliquer. Pour décoder une séquence compressée, un décodeur doit non seulement supporter le standard employé, mais aussi le profil et le niveau.

Le profil supportant les outils de compression les plus simples est habituellement appelé profil de base (*baseline profile*), tandis que le profil supportant les outils de compression les plus couramment utilisés est appelé profil principal (*main profile*).

1.1.3 Contrôle de l'encodage

Un encodeur vidéo doit prendre différentes décisions afin de déterminer les paramètres de codage qui seront utilisés pour encoder la séquence traitée. Les standards de compression vidéo imposent la syntaxe de codage à utiliser ainsi que certaines contraintes, notamment celles définies par les profils et les niveaux. Cependant, aucune règle de décision n'est définie par ces standards. Néanmoins, il existe plusieurs règles de décision et mécanismes de contrôle couramment utilisés par les encodeurs modernes, que ce soit par des encodeurs commerciaux ou des encodeurs de référence (par exemple, l'encodeur HEVC de référence JCT-VC (2014)).

Ainsi, la sélection des paramètres de codage de la séquence traitée s'effectue habituellement en optimisant le débit et la distorsion (la perte de qualité visuelle). Souvent, cette optimisation s'effectue de manière à respecter une contrainte de débit ou de qualité puisque ces deux objectifs sont diamétralement opposés [6]. Par exemple, elle peut consister à minimiser la distorsion en respectant un débit moyen (ou maximal) donné. Ce type d'optimisation s'effectue à l'aide d'un contrôleur de débit (rate control) qui ajuste intelligemment le débit disponible pour l'encodage des prochaines données selon le débit consommé jusqu'à maintenant. Inversement, l'optimisation peut consister à encoder la séquence vidéo de manière à obtenir une qualité relativement constante. Pour ce faire, il est possible d'encoder les trames avec le même paramètre de quantification (*Quantization Parameter*, QP). On parle alors de codage à QP constant.

L'encodage peut aussi être effectué en tenant compte d'autres contraintes. Par exemple, pour une application de vidéoconférence, une contrainte de temps réel sera imposée à l'encodeur. Cette contrainte aura pour effet de limiter la complexité de l'encodage. Pour ce faire, l'encodeur

6. En effet, l'augmentation de la qualité doit s'accompagner d'une augmentation du débit.

vidéo peut opter pour des algorithmes moins efficaces, mais plus rapides. Il peut aussi choisir des paramètres plus restrictifs, par exemple, utiliser une seule trame de référence pour l'estimation de mouvement, au lieu de plusieurs trames.

1.1.3.1 Optimisation débit-distorsion

Pour limiter la complexité de l'encodage, l'optimisation débit-distorsion s'effectue habituellement au niveau d'un superbloc. Elle consiste alors à déterminer le modèle de prédiction optimal. Ce modèle est habituellement constitué d'un mode de partitionnement et d'un mode de prédiction. Le premier mode représente la structure de partitionnement employée pour diviser le superbloc en sous-blocs, tandis que le second type de mode représente, pour chacune des partitions, le type de prédiction (intra ou inter) et les paramètres de prédiction, par exemple, les vecteurs de mouvement.

Pour déterminer le modèle de prédiction optimal d'un superbloc, un encodeur vidéo évalue typiquement plusieurs structures de partitionnement ainsi que plusieurs modes de prédiction pour chacune des partitions évaluées. Parmi les différentes combinaisons évaluées, l'encodeur sélectionne le modèle de prédiction qui minimise une certaine fonction cout. La fonction d'optimisation à employer n'est pas standardisée. Elle correspond généralement à la fonction définie par l'équation suivante :

$$J_{\text{rd}} = \text{SSE}(\mathbf{EC}) + \lambda_{\text{mode}} \times B_{\text{mode}}\,, \tag{1.1}$$

où **EC** contient les différences entre les données originales et les données compressées (les erreurs de compression), B_{mode}, le nombre de bits nécessaires à l'encodage complet (paramètres de prédiction et données résiduelles) du superbloc et λ_{mode} le multiplicateur de Lagrange de cette fonction cout. La fonction SSE, qui représente la somme des erreurs quadratiques (*Sum of Squared Errors*, SSE), est définie par l'équation suivante :

$$\mathrm{SSE}(\mathbf{EC}) = \sum_{i=1}^{M} \sum_{j=1}^{N} (ec_{ij})^2, \tag{1.2}$$

où M et N représentent respectivement la hauteur et la longueur du superbloc traité, et ec_{ij} représente l'échantillon situé à la position (j,i) par rapport au coin supérieur gauche du superbloc traité. La variable λ_{mode} dépend du QP et se calcule habituellement en appliquant l'équation suivante :

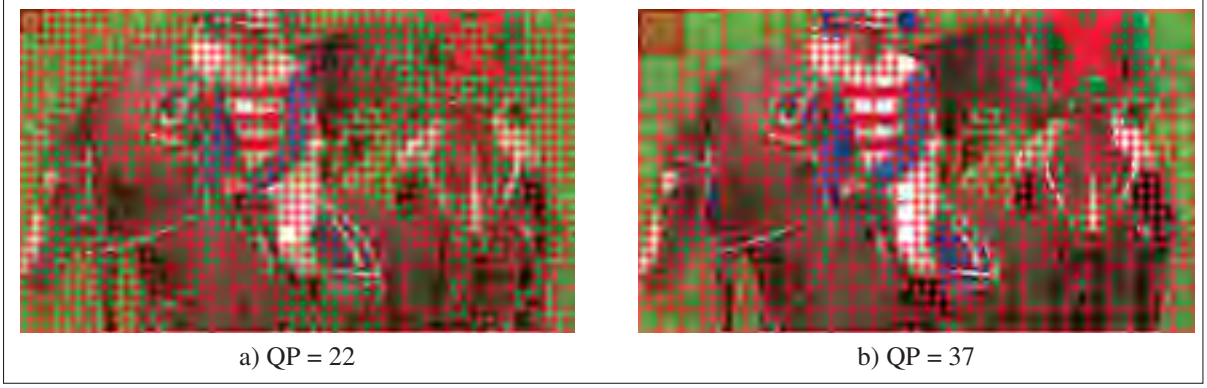
$$\lambda_{\text{mode}} = F * 2^{(QP-12)/3.0}, \tag{1.3}$$

où F représente un facteur qui dépend du standard de compression et de son implémentation. Comme mentionné précédemment, la valeur du QP a un impact important sur le débit et la distorsion. Cette valeur est généralement comprise entre 0 et 51. Lorsque sa valeur est petite, la distorsion résultant de la quantification est plutôt faible. Inversement, l'augmentation de la valeur du QP a généralement pour effet d'augmenter la distorsion et de réduire le débit.

Comme le montre l'exemple sur la figure 1.8, le choix du QP a un impact important sur la structure de partitionnement qui sera sélectionnée par l'encodeur. En effet, le cout du modèle de prédiction dépend en grande partie de sa structure de partitionnement. De manière générale, plus un superbloc est partitionné finement et plus le cout associé à la signalisation de ses paramètres de prédiction sera élevé. En augmentant le QP, on demande à l'encodeur d'attribuer un poids plus important au cout du modèle de prédiction. L'encodeur aura alors tendance à privilégier des partitions plus grossières. L'augmentation du QP a donc pour effet de produire un modèle prédit moins précis (avec une erreur de prédiction plus élevée) afin d'obtenir un débit raisonnable.

1.1.3.2 Évaluation des vecteurs de mouvement

Afin de déterminer le meilleur vecteur de mouvement durant l'estimation de mouvement, les encodeurs vidéos utilisent différentes métriques pour mesurer la distorsion du résiduel (l'erreur de prédiction). Deux métriques sont particulièrement populaires. La première consiste tout



a) QP = 22 b) QP = 37

Figure 1.8 Impact du paramètre de quantification (QP) sur le partitionnement d'une trame HEVC

: les blocs rouges représentent le partitionnement associé au modèle de prédiction, alors que les blocs verts représentant le partitionnement associé aux transformées

simplement à calculer la somme des différences absolues (*Sum of Absolute Differences*, SAD) du résiduel. Ce calcul s'effectue en appliquant l'équation suivante :

$$\mathrm{SAD}(\mathbf{EP}) = \sum_{i=1}^{M} \sum_{j=1}^{N} |(ep_{ij})|, \tag{1.4}$$

où **EP** correspond aux données résiduelles (les différences entre les échantillons du bloc traité et du bloc prédit), aussi appelé l'erreur de prédiction. Pour leurs parts, M et N représentent respectivement la hauteur et la longueur du bloc traité et ep_{ij} représente l'erreur de prédiction à la position (j,i) par rapport au coin supérieur gauche du bloc traité.

La seconde métrique est plus complexe, mais a la propriété d'améliorer l'efficacité du codage en calculant la distorsion dans le domaine fréquentiel. Elle consiste à appliquer la transformée de Hadamard suivie de la SAD. Elle est appelée somme des différences absolues transformées (*Sum of Absolute Transformed Differences*, SATD) et son équation est la suivante :

$$\mathrm{SATD}(\mathbf{EP}) = \mathrm{SAD}(\mathrm{T}(\mathbf{EP})), \tag{1.5}$$

où T représente la transformée de Hadamard définie par l'équation suivante :

$$\mathrm{T}(\mathbf{EP}) = \mathbf{H}_m \cdot \mathbf{EP} \cdot \mathbf{H}_m^T, \tag{1.6}$$

où $\mathbf{H}_m$ représente une matrice de Hadamard de $2^m \times 2^m$. Cette matrice est définie de manière récursive selon l'équation suivante :

$$\mathbf{H}_m = \begin{bmatrix} \mathbf{H}_{m-1} & \mathbf{H}_{m-1} \\ \mathbf{H}_{m-1} & -\mathbf{H}_{m-1} \end{bmatrix}, \tag{1.7}$$

où :

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.8}$$

Afin de tenir compte des bits nécessaires à l'encodage d'un vecteur de mouvement, les encodeurs vidéos utilisent généralement la fonction cout suivante pour évaluer l'efficacité de prédiction d'un vecteur de mouvement :

$$J_{\text{motion}} = \mathrm{D}(\mathbf{EP}) + \lambda_{\text{pred}} \times B_{\text{motion}}, \tag{1.9}$$

où D représente la fonction utilisée pour mesurer l'erreur de prédiction (typiquement, la SAD ou la SATD), B_{motion} correspond au nombre de bits nécessaires à l'encodage de l'information de mouvement du bloc traité et λ_{pred} est le multiplicateur de Lagrange de la fonction cout. La valeur de λ_{pred} correspond généralement à la racine carrée du λ_{mode} et s'obtient en appliquant l'équation suivante :

$$\lambda_{\text{pred}} = \sqrt{\lambda_{\text{mode}}}. \tag{1.10}$$

Nous verrons plus loin comment cette fonction cout a été adaptée à l'encodeur HEVC de référence.

1.1.3.3 Évaluation de l'efficacité de codage

L'efficacité de codage d'un encodeur vidéo est mesurée à l'aide de deux valeurs : le débit et la distorsion. Le débit correspond au nombre de bits moyen par seconde nécessaires à l'encodage d'une séquence vidéo. Pour sa part, la distorsion mesure les différences entre les pixels de la séquence originale et ceux de la séquence reconstruite. Différentes métriques peuvent être utilisées pour évaluer cette distorsion. Une métrique simple consiste à calculer l'erreur quadratique moyenne (EQM) sur la séquence entière en appliquant l'équation suivante :

$$\mathrm{EQM}(\psi_o, \psi_r) = \frac{1}{F \cdot H \cdot W} \sum_{k=1}^{F} \sum_{i=1}^{H} \sum_{j=1}^{W} (\psi_o(i,j,k)) - \psi_r(i,j,k))^2 , \tag{1.11}$$

où H et W représentent respectivement la hauteur et la longueur de l'image, mesurée en pixels, ψ_o la séquence originale, ψ_r la séquence reconstruite et F le nombre d'images dans la séquence. Le rapport signal à bruit de crête (*Peak Signal-to-Noise Ratio*, PSNR) est une mesure beaucoup plus courante en compression vidéo. Elle utilise une échelle logarithmique pour réduire l'étendue des valeurs et elle tient compte de la valeur maximale ψ_{max} de la plage dynamique. L'équation du PSNR est la suivante :

$$\mathrm{PSNR}(\psi_o, \psi_r) = 10 \log_{10} \left(\frac{\psi_{max}^2}{\mathrm{EQM}(\psi_o, \psi_r)} \right) . \tag{1.12}$$

La valeur de ψ_{max} dépend du nombre de bits utilisés pour encoder une composante d'un pixel. Pour 8 bits, cette valeur est égale à 255. Un PSNR élevé est signe d'une faible distorsion.

La relation entre le débit et la distorsion est souvent illustrée sous la forme d'une courbe débit-distorsion. Pour créer cette courbe, on doit encoder plusieurs fois une séquence vidéo avec les mêmes paramètres à l'exception du QP, qu'on modifie d'un encodage à l'autre pour mesurer son influence sur le débit et la distorsion. Le résultat de chaque encodage est illustré sous la forme d'un point dans le graphique. Ces points sont par la suite reliés pour former la courbe débit-distorsion. La figure 1.9 montre l'exemple d'une courbe débit-distorsion produite à l'aide

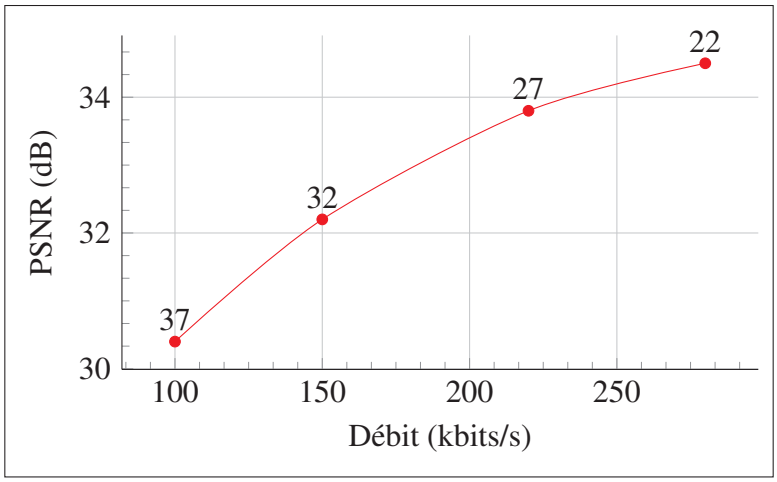


Figure 1.9 Exemple de courbe débit-distorsion

de 4 points (•) obtenus respectivement pour les QPs $22, 27, 32, 37$; des QPs couramment utilisés pour évaluer les performances d'un encodeur.

Pour comparer l'efficacité de codage de deux encodeurs (ou de deux configurations d'encodage), il est commun de dessiner les courbes débit-distorsion de ces encodeurs sur le même graphique. Lorsqu'un encodeur obtient une courbe débit-distorsion située au-dessus de la courbe d'un autre encodeur, cela signifie qu'il a une meilleure efficacité de codage. Deux mesures basées sur les courbes débit-distorsion ont été proposées par Bjøntegaard (2001). Ces mesures sont appelées le *Bjøntegaard delta* (BD)-PSNR et le BD-Rate. Ces deux fonctions calculent respectivement la différence de qualité moyenne (en dB) et la différence de débit moyenne (en pourcentage) entre deux courbes débit-distorsion. Pour ce faire, seule la région illustrée en gris sur la figure 1.10 est utilisée pour faire le calcul. Les détails sur les calculs à effectuer sont présentés dans Bjøntegaard (2001). Le BD-Rate représente l'augmentation moyenne du débit pour une qualité vidéo similaire. Inversement, le BD-PSNR représente la réduction moyenne de la qualité pour un débit similaire.

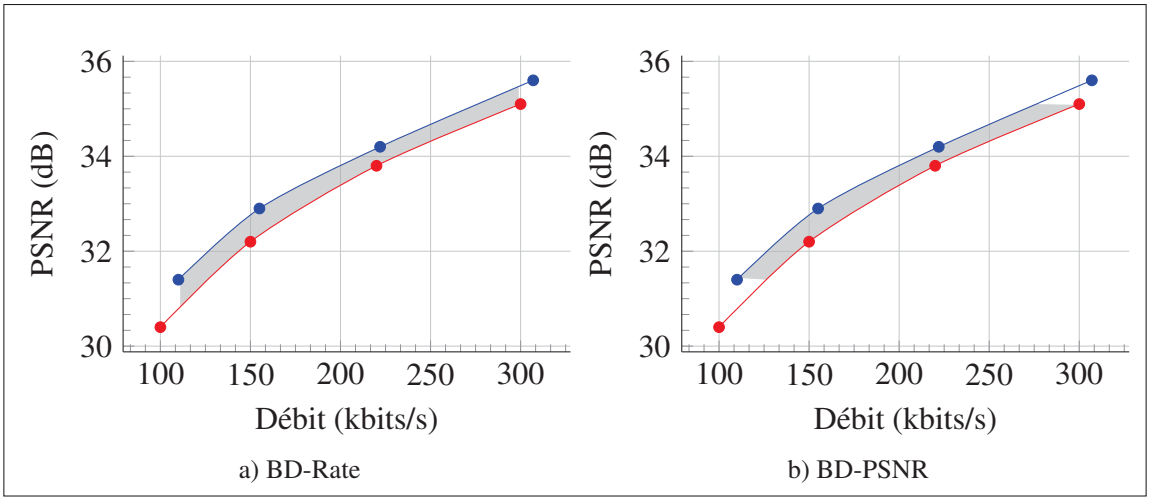


Figure 1.10 Mesure du delta entre deux courbes débit-distorsion

1.2 Le standard de compression vidéo H.264

Le standard de compression vidéo H.264 a été développé conjointement par le *Video Coding Experts Group* (VCEG) de l'ITU et le *Moving Picture Experts Group* (MPEG) de l'ISO, connu sous le nom de la *Joint Collaborative Team on Video Coding* (JCT-VC). Sa publication remonte à mai 2003. Comparativement à son prédécesseur MPEG-4 partie 2, il augmente le taux de compression par un facteur de 2 pour une qualité vidéo similaire. De plus, il offre une interface simple et efficace pour supporter l'intégration de différents types de réseaux (Ethernet, réseau local (*Local Area Network*, LAN), réseaux mobiles, etc.) et différentes formes de stockage. Enfin, il présente un ensemble d'outils et de profils suffisamment large pour supporter un vaste éventail d'applications (vidéophonie, mobile, télévision, etc.) et de conditions réseau (perte et corruption de paquets, taille des paquets, etc.).

Par rapport à ses prédécesseurs, H.264 est constitué de modèles de prédiction plus précis et, généralement, nettement plus efficaces. Il supporte un partitionnement plus fin. Par exemple, sa plus petite partition couvre une région de 4×4 pixels. Ceci lui permet de découper plus finement la trame traitée, notamment pour mieux suivre le mouvement dans les régions complexes. Sa prédiction inter supporte des vecteurs de mouvement précis au quart de pixel et plusieurs trames de référence peuvent être utilisées. De plus, le vecteur de mouvement à encoder est prédit à

l'aide d'un prédicteur médian, ce qui réduit le cout d'encodage des vecteurs de mouvement. Pour sa part, la prédiction intra supporte jusqu'à neuf modes de prédiction.

Sur le plan du codage du résiduel, la DCT a été remplacée par deux nouvelles transformées plus performantes : la transformée entière (une approximation de la DCT) et la transformée de Hadamard. Deux modes d'encodage entropique alternatifs sont supportés par H.264 : le codage à longueur variable adaptable au contexte (*Context-Adaptive Variable-Length Coding*, CAVLC) et le codage arithmétique binaire adaptable au contexte (*Context-Adaptive Binary Arithmetic Coding*, CABAC). Le premier type de codage est peu complexe et se base sur du codage de type Huffman, tandis que le second codage est plus complexe, mais aussi plus efficace. Ces deux modes de codage utilisent des statistiques adaptées au contexte, c'est-à-dire, collectées à l'aide des données encodées précédemment dans la séquence traitée. Enfin, H.264 comporte un filtre de déblocage qui rehausse la qualité des trames reconstruites en atténuant les discontinuités causées par la quantification aux frontières des blocs.

1.2.1 Structures de codage

Comme mentionné précédemment, H.264 partitionne chaque image en superblocs de 16×16 pixels appelés macroblocs. Un macrobloc peut être de type intra, inter ou skip (un cas spécial de prédiction inter). Dans H.264, la combinaison d'un type de prédiction et d'un schéma de partitionnement est appelée mode. La figure 1.11 illustre tous les modes supportés par H.264. Ainsi, H.264 définit deux modes intra : les modes I16×16 et I4×4 qui représentent respectivement un macrobloc intra non partitionné (un bloc de 16×16 pixels) et un macrobloc partitionné en blocs de 4×4 pixels. Quatre modes inter sont supportés, soient les modes P16×16, P16×8, P8×16 et P8×8. Lorsque le mode sélectionné est le mode P8x8, chaque bloc 8×8 composant le macrobloc peut être sous-partionné en blocs de 8×8, 8×4, 4×8 ou 4×4 pixels. Enfin, le mode skip n'est jamais partitionné.

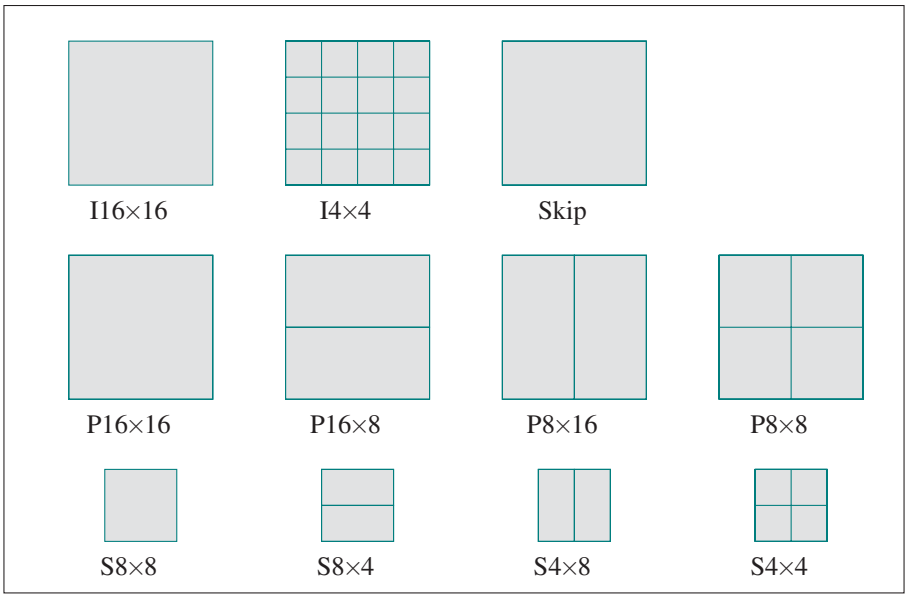


Figure 1.11 Les modes de partitionnement H.264

1.2.2 Prédiction inter

H.264 supporte une prédiction inter précise au quart de pixel. Plusieurs images de référence peuvent être utilisées pour estimer le mouvement. Enfin, le vecteur de mouvement à encoder est prédit par un prédicteur qui sélectionne la médiane de trois vecteurs de mouvement voisins qui ont été encodés précédemment. H.264 définit un mode de prédiction inter spécial appelé mode skip. Ce mode est encodé sans données résiduelles et son vecteur correspond au vecteur prédit. Il est très efficace pour encoder un macrobloc situé dans une région fixe ou simple en mouvement.

1.2.2.1 Calcul des valeurs des pixels à des positions fractionnaires

Les valeurs des pixels situés à des positions fractionnaires sont calculées par interpolation. Un filtre à 6 coefficients est d'abord appliqué sur les pixels aux positions entières afin d'interpoler les demis pixels. Ces derniers sont par la suite utilisés par un filtre linéaire simple afin de produire les positions quarts de pixels.

1.2.2.2 Prédiction de vecteurs de mouvement

Comme mentionné précédemment, les vecteurs de mouvement sont habituellement très fortement corrélés spatialement. H.264 exploite cette corrélation en prédisant le prochain vecteur à encoder (ou décoder) à l'aide d'un prédicteur médian. Seule la différence entre le vecteur prédit et le vecteur trouvé par l'algorithme d'estimation de mouvement est signalée. Globalement, ce mécanisme de prédiction réduit le nombre de bits nécessaires à la signalisation des vecteurs de mouvement. Il est plus particulièrement efficace dans les régions où le mouvement est constant spatialement.

L'algorithme de prédiction extrait d'abord les vecteurs de mouvement de trois blocs qui sont situés dans le voisinage du bloc traité E tel qu'illustré à la figure 1.12. La sélection de ces trois blocs dépend de la structure de partitionnement. La figure montre les blocs ciblés par l'algorithme lorsque les régions voisines de E sont a) non-partitionnées et b) partitionnées en blocs de différentes tailles. Dans le premier cas, le bloc A correspond au bloc situé immédiatement à gauche de E ; le bloc B correspond au bloc situé immédiatement au-dessus de E ; et le bloc C correspond au bloc situé immédiatement à droite du bloc B. S'il y a plus d'une partition au dessus du bloc E, le bloc B correspond alors à la partition la plus à gauche. Une règle similaire est appliquée pour le bloc C. Pour le bloc A, la partition la plus haute est sélectionnée. Une fois les blocs sélectionnés, H.264 calcule la valeur de la prédiction selon les règles suivantes :

- La médiane des vecteurs de mouvement des partitions A, B et C est utilisée pour les partitions qui ne sont pas de format 16×8 et 8×16 ;
- le vecteur de mouvement B est utilisé pour la prédiction d'une partition 16×8 située au niveau supérieur d'un macrobloc et le vecteur de mouvement A est utilisé pour la prédiction de la partition 16×8 inférieure ;
- le vecteur de mouvement A est utilisé pour la prédiction d'une partition 8×16 située à gauche d'un macrobloc et le vecteur de mouvement C pour la prédiction d'une partition 8×16 située à droite.

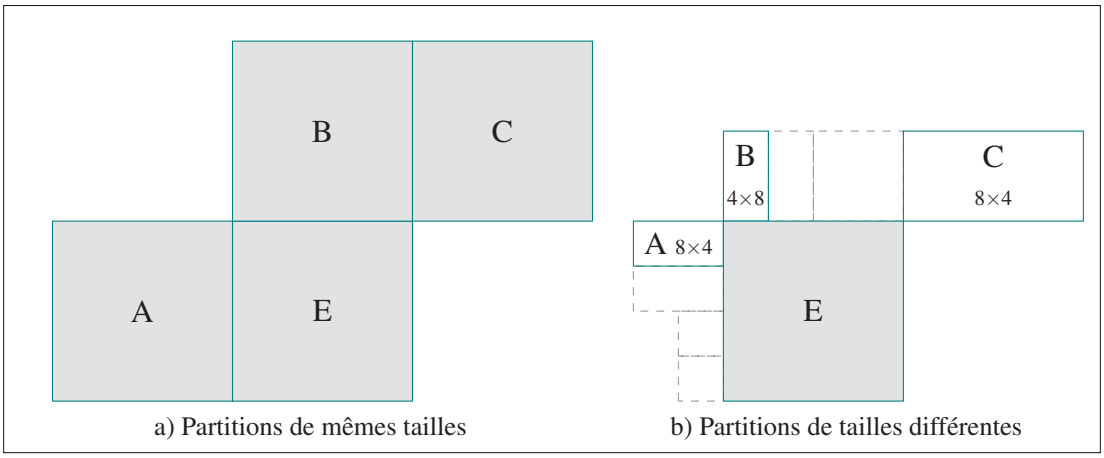


Figure 1.12 Sélection des blocs utilisés par le prédicteur médian de H.264

1.2.3 Prédiction intra

La prédiction intra de H.264 consiste à prédire les pixels du bloc traité à l'aide des pixels voisins immédiats situés dans des blocs qui ont été précédemment encodés puis reconstruits. Le mode de partitionnement I4×4 supporte jusqu'à 9 modes de prédiction (voir la figure 1.13), soit 8 prédicteurs directionnels et un prédicteur non directionnel appelé prédicteur DC. Ce dernier calcule la moyenne des pixels prédicteurs et copie ces valeurs dans le bloc prédit. Pour leur part, les prédicteurs directionnels prolongent les pixels des blocs voisins dans le bloc prédit. Ce prolongement nécessite l'emploi d'une méthode d'interpolation pour les modes 3 à 8. Enfin, certains prédicteurs sont désactivés lorsque le bloc traité est situé immédiatement en dessous de la bordure supérieure de l'image ou immédiatement à droite de la bordure gauche de l'image. Pour sa part, le mode de partitionnement I16×16 supporte jusqu'à 4 modes de prédiction seulement : deux prédicteurs directionnels (un horizontal et l'autre vertical), un prédicteur DC et un prédicteur planaire.

1.3 Le standard de compression vidéo HEVC

Le standard de compression HEVC a été publié en 2013 (ITU-T (2013)). Tout comme son prédécesseur H.264, il a été développé conjointement par le VCEG de l'ITU et le MPEG de l'ISO. Il s'inspire largement des standards précédents, et plus spécifiquement, du standard

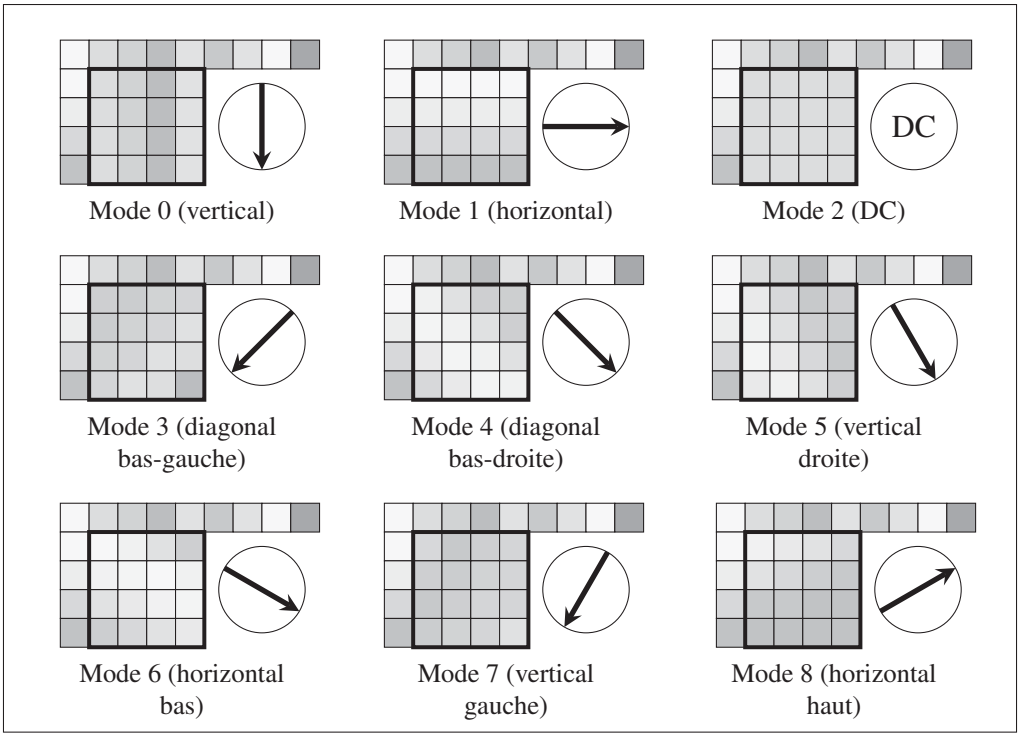


Figure 1.13 Exemples de modes de prédiction intra H.264 : les 9 modes de prédiction intra associés au mode de partitionnement I4×4

H.264. À ce titre, le standard HEVC peut être vu comme une évolution incrémentale de H.264, où des outils existants ont été améliorés, au prix d'une complexité plus grande, pour obtenir un codage plus efficace. Cette évolution est aussi basée sur l'ajout de nouveaux outils de compression et sur l'emploi de nouvelles structures de données.

Plus spécifiquement, le macrobloc a été remplacé par une structure beaucoup plus flexible et efficace appelée unité de codage arborescent (*Coding Tree Unit*, CTU). La prédiction intra supporte maintenant jusqu'à 34 modes de prédiction (comparativement à 9 pour H.264). Les vecteurs de mouvement sont toujours précis au quart de pixel, cependant les filtres d'interpolation ont été améliorés par rapport à ceux de H.264. Le prédicteur médian utilisé par H.264 a été remplacé par deux prédicteurs du mouvement avancés. Un nouvel outil a été ajouté à HEVC pour copier l'information de mouvement d'un bloc voisin à l'intérieur du bloc traité. Cet outil, appelé fusion du mouvement (*motion merge*), supporte jusqu'à cinq candidats. Il est particulièrement efficace dans les régions où le mouvement est uniforme.

Sur le plan du codage résiduel, HEVC supporte deux types de transformées : la DCT entière et la DST. La première peut-être appliquée sur des blocs aussi gros que 32×32 pixels et aussi petits que 4×4 pixels. La seconde est seulement utilisée sur des blocs de 4×4 pixels pour des raisons de complexité et d'efficacité de codage. Contrairement à H.264, seul le codage entropique de type CABAC est supporté par HEVC. Le codage de type CAVLC a été éliminé pour des raisons d'efficacité de codage. Enfin, HEVC supporte deux filtres appliqués successivement pour rehausser la qualité des images reconstruites. Le premier est un filtre de déblocage similaire à celui de H.264, mais adapté à la nouvelle structure de données de HEVC. Le second est un filtre de décalage adaptatif des échantillons (*Sample Adaptative Offset*, SAO). Ce nouveau filtre corrige la valeur de certains pixels (par exemple, ceux situés sur ou à côté d'une bordure) en appliquant un décalage de valeurs sur certains groupes de pixels.

Le tableau 1.1 résume les principales différences entre les standards H.264 et HEVC. Dans le reste de cette section, nous présentons plus en détail la structure de codage de HEVC ainsi que ses méthodes de prédiction inter et intra.

Tableau 1.1 Comparaison des outils de compression H.264 et HEVC

Outils de compression	H.264	HEVC
Taille des superblocs	16×16 (macrobloc)	64×64[1] (CTU)
Partitionnement inter	16×16 à 4×4	64×64 à 8×4 et 4×8
Prédiction du mouvement	Un prédicteur médian	Deux prédicteurs avancés
Copie du mouvement	Un candidat[2]	Cinq candidats[3]
Précision du mouvement	Quart de pixel	Quart de pixel
Partitionnement intra	16×16, 4×4	De 64×64 à 4×4
Prédiction intra	Jusqu'à 9 prédicteurs	Jusqu'à 35 prédicteurs

[1] La taille d'une CTU est configurable de 8×8 à 64×64 pixels. La CTU est de 64×64 pixels lorsque sa taille n'est pas précisée dans le texte.
[2] Pour H.264, la copie du mouvement s'effectue en employant le mode skip. Ce mode copie le mouvement du prédicteur du mouvement et n'encode pas le résiduel.
[3] Pour HEVC, la copie du mouvement s'effectue à l'aide du mode de prédiction appelé mode merge. Ce mode supporte jusqu'à cinq candidats. Lorsque l'encodage des données résiduelles est « sauté », ce mode est plutôt appelé mode skip.

1.3.1 Structures de codage

Le standard HEVC divise le contenu de chaque image en unités de traitement élémentaires appelées CTUs. Cette nouvelle unité est comparable au macrobloc utilisé par les standards

antérieurs. Cependant, elle est beaucoup plus flexible et efficace. Comme nous le verrons plus loin, elle a notamment la particularité de représenter la région traitée sous la forme d'un arbre quaternaire. De plus, sa taille peut être configurée dans l'entête d'une trame. Elle peut correspondre à une région de 64×64, 32×32, 16×16 ou même de 8×8 pixels, alors que la taille du macrobloc est fixée à 16×16 pixels.

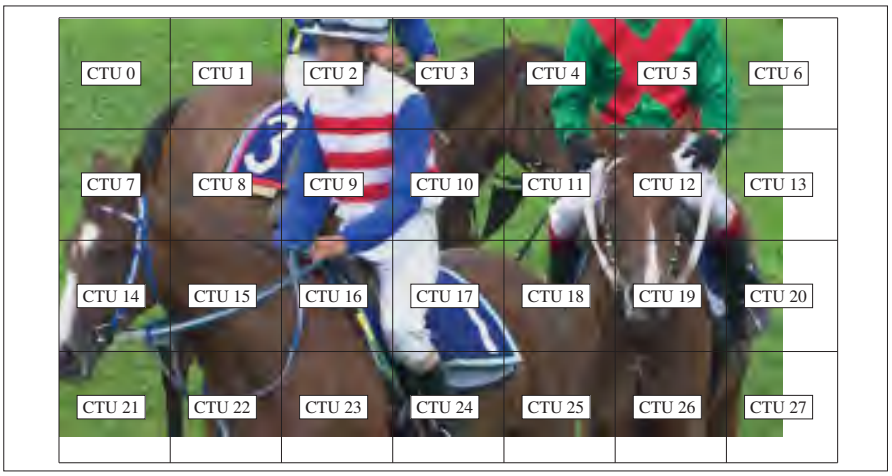
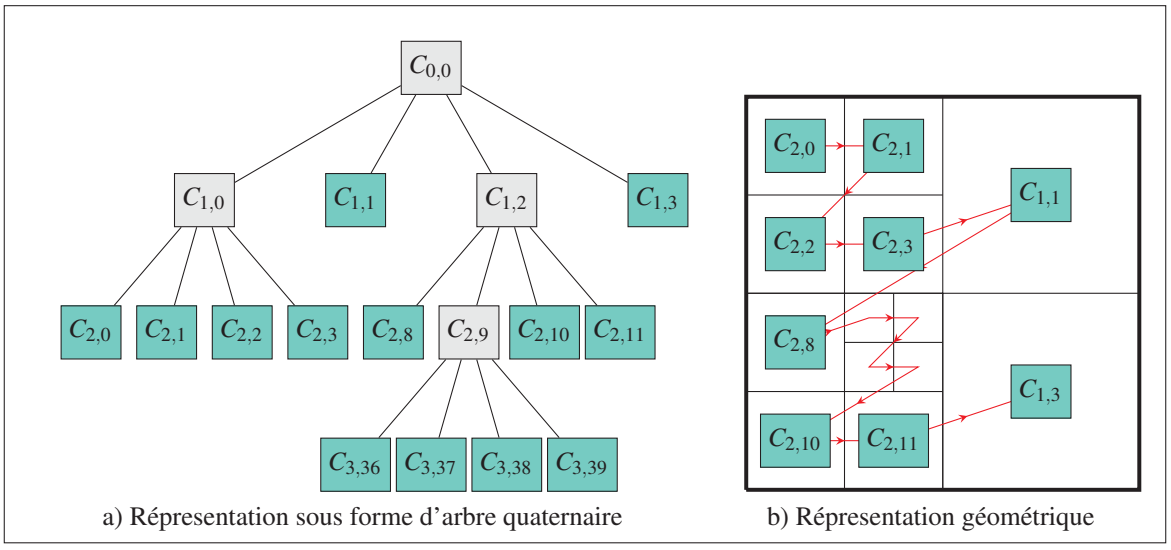


Figure 1.14 Partitionnement d'une trame HEVC de 416×240 pixels en unités de codage arborescent (CTUs) de 64×64 pixels

Les CTUs de grande taille, par exemple, de 64×64 échantillons, sont très efficaces pour représenter des régions où le mouvement est uniforme. Leur efficacité augmente généralement avec la résolution de la séquence traitée. Dans certaines conditions que nous verrons plus loin, la taille d'une CTU peut dépasser le cadre de l'image comme le montre la figure1.14. Enfin, il est à noter que la majorité des travaux de recherche portant sur HEVC sont effectués en considérant seulement des CTUs de 64×64 pixels, puisque c'est la taille qui obtient en moyenne la plus grande efficacité de codage. Pour cette raison, nous utilisons aussi cette taille pour nos travaux de recherche.

1.3.1.1 L'unité de codage

La CTU est partitionnée hiérarchiquement en sous-blocs par l'entremise d'un arbre quaternaire. Dans ce type d'arbre, chaque nœud possède zéro ou quatre enfants. Rappelons qu'un nœud sans enfant correspond à une feuille de l'arbre, alors qu'un nœud sans parent correspond à la racine de l'arbre. Un exemple de partitionnement hiérarchique est illustré sur la figure 1.15. Cet exemple montre l'arbre quaternaire qui est associé à la CTU ainsi que la représentation géométrique correspondant au partitionnement. Chaque nœud de l'arbre correspond à une unité de codage (*Coding Unit*, CU), notée $C_{i,j}$. L'index i représente le niveau de profondeur de la CU dans l'arbre, alors que l'index j représente la j-ième CU située au niveau i. Lorsqu'une CU est partitionnée, ses quatre enfants correspondent aux sous-CUs $C_{i+1,4j+k}$, avec $k = 0 \ldots 3$.



a) Répresentation sous forme d'arbre quaternaire b) Répresentation géométrique

Figure 1.15 Exemple de partitionnement d'une unité de codage arborescent

Le standard HEVC impose différentes contraintes sur la structure de l'arbre. Notamment, le nombre de niveaux de profondeur maximal est établi à 4. De plus, la taille minimale d'une CU est fixée à 8×8 pixels [7]. Par ailleurs, HEVC impose un parcours précis pour l'encodage (et le

7. La longueur et la hauteur des trames d'une séquence doivent être un multiple de la taille de la plus petite CU supportée par l'encodage. Cette taille correspond normalement à 8, mais elle peut être plus grande, c'est selon la configuration de l'encodeur.

décodage) d'une CTU, comme l'illustre en exemple le tracé rouge sur la figure 1.15.b. C'est l'ordre de ce parcours qui définit les règles de dépendances qui existent entre les différents CUs composant la CTU traitée. Deux règles contraignent le parcours de l'arbre quaternaire, comme celui illustré sur la figure 1.15.a. Ces règles sont les suivantes :

a. Un nœud situé à un niveau de profondeur *i* ne peut être traité avant les nœuds de même profondeur (ses frères) situés à sa gauche (dans l'arbre quaternaire) ainsi que leurs descendants.

b. Un nœud ne peut être encodé après ses descendants.

Ces deux règles doivent être appliquées strictement durant l'encodage d'une CTU. Comme nous le verrons plus loin, il est cependant possible de traiter les CUs d'une CTU dans un ordre qui ne respecte pas la règle b.

1.3.1.2 L'unité de prédiction

Chaque CU représentant une feuille de l'arbre est associée à une unité de prédiction (*Prediction Unit*, PU). Cette unité contient l'information de prédiction qui est associée à la CU. Cette information inclut le type de prédiction, le mode de partitionnement et les paramètres de prédiction pour chacune des partitions de la PU. Ces paramètres correspondent au mode de prédiction pour une PU intra, alors qu'ils correspondent à l'information de mouvement (vecteur de mouvement, index de la trame de référence, etc.) lorsque la PU est de type inter. Contrairement à la CTU, qui supporte plusieurs niveaux hiérarchiques de partitionnement, la PU supporte un seul niveau de partitionnement. Jusqu'à 8 schémas de partitionnement sont disponibles (voir figure 1.16).

Comme pour H.264, les schémas de partitionnement disponibles dépendent du type de prédiction. Pour une PU intra, seul le partitionnement 2Nx2N est disponible, sauf lorsque la PU est associée à une CU située au dernier niveau de profondeur de la CTU. Dans un tel cas, la structure de partitionnement NxN est aussi disponible. En somme, la prédiction intra supporte des partitions carrées de 64×64, 32×32, 8×8 ou 4×4 pixels.

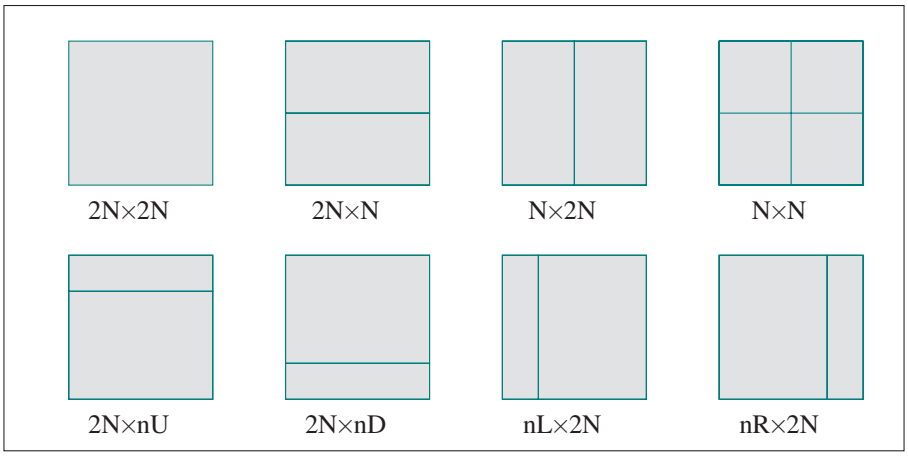


Figure 1.16 Les modes de partitionnement de l'unité de prédiction HEVC

Une PU de type inter supporte toujours au moins trois schémas de partitionnement, soient les modes 2N×2N, 2N×N, et N×2N. Ces schémas de partitionnement sont dits symétriques. Lorsque la PU a une taille égale ou supérieure à 16×16 pixels, quatre schémas supplémentaires sont disponibles. Ces schémas sont dits asymétriques et correspondent aux modes suivants : 2N×nU, 2N×nD, nL×2N et nR×2N. Ainsi, les plus petites partitions supportées par la prédiction inter ont une taille de 8×4 ou 4×8. Nous rappelons que H.264 supporte des partitions inter aussi petites que 4×4 pixels.

1.3.1.3 L'unité de transformée

En plus d'être associée à une PU, chaque CU représentant une feuille de l'arbre est associée à un arbre quaternaire d'unités de transformée (*Transform Unit*, TUs). Ce dernier type d'unité représente la structure de partitionnement utilisée pour appliquer les transformées sur le résiduel. La figure 1.17 montre l'exemple d'un tel arbre.

Jusqu'à trois niveaux de profondeur sont supportés par un arbre de TUs. Chaque partition de l'arbre est associée à une transformée. Les plus grandes partitions (et transformées) supportées sont de 32×32 pixels alors que les plus petites sont de 4×4 pixels.

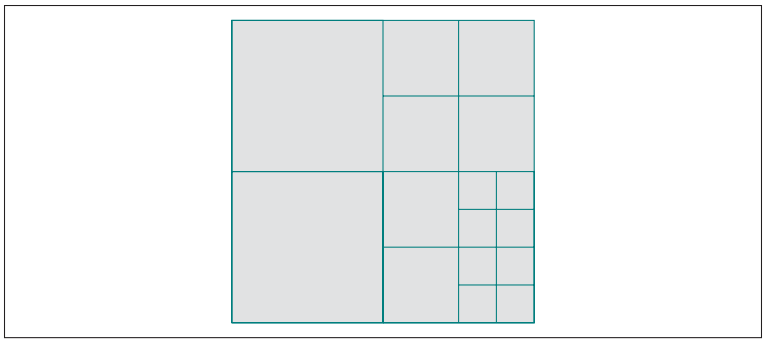


Figure 1.17 Partitionnement hiérarchique d'une unité de transformée

1.3.2 Prédiction inter

HEVC supporte deux modes de prédiction inter : le mode prédiction avancée du vecteur de mouvement (*Advanced Motion Vector Prediction*, AMVP) et le mode fusion du mouvement (*motion merge*). Le premier mode correspond à un modèle de prédiction inter classique, c'est-à-dire, un modèle basé sur l'utilisation d'un algorithme d'estimation de mouvement. Cette méthode prédit le vecteur de mouvement du bloc traité à l'aide d'une méthode de prédiction avancée supportant jusqu'à deux prédicteurs candidats. Lorsque ce mode est utilisé, les paramètres de prédiction à encoder correspondent à l'index du prédicteur candidat retenu, le vecteur de mouvement différentiel (la différence entre le mouvement trouvé par l'algorithme de recherche et le vecteur de mouvement prédit) et l'index de l'image de référence.

Pour sa part, le second mode de prédiction consiste à copier l'information de mouvement d'un bloc voisin candidat dans le bloc traité. Ce mode supporte jusqu'à 5 candidats. L'index du candidat retenu constitue le seul paramètre de prédiction à encoder. Sur le plan du codage des données résiduelles, ce mode supporte un drapeau skip qui permet de sauter l'encodage des données résiduelles lorsqu'il est à vrai.

De manière générale, le mode de prédiction AMVP est efficace dans les régions où le mouvement est changeant, alors que le mode fusion du mouvement est pour sa part efficace dans les régions sans mouvement puisque le signalement de ses paramètres de prédiction

nécessite généralement moins de bits que le mode AMVP pour un même vecteur de mouvement.

1.3.2.1 Copie du mouvement

Le mode merge/skip, utilisé pour copier l'information du mouvement d'un autre bloc, supporte un maximum de cinq vecteurs de mouvement candidats. La liste des candidats est créée en appliquant la procédure définie par HEVC. Cette procédure est appliquée autant par l'encodeur que le décodeur afin d'assurer la cohérence des données prédites. Pour une trame inter de type P [8], cette procédure consiste d'abord à extraire quatre candidats spatiaux uniques et un candidat temporel. Les doublons sont par la suite éliminés. Puis, si la liste est incomplète, un vecteur de mouvement zéro est ajouté.

1.3.2.1.1 Candidats spatiaux

Les quatre candidats spatiaux sont extraits des régions illustrées sur la figure 1.18. Sauf pour les exceptions mentionnées plus bas, l'ordre d'extraction est le suivant : A_1, B_1, B_0, A_0 et B_2. La région B_2 est seulement considérée lorsqu'au moins une des quatre autres régions est non disponible (par exemple, lorsque le bloc traité n'a pas de voisins inférieurs) ou est de type intra.

Pour la seconde partition d'une PU ayant pour mode de partitionnement N×2N, nL×2N ou nR×2N, l'ordre d'extraction est le suivant : B_1, B_0, A_0 et B_2. Pour ce cas, la région A_1 n'est pas considérée puisque cela aura pour effet de créer deux partitions rectangulaires ayant le même vecteur de mouvement. Pour des raisons similaires, la procédure d'extraction de la deuxième partition d'un PU ayant pour mode de partitionnement 2N×N, 2N×nU ou 2N×nD est changée. L'ordre d'extraction devient alors le suivant : A_1, B_0, A_0 et B_2.

8. La procédure est différente pour une trame inter de type B. Ce dernier type de trame n'est pas considéré dans nos travaux.

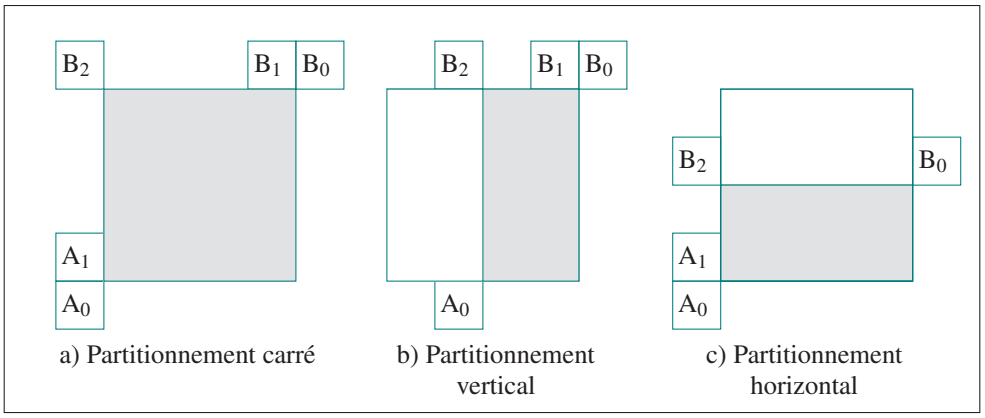


Figure 1.18 Candidats spatiaux pour la prédiction du mouvement : la partition HEVC active est illustrée en gris

1.3.2.1.2 Candidat temporel

Deux régions sont considérées pour extraire le candidat temporel, soient les régions C3 et H illustrées sur la figure 1.19. Ces régions sont situées dans l'image de référence. Le candidat est extrait de la région H si la PU située sur cette région est : disponible, située à l'intérieur de la CTU et de type inter. Sinon, le candidat est extrait de la région C3.

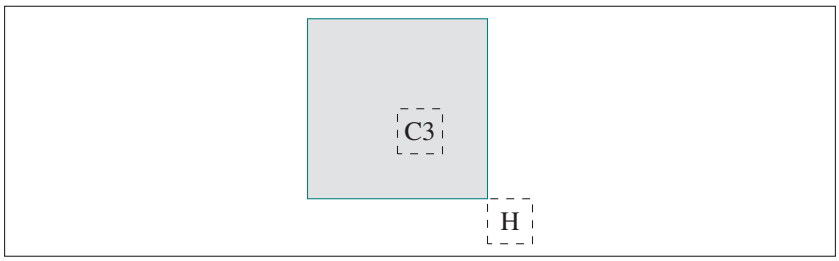


Figure 1.19 Candidats temporels pour la prédiction du mouvement : le bloc gris représente la région de l'image de référence couvrant la partition HEVC active

1.3.2.2 Prédiction avancée du mouvement

Pour sa part, le mode AMVP supporte seulement jusqu'à deux candidats. Cependant, ce mode permet de signaler un vecteur de mouvement différentiel en plus du candidat retenu. Comme

pour le mode merge, ce mode applique la procédure définie par HEVC pour générer sa liste de deux candidats. Cette procédure consiste d'abord à extraire deux candidats spatiaux uniques. Si la liste n'est pas pleine, un candidat temporel est par la suite extrait. Si la liste n'est toujours pas pleine, le vecteur de mouvement zéro est ajouté.

La procédure pour sélectionner les deux candidats spatiaux est légèrement différente de celle du mode merge. Les mêmes régions d'extraction sont considérées, cependant la procédure applique deux parcours : un premier parcours pour extraire un candidat situé à gauche du bloc traité et un second parcours pour extraire un candidat situé au dessus du bloc traité. L'ordre d'extraction du premier parcours est le suivant : A_0 et A_1. Celui du second est pour sa part le suivant : B_0, B_1 et B_2. La procédure pour sélectionner le candidat temporel est la même que celle utilisée par le mode merge.

1.3.3 Prédiction intra

Tout comme H.264, la prédiction intra de HEVC consiste à prédire les pixels du bloc traité à l'aide des pixels voisins immédiats situés dans des blocs qui ont été précédemment encodés puis reconstruits. HEVC définit jusqu'à un maximum de 35 modes de prédiction intra : un mode DC, un mode planaire ainsi que 33 modes de prédiction directionnelle. Ces modes sont illustrés sur la figure 1.20. Pour les modes de prédiction directionnelle, les pixels de prédiction sont filtrés dans certains cas afin d'améliorer la prédiction. L'activation du filtrage dépend de la taille du bloc traité ainsi que de la direction. Sur la figure, les croix représentent les modes non filtrés alors que les points représentent les modes filtrés.

1.4 Encodeur HEVC de référence

Les standards de compression vidéo développés par l'ITU et l'ISO comprennent habituellement l'implémentation d'un encodeur et d'un décodeur de référence. Ces implémentations sont notamment utilisées pour effectuer des tests de conformité. Par exemple, l'implémentation

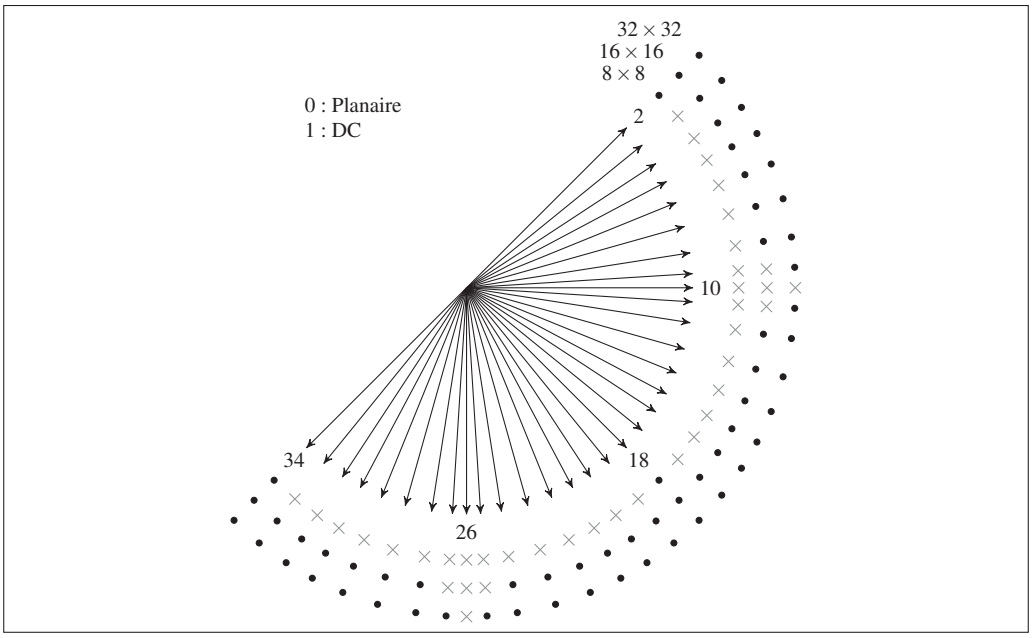


Figure 1.20 Les 35 modes de prédiction intra définis par HEVC

d'un décodeur HEVC doit être en mesure de lire un train de bits qui a été généré par l'encodeur HEVC de référence.

Il est courant en recherche d'implémenter ses propositions à l'intérieur d'un encodeur de référence. D'abord, parce que l'encodeur de référence est la première implémentation complète d'un standard disponible. Cette implémentation est distribuée gratuitement, repose généralement sur une architecture portable et stable (qui change peu d'une version à l'autre). Ensuite, parce les différents encodeurs implémentant un même standard peuvent obtenir des performances très différentes, et cela autant sur le plan de l'efficacité de codage qu'en termes de temps d'exécution. En implémentant leurs propositions dans un même encodeur (l'encodeur de référence, en l'occurrence), les chercheurs facilitent la comparaison de leurs travaux.

Pour ces raisons, nous avons choisi d'implémenter la partie HEVC de notre approche de transcodage dans l'encodeur HEVC de référence, connu sous l'acronyme HM (HEVC Test Model). Dans cette section, nous présentons des aspects du HM qui seront utiles plus loin, notamment pour mieux expliquer nos contributions. D'abord, nous présentons les mesures de distorsion et les fonctions de cout du HM. Puis, nous abordons deux modules que nous allons

remplacer plus loin dans notre approche de transcodage par nos propres méthodes. Le premier module porte sur l'algorithme d'estimation de mouvement du HM, tandis que le second module concerne l'algorithme utilisé pour évaluer les différents modes (structures de partitionnement et modèles de prédiction) d'une CTU.

1.4.1 Mesures de distorsion

Comme nous le verrons plus loin, le HM utilise deux mesures de distorsion durant l'estimation de mouvement pour mesurer la distorsion du résiduel (l'erreur de prédiction). La première est la SAD que nous avons présentée précédemment et qui est définie par l'équation 1.4. La seconde mesure est la SATD que nous avons aussi vue précédemment et qui est définie par l'équation 1.5. Afin de limiter la complexité de cette dernière mesure, le HM limite l'application de la transformée de Hadamard à des régions de 8×8 pixels (ou 4×4 pour les régions plus petites). Pour ce faire, l'encodeur de référence partitionne d'abord le bloc **EP** en blocs de $S \times S$ échantillons, où S est déterminé par l'équation suivante :

$$S = \min(M, N, 8), S \in \{4, 8\}, \tag{1.13}$$

où M et N représentent respectivement la hauteur et la longueur du bloc traité. Par la suite, la SATD (voir équation 1.5) est appliquée sur chacune des partitions, notées $\mathbf{EP}_{i,j}$, à l'aide de l'équation suivante :

$$\mathrm{SATD}(\mathbf{EP}_{i,j}) = \mathrm{SAD}(\mathrm{T}(\mathbf{EP}_{i,j})), \tag{1.14}$$

où T représente la transformée de Hadamard telle que définie précédemment par l'équation 1.6. Enfin, les SATD calculées sur chacune des partitions sont sommées. On obtient alors l'équation suivante :

$$\mathrm{SATD}(\mathbf{EP}) = \sum_{i=1}^{M/S} \sum_{j=1}^{N/S} \mathrm{SAD}(\mathrm{T}(\mathbf{EP}_{i,j})). \tag{1.15}$$

1.4.2 Fonctions cout

Pour évaluer la performance débit-distorsion d'un vecteur de mouvement durant l'estimation de mouvement, le HM utilise la fonction cout J_{motion} définie précédemment par l'équation 1.9. De même, pour évaluer la performance d'un mode, le HM utilise la fonction cout définie J_{rd} par l'équation 1.1. Ce sont des couts qu'il faut minimiser.

1.4.3 Estimation du mouvement

Tel que mentionné à la section 1.3.2, le HM implémente les deux modèles de prédiction inter définis par le standard HEVC, soient le mode merge du mouvement et le mode AMVP. L'implémentation du premier mode consiste à évaluer successivement le cout J_{rd} des 5 vecteurs de mouvement candidats et à sélectionner celui ayant obtenu le plus petit cout. Pour chaque candidat, le cout est évalué avec le drapeau skip activé (ce qui a pour effet de désactiver le codage du résiduel) et désactivé. Pour sa part, le mode AMVP emploie l'algorithme de recherche décrit plus bas. Le traitement de ces deux modes est effectué pour chacune des partitions inter évaluées durant le traitement d'une CTU.

1.4.3.1 Algorithme de recherche de vecteurs de mouvement du HM

Pour chacune des partitions AMVP évaluées, le HM applique un algorithme de recherche du meilleur vecteur de mouvement constitué des quatre étapes suivantes :

- **Étape 1** : Sélection du vecteur de mouvement prédit ;
- **Étape 2** : Recherche du meilleur vecteur de mouvement au pixel entier ;
- **Étape 3** : Raffinement du meilleur vecteur de mouvement au demi et au quart de pixel ;
- **Étape 4** : Mise à jour du vecteur de mouvement prédit.

La première étape a pour objectif d'initialiser le vecteur de mouvement prédit, noté $\mathbf{v}_{\text{p}}$, qui sera utilisé par l'algorithme de recherche durant la seconde étape. Rappelons que HEVC supporte jusqu'à deux vecteurs de mouvement prédits, notés $\mathbb{v}_{\text{AMVP}} = \{\mathbf{V_1}, \mathbf{V_2}\}$. Parmi ces

deux vecteurs, l'algorithme sélectionnera celui obtenant le cout du mouvement le plus bas comme vecteur $\mathbf{v}_p$.

La deuxième étape a pour objectif de trouver le meilleur vecteur de mouvement dans la zone de recherche en considérant seulement les pixels entiers. Deux algorithmes de recherche sont implémentés dans le HM : un algorithme de recherche exhaustive et un algorithme de recherche rapide appelé TZS (*Test Zonal Search*). Comme mentionné précédemment, l'algorithme de recherche exhaustive évalue l'ensemble des vecteurs de mouvement compris dans la zone de recherche. Cet algorithme trouve une solution optimale, cependant il est beaucoup trop complexe pour être utilisé dans une application commerciale. Pour cette raison, nous ne l'étudierons pas davantage. Comme nous le verrons plus bas, le second algorithme effectue une recherche en quatre étapes et comprend deux patrons de recherche. Cet algorithme est nettement plus rapide que l'algorithme de recherche exhaustive, particulièrement lorsque la plage de recherche est grande. Plus loin, nous comparerons les performances de cet algorithme avec les performances de l'algorithme de propagation de mouvement proposé, le tout appliqué dans un contexte de transcodage. Il est donc important de bien comprendre le fonctionnement de l'algorithme TZS ainsi que sa complexité.

La troisième étape consiste à raffiner le vecteur de mouvement entier, trouvé à l'étape précédente, afin d'obtenir un vecteur de mouvement précis au quart de pixel. Ce raffinement s'effectue en deux étapes successives : un raffinement au demi-pixel, puis un raffinement au quart de pixel. Le nombre de positions évaluées durant cette étape est constant et inférieur au nombre de positions évaluées durant la seconde étape. La recherche est d'abord effectuée au pixel entier (étape 2) parce l'évaluation d'une position non entière est beaucoup plus complexe que l'évaluation d'une position entière, notamment parce qu'elle nécessite l'application d'une méthode d'interpolation sur le bloc prédit.

Une fois le meilleur vecteur de mouvement trouvé, la quatrième détermine quel vecteur de mouvement prédit obtient le cout le plus bas. Le vecteur $\mathbf{v}_p$ est mis à jour si nécessaire.

1.4.3.1.1 Recherche TZS

L'algorithme de recherche TZS se divise en quatre étapes et est constitué de deux principaux patrons de recherche : un patron de recherche en losange et un patron de recherche carré couvrant la zone de recherche.

La première étape de l'algorithme sélectionne le meilleur point de départ, noté $\mathbf{v}_{\mathrm{i}}$, parmi les candidats suivants : les deux vecteurs de mouvement prédits et le vecteur de mouvement zéro, noté $\mathbf{v}_{\mathrm{zero}}$. Le candidat minimisant la fonction cout J_{motion} du mouvement (avec pour mesure de distorsion la SATD) est sélectionné comme vecteur de mouvement initial.

La seconde étape effectue une recherche itérative en losange sur des pixels entiers. Le cout d'une position est évalué à l'aide de la fonction J_{motion} (avec pour mesure de distorsion la SAD). La figure 1.21.a montre les vecteurs de mouvement évalués durant les trois premières itérations. À l'exception de la première itération, le patron en losange est composé de 8 vecteurs de mouvement situés à une distance de Manhattan de d pixels du vecteur de mouvement $\mathbf{v}_{\mathrm{i}}$. La valeur de d est égale à 1 à la première itération et double de valeur à chaque nouvelle itération. Le traitement itératif se termine lorsque d est égal à la plage de recherche (64 pixels, par défaut) ou lorsque les trois itérations les plus récentes n'ont pas permis de trouver un nouveau meilleur vecteur de mouvement. Ainsi, le nombre d'itérations minimal est égal à trois et correspond à 20 vecteurs de mouvement testés. Pour une plage de recherche de 64 pixels, le nombre d'itérations maximal est de 7 et correspond à 52 vecteurs de mouvement évalués.

La troisième étape effectue une recherche par balayage sur la zone de recherche complète. Cette recherche est seulement activée lorsque le meilleur vecteur de mouvement a été trouvé après la troisième itération (lorsque le meilleur vecteur de mouvement est situé à distance d plus grande que 5 du vecteur de mouvement initial). Dans le cas contraire, l'algorithme suppose que le meilleur vecteur de mouvement est situé près du vecteur de mouvement initial et qu'il est donc inutile d'effectuer une recherche couvrant l'ensemble de la zone de recherche. La figure 1.21.b montre un exemple de recherche par balayage pour une plage de recherche de 8 pixels. La recherche débute dans le coin supérieur gauche. Par la suite, la méthode parcourt la

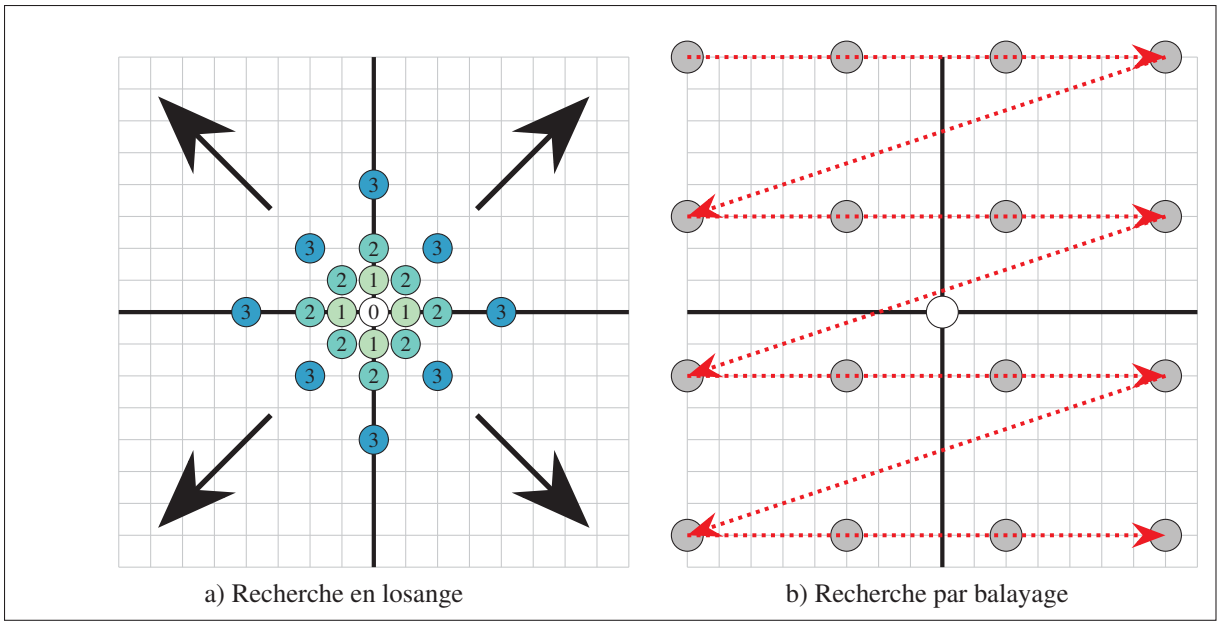


Figure 1.21 Méthodes de recherche de l'algorithme TZS du HM : a) le cercle numéroté «0» représente le vecteur de mouvement initial ; les autres cercles, numérotés de «1» à «3», représentent les vecteurs de mouvement évalués par les trois premières itérations de l'algorithme

région de recherche de la gauche vers la droite et de haut en bas. La méthode se déplace par bonds de 5 pixels et couvre l'ensemble de la zone de recherche. Pour une plage de recherche de 64 pixels, 169 positions sont évaluées.

Enfin, la quatrième étape effectue un raffinement du meilleur vecteur de mouvement à l'aide de la recherche en losange. Cette fois-ci, le vecteur de mouvement initial de la recherche est le meilleur vecteur de mouvement trouvé durant les étapes précédentes. Aussi, le traitement itératif se termine lorsque le meilleur vecteur de mouvement n'a pas été trouvé dans l'une des deux dernières itérations (et non trois, comme pour la deuxième étape).

1.4.3.1.2 Raffinement au demi et quart de pixel

Le raffinement du meilleur vecteur de mouvement entier consiste à effectuer successivement un raffinement au demi et au quart de pixel. Pour ce faire, une recherche est réalisée avec un patron carré d'un demi-pixel de distance du meilleur vecteur de mouvement entier. Par la suite,

le meilleur vecteur au demi-pixel trouvé durant ce raffinement devient le point central d'une recherche carrée d'un quart de pixel. La figure 1.22 montre un exemple de raffinement. Enfin, il est important de noter que le cout des positions évaluées est obtenu en employant la fonction J_{motion} avec pour mesure de distorsion la SATD.

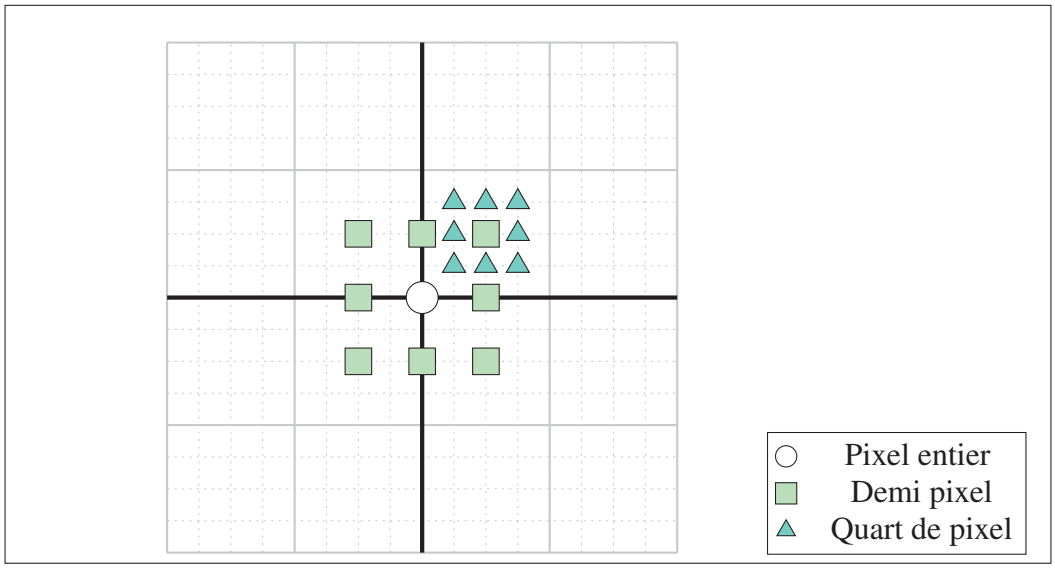


Figure 1.22 Exemple du raffinement au quart de pixel effectué par le HM

1.4.4 Évaluation des modes

Le processus d'évaluation des modes a pour rôle d'évaluer les différents modes (modes de partitionnement, modes de prédiction) d'une CTU afin de déterminer le meilleur mode (les paramètres de codage qui seront encodés). C'est un processus de haut-niveau très important puisqu'il a un impact sur l'efficacité de codage et sur la complexité de l'encodeur. Ce processus doit notamment déterminer : l'ordre de parcours de la CTU traitée, l'ordre d'évaluation des PUs, les conditions d'arrêt du traitement s'il y a lieu et les critères d'évaluation.

Pour sa part, le HM visite les CUs de la CTU traitée selon un parcours préfixe (aussi appelé parcours préordre) comme le montre la figure 1.23. Lorsqu'une CU est visitée, ses PUs sont évaluées dans l'ordre suivant : PUs merge/skip, PUs AMVP et PUs intra. Les PUs AMVP

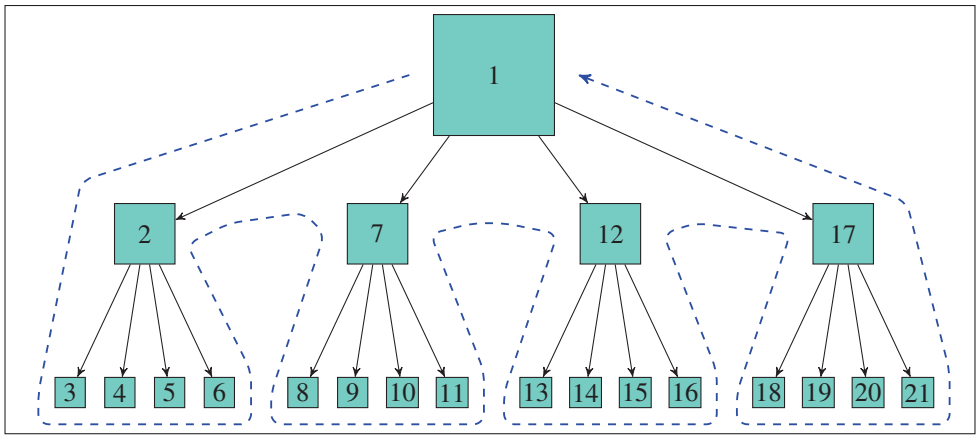


Figure 1.23 Traversée d'un arbre quaternaire selon un parcours préfixe

sont évaluées selon leur mode de partitionnement dans l'ordre suivant : 2N×2N, 2N×N, N×2N, 2N×nU, 2N×nD, nL×2N, nR×2N [9]. L'évaluation d'une PU consiste à déterminer ses paramètres de prédiction et à calculer son cout J_{rd}. La PU ayant obtenu le cout le plus bas est sélectionnée comme meilleur mode.

Une fois l'ensemble des PUs évaluées, le processus traite récursivement les descendants de la CU en appliquant le parcours préfixe. Puis, lorsque l'ensemble des descendants a été traité, le cout J_{rd} de la CU est comparé avec les couts J_{rd} combinés de ses quatre enfants. Pour réduire la complexité de ce processus, le HM bloque le parcours récursif des enfants d'une CU lorsque le meilleur mode ne contient aucun résiduel encodé.

1.5 Résumé

Dans ce premier chapitre, nous avons présenté les fondements du codage vidéo qui seront nécessaires à la compréhension des prochains chapitres. Nous avons d'abord abordé les notions fondamentales du codage vidéo en portant une attention particulière aux modèles de prédiction spatiale et temporelle. Nous verrons au prochain chapitre que plusieurs approches de transcodage exploitent ces modèles pour accélérer le traitement. Par la suite, nous avons étudié les spécificités des standards de codage H.264 et HEVC en présentant notamment les

9. Les partitions asymétriques sont seulement évaluées dans certaines conditions.

différences entre les modèles de prédiction de ces deux standards. Enfin, nous avons présenté l'encodeur HEVC de référence, appelé le HM, en décrivant notamment ses algorithmes d'estimation du mouvement et son approche pour évaluer les modes HEVC.

CHAPITRE 2

LE TRANSCODAGE VIDÉO

Ce chapitre présente, dans un premier temps, une brève introduction au domaine du transcodage vidéo. Cette introduction porte sur les motivations du transcodage, les opérations de transcodage, les principales architectures de transcodage ainsi que quelques techniques communes de transcodage. Dans un second temps, ce chapitre présente l'état de l'art du transcodage vidéo H.264 et HEVC. Cette revue de la littérature porte plus spécifiquement sur les deux problèmes qui seront traités par la suite dans cette thèse, soit la réduction des modes à tester durant l'évaluation d'une CTU et l'accélération de l'estimation de mouvement.

2.1 Les principes du transcodage vidéo

Le transcodage vidéo consiste à convertir une séquence vidéo source, en lui appliquant un nouvel ensemble de paramètres (format vidéo, débit, etc.), afin de produire la séquence cible (aussi appelée séquence transcodée). Pour ce faire, un transcodeur vidéo peut appliquer différentes opérations de transcodage comme l'illustre la figure 2.1. Les opérations plus courantes sont : la réduction du débit, la réduction de la résolution spatiale, la réduction de la résolution temporelle et le changement de format de codage. D'autres opérations, qui ne modifient pas les paramètres de codage, peuvent être appliquées, comme l'ajout d'un tatouage numérique ou d'un logo.

Le transcodage est dit homogène lorsque les fichiers source et cible ont le même format de codage. Lorsque ce format diffère, on parle plutôt de transcodage hétérogène. Ce dernier type de transcodage est souvent appliqué pour convertir une séquence existante vers un format plus récent, par exemple, pour passer d'un format H.264 à un format HEVC. La séquence produite par le transcodeur bénéficie alors des avantages du nouveau format, en particulier d'une plus grande efficacité de codage. Dans d'autres situations, le transcodage hétérogène est plutôt utilisé pour convertir la séquence source vers un format de codage plus ancien, souvent pour des raisons de compatibilité.

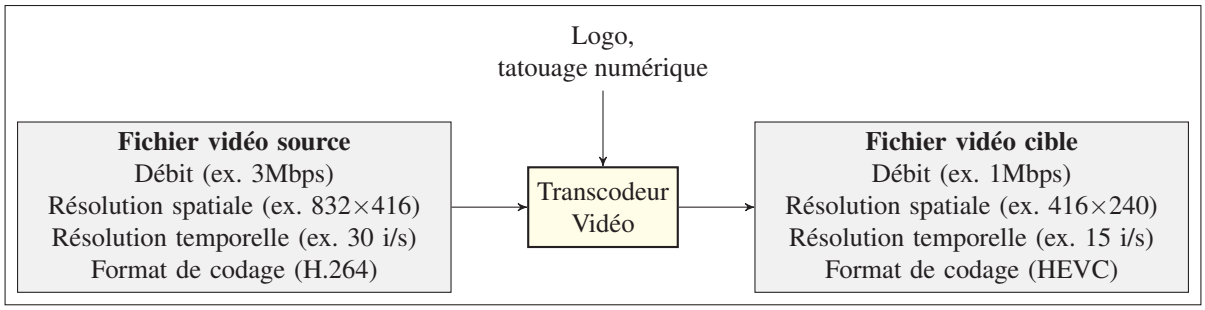


Figure 2.1 Opérations de transcodage vidéo
Adaptée de Ahmad (2005, p.1)

Sur le plan applicatif, le transcodage vidéo a pour objectif premier d'assurer l'interopérabilité entre les différents systèmes. Comme le souligne Dogan *et al.* (2004), le transcodage assure l'accès multimédia universel (*Universal Multimedia Access*, UMA) en permettant la conversion d'un fichier multimédia existant vers un nouveau format qui respecte les contraintes imposées par le réseau et l'appareil d'un client cible. Ce transcodage peut être effectué sur demande, en adaptant une séquence vidéo après avoir reçu les contraintes (débit, résolution spatiale, etc.) du client. Il peut aussi être effectué préalablement, en créant plusieurs versions d'une même séquence. Par la suite, le client fournit ses contraintes au serveur et ce dernier choisit pour lui la version la plus appropriée. Alternativement, le client peut interroger le serveur sur les différentes versions disponibles, puis choisir la version qu'il désire obtenir. Ce dernier type d'approche est très populaire de nos jours (Stockhammer (2011)).

2.1.1 Les architectures de transcodage vidéo

Il existe plusieurs architectures de transcodage vidéo. Certaines sont très générales et permettent d'effectuer différents types d'opérations de transcodage, c'est notamment le cas pour l'architecture en cascade. Au contraire, d'autres architectures ont été conçues pour répondre à des opérations de transcodage spécifiques. Par exemple, l'architecture de transcodage en boucle fermée est seulement utilisée pour effectuer un transcodage homogène qui réduit le débit. En se limitant ainsi à une seule opération de transcodage, cette architecture est en mesure de réduire la complexité du transcodage significativement.

Comme le mentionne Ahmad *et al.* (2005), les architectures de transcodage doivent idéalement répondre aux trois exigences suivantes :

- L'information de codage (vecteur de mouvement, résiduel, modes, etc.) du fichier source doit être exploitée au mieux pour encoder le fichier cible ;
- La qualité de la séquence cible doit être proche de celle de la séquence source ;
- La complexité du transcodage doit être faible. Idéalement, on doit cibler une complexité qui permet d'effectuer du traitement en temps réel.

2.1.1.1 Architecture de transcodage en cascade

L'architecture de transcodage la plus simple est appelée transcodage en cascade (figure 2.2). Dans ce type d'architecture, la séquence source est entièrement décodée, puis réencodée de manière à respecter l'ensemble des nouveaux paramètres. Notons que le mouvement est reconstruit au décodage de la séquence source à l'aide de la Compensation du mouvement (CM), puis qu'une nouvelle Estimation du mouvement (EM) est effectuée durant l'encodage de la séquence cible. Cette architecture est très flexible puisqu'elle n'impose aucune contrainte sur l'encodage, à l'exception des contraintes définies par l'utilisateur. C'est aussi l'architecture obtenant la meilleure efficacité de codage puisqu'elle n'utilise aucun raccourci pour accélérer le traitement. Cependant, c'est aussi l'architecture obtenant les temps d'exécution les plus lents. Dans les faits, sa complexité calculatoire correspond à la complexité combinée d'un décodage et d'un encodage conventionnels.

2.1.1.2 Architecture de transcodage en boucle ouverte

À l'opposé du transcodage en cascade, on retrouve le transcodage en boucle ouverte (figure 2.3). Dans ce type d'architecture, la séquence source est décodée partiellement de manière à extraire les données résiduelles quantifiées $R_n^{'f}$. Ces données sont par la suite re-quantifiées, pour former les données résiduelles $R_n^{''f}$, puis encodées conjointement avec l'information de codage originale. En sautant ainsi plusieurs étapes de codage, cette architecture enregistre les

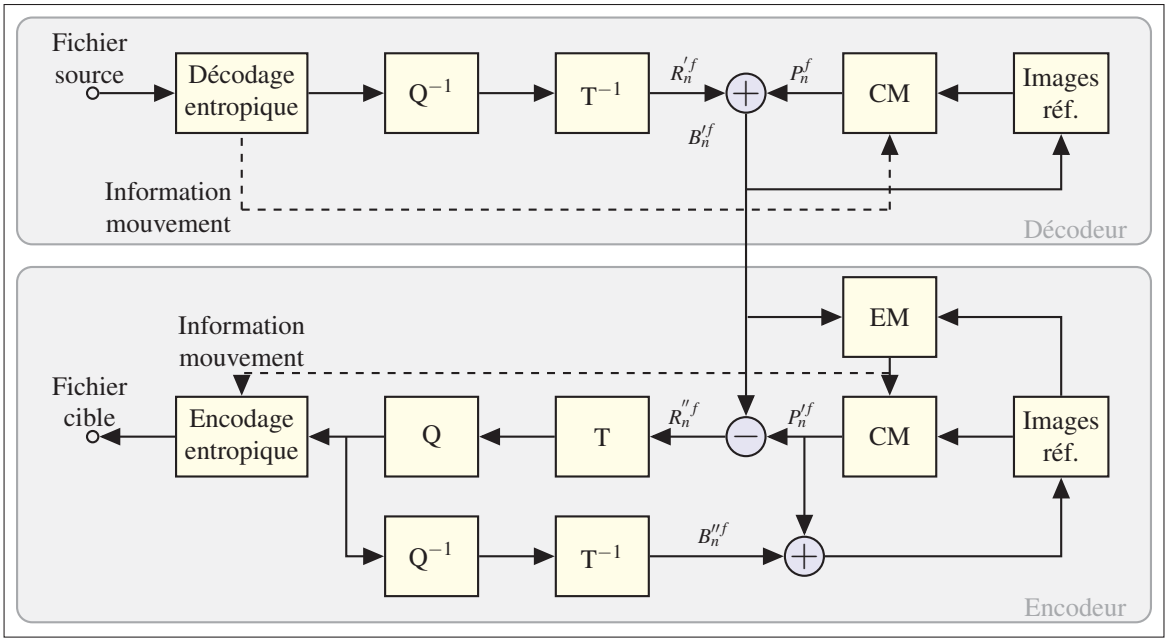


Figure 2.2 Architecture de transcodage en cascade
Adaptée de Negi (2013, p.353)

temps d'exécution les plus faibles. En contrepartie, elle obtient aussi la pire dégradation de la qualité visuelle. Ce type de transcodage est surtout utilisé pour réduire le débit d'une séquence existante.

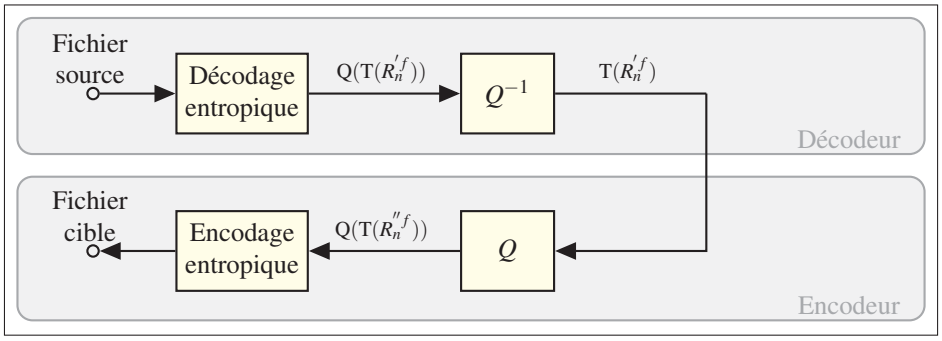


Figure 2.3 Architecture de transcodage en boucle ouverte
Adaptée de Negi (2013, p.354)

Cette dégradation est causée en partie par l'incohérence qui existe entre les images de référence considérées par le transcodeur et celles qui sont réellement utilisées pour décoder la séquence

transcodée. En effet, en modifiant les données résiduelles de la séquence source, le transcodeur vient aussi modifier les images qui seront reconstruites durant le décodage. Comme nous le savons, ces images sont par la suite utilisées par les différentes méthodes de prédiction. Cependant, le transcodage ne tient pas compte de ces modifications, puisqu'il continue à utiliser les données résiduelles originales (celles basées sur les données prédites P_n^f), et non les données résiduelles recalculées selon la nouvelle prédiction $P_n'^f$, comme le fait l'architecture de transcodage en cascade. Cette incohérence a pour effet de produire une erreur, appelée erreur de dérive (*drift error*) (Qian *et al.* (2006)), qui se propage et s'accumule de trame en trame. Cette propagation s'arrête seulement à la prochaine trame intra.

2.1.1.3 Architecture de transcodage en boucle fermée

L'architecture de transcodage en boucle fermée (figure 2.4) est une solution intermédiaire entre le transcodage en cascade et le transcodage en boucle ouverte. Dans ce type d'architecture, le résiduel $P_n'^f$ de la séquence source est corrigé de manière à tenir compte de la re-quantifacation effectuée durant le transcodage. En corrigeant ainsi le résiduel, cette architecture élimine l'erreur de dérive. Contrairement à l'architecture de transcodage en cascade, qui reconstruit les trames de référence au décodage de la séquence source et à l'encodage de la séquence cible, l'architecture en boucle fermée reconstruit seulement les images de référence durant le réencodage. En procédant ainsi, la complexité est réduite, cependant le transcodeur n'est pas en mesure de ré-estimer le mouvement, ce qui affecte négativement l'efficacité de codage.

2.1.2 Le transcodage homogène

Comme mentionné précédemment, le transcodage homogène consiste à appliquer des opérations de transcodage sur une séquence source sans modifier son format de codage. Pour chaque opération de transcodage, on dénombre un grand nombre de techniques de transcodage homogènes dans la littérature. Plusieurs de ces techniques sont aussi applicables, ou adaptables, au transcodage hétérogène. Nous présentons sommairement quelques-unes de ces techniques afin de donner au lecteur un aperçu des différents travaux de recherche sur le transcodage.

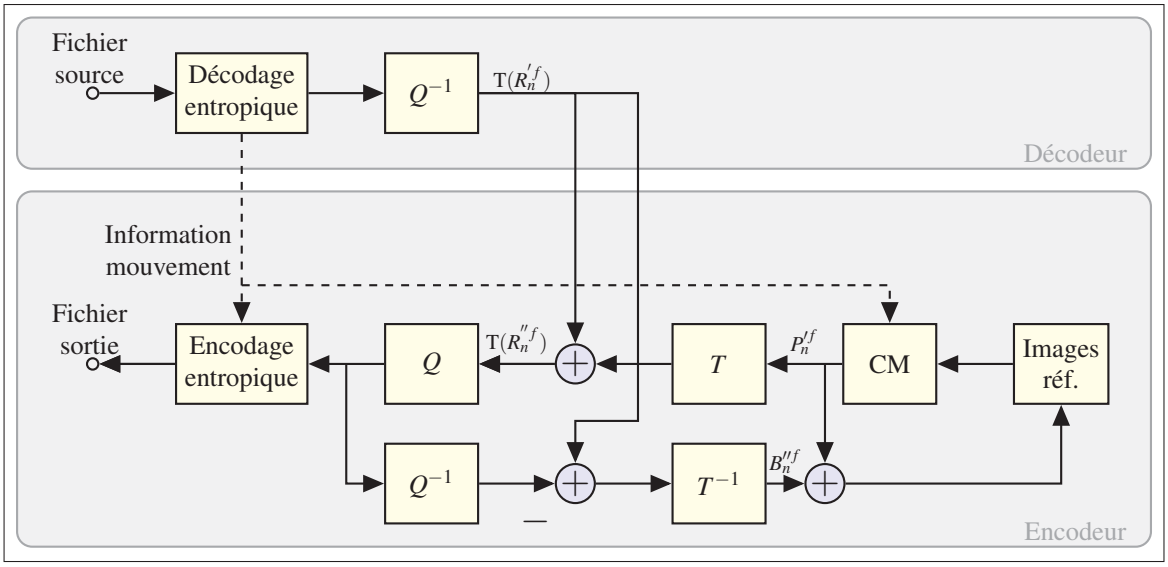


Figure 2.4 Architecture de transcodage en boucle fermée
Adaptée de Negi (2013, p.354)

2.1.2.1 Réduction du débit

L'utilisation la plus courante du transcodage homogène est la réduction du débit. Cette réduction peut s'effectuer en réencodant la séquence vidéo source à l'aide d'un QP plus élevé, en employant un transcodeur en cascade, par exemple. Cependant, on compte deux autres techniques de transcodage qui réduisent le débit en manipulant seulement les données résiduelles. Une première technique consiste à tronquer les coefficients de hautes fréquences (Eleftheriadis et Anastassiou (1995)). Cette technique est simple d'implémentation, mais produit des artéfacts visibles dans les régions détaillées (texturées). Une seconde technique consiste à re-quantifier les coefficients (Nakajima *et al.* (1995)) afin de produire des artefacts moins visibles. Pour bénéficier d'un faible temps d'exécution, ces deux techniques peuvent être implémentées dans une architecture en boucle fermée ou en boucle ouverte.

2.1.2.2 Réduction de la résolution spatiale

Une autre utilisation courante du transcodage homogène est la réduction de la résolution spatiale. Cette opération a non seulement pour effet de réduire la taille des images sources,

mais aussi de réduire le débit. Elle est généralement utilisée pour respecter une contrainte de résolution maximale, par exemple, imposée par l'appareil cible. Cependant, elle est aussi utilisée pour offrir un bon rapport qualité/débit, comme le mentionnent (Joset et Coulombe (2013)).

En pratique, les images sources sont généralement réduites dans le domaine spatial, par exemple, en appliquant un filtre passe-bas suivi par un sous-échantillonnage des pixels. D'autres méthodes moins complexes ont été proposées pour effectuer le redimensionnement dans le domaine fréquentiel (Chang et Messerschmitt (1995) et Shen *et al.* (1999)). Cependant, ces méthodes sont adaptées à une architecture de transcodage effectué dans le domaine fréquentiel. Ce type d'architecture est reconnu pour produire un codage peu efficace (Ahmad *et al.* (2005)).

Le redimensionnement des images sources a pour effet de rendre les vecteurs de mouvement sources inutilisables pour l'encodage de la séquence cible. En effet, ces vecteurs doivent être adaptés par le transcodeur, d'une part, pour tenir compte du facteur de redimensionnement ; d'autre part, pour respecter les structures de partitionnement disponibles. Prenons l'exemple d'un format vidéo qui supporte des partitions aussi petites que 4×4 pixels et d'une réduction spatiale diminuée par un facteur de 2 :1. Dans une telle situation, jusqu'à 4 vecteurs de mouvement source peuvent couvrir une partition cible de 4×4 pixels. Le transcodeur doit alors calculer un nouveau vecteur de mouvement à l'aide de ces 4 vecteurs.

Pour ce faire, plusieurs méthodes ont été proposées, dont l'utilisation du vecteur de mouvement dominant, de la moyenne des 4 vecteurs de mouvement et de la médiane des 4 vecteurs de mouvement (Björk et Christopoulos (1998)). Une fois le vecteur de mouvement calculé par l'une de ces méthodes, son amplitude est réduite de moitié pour tenir compte du facteur de redimensionnement. Enfin, un algorithme de raffinement [1] peut être appliqué sur le vecteur,

1. Un algorithme de raffinement du mouvement est un algorithme de recherche qui cherche un meilleur vecteur de mouvement dans une petite zone de recherche entourant le vecteur actuel. Nous verrons plus en détail ce concept à la section 2.2.2.

notamment pour corriger une erreur de précision du redimensionnement ou un mauvais sous-échantillonnage (la sélection d'un vecteur H.264 traduisant mal le mouvement).

2.1.2.3 Réduction de la résolution temporelle

Une opération moins courante est la réduction de la résolution temporelle. En plus de réduire le nombre d'images par seconde, cette opération a pour effet de réduire le débit. Elle est souvent appliquée pour assurer un décodage en temps réel sur des appareils ayant un processeur peu performant. De manière générale, la réduction temporelle est effectuée en éliminant certaines trames de la séquence source. Par exemple, il est courant d'éliminer une trame sur deux pour réduire la résolution temporelle de moitié.

Cette élimination a pour effet de rendre la prédiction de plusieurs trames inter invalides. Puisque leurs vecteurs pointent maintenant sur des trames de référence inexistantes (qui ont été retranchées de la séquence). Le mouvement doit donc être de nouveau estimé en utilisant cette fois-ci les nouvelles trames de référence. Pour ne pas ré-estimer le mouvement au complet, plusieurs méthodes ont été proposées dans la littérature, dont : l'interpolation bilinéaire du mouvement (Hwang et Wu (1998)), la reconstruction du parcours des vecteurs de mouvement dominants (Youn *et al.* (1999)), la composition vectorielle télescopique (Shanableh et Ghanbari (2000)).

2.1.3 Le transcodage hétérogène

Le transcodage hétérogène emploie des techniques similaires à celles utilisées par le transcodage homogène. Il se distingue de ce dernier type de transcodage essentiellement par le besoin de supporter une nouvelle syntaxe d'encodage, ainsi que de nouvelles structures de données et de nouveaux outils de compression. Plusieurs approches ont été développées pour ce type de transcodage, notamment pour convertir des séquences MPEG-4 en séquences H.264, et des séquences MPEG-2, en séquences H.264. Certaines de ces techniques sont présentées dans la prochaine section, qui est consacrée au transcodage H.264 à HEVC.

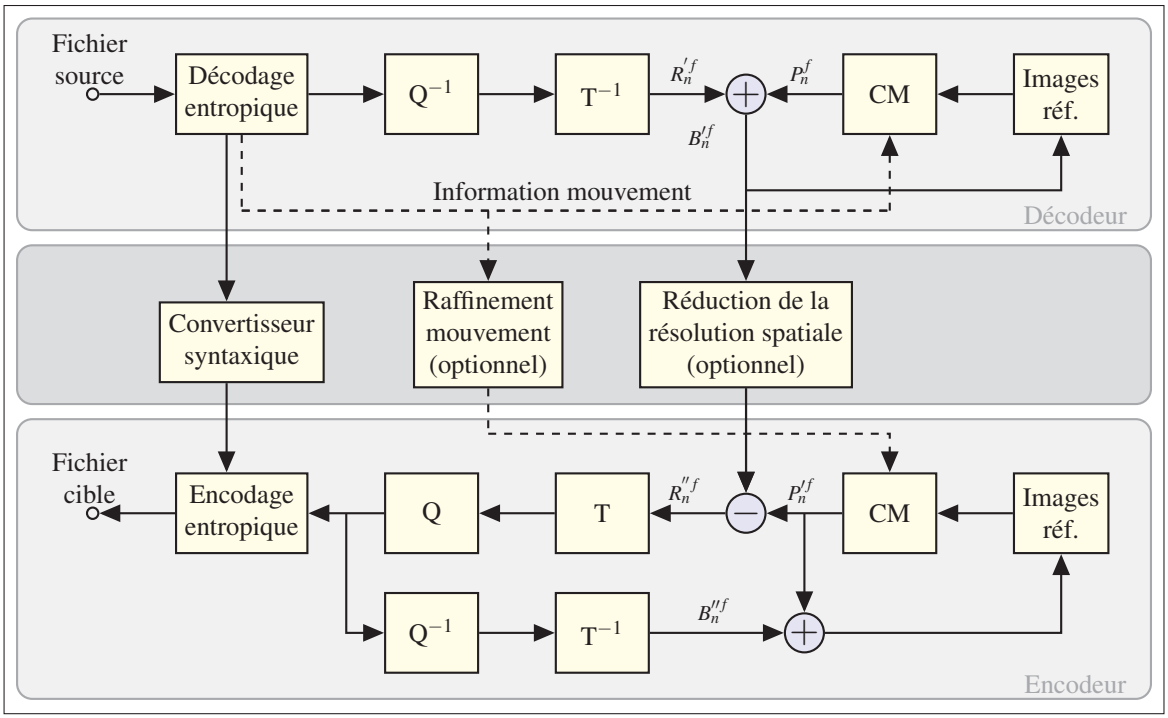


Figure 2.5 Architecture typique de transcodage hétérogène
Adaptée de Negi (2013, p.354)

La figure 2.5 illustre une architecture de transcodage hétérogène typique. Dans ce type d'architecture, le fichier source est entièrement décodé ; d'une part, pour récupérer les données reconstruites, comme c'est le cas pour du transcodage en cascade ; d'autre part, pour récupérer de l'information de codage, tel que les vecteurs de mouvement, les modes de prédiction, la structure de partitionnement et les données résiduelles. Comme nous le verrons plus loin, cette information est par la suite utilisée pour accélérer l'encodage de la séquence cible. Entre autres, plusieurs méthodes réutilisent cette information pour accélérer l'estimation de mouvement et pour réduire le nombre de modes à évaluer. Nous reviendrons sur ces concepts dans la prochaine section.

2.2 L'état de l'art sur le transcodage vidéo H.264 à HEVC

Dans cette section, nous présentons une revue de la littérature de deux sous-problèmes portant sur le transcodage H.264 à HEVC. Nous traitons d'abord des méthodes qui ont été développées

pour réduire le nombre de modes à évaluer durant le traitement d'une CTU, plus spécifiquement pour des trames inter. Par la suite, nous abordons les méthodes qui ont été développées pour accélérer l'estimation de mouvement de l'encodage HEVC. Dans les deux cas, nous verrons comment ces méthodes exploitent l'information de codage H.264 pour accélérer le transcodage.

2.2.1 Réduction des modes à évaluer

Comme nous l'avons vu précédemment, les encodeurs vidéos évaluent plusieurs modes de partitionnement et de prédiction afin de déterminer le mode à encoder pour la CTU traitée. C'est une étape cruciale de l'encodage puisque différentes stratégies peuvent être employées à ce moment. Ces stratégies doivent notamment déterminer : de quelle manière est parcourue la CTU traitée ; de quelle manière les modes à évaluer sont sélectionnés et évalués ; et, s'il y a lieu, quelles sont les conditions pour arrêter prématurément le traitement de la CTU.

Les stratégies choisies ont une incidence directe sur la complexité et l'efficacité du codage. Généralement, ces stratégies visent à réduire le nombre de modes à évaluer, en affectant le moins possible l'efficacité du codage. C'est une tâche complexe, puisque jusqu'à 85 CUs peuvent être évaluées durant le traitement d'une CTU. Et pour chaque CU traitée, jusqu'à 12 PUs inter et 5 PUs intra peuvent être évaluées.

À la section 1.4.4, nous avons décrit l'approche adoptée par l'encodeur HEVC de référence (appelée le HM) pour évaluer les modes. Cette approche privilégie l'efficacité du codage plutôt que la complexité. Elle a donc tendance à évaluer un très grand nombre de modes. D'autres approches moins complexes ont été proposées dans la littérature. Plusieurs de ces approches réduisent le nombre de modes évalués en arrêtant prématurément l'évaluation des modes lorsqu'une prédiction efficace a été trouvée (Gweon et Yung-Lyul (2012); Yoo et Suh (2013); Shen et Yu (2013)). Néanmoins, le nombre de modes éliminés par ces approches est souvent faible, particulièrement pour les séquences complexes, puisque ces méthodes ont seulement accès à l'information produite par l'encodeur HEVC.

Pour ce qui est des transcodeurs vidéos, ces derniers ont accès à l'information de codage de la séquence source pour réduire davantage le nombre de modes à évaluer. En ce qui touche plus spécifiquement au transcodage vidéo H.264 à HEVC, plusieurs méthodes ont été proposées dans la littérature. Ces méthodes exploitent couramment les modes, le résiduel ainsi que les vecteurs de mouvement H.264 pour accélérer l'encodage HEVC. Certaines de ces méthodes prennent leurs décisions au niveau de la CTU, mais la majorité d'entre elles les prennent plutôt au niveau de la CU pour avoir un meilleur contrôle sur les modes à tester.

2.2.1.1 Décisions au niveau des unités de codage arborescent

Dans la littérature, on retrouve quelques méthodes agissant au niveau des CTUs. Ces méthodes analysent la CTU traitée à l'aide de l'information de codage H.264. Puis, selon le résultat de l'analyse, la CTU se voit attribuer un ensemble de paramètres qui simplifient l'évaluation de ces modes, notamment en réduisant le nombre de modes à évaluer.

Dans leur approche, Jiang *et al.* (2013) classent chaque CTU en trois catégories de complexité : faible, moyenne et forte. La catégorie est déterminée en fonction du nombre de bits qui ont été utilisés pour encoder les macroblocs H.264 couvrant la région traitée. Plus une région H.264 contient de bits, plus elle est considérée comme complexe par les auteurs. Basée sur l'observation que les régions complexes nécessitent une structure de partitionnement plus profonde que les régions moins complexes, les auteurs déterminent par la suite les niveaux de profondeur (voir section 1.3.1.1) à évaluer pour chaque CTU. Ainsi, les CTUs classées comme de faible complexité évaluent les niveaux 0 à 2 ; ceux de complexité moyenne, les niveaux 0 à 3 ; et ceux de forte complexité, les niveaux 1 à 3.

Une approche similaire a été proposée par Luo *et al.* (2014) pour un transcodage AVS (standard chinois très similaire à H.264) vers HEVC. Ces auteurs classent les CTUs en trois régions d'intérêt (faible, moyen et élevé), selon les vecteurs de mouvement et les données résiduelles H.264 couvrant la région de la CTU traitée. Les régions contenant peu de données résiduelles et peu de variance dans le mouvement sont considérées comme de faible intérêt ; celles contenant

beaucoup de données résiduelles ou une variance du mouvement élevée, comme d'intérêt moyen ; et celles contenant à la fois beaucoup de données résiduelles et une variance élevée, comme d'intérêt élevé. Par la suite, les CTUs classées comme de faible intérêt évaluent les niveaux 0 à 1 ; ceux d'intérêt moyen, les niveaux 0 à 2 ; et ceux de fort intérêt, les niveaux 1 à 2. La figure 2.6 montre un exemple de classification obtenue par cette approche.

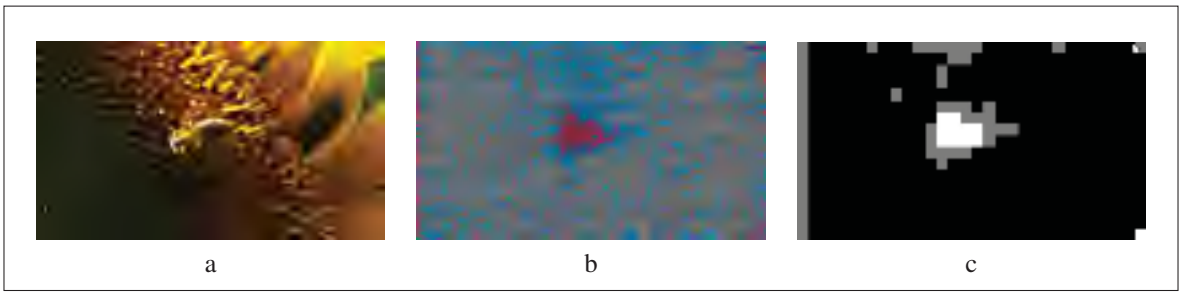


Figure 2.6 Exemples de classification de la complexité des régions de la séquence source : a) séquence à transcoder b) information de codage : les blocs rouges représentent les régions intra, les gris, les régions skip et les bleus, les régions inter. c) régions détectées : les blocs blancs représentent les régions d'intérêt élevé, les gris, d'intérêt moyen et les noirs, d'intérêt faible
Tirée de Luo *et al.* (2014, p. 3) © 2014 IEEE

Les deux méthodes précédentes supposent que la complexité des régions H.264 est corrélée avec la structure de partitionnement à employer dans HEVC, ce qui est généralement vrai. Malheureusement, l'analyse est effectuée à un niveau beaucoup trop élevé pour être en mesure de réduire significativement le nombre de blocs à évaluer. C'est pourquoi ces méthodes sont souvent utilisées conjointement avec d'autres méthodes.

2.2.1.2 Décisions au niveau des unités de codage

Pour prendre des décisions plus précises, il est possible d'analyser les CUs qui sont visitées durant le parcours d'une CTU. Dans la littérature, on retrouve plusieurs méthodes conçues pour un parcours préfixe de la CTU (voir section 1.4.4). Ces méthodes analysent la CU traitée afin de déterminer les PUs à évaluer, en plus de décider si les enfants de la CU doivent être visités (si le mode split doit être activé).

2.2.1.2.1 Méthodes basées sur les modes H.264

Plusieurs auteurs définissent des règles de correspondance (*mapping*) pour déterminer les PUs HEVC à évaluer selon les modes H.264 qui ont été encodés dans la même région. La majorité de ces règles sont simples et intuitives pour les spécialistes du domaine. Elles sont basées sur l'idée que H.264 et HEVC utilisent généralement des modes de partitionnement et de prédiction similaires pour encoder une même région. Autrement dit, un modèle de prédiction qui est efficace dans H.264 le sera probablement aussi dans HEVC. Ainsi, une région qui a été encodée avec un partitionnement fin dans H.264, le sera généralement aussi dans HEVC. De même, une région qui a été encodée avec une méthode de prédiction temporelle (inter) dans H.264, le sera probablement aussi dans HEVC.

L'usage de ce type de règles trouve ses origines dans d'autres cas de transcodage, notamment le transcodage MPEG-4 à H.264 (Lee *et al.* (2006)). Pour ce dernier cas, les résultats obtenus sont intéressants puisque les standards impliqués supportent des modèles de prédiction similaires et que le nombre de modes à évaluer est petit. Pour du transcodage H.264 à HEVC, il est plus difficile d'établir des règles de correspondance efficaces puisque HEVC supporte beaucoup plus de modes de partitionnement et de prédiction que H.264, en plus de supporter des partitions plus grandes. On retrouve tout de même plusieurs règles efficaces pour des CUs 16×16 et 8×8, puisque H.264 supporte des structures de partitionnement et de prédiction similaires à celles de HEVC. À titre d'exemple, le tableau 2.1 montre les règles proposées par Peixoto *et al.* (2014b) pour déterminer les PUs HEVC à tester pour une CU 16×16, selon le mode du macrobloc H.264 couvrant la même région. On remarque que ces règles réduisent significativement le nombre de modes à tester, sauf lorsque le macrobloc H.264 est de mode P8x8 ou intra. Les auteurs justifient ce choix par le fait qu'il est difficile de prédire la meilleure PU pour ces cas. Nos expériences corroborent cette affirmation.

Pour des CUs 64×64 et 32×32, il est plus difficile d'établir des règles efficaces basées exclusivement sur les modes H.264, puisque ce dernier format ne supporte pas des partitions aussi grandes. Inspirés par les travaux de recherche sur le transcodage visant à réduire la

Tableau 2.1 Exemples de règles pour déterminer les unités de prédiction (PUs) à tester, pour une unité de codage (CU) de 16×16 pixels, selon le type du macrobloc
Adapté de Peixoto *et al.* (2014, p.103) © 2014 IEEE

		Type du macrobloc H.264					
		Skip	P16×16	P16×8	P8×16	P8×8	Intra
PUs HEVC testées	Skip/Merge	✓	✓	✓	✓	✓	✓
	P2N×2N		✓	✓	✓	✓	✓
	P2N×N			✓		✓	✓
	PN×N				✓	✓	✓
	P2N×nU			✓		✓	✓
	P2N×nD			✓		✓	✓
	PnR×2N				✓	✓	✓
	PnL×2N				✓	✓	✓
	Intra						✓
	PSkip					✓	✓

résolution spatiale, Shen *et al.* (2013) ont développé les 4 règles suivantes pour déterminer les PUs à tester pour une CU 32×32 :

- Le mode P2N×2N est activé lorsqu'il a au moins deux macroblocs de mode P16×16.
- Le mode PN×2N est activé lorsqu'il les deux macroblocs de gauche, ou de droite, sont de mode P16×16.
- Le mode P2N×N est activé lorsque les deux macroblocs du haut, ou du bas, sont de mode P16×16.
- L'intra (le mode I2N×2N) est activé lorsqu'il au moins un macrobloc de mode intra.

En plus de ces règles, les auteurs testent toujours les modes merge/skip puisque ces modes sont peu complexes à tester et peuvent avoir un impact significatif sur l'efficacité de codage. Inversement, les auteurs désactivent les modes asymétriques, puisqu'ils sont complexes à évaluer et apportent des gains d'efficacité de codage négligeable (sous les 1% selon Kim *et al.* (2012)). Malheureusement, l'approche complète de transcodage (qui inclut d'autres méthodes) proposée par les auteurs réduit significativement l'efficacité de codage. Il est donc difficile de juger de l'efficacité de ces règles. De leur côté, Chen *et al.* (2013) ont proposé des règles

qui tiennent compte notamment du nombre de macroblocs de mode skip et de mode P8×8. Cependant, ces règles sont très conservatrices puisque le nombre de modes testés reste élevé.

2.2.1.2.2 Méthodes considérant le résiduel H.264

Afin de mieux déterminer les modes HEVC à tester, d'autres auteurs considèrent le résiduel, en plus des modes H.264. Comme nous l'avons mentionné antérieurement, le résiduel correspond à l'erreur de prédiction. Ainsi, un résiduel élevé signifie que la prédiction qui lui est associée est peu efficace (peu précise). Pour tenter de réduire cette erreur, le transcodeur a intérêt à tester davantage de modes, en particulier des plus petits blocs, puisqu'ils sont reconnus pour diminuer l'erreur de prédiction. Inversement, une prédiction efficace produira un résiduel faible, voire nul. Dans un tel cas, le transcodeur a souvent intérêt à tester des partitions plus grossières afin de simplifier le modèle de prédiction (réduire le nombre de bits nécessaires à la signalisation des paramètres de prédiction).

Conformément à ce qui précède, Chen *et al.* (2013) calculent l'EQM de la trame H.264. Ensuite, ils activent le mode split pour une CU 16×16 lorsque le ratio entre l'EQM du macrobloc H.264 et l'EQM de la trame H.264 est supérieur à 1.5. Autrement dit, les auteurs activent le mode split lorsque le résiduel H.264 de la CU est relativement élevé. En plus de cette règle, les auteurs activent aussi le mode split lorsque le macrobloc H.264 est de mode P8×8.

Pour leur part, Fang *et al.* (2014) proposent plusieurs euristiques qui désactivent des modes lorsque le résiduel H.264 est nul. En plus de ces règles, les auteurs proposent des règles pour déterminer les modes asymétriques à tester selon où se concentre le résiduel dans H.264.

2.2.1.2.3 Méthodes basées sur l'analyse du mouvement H.264

En plus des modes et du résiduel, d'autres chercheurs exploitent aussi les vecteurs de mouvement H.264 pour réduire le nombre de modes à évaluer. Ainsi, Jiang et Chen (2013) proposent de segmenter en deux régions le mouvement H.264 de la CU traitée en adaptant un

algorithme appelé Simple Linear Iteractive Clustering (SLIC) (Achanta *et al.* (2012)), utilisé originellement pour segmenter une image en zones uniformes. Par la suite, les auteurs utilisent les formes des deux régions pour classer la CTU dans l'une des trois catégories suivantes : 1) région uniforme 2) région à dominance verticale et 3) région à dominance horizontale. Finalement, les PUs inter à tester sont déterminées en fonction de la catégorie selon les règles définies par le tableau 2.2).

Tableau 2.2 Règles pour déterminer les modes HEVC à tester selon la segmentation du mouvement
Adapté de Jiang et Chen (2013, p.1224)

Résultat clustering	**Caractéristiques de la région**	**Mode split évalué**	**PUs évalués**
	Uniforme	Non	P2N×2N P2N×N PN×2N
	Dominance verticale	Oui	P2N×2N PN×2N PnL×2N PnR×2N
	Dominance horizontale	Oui	P2N×2N P2N×N P2N×nU P2N×nD

De leur côté, Peixoto *et al.* (2014a) utilisent un classifieur linéaire pour déterminer si une CU doit être divisée (via le mode split). Le classifieur proposé prend ses décisions selon les caractéristiques suivantes : (i) la variance des distances des vecteurs de mouvement ; (ii) la variance des angles des vecteurs de mouvement ; (iii) le nombre de coefficients DCT et (iv) la distribution des modes H.264.

Pour leur part, Xing *et al.* (2013) ont développé une méthode de classification spécifique à la vidéosurveillance. Cette méthode modélise l'arrière-plan afin de classer chaque CU en trois

classes : arrière-plan, avant-plan et hybride. Ces classes sont par la suite utilisées non seulement pour déterminer les modes à évaluer, mais aussi pour déterminer les paramètres de l'estimation de mouvement. Par exemple, une plus petite zone de recherche est utilisée pour les CU classées comme appartenant à l'arrière-plan.

2.2.1.2.4 Méthodes basées sur les données HEVC

En plus de l'information H.264, plusieurs méthodes considèrent aussi l'information de codage HEVC afin de réduire davantage le nombre de modes évalués. Cette information peut inclure les couts et la distorsion des modes qui ont été jusqu'à maintenant testés par la CTU traitée. Elle peut aussi inclure des données sur les CTUs voisines qui ont été traitées, ou encore des données sur les trames HEVC précédentes, puisque les modes sont spatialement et temporellement corrélés comme l'ont observé Shen *et al.* (2013).

Tableau 2.3 Distribution de la profondeur des CUs
Adapté de Shen *et al.* (2013, p.244) © 2014 IEEE

Séquence	QP	Distribution profondeur (%)				Séquence	QP	Distribution profondeur (%)			
		0	1	2	3			0	1	2	3
BasketballDrill 832×480	22	21.85	32.92	26.88	18.35	BQSquare 416×240	22	0.39	17.14	50.93	31.51
	27	32.24	32.69	23.48	11.59		27	6.46	38.86	35.96	18.72
	32	40.05	34.57	19.08	6.30		32	20.50	46.61	21.07	11.82
	37	43.84	35.48	14.40	3.25		37	31.43	46.11	16.11	6.35
ChinaSpeed 1024×768	22	17.14	33.32	28.29	21.25	Chromakey 720×480	22	15.75	46.76	30.30	7.19
	27	27.05	35.35	23.34	14.26		27	31.80	45.30	20.12	2.78
	32	38.23	33.98	18.59	9.20		32	48.49	35.94	14.27	1.30
	37	51.65	31.08	12.33	4.94		37	59.46	28.45	11.68	0.41
Kimono 1920×1080	22	15.97	59.41	22.13	2.49	Traffic 2560×1600	22	25.16	32.21	29.98	12.65
	27	26.97	55.69	16.09	1.25		27	44.81	30.13	18.95	6.11
	32	39.05	49.23	11.10	0.62		32	58.82	26.63	11.73	2.82
	37	56.28	40.33	6.15	0.24		37	70.10	22.07	6.73	1.10

Ainsi, Shen *et al.* (2013) ont développé deux conditions pour arrêter la division (l'activation du mode split) d'une CU. La première condition arrête la division lorsque le meilleur cout J_{rd} de la CU testée est plus grand que $0 < \alpha < 1$ fois le cout J_{rd} de la CU parent. La valeur de α est déterminée empiriquement en fonction de la résolution de la séquence vidéo. Pour sa part, la deuxième condition tient compte notamment de la résolution de la séquence vidéo, du QP et des modes H.264 couvrant la CTU. Les auteurs ont observé que l'augmentation de la résolution et

du QP a pour effet de réduire le nombre de petites partitions utilisées pour l'encodage comme le montre le tableau 2.3.

De leur côté, Peixoto *et al.* (2014a) ont proposé une méthode pour arrêter prématurément le traitement d'une CU 32×32 ou 16×16 lorsqu'un mode a obtenu un cout J_{rd} satisfaisant. Plus spécifiquement, cette méthode calcule d'abord le cout du meilleur mode merge/skip, noté C_{SKIP}. Si ce cout est inférieur à un seuil T_{SKIP}, le traitement de la CU est arrêté. Sinon, le traitement se poursuit en appliquant des règles similaires aux modes inter et intra. Ainsi, la méthode calcule le meilleur cout inter, noté C_{Inter}. Si ce cout est inférieur à un seuil T_{Inter}, le traitement de la CU est arrêté. Sinon, le meilleur cout intra C_{Intra} est calculé. Si ce cout est inférieur au seuil C_{Intra}, le traitement est arrêté. Dans le cas contraire, le mode split est activé est les sous-CUs sont visitées. Les auteurs déterminent préalablement les seuils en modélisant le cout et le type des meilleurs modes obtenus pour un transcodage complet selon un QP donné.

2.2.2 Accélération de l'estimation du mouvement

L'accélération de l'estimation de mouvement constitue un autre problème de transcodage abondamment couvert par la littérature. Pour la majorité des encodeurs vidéos, l'estimation du mouvement est l'étape qui consomme le plus grand pourcentage du temps d'exécution. Par exemple, Bossen *et al.* (2012) ont mesuré le temps d'exécution passé dans les principales classes de l'encodeur HEVC de référence [2] pour différents scénarios de codage. Dans le cas d'un encodage à accès aléatoire [3], leurs tests montrent qu'environ 40% du temps d'exécution est consommé par la classe TComRdCost. Les auteurs estiment que l'essentiel de ce temps est utilisé par l'estimation du mouvement pour calculer la distorsion des positions testées (via la SAD, pour les pixels entiers, et la SATD, pour les pixels fractionnaires). De plus, ils estiment à environ 20% le temps d'exécution consommé par l'interpolation des pixels fractionnaires. Encore une fois, cette tâche est principalement utilisée par l'estimation du mouvement.

2. Le HM version 8.0.
3. En utilisant l'algorithme de recherche TZS (voir section 1.4.3.1).

Comme nous l'avons vu à la section 1.1.1.2, les algorithmes de recherche employés par les encodeurs vidéos conventionnels utilisent généralement une grande zone de recherche pour estimer le mouvement de la région traitée. Puisque le mouvement est alors inconnu, cette large zone de recherche permet à ces algorithmes de détecter et de signaler une discontinuité dans le mouvement. Dans un contexte de transcodage vidéo H.264 à HEVC, la situation est différente, car le transcodeur peut accéder aux vecteurs de mouvement de la séquence H.264. Même si cette représentation du mouvement constitue un bon point de départ, elle doit être adaptée et raffinée durant l'encodage HEVC, notamment pour tenir compte des différences entre les deux standards. On parle alors de raffinement du mouvement, plutôt que d'estimation de mouvement.

Les méthodes de raffinement proposées dans la littérature sont normalement constituées de deux paramètres de recherche : le vecteur de mouvement de départ et la plage de raffinement, utilisée pour définir la zone de raffinement. Le vecteur de mouvement de départ correspond habituellement à un vecteur de mouvement H.264 ou à un vecteur AMVP de HEVC. On distingue deux catégories de méthodes pour déterminer la zone de raffinement : les méthodes statiques, qui définissent une plage de recherche constante pour l'ensemble des PUs traitées ; et les méthodes dynamiques, qui adaptent la plage de recherche dynamiquement. Enfin, les méthodes présentées effectuent toutes au moins un raffinement au demi pixel et la plupart, un raffinement au quart de pixel.

2.2.2.1 Méthodes de raffinement avec une zone de recherche dynamique

La méthode de raffinement proposée par Peixoto *et al.* (2014a) sélectionne le meilleur vecteur de mouvement H.264 couvrant la PU traitée comme point de départ. Ce vecteur de mouvement est par la suite directement raffiné au quart de pixel. Les auteurs annoncent un transcodage de 1.40 à 1.77 fois plus rapide qu'un transcodage en cascade. Cependant ils obtiennent une augmentation du BD-Rate pouvant aller jusqu'à 7.71%. Ces résultats suggèrent donc que la zone de raffinement est trop restreinte, ou que les vecteurs H.264 utilisés par la méthode sont insuffisants.

De leur côté, Shen *et al.* (2013) proposent d'utiliser la médiane des vecteurs du AMVP et de H.264 couvrant la PU traitée comme point de départ. Par la suite, ce vecteur de mouvement est raffiné selon une plage de raffinement fixée à 4 pixels. Malheureusement, les auteurs ne fournissent aucun résultat sur les performances de ces méthodes (ils fournissent des résultats globaux pour leur approche de transcodage complète).

2.2.2.2 Méthodes de raffinement avec une zone de recherche dynamique

Zong-Yi *et al.* (2013) définissent le vecteur de mouvement de départ comme la médiane des vecteurs H.264 couvrant la PU traitée, mais seulement pour les PUs plus grandes que 16×16 pixels. Pour les autres PUs, les auteurs utilisent directement le vecteur de mouvement H.264 le plus près du centre de la PUs traitée. Les auteurs utilisent une plage de raffinement égale à la distance entre le vecteur de mouvement de départ et le meilleur vecteur de mouvement AMVP. Cependant, afin de réduire les mauvais résultats, les auteurs fixent la plage de recherche minimale à 2 pixels, une valeur déterminée statiquement. Les auteurs annoncent une réduction du temps de transcodage moyen de 15.19% (une accélération d'environ 1.18) pour un impact négligeable sur l'efficacité de codage

Pour leur part, Fang *et al.* (2014) définissent le meilleur vecteur AMVP comme point de départ. La plage de raffinement est adaptée en fonction de la plus grande distance entre ce point de départ et tous les vecteurs de mouvement H.264 couvrant la région traitée. Par la suite, les auteurs étendent ou réduisent cette plage de recherche en fonction de la probabilité que le mode traité soit le meilleur. Les auteurs annoncent une réduction du temps de transcodage global d'environ 15 % (une accélération d'environ 1.18) pour une perte d'efficacité de codage négligeable.

2.3 Résumé

Dans ce chapitre, nous avons dans un premier lieu abordé le domaine du transcodage vidéo de manière générale. Plus spécifiquement, nous avons discuté des motivations du transcodage,

des opérations de transcodage les plus courantes et de quelques architectures de transcodage populaires, dont l'architecture typiquement employée pour du transcodage hétérogène. Dans un second temps, nous avons présenté une revue de la littérature portant spéciquement sur le transcodage vidéo H.264 à HEVC. Nous avons divisé cette revue en deux sections, l'une traitant du problème de l'accélération de l'estimation de mouvement et l'autre portant sur la réduction du nombre de modes HEVC à tester durant le traitement d'une CTU. Dans les deux prochains chapitres, nous présentons les solutions que nous avons développées pour résoudre ces deux problèmes importants.

CHAPITRE 3

TRANSCODAGE H.264 À HEVC BASÉ SUR LA PROPAGATION DU MOUVEMENT

Dans ce chapitre, nous proposons une approche de propagation du mouvement appliquée à un contexte de transcodage H.264 à HEVC. Comme nous le démontrons plus loin, cette approche constitue une excellente alternative aux méthodes de raffinement de mouvement qui sont couramment utilisées dans un contexte de transcodage. L'approche proposée repose sur l'idée que l'information du mouvement existante est suffisamment précise pour être réutilisée durant l'encodage HEVC sans devoir appliquer un algorithme de raffinement du mouvement. En contrepartie, cette information se propage de manière différente dans la séquence HEVC, d'où le nom de notre approche. Ainsi, nous verrons que le meilleur vecteur de mouvement pour la région traitée dans HEVC est généralement localisé dans la région H.264 correspondante, comme le supposent plusieurs auteurs. Cependant, nous montrons aussi qu'un meilleur vecteur de mouvement peut parfois être situé à l'extérieur de cette région, par exemple, dans le voisinage de la région H.264 correspondante, ou encore, dans des régions HEVC qui ont été précédemment encodées.

Le reste de ce chapitre est structuré de la manière suivante. D'abord, la section 3.1 fait un retour sur les méthodes de raffinement du mouvement et décrit les problèmes de ce type de méthode lorsqu'elles sont appliquées dans un contexte de transcodage H.264 à HEVC. Par la suite, la section 3.2 présente l'approche de propagation du mouvement proposée, puis la section 3.3 analyse de quelle manière le mouvement existant se propage dans la séquence HEVC lorsque notre approche est utilisée. Enfin, la section 3.4 décrit les expériences qui ont été menées pour évaluer les performances de notre approche et présente les résultats expérimentaux.

3.1 Problèmes du raffinement du mouvement

Comme nous l'avons vu au chapitre précédent, le raffinement du mouvement consiste à réutiliser les vecteurs de mouvement d'une séquence source (à transcoder) afin de réduire la complexité de l'estimation de mouvement utilisée pour déterminer les vecteurs de mouvement

de la séquence cible (transcodée). Par exemple, pour du transcodage H.264 à HEVC, il consiste généralement à déterminer d'abord le vecteur de mouvement initial, souvent un vecteur de mouvement H.264, puis à raffiner ce vecteur en effectuant une recherche sur une zone plus ou moins petite (par exemple, sur une fenêtre de recherche de 9×9 pixels) ; c'est selon l'algorithme de raffinement et le contexte.

Dans plusieurs cas de transcodage, le raffinement est nécessaire pour atteindre une bonne efficacité de codage, idéalement une efficacité similaire à celle obtenue en employant un transcodeur en cascade. Ainsi, lorsque le transcodage a pour but de réduire la résolution temporelle, le raffinement permet de corriger et d'affiner le mouvement qui a été reconstruit, par exemple, par interpolation bilinéaire (voir section 2.1.2.3). De même, le raffinement permet de corriger des erreurs de sous-échantillonnage et de redimensionnement des vecteurs de mouvement lorsque le transcodage a pour but de réduire la résolution spatiale (voir section 2.1.2.2).

En ce qui concerne le transcodage hétérogène, le raffinement permet d'améliorer la représentation du mouvement lorsque le format cible supporte des outils de prédiction du mouvement plus précis. Par exemple, pour du transcodage H.263 à H.264, le raffinement permet notamment de passer d'une représentation du mouvement précise au demi-pixel à une représentation précise au quart de pixel (Bialkowski *et al.* (2006)). Il permet aussi d'améliorer la résolution du mouvement (la densité du champ de vecteurs) en exploitant des partitions plus petites que celles supportées par H.263.

Pour du transcodage H.264 à HEVC, le raffinement du mouvement semble moins intéressant puisque les deux formats supportent des vecteurs de mouvement précis au quart de pixel et les plus petites partitions inter supportées sont passées de 4×4 pixels pour H.264 à 4×8 et 8×4 pixels pour HEVC. Cependant, d'autres facteurs peuvent motiver l'utilisation d'un algorithme de raffinement pour ce cas de transcodage spécifique. Par exemple, Zong-Yi *et al.* (2013) ont constaté que le raffinement du mouvement améliore l'efficacité de codage lorsque le vecteur

de mouvement H.264 sélectionné et le meilleur vecteur de mouvement prédit par HEVC (le meilleur $\mathbf{v}_p$) sont différents.

Selon les limites de nos connaissances, aucun auteur n'a cependant démontré si les algorithmes de raffinement sont efficaces parce qu'ils permettent de trouver des nouveaux vecteurs de mouvement (comme c'est le cas pour plusieurs autres cas de transcodage) ou, au contraire, s'ils le sont parce qu'ils permettent de propager, indirectement, l'information de mouvement d'autres régions, par exemple, les vecteurs de mouvement H.264 situés dans le voisinage de la région traitée, comme nous le démontrons plus loin.

Malgré cette absence d'étude, l'emploi d'un algorithme de raffinement pour du transcodage H.264 vers HEVC est tout de même recommandé. En effet, comparativement à un algorithme de recherche conventionnel (avec une grande zone de recherche), plusieurs algorithmes de raffinement du mouvement obtiennent une efficacité de codage similaire, tout en réduisant significativement la complexité des calculs (en utilisant une fenêtre de recherche plus petite). Cependant, les algorithmes de raffinement présentés dans la littérature affectent seulement les performances de la recherche du meilleur vecteur de mouvement effectuée pour des pixels situés à des positions entières. Ces algorithmes ne modifient donc pas la recherche effectuée à des positions fractionnaires. Comme nous l'avons mentionné précédemment (voir section 2.2.2), cette dernière étape est très complexe, notamment parce qu'elle doit interpoler les pixels de référence. En plus de ce problème, les algorithmes de raffinement ont pour inconvénient d'effectuer des calculs redondants, comme nous l'expliquerons plus loin.

3.2 Approche de propagation du mouvement proposée

Contrairement aux approches de raffinement présentées dans la littérature, l'approche de propagation de mouvement que nous proposons suppose que l'information de mouvement existante est suffisamment précise pour être réutilisée durant l'encodage de la séquence HEVC. Cependant, cette information ne se propage pas de la même manière durant l'encodage de la séquence HEVC. Par exemple, le meilleur vecteur de mouvement pour la région HEVC peut

être absent de la région H.264 correspondante, mais être présent dans le voisinage de cette dernière région, comme nous le verrons plus loin.

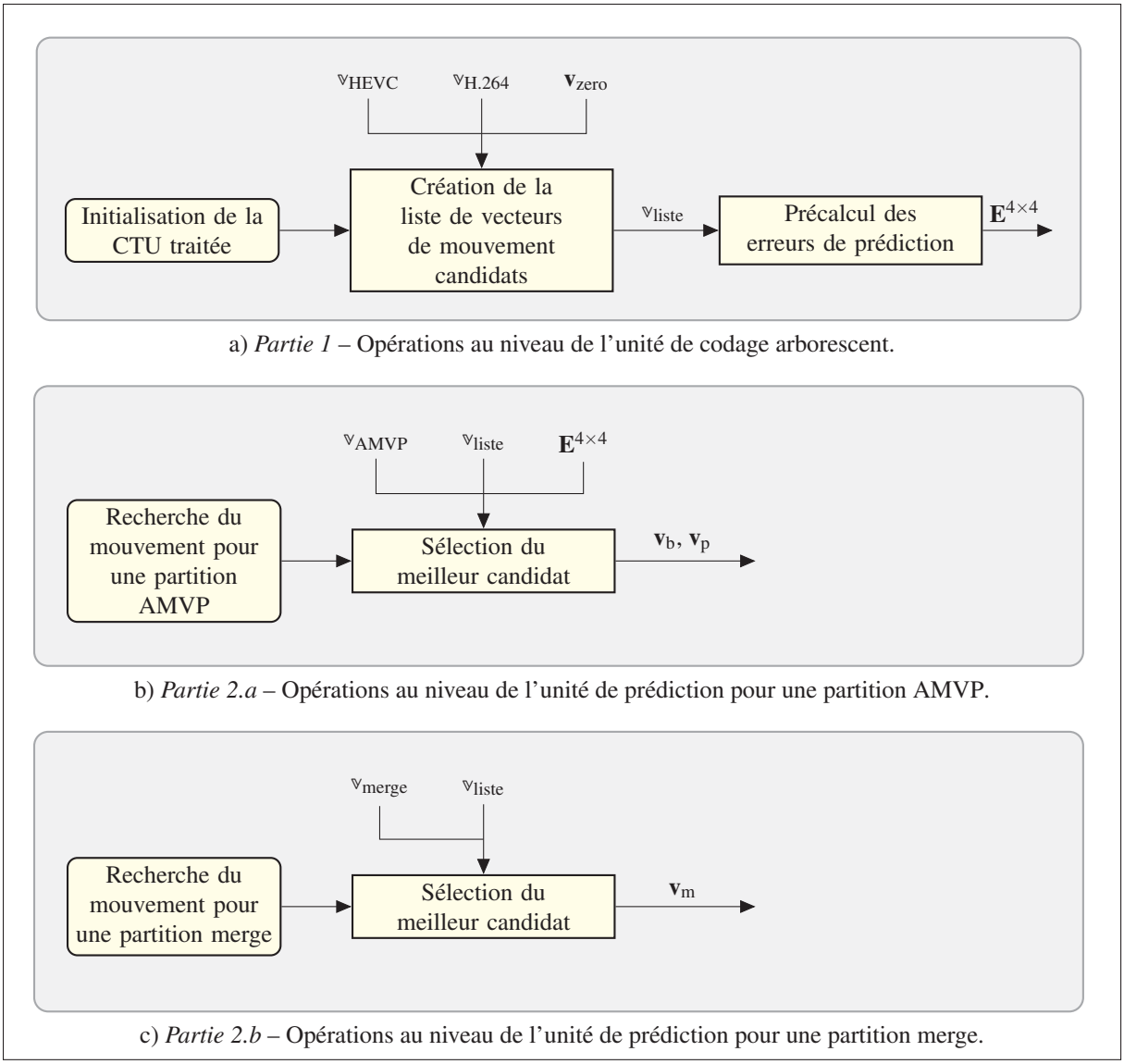
Afin de tenir compte de cette notion de propagation, l'algorithme proposé crée une liste des vecteurs de mouvement candidats composée de vecteurs de mouvement H.264 et HEVC qui ont une probabilité élevée de se propager dans la région HEVC traitée. Par la suite, l'algorithme sélectionne, parmi ces candidats, le meilleur vecteur de mouvement pour chacune des partitions inter testées. Les prochaines sous-sections décrivent plus en détail les différents aspects de cette approche.

3.2.1 Vue sommaire de l'approche proposée

Comme l'illustre la figure 3.1, l'approche proposée est constituée de deux parties opérant à des niveaux et des moments différents. La *Partie 1* est appelée à l'initialisation de la CTU traitée. Elle a pour rôle de générer les données qui seront exploitées par la seconde partie. La *Partie 2* opère au niveau des PUs inter. Elle se divise en deux sous-parties, la *Partie 2.a* et la *Partie 2.b*, qui sont appelées pour déterminer le meilleur vecteur de mouvement de chacune des partitions testées durant le traitement des modes, et cela, respectivement pour les modes AMVP et merge.

Comme nous le décrivons plus loin, la *Partie 1* crée deux types d'information : une liste des vecteurs de mouvement candidats, notée $\forall_{\text{liste}}$; et les erreurs de prédiction précalculées, représentée par la matrice $\mathbf{E}^{4\times4}$, associées à ces candidats. La liste est constituée du vecteur nul, noté $\mathbf{v}_{\text{zero}}$, et de vecteurs de mouvement H.264 et HEVC ayant une probabilité élevée d'être sélectionnés par au moins une des partitions qui sera testée plus tard durant le traitement des PUs. Il est important de noter que les partitions AMVP évaluent tous les candidats de la liste, et uniquement ces candidats. Étant donné que ces candidats sont connus à l'initialisation de la CTU, les erreurs de prédiction sont précalculées au même moment afin d'éliminer des calculs répétitifs, comme nous le verrons à la section 3.2.3.

Le traitement effectué par la *Partie 2* dépend du mode. Pour le mode AMVP, chaque partition testée évalue le cout des vecteurs de mouvement candidats de $\forall_{\text{liste}}$ en exploitant les erreurs



a) *Partie 1* – Opérations au niveau de l'unité de codage arborescent.

b) *Partie 2.a* – Opérations au niveau de l'unité de prédiction pour une partition AMVP.

c) *Partie 2.b* – Opérations au niveau de l'unité de prédiction pour une partition merge.

Figure 3.1 Diagramme de blocs de l'approche de propagation du mouvement proposée

de prédiction précalculées $\mathbf{E}^{4\times4}$ et les vecteurs de mouvement prédits ∇_{AMVP}. La méthode retourne la combinaison du vecteur de mouvement HEVC prédit, noté $\mathbf{v}_p$, et du vecteur de mouvement, noté $\mathbf{v}_b$, qui minimise le cout J_{motion}. Pour le mode merge, les couts des candidats merge, notés ∇_{merge}, sont calculés en exploitant les erreurs de prédiction précalculées $\mathbf{E}^{4\times4}$. Le reste du traitement de ces modes est inchangé par rapport à l'encodeur HEVC de référence.

3.2.2 Création de la liste des vecteurs de mouvement candidats

La figure 3.2 montre les régions d'où sont extraits les vecteurs de mouvement H.264 et HEVC pour former la liste des vecteurs de mouvement candidats, notée $\mathbb{v}_{\text{liste}}$. Sur cette figure, les cellules des grilles représentent les vecteurs de mouvement qui sont stockés dans l'implémentation de notre approche de propagation de mouvement. Pour la trame H.264 et la trame HEVC courante, chaque cellule représente une région de 4×4 pixels et est associée à un vecteur de mouvement. L'utilisation de ces grilles a pour avantage de représenter le mouvement de manière uniforme sans tenir compte de la structure de partitionnement encodée, respectivement par la trame H.264 et la trame courante HEVC. De plus, cette représentation n'entraine aucune perte d'information sur le mouvement, puisque le plus petit bloc supporté par H.264 couvre une région de 4×4 pixels, et celui de HEVC, une région de 4×8 ou 8×4 pixels. Pour la trame HEVC de référence, chaque cellule représente une région de 16×16 pixels, puisque HEVC sous-échantillonne les vecteurs de mouvement situés dans les trames de référence, de manière à conserver un seul vecteur par bloc 16×16, afin de réduire l'espace mémoire nécessaire au stockage de l'information de mouvement.

Les vecteurs de mouvement H.264, notés $\mathbb{v}_{\text{H.264}}$, sont extraits de la trame H.264 n. La région d'extraction de base est la même que la région de la CTU traitée. Cette région de base peut cependant être étendue dans son voisinage pour extraire davantage de vecteurs de mouvement H.264. C'est le paramètre α qui définit l'étendue de ce voisinage. Pour être conforme avec notre représentation de vecteurs de mouvement H.264, l'unité de mesure de ce paramètre est le bloc 4×4. À noter que seuls les vecteurs H.264 situés dans la région d'extraction de base sont extraits lorsque $\alpha = 0$.

Les vecteurs de mouvement HEVC, notés $\mathbb{v}_{\text{HEVC}}$, sont extraits depuis la trame courante et la trame HEVC de référence. Dans la trame courante, la région d'extraction correspond aux blocs situés dans le voisinage de la CTU traitée et qui ont déjà été encodés. À noter que certains de ces vecteurs peuvent avoir été extraits précédemment de la région d'extraction H.264, étant donné la corrélation entre les vecteurs H.264 et HEVC. Dans la trame de référence, la zone

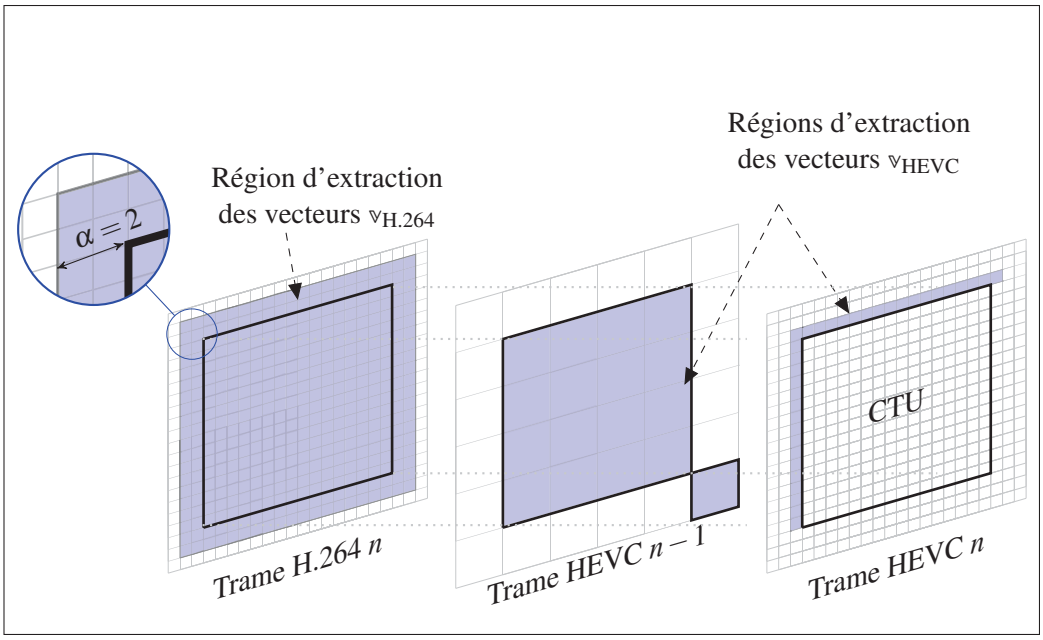


Figure 3.2 Extraction des vecteurs de mouvement candidats

d'extraction couvre la même région que la CTU traitée. Elle inclut aussi une autre région située en bas à droite de cette dernière, comme le montre la figure. Cette région supplémentaire correspond à un bloc qui sert parfois de candidat temporel au mode merge et AMVP (voir section 1.3.2.1.2).

En plus des vecteurs de mouvement H.264 et HEVC extraits des régions illustrées plus haut, la liste des vecteurs de mouvement candidats contient aussi le vecteur nul, noté $\mathbf{v}_{zero}$. C'est un vecteur couramment présent dans les régions sans mouvement. Enfin, les doublons sont enlevés de la liste et le nombre total de candidats est noté K.

3.2.3 Élimination de la redondance des calculs

Comme mentionné précédemment, le mouvement HEVC peut être représenté sous la forme d'une grille de blocs de 4×4 pixels, où chaque bloc est associé à un vecteur de mouvement. Dans une CTU, jusqu'à 24 modes AMVP peuvent couvrir le même bloc 4×4 [1]. Ce nombre

1. Jusqu'à 3 modes symétriques par niveau de profondeur pour les niveaux 0 à 3 ; et jusqu'à 4 modes asymétriques par niveau de profondeur pour les niveaux 0 à 2 ; pour un total de 24 modes.

augmente jusqu'à 28 lorsque les modes merge sont aussi considérés. Ainsi, pour un bloc de 4×4 pixels, l'erreur de prédiction (SATD) peut être recalculée jusqu'à 28 fois pour un même vecteur de mouvement, mais pour des partitions de tailles différentes. Puisque le résultat est toujours le même, pour chaque calcul de la contribution de ces blocs, on parle alors de redondance des calculs. Cette redondance peut avoir un impact significatif sur la complexité d'un encodeur HEVC puisqu'elle nécessite la répétition de calculs complexes, tels que l'interpolation des pixels à des positions fractionnaires (l'application de filtres) et le calcul de la SATD. Le degré de redondance exacte d'un encodage vidéo HEVC dépend de plusieurs facteurs, dont l'activité de mouvement dans la CTU traitée, les stratégies implémentées pour éliminer des partitions inter à tester, et l'approche d'estimation de mouvement implémentée (recherche du mouvement, raffinement du mouvement ou propagation du mouvement).

Lorsqu'un algorithme de recherche du mouvement (comme celui présenté à la section 1.4.3.1), ou un algorithme de raffinement du mouvement est employé, la redondance des calculs est difficile à éliminer puisque les vecteurs de mouvement à tester sont généralement déterminés au niveau des PUs durant le traitement d'une partition. Ainsi, les vecteurs de mouvement testés peuvent varier d'une partition à une autre et sont déterminés de manière dynamique, au fur et à mesure que les partitions de la CTU sont traitées. Certaines approches parallèles ont été proposées dans la littérature de la compression vidéo pour déterminer les vecteurs de mouvement à l'initialisation de la CTU (ou du macrobloc, pour H.264) de manière à éliminer cette redondance (Chen et Hang (2008); Gao et Zhou (2014)). Cependant, ces approches réduisent l'efficacité de codage. De plus, elles sont basées sur une recherche du mouvement exhaustive. Elles sont donc beaucoup trop complexes pour être utilisées par des applications séquentielles.

En revanche, notre approche de propagation du mouvement peut facilement éliminer cette redondance puisque les partitions de la CTU évaluent les mêmes vecteurs, soit ceux de la liste des vecteurs de mouvement candidats. Comme nous le verrons plus en détail dans les prochaines sous-sections, l'approche précalcule l'erreur de prédiction de chacun des blocs 4×4 couvrant la CTU traitée pour chaque candidat de la liste $\mathbb{v}_{\text{liste}}$. Au niveau des PUs, les erreurs

précalculées des blocs 4×4 couvrant la partition traitée sont sommées afin d'obtenir l'erreur de prédiction totale d'un vecteur de mouvement donné, et cela, pour l'ensemble des candidats de la liste $\triangledown_{\text{liste}}$. Aucun autre vecteur ne sera évalué au niveau des PUs.

3.2.3.1 Précalcul des erreurs de prédiction

Pour chaque vecteur de mouvement candidat, les erreurs de prédiction sont précalculées en deux étapes. La première étape interpole la région prédite de la CTU, habituellement une région de 64×64 pixels, puisque les vecteurs de mouvement sont précis au quart de pixel. La seconde étape calcule l'erreur de prédiction pour chaque bloc 4×4 couvrant la région de la CTU traitée. Pour un bloc 4×4 situé à la position $(4x, 4y)$ par rapport au coin gauche supérieur de la CTU, l'erreur de prédiction du k-ième candidat se précalcule en appliquant l'équation suivante :

$$e^{4\times4}_{x,y,k} = \text{SATD}(B^{\text{luma}}_{x,y,k}) + \text{SATD}(B^{\text{Cb}}_{x,y,k}) + \text{SATD}(B^{\text{Cr}}_{x,y,k}) \tag{3.1}$$

où $B^{\text{luma}}_{x,y,k}$ contient la différence entre le bloc luma [2] 4×4 courant et le bloc luma 4×4 prédit, avec $x, y \in \mathbb{N}$. Les blocs $B^{\text{Cb}}_{x,y,k}$ et $B^{\text{Cr}}_{x,y,k}$ contiennent des différences équivalentes pour les blocs chroma [3] 2×2 correspondants. Contrairement à l'approche originale du HM, les blocs chroma sont considérés dans l'équation, en addition au bloc luma, pour améliorer la précision de la prédiction. Les figures 3.3.a à 3.3.c montrent des exemples d'erreurs de prédiction précalculées selon trois vecteurs de mouvement pour une CTU 32×32.

3.2.3.2 Calcul de l'erreur de prédiction d'une partition

Au niveau des PUs, les erreurs de prédiction des blocs 4×4 couvrant la partition traitée sont sommées afin d'obtenir l'erreur de prédiction totale d'un vecteur de mouvement candidat donné. Ainsi, pour une partition de $4N \times 4M$ pixels située à la position ($4x$, $4y$) relative au

2. le luma correspondant à l'information de luminance d'un signal vidéo.
3. le chroma correspondant à l'information de chrominance (couleur) d'un signal vidéo.

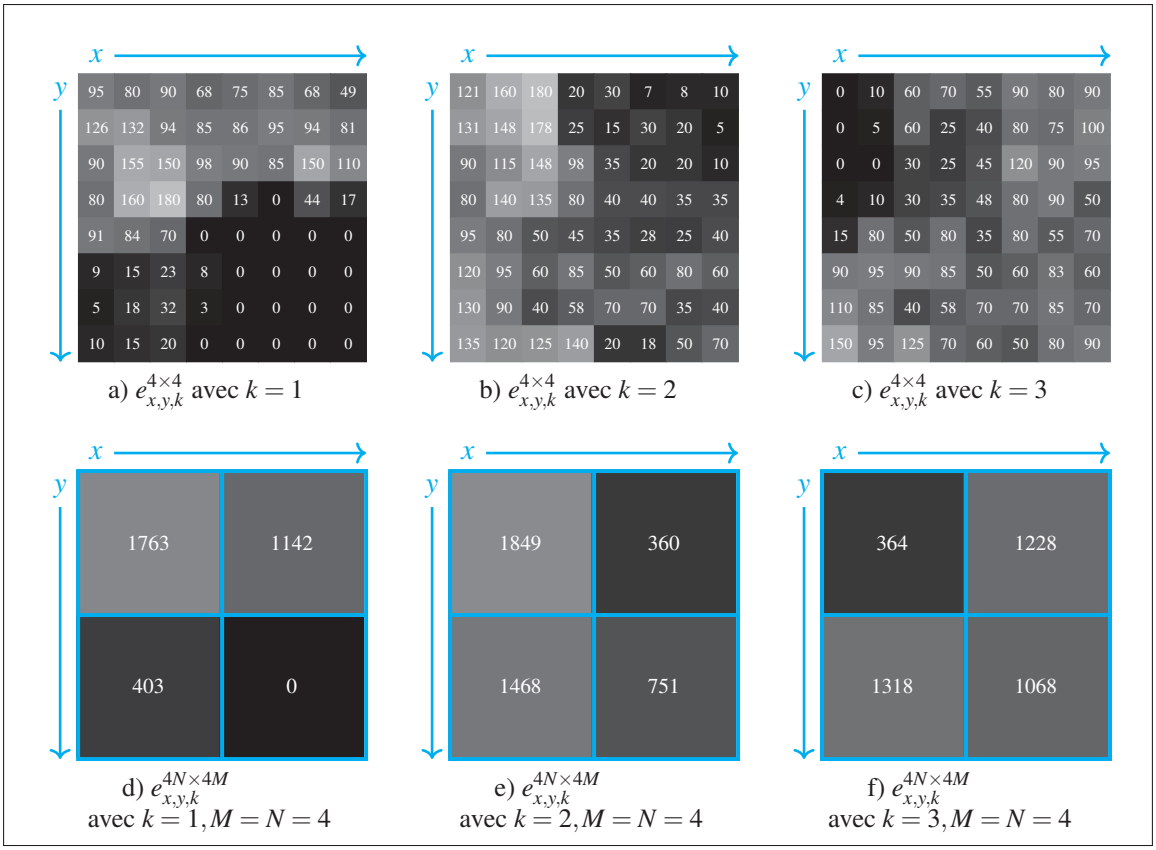


Figure 3.3 Exemples d'erreurs de prédiction : a-c) erreurs précalculées à l'initialisation d'une CTU et d-f) erreurs sommées durant le traitement des PUs

coin supérieur gauche de la CTU, l'erreur de prédiction du k-nième candidat se calcule en appliquant l'équation suivante :

$$e^{4N\times4M}_{x,y,k} = \sum_{i=x}^{(x+N-1)} \sum_{j=y}^{(y+M-1)} e^{4\times4}_{x,y,k}\,. \tag{3.2}$$

Les figures 3.3.d à 3.3.f montrent des exemples d'erreurs de prédiction sommées pour des partitions de 16×16 pixels appartenant à une CTU 32×32. Pour les partitions merge, seuls les vecteurs de mouvement représentant des candidats merge sont considérés.

3.2.3.3 Sélection du meilleur vecteur de mouvement

Une fois l'erreur de prédiction d'un vecteur de mouvement candidat calculée, le cout de la partition prédite est obtenu en ajoutant le cout du mouvement à l'équation 3.2. Pour une partition AMVP, ce cout est calculé à l'aide de l'équation suivante :

$$\mathrm{J}^{\mathrm{AMVP}}_{4N\times 4M}(x,y,k,l) = e^{4N\times 4M}_{x,y,k} + \lambda_{\mathrm{pred}} \times B^{\mathrm{AMVP}}_{\mathrm{motion}}(k,l)\,, \tag{3.3}$$

où k représente l'index du vecteur de mouvement testé et l, l'index du vecteur de mouvement prédit, un vecteur appartenant à $\mathbb{V}_{\mathrm{AMVP}}$. La sélection de la meilleure combinaison k et l s'effectue en minimisant la fonction cout précédente. On obtient donc les index optimaux, notés k^* l^*, de la manière suivante :

$$(k^*,l^*) = \underset{k=1\ldots K, l=1\ldots L}{\arg\min} \left(\mathrm{J}^{\mathrm{AMVP}}_{4N\times 4M}(x,y,k,l)\right). \tag{3.4}$$

Pour les partitions merge, le cout est calculé d'une manière similaire en considérant seulement les index k de la liste $\mathbb{V}_{\mathrm{liste}}$ correspondant à des vecteurs de mouvement associés aussi à la liste $\mathbb{V}_{\mathrm{merge}}$. La fonction cout à minimiser est alors la suivante :

$$\mathrm{J}^{\mathrm{Merge}}_{4N\times 4M}(x,y,k) = e^{4N\times 4M}_{x,y,k} + \lambda_{\mathrm{pred}} \times B^{\mathrm{merge}}_{\mathrm{motion}}(k)\,. \tag{3.5}$$

Il est important de noter que tous les vecteurs de mouvement associés à la liste $\mathbb{V}_{\mathrm{merge}}$ sont toujours inclus dans la liste $\mathbb{V}_{\mathrm{liste}}$ puisque cette dernière est constituée de tous les vecteurs de mouvement HEVC situé dans le voisinage de la CTU traitée, en plus des vecteurs H.264.

3.2.4 Parallélisation des opérations effectuées au niveau de l'unité de codage arborescent

Nous avons développé une version parallèle de notre approche de propagation de mouvement. Comme le montre la figure 3.4, Cette version est constituée de deux fils d'exécution (*threads*) : un fil principal, noté f_1, et un fil supplémentaire, noté f_2. Le fil principal f_1 effectue l'ensemble des tâches de transcodage, à l'exception de la tâche attribuée au fil f_2. Ce dernier est lancé à l'initialisation de la CTU courante, notée c_1. Il a pour rôle d'initialiser la liste des vecteurs de mouvement candidats et de précalculer les erreurs de prédiction de la prochaine CTU, notée c_2, en utilisant les vecteurs de mouvement disponibles. Puisque le fil f_1 traite parallèlement la CTU courante, les vecteurs ∇_{HEVC} ne sont pas entièrement connus étant donné qu'ils dépendent de l'encodage de la CTU c_1. De ce fait, seuls les vecteurs $\nabla_{\text{H.264}}$ et le vecteur $\mathbf{v}_{\text{zero}}$ sont initialisés par le fil f_2. Les vecteurs ∇_{HEVC} seront initialisés par la suite par le fil f_1, lorsque ce dernier aura terminé le traitement de la CTU c_1 et commencé celui de la CTU c_2 (tout en relançant le fil f_2 pour que ce dernier commence le traitement de la prochaine CTU, notée c_3). Le processus se répète de cette manière pour les autres CTUs.

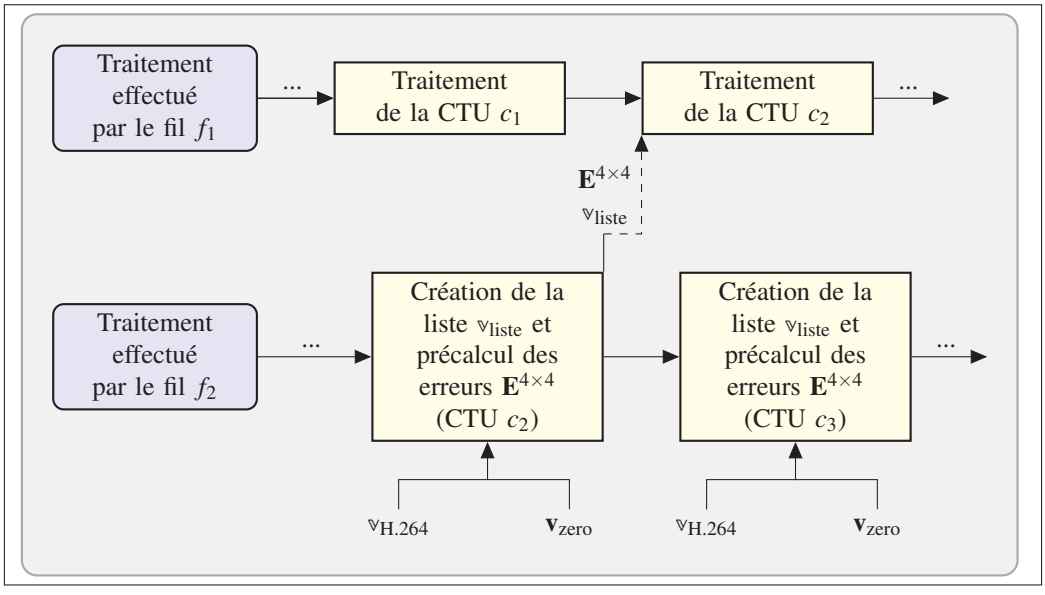


Figure 3.4 Diagramme de blocs de la version parallèle de l'approche de propagation de mouvement proposée

Dans notre implémentation, la première CTU de chaque trame est traitée de manière séquentielle étant donnée la structure de l'encodeur HEVC de référence. De même, la dernière CTU de chaque trame effectue seulement son traitement, et ne lance donc pas le fil f_2 pour commencer le prétraitement de la prochaine CTU (la première CTU de la prochaine trame). Plus loin, nous comparons les résultats de cette version avec la version séquentielle.

3.3 Analyse de la propagation du mouvement

L'approche présentée dans la section précédente repose sur l'idée que les vecteurs de mouvement H.264 sont suffisamment précis pour être réutilisés dans HEVC sans raffinement du mouvement, mais que ces vecteurs se propagent de manière différente dans HEVC. Autrement dit, la représentation du mouvement HEVC peut être obtenue en employant un algorithme de propagation du mouvement, comme l'approche que nous proposons dans cette section, plutôt qu'un algorithme de raffinement du mouvement, comme ceux proposés dans la littérature. Pour valider cette idée, nous avons posé les deux hypothèses suivantes :

- *Hypothèse 1* : Les vecteurs de mouvement H.264 sont suffisamment précis pour être réutilisés dans HEVC sans l'emploi additionnel d'un algorithme de raffinement du mouvement.
- *Hypothèse 2* : Le meilleur vecteur de mouvement pour la région traitée dans HEVC n'est pas toujours localisé dans la région H.264 correspondante (la région d'extraction de base H.264) ; dans certains cas, un meilleur vecteur peut être trouvé dans le voisinage de cette région pour différente raisons (différences entre les standards H.264 et HEVC, différences entre les images de référence H.264 et HEVC, différences entre les vecteurs de mouvement prédits par H.264 et HEVC)

Afin de valider ces hypothèses et mesurer la performance de l'approche proposée, nous avons mené plusieurs expériences sur un transcodeur H.264 à HEVC. Dans la section 3.4, nous décrivons la méthodologie complète de nos expériences et nous montrons que notre algorithme de propagation de mouvement a un impact négligeable sur l'efficacité de codage

(comparé au même transcodage effectué avec l'algorithme d'estimation de mouvement du HM). Conséquemment, ces résultats valident l'*Hypothèse 1*. Cependant, ils sont insuffisants pour valider l'*Hypothèse 2*, puisqu'ils ne fournissent aucune information sur les vecteurs de mouvement sélectionnés par HEVC durant le transcodage.

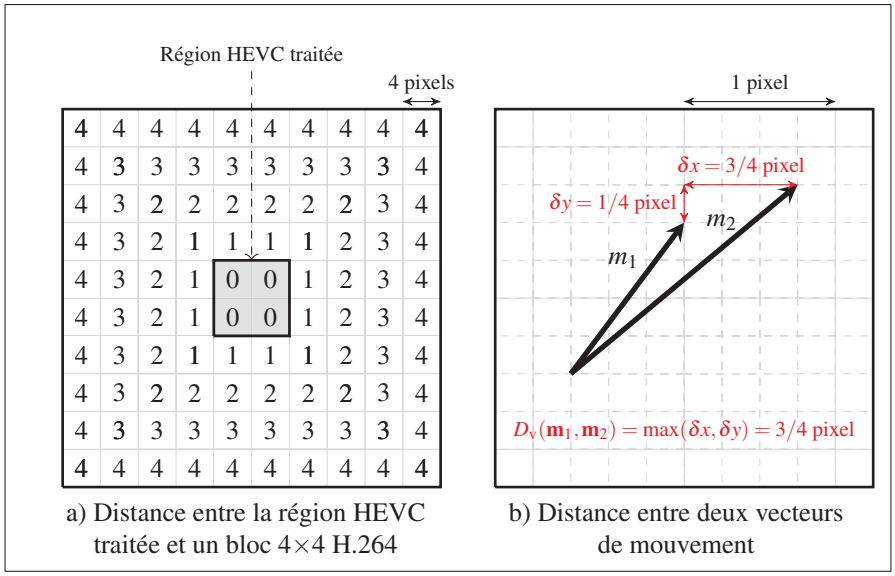
Pour résoudre ce problème, nous analysons dans cette section les relations entre les vecteurs de mouvement H.264 et les vecteurs de mouvement HEVC qui ont été encodés à l'aide de notre approche de propagation du mouvement. Plus spécifiquement, nous désirons déterminer où se trouve le vecteur HEVC encodé dans la région d'extraction H.264 (quels blocs 4×4 H.264 contiennent ce vecteur). Ainsi, si le vecteur de mouvement HEVC encodé est toujours trouvé à l'intérieur de la région d'extraction de base (la même région que la partition HEVC traitée), l'*Hypothèse 2* est alors invalide, et le paramètre α, qui définit la région étendue d'extraction des vecteurs H.264 (voir figure 3.2), doit être égal à 0. Inversement, si le bloc HEVC encodé se trouve, dans certains cas, uniquement à l'extérieur de la région HEVC traitée, l'*Hypothèse 2* est alors valide et le paramètre α doit être déterminé.

3.3.1 Impact de la zone d'extraction étendue H.264

Pour déterminer l'impact de la zone d'extraction étendue H.264 sur les vecteurs de mouvement HEVC encodés, nous avons effectué une expérience avec le paramètre α fixé à 16 blocs 4×4 (64 pixels). L'expérience calcule la distance D_{bloc} (en blocs 4×4) entre la partition HEVC encodée et le bloc 4×4 H.264 le plus près ayant un vecteur de mouvement similaire. Comme le montre la figure 3.5.a, la distance D_{bloc} est mesurée en employant la distance de Tchebychev. Cette mesure, aussi appelée la distance de l'échiquier, calcule la distance entre deux blocs, situés respectivement aux positions $\mathbf{b}_1$ et $\mathbf{b}_2$, en appliquant l'équation suivante :

$$D_{\text{bloc}}(\mathbf{b}_1, \mathbf{b}_2) = \max(|b_{1_x} - b_{2_x}|, |b_{1_y} - b_{2_y}|)\,. \tag{3.6}$$

La distance D_{bloc} obtenue a pour avantage d'être directement reliée au paramètre α. Par exemple, une distance D_{bloc} de 2 entre un bloc H.264 et la région traitée HEVC signifie que le paramètre α doit être fixé à 2 pour atteindre ce bloc.



a) Distance entre la région HEVC traitée et un bloc 4×4 H.264

b) Distance entre deux vecteurs de mouvement

Figure 3.5 Distance de Tchebychev

Pour des raisons similaires, la distance entre les vecteurs de mouvement de deux blocs est aussi calculée à l'aide de la distance de Tchebychev, en appliquant l'équation suivante :

$$D_{\text{v}}(\mathbf{v}_1, \mathbf{v}_2) = \max(|v_{1_x} - v_{2_x}|, |v_{1_y} - v_{2_y}|)\,. \tag{3.7}$$

Lorsque cette distance est égale ou plus petite que la valeur d'un paramètre de tolérance τ, les vecteurs de mouvement sont considérés comme similaires. Ce paramètre de tolérance a pour but de déterminer si la zone étendue d'extraction H.264 est avantageuse surtout parce qu'elle donne accès à des vecteurs de mouvement H.264 différents des vecteurs situés dans la zone d'extraction de base ou, au contraire, si la zone d'extraction est avantageuse parce qu'elle permet d'accéder à des vecteurs de mouvement similaires, mais plus précis (qui réduisent

davantage l'erreur de prédiction), à ceux déjà situés à l'intérieur de la région d'extraction de base.

3.3.1.1 Distance du meilleur vecteur de mouvement H.264

Notre analyse principale consiste à calculer, pour chaque mode de prédiction inter (AMVP et merge), la fonction de distribution cumulative de la distance entre le bloc HEVC encodé et le bloc H.264 le plus près (celui minimisant D_{bloc}) et respectant la contrainte $D_{\text{v}} \leq \tau$.

La figure 3.6 montre en exemple les distributions obtenues pour trois séquences et trois valeurs de tolérance τ (1/4, 1/2, 3/4 pixels). Nous observons sur cette figure deux tendances différentes pour les modes AMVP et merge. Pour le mode AMVP avec $\tau = 0$, le vecteur de mouvement H.264 le plus près est souvent localisé dans la région d'extraction de base. De plus, la fonction converge rapidement vers 1. Pour le mode merge, le vecteur de mouvement H.264 est moins souvent localisé dans la région d'extraction de base H.264, en particulier lorsque le QPs est élevé. De plus, les fonctions de distribution des modes merge n'atteignent pas toujours 1. Ces deux tendances s'expliquent par le fait que les modes AMVP et merge jouent des rôles complémentaires. Le mode merge est un mécanisme utilisé pour propager des vecteurs de mouvement HEVC dans des blocs (spatialement et temporellement) voisins. Il est donc très efficace dans des régions où le mouvement est uniforme ou continu. Pour sa part, le mode AMVP est un mécanisme surtout utilisé pour signaler un changement dans le mouvement. Il est donc souvent utilisé dans les régions où le mouvement est discontinu. Dans un tel cas, le mode AMVP favorise les vecteurs de mouvement H.264, puisque ces derniers contiennent généralement de l'information pertinente sur un changement de mouvement qui est alors inconnu de HEVC.

Ces résultats valident l'*Hypothèse 2* pour les deux modes. En effet, le vecteur de mouvement HEVC encodé n'est pas toujours présent dans la région H.264 correspondante puisque $P(D_{\text{bloc}} = 0) < 1$. Il est cependant présent dans le voisinage rapproché puisque $P(D_{\text{bloc}} = 0) < P(D_{\text{bloc}} \leq d)$ pour des petites valeurs de d. Il est important de noter que le mode merge

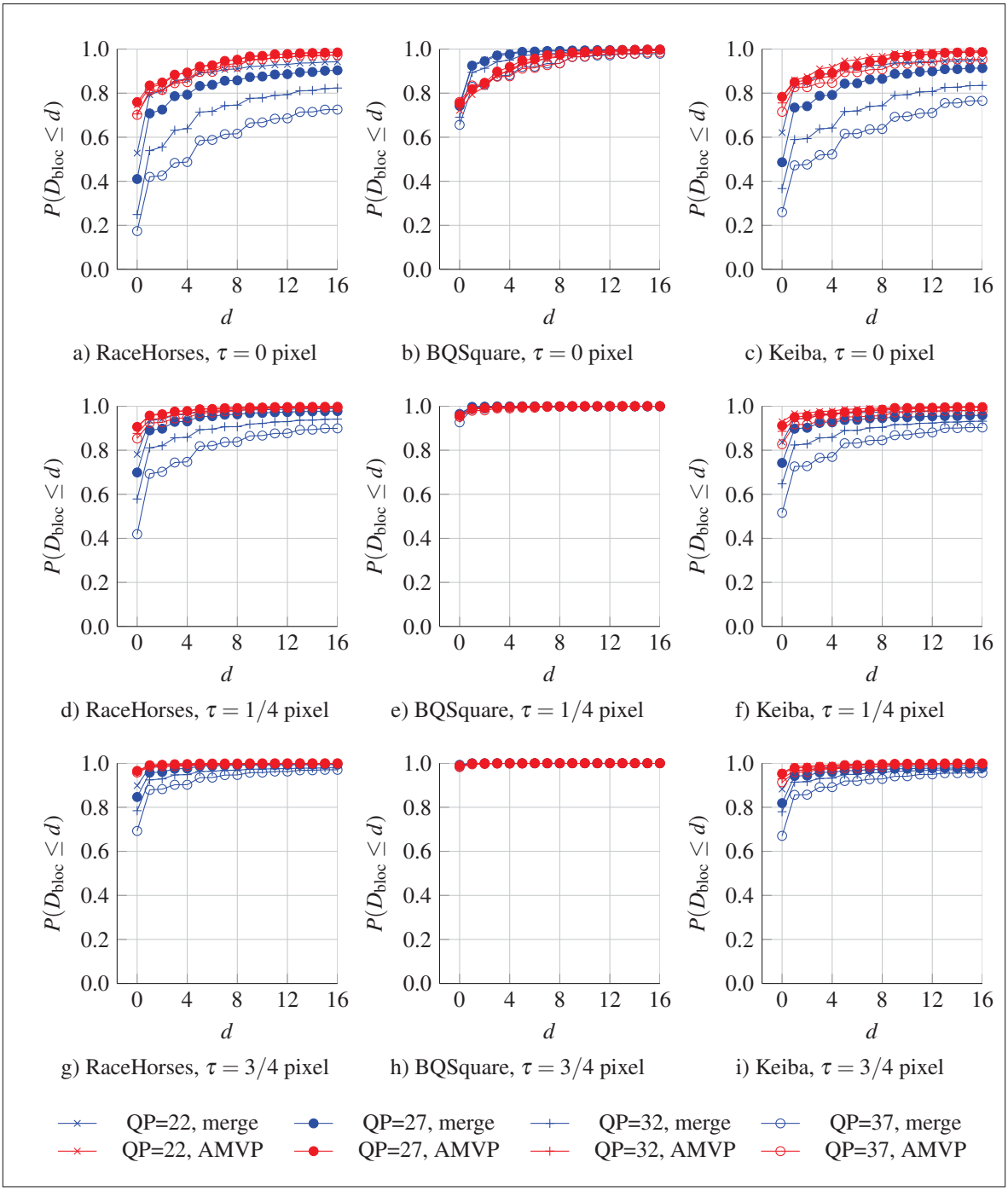


Figure 3.6 Fonctions des distributions cumulatives des distances entre le bloc HEVC traité et le bloc H.264 le plus près ayant un vecteur de mouvement similaire à celui du bloc HEVC encodé

n'a aucune influence sur le paramètre α, puisque ce mode utilise uniquement des vecteurs de mouvement déjà présents dans $\mathbb{V}_{\text{HEVC}}$. Enfin, même si l'augmentation du paramètre α permet de trouver un meilleur vecteur de mouvement H.264, nous avons trouvé expérimentalement que le meilleur compromis entre la complexité de l'approche et l'efficacité de codage est obtenue lorsque α est fixé à 1.

Nous pouvons également observer sur la figure 3.6 que l'augmentation de la tolérance a pour effet d'élever les courbes des fonctions de distribution. Plus particulièrement, on remarque que les fonctions des modes AMVP sont très proches de 1 pour $d = 0$ lorsque la tolérance τ est fixée à 3/4 pixel. Par conséquent, les vecteurs de mouvement H.264 trouvés dans la région d'extraction étendue H.264 (mais absent de la région de base) sont généralement très similaires à l'un des vecteurs H.264 situés à l'intérieur de la région d'extraction de base. Ces résultats suggèrent donc que l'augmentation du paramètre α (pour une tolérance $\tau = 0$) a généralement pour effet de raffiner au pixel fractionnaire un vecteur de mouvement H.264 se trouvant déjà dans la région d'extraction de base.

En contrepartie, et toujours pour une tolérance de 3/4 pixel, on note une petit augmentation des fonctions de distribution lorsque la distance d passe de 0 à 1. Ces résultats indiquent que les blocs H.264 situés dans le voisinage immédiat de la région traitée par HEVC sont susceptibles de contenir des vecteurs de mouvement différents (d'au moins 1 pixel) pouvant réduire le cout du mouvement par rapport à des vecteurs de mouvement H.264 situés dans la région d'extraction de base H.264.

3.3.1.2 Propagation des vecteurs de mouvement H.264 dans HEVC

Jusqu'à maintenant, nous avons analysé la distance entre le bloc HEVC encodé et le bloc H.264 le plus près ayant un vecteur de mouvement similaire. Cette analyse a permis de conclure qu'il est pertinent d'extraire des vecteurs de mouvement au-delà de la région traitée dans HEVC ($\alpha > 0$). Cependant, elle ne valide pas si le voisinage au complet du bloc traité doit être

considéré ou, autrement dit, si la zone d'extraction (le rectangle) doit couvrir les régions situées au-dessus, en dessous, à gauche et à droite du bloc traité.

Comme nous l'avons vu précédemment, le vecteur de mouvement du bloc courant est prédit à l'aide de blocs qui ont été précédemment encodés, généralement des blocs situés au-dessus et à gauche de la région traitée, mais il peut aussi s'agir, pour HEVC, d'un bloc situé au même endroit que le bloc traité dans la trame de référence HEVC. Les vecteurs prédits influencent grandement les algorithmes d'estimation de mouvement. D'une part, les régions où le mouvement est continu sont souvent encodées directement avec un vecteur de mouvement prédit [4]. D'autre part, le cout du vecteur différentiel augmente lorsqu'il s'éloigne du vecteur prédit. De ce fait, les vecteurs prédits sont favorisés dans les régions simples, parfois même lorsqu'ils ne décrivent pas le mouvement réel, en particulier lorsque le QP est élevé. Pour ces différentes raisons, nous croyons les vecteurs de mouvement H.264 situés au-dessus ou à gauche du bloc traité pourraient avoir une probabilité plus élevée d'être sélectionnés par notre approche de propagation que les blocs H.264 situés en dessous ou à droite de la région traitée.

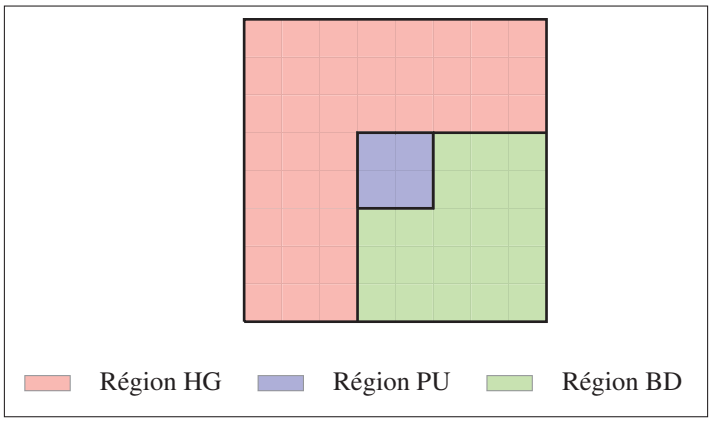


Figure 3.7 Les trois sous-régions de la région d'extraction H.264

Afin de valider cette hypothèse, nous avons divisé la région d'extraction H.264 en trois sous-régions de base, les régions PU, HG et BD, comme le montre la figure 3.7. La région PU

4. On considère qu'une région est encodée avec un vecteur prédit lorsque le mode merge est utilisé (ce mode n'autorise aucun vecteur différentiel) ou lorsque le mode AMVP est utilisé et associé à un vecteur différentiel nul.

représente la région H.264 située à l'intérieur de la région HEVC traitée. La région HG représente une région H.264 située en haut ou à gauche de la région HEVC traitée. La région BD représente une région H.264 située en bas ou à droite de la région HEVC traitée, mais absente de la région HG. Les ensembles de vecteurs de mouvement situés dans les régions HG, PU, BD sont respectivement notés $\mathcal{HG}$, $\mathcal{PU}$ et $\mathcal{BD}$. Par exemple, $\mathcal{PU}$ est défini par :

$$\mathcal{PU} = \{\mathbf{v} \,|\, \mathbf{v} \text{ est un vecteur de la région PU}\}\,, \tag{3.8}$$

où $\mathbf{v}$ est un vecteur de mouvement H.264 encodé. Le vecteur de mouvement HEVC encodé peut être présent dans zéro jusqu'à trois de ces régions (ensembles). Les combinaisons d'intérêts pour notre analyse sont les suivantes :

- $\mathcal{HG} \setminus (\mathcal{PU} \cup \mathcal{BD}) \Rightarrow$ Seulement $\mathcal{HG}$.
- $\mathcal{BD} \setminus (\mathcal{HG} \cup \mathcal{PU}) \Rightarrow$ Seulement $\mathcal{BD}$.
- $\mathcal{PU} \setminus (\mathcal{HG} \cup \mathcal{BD}) \Rightarrow$ Seulement $\mathcal{PU}$.
- $(\mathcal{HG} \cap \mathcal{PU}) \setminus \mathcal{BD} \Rightarrow$ Intersection de $\mathcal{HG}$ et $\mathcal{PU}$.
- $(\mathcal{PU} \cap \mathcal{BD}) \setminus \mathcal{HG} \Rightarrow$ Intersection de $\mathcal{PU}$ et $\mathcal{BD}$.
- $(\mathcal{HG} \cap \mathcal{PU} \cap \mathcal{BD}) \Rightarrow$ Intersection de $\mathcal{HG}$, $\mathcal{PU}$ et $\mathcal{BD}$.
- $\emptyset \Rightarrow$ Lorsque le vecteur n'a pas été trouvé dans la région d'extraction H.264.

Étant donné les rôles complémentaires des modes merge et AMVP, nous avons analysé ces modes de manière séparée. Notre analyse considère seulement les modes qui ont été encodés par le transcodeur (les modes gagnants). Les autres modes testés ne sont pas considérés puisqu'ils constituent des modes plus couteux, donc moins intéressant d'un point de vue d'efficacité de codage. Comme le montre la figure 3.8, le transcodeur vidéo encode beaucoup plus fréquemment des modes merge que des modes AMVP, en particulier pour des séquences qui ont été encodées et transcodées avec un QP élevé (l'Annexe I décrit les différences séquences vidéos). Ce phénomène s'explique par le fait que le mode merge est très efficace dans les régions où le mouvement est continu. Par exemple, on observe une présence plus

élevée de modes merge dans les séquences *Mobisode2* et *BQSquare*, deux séquences où le mouvement est très simple (voir tableau I-1). Inversement, les séquences *Racehorses* et *Keiba*, deux séquences où le mouvement est moins régulier, le nombre de modes merge est moins élevé. Enfin, l'augmentation du QP a pour effet d'augmenter la proportion des modes merge parce qu'une contrainte plus forte est appliquée sur le cout du mouvement, ce qui favorise les modes où le signalement du mouvement est peu couteux, comme c'est le cas pour les modes merge.

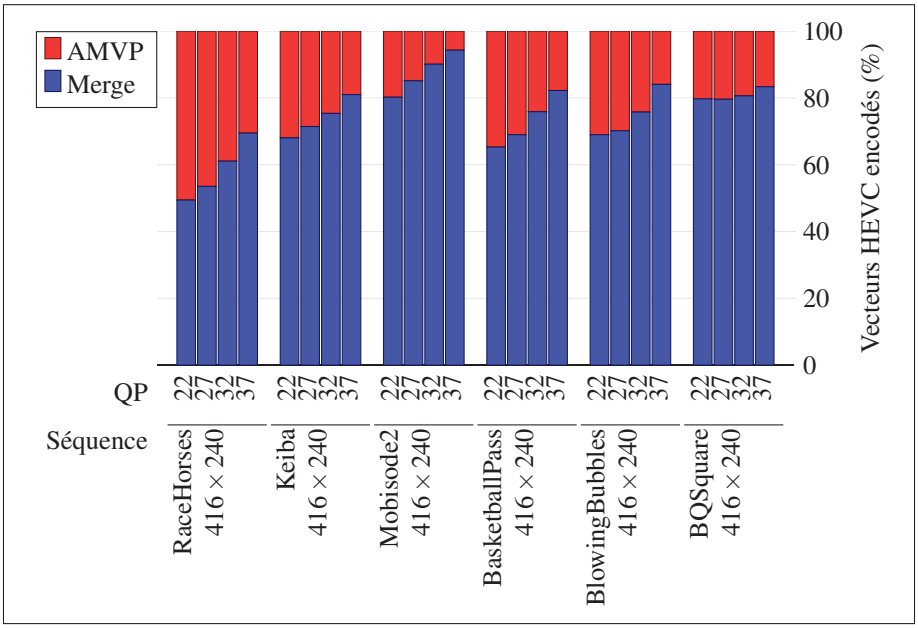


Figure 3.8 Distribution des modes AMVP et merge

Sur la figure 3.9, nous observons de nouveau deux tendances différentes pour les modes AMVP et merge. Ces tendances corroborent l'idée que les deux modes jouent des rôles distincts. Ainsi, la proportion des ensembles $\mathcal{PU} \setminus (\mathcal{HG} \cup \mathcal{BD})$, $(\mathcal{PU} \cap \mathcal{BD}) \setminus \mathcal{HG}$ et $\mathcal{BD} \setminus (\mathcal{HG} \cup \mathcal{PU})$ (représentés par les couleurs de beige à brun) est nettement plus élevée pour le mode AMVP que le mode merge. Ces ensembles représentent l'apparition d'un nouveau vecteur de mouvement dans H.264 (un vecteur qui n'est pas présent dans la région HG). Le mode merge est inapproprié pour représenter cette discontinuité dans le mouvement. Inversement, les ensembles qui incluent la localisation HG sont beaucoup plus fréquents pour le mode merge

puisque ce mode propage le mouvement prédit, généralement par un vecteur de mouvement situé au-dessus ou à gauche de la région HEVC traitée. De plus, le mode merge inclut davantage d'occurrences de l'ensemble $\emptyset$ (représentées par la couleur mauve). Ce dernier résultat s'explique par le fait que le mode merge supporte des candidats HEVC temporels. Ces candidats sont parfois uniques et absents des données H.264 de la trame courante.

3.4 Expérimentation

Jusqu'à maintenant, nous avons démontré que le meilleur vecteur de mouvement H.264 pour la région HEVC traitée n'est pas toujours situé dans la région H.264 correspondante (la région d'extraction de base). Il peut être situé dans le voisinage de cette région (la région d'extraction étendue). Ainsi, l'*Hypothèse 2* est validée et l'emploi d'un algorithme de propagation est justifié. Cependant, l'analyse effectuée précédemment n'indique pas si l'approche proposée produit un codage aussi efficace qu'un algorithme d'estimation de mouvement conventionnel, ou qu'un algorithme de raffinement tel que proposé dans la littérature. Afin de vérifier si c'est le cas et pour valider l'*Hypothèse 1*, nous présentons dans cette section les expériences que nous avons effectuées pour analyser l'efficacité de codage de notre approche. Nous présentons notre méthodologie à la section 3.4.1 et nos résultats expérimentaux à la section 3.4.2.

3.4.1 Méthodologie

Pour valider les performances de l'approche de propagation de mouvement proposée, nous avons implémenté deux transcodeurs H.264 à HEVC. Les deux implémentations sont basées sur le décodeur H.264 de référence, la JM 18.2 (JCT-VC (2013)), et l'encodeur HEVC de référence, le HM 12.1 (JCT-VC (2014)).

Le premier transcodeur effectue un transcodage en cascade, c'est-à-dire, qu'il décode entièrement la séquence H.264 à l'entrée pour la réencoder au complet afin de produire la séquence HEVC de sortie. Ce transcodeur estime le mouvement à l'aide de l'algorithme de recherche TZS (voir section 1.4.3.1.1) du HM.

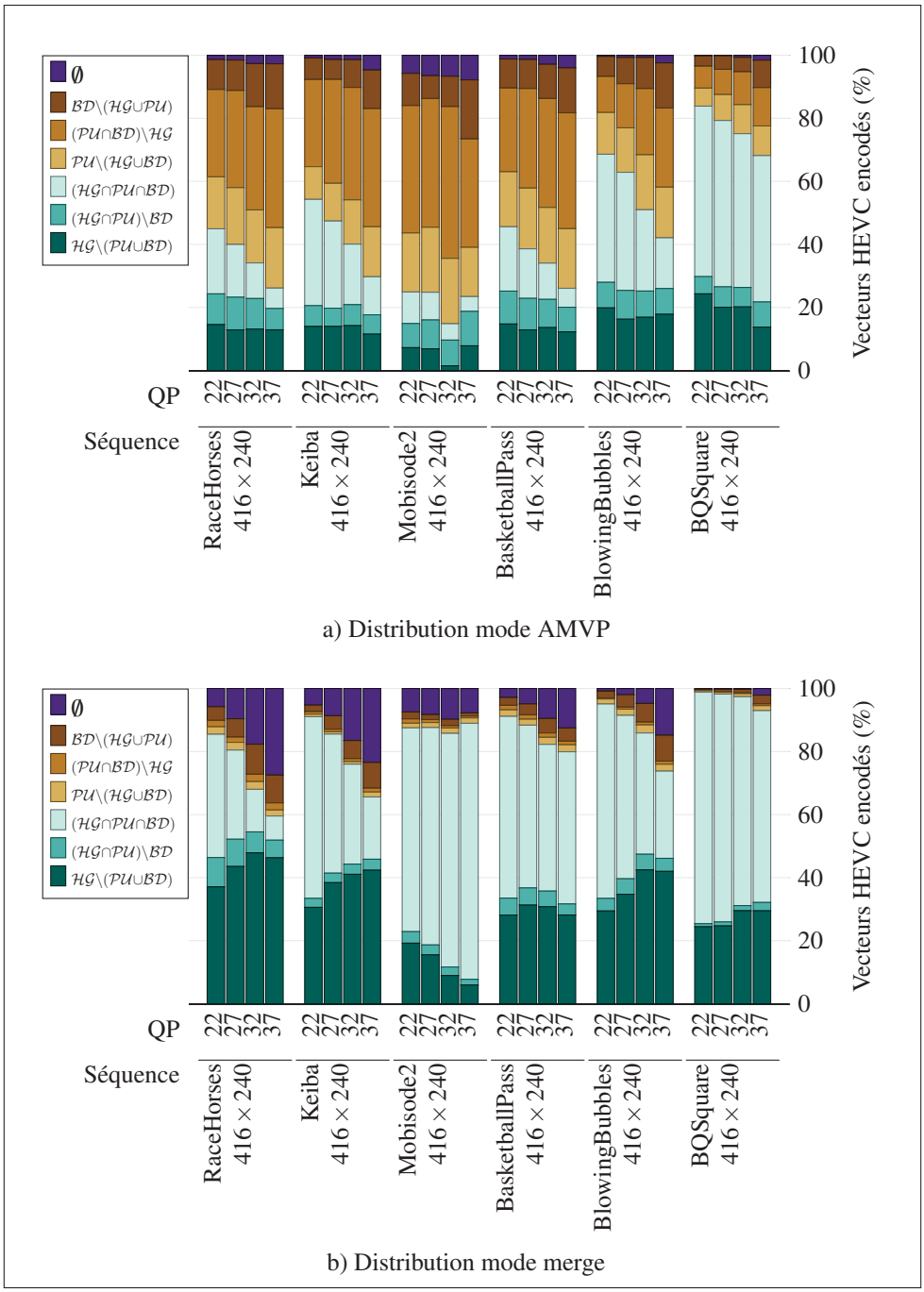


Figure 3.9 Distribution des localisations H.264

Le deuxième transcodeur est une version du premier transcodeur qui a été modifié afin de supporter notre algorithme de propagation de mouvement. En plus de décoder entièrement la séquence H.264, ce transcodeur extrait l'information de mouvement H.264. Cette information

est par la suite transmise au module implémentant notre algorithme de propagation de mouvement (qui remplace la recherche TZS) afin de générer la liste des vecteurs de mouvement candidats tel que décrit précédemment. À noter que deux versions de ce transcodeur sont évaluées : une version séquentielle et une version parallèle (voir section 3.2.4).

3.4.1.1 Description et configuration des expériences

Afin de comparer les deux transcodeurs, nous avons encodé et transcodé plusieurs séquences vidéos. Comme plusieurs autres travaux de recherche, nous avons utilisé des séquences vidéos recommandées par la JCT-VC dans un document intitulé « *Common test conditions and software reference configurations* » (Bossen (2012)). Nous avons opté pour des séquences ayant des résolutions de 416×240 (des séquences de classe D, selon Bossen (2012)), 832×480 (classe C) et 1920×1080 (classe B) pixels afin de limiter la durée des simulations. Les séquences sélectionnées sont constituées de contenus vidéo variés et représentent des cas d'encodage vidéo réalistes. Le tableau I-1 présente ces séquences et décrit leur contenu selon quatre aspects : le mouvement des objets, le mouvement de la caméra, les occlusions et la variation de l'éclairage. Ces descriptions sont particulièrement utiles pour l'analyse des résultats.

Comme recommandé par la JCT-VC pour évaluer les performances d'un encodage HEVC, chaque séquence du tableau I-1 est successivement encodée puis transcodée en employant les quatre QPs suivants : 22, 27, 32, 37. Le transcodage est donc effectué avec un QP fixe, comme c'est le cas pour plusieurs autres travaux de recherche. L'utilisation d'un QP fixe permet d'évaluer les performances d'une approche de transcodage sans être biaisée par un contrôleur de débit (*rate control*). Une structure de codage de type IPPP [5] avec une seule trame intra, insérée au début de la séquence, est utilisée pour effectuer le codage et le transcodage. L'encodage HEVC est effectué avec la configuration faible délai P (*low delay P*).

5. Dans ce type de structure, une trame intra est suivi par plusieurs trames inter de type P.

Les séquences H.264 sont générées à l'aide de l'encodeur H.264 de référence, la JM 18.2. L'encodeur H.264 est configuré pour produire une séquence compatible avec le profil *baseline* et emploie l'algorithme de recherche *Enhanced Predictive Zonal Search* (EPZS) (Tourapis (2002)). L'encodeur H.264 et le transcodeur en cascade sont configurés pour utiliser une plage de recherche de ± 64 pixels (cette plage est utilisée pour contraindre la recherche à l'intérieur d'une zone de recherche telle que décrit à la section 1.1.1.2).

3.4.1.2 Mesures de performance

Les performances du transcodeur basé sur notre approche de propagation sont comparées aux performances du transcodeur en cascade à l'aide de deux mesures. D'abord, l'efficacité de codage est évaluée à l'aide du BD-Rate (voir section 1.1.3.3). Ensuite, la réduction de la complexité est calculée à l'aide de la mesure d'accélération définie par l'équation suivante :

$$A = \frac{T_r}{T_m}, \tag{3.9}$$

où T_r représente le temps d'exécution de la partie encodage HEVC du transcodeur en cascade et T_m, le temps d'exécution pour la méthode proposée. Cette mesure a l'avantage d'évaluer l'impact global de l'algorithme de propagation de mouvement sur l'encodage HEVC effectué durant le transcodage. C'est important d'analyser cet impact, puisque les vecteurs de mouvement sélectionnés ont une influence sur les autres étapes de l'encodage HEVC.

3.4.2 Résultats

Comme le montre le tableau 3.1, notre approche a tendance à préserver l'efficacité de codage, comparativement à un transcodeur en cascade. Dans plusieurs cas, on observe une légère réduction du BD-Rate, c'est-à-dire, une légère amélioration de l'efficacité de codage. Pour les autres cas, l'augmentation du BD-Rate est inférieure à 1%, ce qui est plutôt marginal. Ces résultats démontrent donc que les vecteurs de mouvement H.264 sont suffisamment précis pour

être réutilisés par notre algorithme de propagation du mouvement, sans l'emploi additionnel d'un algorithme de raffinement dans la configuration testée, puisque l'algorithme proposé préserve l'efficacité de codage.

Tableau 3.1 Performances du transcodeur proposé relatives aux performances d'un transcodeur en cascade

		Accélération version séquentielle (par QP)					Accélération version parallèle (par QP)					
Séquence	Résolution	22	27	32	37	Moy.	22	27	32	37	Moy.	BD-Rate(%)
BasketballDrive	1920×1080	1.40	1.48	1.52	1.54	1.49	1.43	1.52	1.55	1.57	1.52	−1.57
BQTerrace	1920×1080	1.30	1.38	1.47	1.48	1.41	1.32	1.41	1.50	1.52	1.44	−0.87
Cactus	1920×1080	1.27	1.33	1.39	1.43	1.36	1.29	1.36	1.41	1.45	1.38	0.15
Kimono	1920×1080	1.32	1.33	1.39	1.43	1.37	1.35	1.39	1.43	1.48	1.41	0.96
ParkScene	1920×1080	1.30	1.37	1.44	1.46	1.39	1.32	1.40	1.47	1.49	1.42	0.58
BasketballDrill	832×480	1.31	1.35	1.42	1.46	1.39	1.33	1.38	1.44	1.49	1.41	−0.66
BQMall	832×480	1.30	1.36	1.40	1.46	1.38	1.33	1.39	1.45	1.50	1.42	−0.34
PartyScene	832×480	1.27	1.30	1.35	1.42	1.34	1.28	1.33	1.40	1.46	1.37	−0.15
RaceHorses	832×480	1.30	1.33	1.41	1.49	1.38	1.32	1.37	1.45	1.54	1.42	−0.99
BasketballPass	416×240	1.30	1.30	1.37	1.43	1.35	1.32	1.34	1.39	1.48	1.38	0.09
BQSquare	416×240	1.26	1.29	1.36	1.48	1.35	1.28	1.30	1.38	1.50	1.37	0.17
BlowingBubbles	416×240	1.27	1.31	1.40	1.42	1.35	1.28	1.34	1.42	1.47	1.38	0.38
RaceHorses	416×240	1.27	1.28	1.36	1.42	1.33	1.31	1.33	1.41	1.49	1.39	−0.87
Moyenne	**N.A**	**1.30**	**1.34**	**1.41**	**1.46**	**1.38**	**1.32**	**1.37**	**1.44**	**1.50**	**1.49**	**-0.25**

En plus de préserver l'efficacité de codage, l'approche proposée réduit le temps d'exécution de manière significative. Selon les résultats montrés par le tableau, l'accélération obtenue par notre approche de transcodage a tendance à légèrement augmenter avec le QP. Deux principales raisons expliquent cette amélioration de l'accélération. D'une part, le pourcentage du temps consacré à l'estimation du mouvement par le transcodeur en cascade augmente avec le QP. D'autre part, ce pourcentage de temps diminue pour notre approche de transcodage. Dans les deux cas, l'augmentation du QP entraine une quantification plus agressive. Davantage de coefficients sont donc éliminés, et plusieurs CUs sont encodées sans données résiduelles. Cette réduction de l'information résiduelle simplifie donc le processus d'encodage, en particulier l'évaluation des couts débit-distorsion. L'augmentation du QP réduit aussi la complexité de l'estimation de mouvement du transcodeur en cascade (en réduisant le nombre de vecteurs de mouvement évalués), mais dans une moindre mesure. Globalement, l'augmentation du QP a

donc pour effet d'augmenter la proportion du temps que le transcodeur en cascade consacre à l'estimation du mouvement. Inversement, l'augmentation du QP réduit généralement la proportion du temps que notre transcodage consacre à la propagation du mouvement. Ces deux phénomènes ont donc pour conséquence d'améliorer l'accélération avec l'augmentation du QP.

Nous remarquons également que la version parallèle de notre approche améliore légèrement les accélérations en obtenant la même efficacité de codage que la version séquentielle. Les accélérations sont certes modestes, cependant l'implémentation de cette version parallèle est simple. Elle montre donc l'un des avantages de déterminer les vecteurs de mouvement candidats durant l'initialisation d'une CTU.

En somme, l'algorithme de propagation du mouvement proposé peut remplacer avantageusement, dans un contexte de transcodage H.264 à HEVC, un algorithme de recherche conventionnel, tel que l'algorithme de recherche TZS du HM, pour les configurations utilisées. Cependant, notre analyse ne démontre pas si une approche basée sur la propagation du mouvement est aussi efficace pour d'autres configurations de transcodage, notamment pour des configurations supportant plusieurs images de référence et des configurations exploitant la prédiction bidirectionnelle (les trames B).

3.4.2.1 Comparaison avec l'état de l'art

Il est difficile de déterminer avec certitude si notre approche obtient de meilleurs résultats que les approches de raffinement proposées dans la littérature, puisque la plupart des chercheurs évaluent uniquement les performances globales de leur transcodeur, c'est-à-dire, qu'ils tiennent compte d'autres méthodes durant l'évaluation des performances. Néanmoins, certains auteurs présentent les performances spécifiques de leur algorithme de raffinement de mouvement. Une comparaison équitable avec ces méthodes demeure cependant difficile, puisque la méthodologie employée diffère d'un chercheur à un autre, malgré plusieurs similitudes. Ces méthodologies se distinguent sur plusieurs aspects, dont les plus importants sont : les séquences utilisées, la version du HM employé, les configurations de transcodage évaluées ainsi que les

métriques de performances employées. Ces restrictions étant mentionnées, nous comparons maintenant notre approche avec deux autres approches présentées dans la littérature.

D'abord, Peixoto et Izquierdo (2012) annoncent une augmentation moyenne du BD-Rate de 3.60 % pour une accélération moyenne de 1.46. Les résultats obtenus par ces auteurs varient beaucoup d'une séquence à une autre. Par exemple, les auteurs obtiennent respectivement pour les séquences RaceHorses et BasketballPass, des accélérations de 1.77 et 1.40 et des augmentations du BD-Rate de 2.78 et 7.71 %. Les auteurs obtiennent donc une accélération moyenne (1.46) légèrement supérieure à la nôtre (1.38), au prix d'une perte d'efficacité de codage plus grande. Notons cependant que les auteurs effectuent leurs tests avec quatre trames de référence, alors que notre approche est implémentée pour supporter une seule trame de référence. Afin de fournir une comparaison plus juste, nous avons comparé les résultats de notre approche, configurée avec une seule trame de référence, avec les résultats d'un transcodage en cascade configuré avec quatre trames de référence. Les résultats obtenus sont présentés dans le tableau 3.2.

Tableau 3.2 Performances du transcodeur proposé, configuré avec une trame de référence, relatives aux performances d'un transcodeur en cascade, configuré avec quatre trames de référence

		Accélération (par QP)					
Séquence	Résolution	22	27	32	37	Moy.	BD-Rate(%)
BasketballDrive	1920×1080 (Classe B)	2.51	2.89	3.13	3.23	2.94	−1.17
BQTerrace	1920×1080 (Classe B)	1.71	2.00	2.35	2.50	2.14	0.83
Cactus	1920×1080 (Classe B)	1.75	2.14	2.40	2.58	2.22	0.53
Kimono	1920×1080 (Classe B)	2.32	2.53	2.80	2.99	2.66	0.66
ParkScene	1920×1080 (Classe B)	1.87	2.20	2.52	2.65	2.31	0.80
BasketballDrill	832×480 (Classe C)	2.17	2.56	2.94	3.20	2.72	3.34
BQMall	832×480 (Classe C)	2.12	2.47	2.79	3.07	1.41	2.26
PartyScene	832×480 (Classe C)	1.73	1.92	2.30	2.67	2.16	5.57
RaceHorses	832×480 (Classe C)	2.45	2.84	3.37	3.82	3.12	0.33
BasketballPass	416×240 (Classe D)	2.20	2.44	2.82	3.10	2.64	0.83
BQSquare	416×240 (Classe D)	1.58	1.67	1.99	2.50	1.94	13.20
BlowingBubbles	416×240 (Classe D)	1.71	1.95	2.37	2.60	2.16	3.91
RaceHorses	416×240 (Classe D)	2.24	2.59	3.04	3.45	2.83	−0.01
Moyenne	**N.A**	**2.03**	**2.32**	**2.68**	**2.95**	**2.50**	**2.39**

Ce tableau montre que notre approche obtient des accélérations nettement supérieures aux accélérations obtenues par Peixoto et Izquierdo (2012) et celles obtenues dans le tableau précédent, où le transcodeur en cascade emploie une seule trame de référence. Cette augmentation de l'accélération s'accompagne d'une petite augmentation du BD-Rate pour la plupart des séquences. Les résultats suggèrent donc que notre approche est beaucoup plus performante que l'approche proposée par Peixoto et Izquierdo (2012). Enfin, Fang *et al.* (2014) annoncent pour leurs parts des accélérations situées entre 1.07 et 1.33 pour un impact négligeable sur l'efficacité de codage. Les auteurs ne spécifient cependant pas le nombre de trames utilisées pour effectuer les tests. Quoi qu'il en soit, les performances annoncées suggèrent encore une fois que notre approche de transcodage est nettement plus performante. Comme mentionné précédemment, les performances des autres approches présentées dans l'état de l'art sont mesurées globalement, en incluant d'autres méthodes de transcodage. Nous reviendrons sur les performances de ces approches au prochain chapitre.

3.5 Résumé

Dans ce chapitre, nous avons proposé une approche de propagation du mouvement appliquée à du transcodage vidéo H.264 à HEVC. L'idée principale de cette approche est que les vecteurs de mouvement H.264 sont assez précis pour être réutilisés durant le transcodage vers la séquence HEVC. Cependant, ces vecteurs se propagent dans certains cas de manière différente dans HEVC que dans H.264. Nous avons validé ces deux hypothèses en analysant les vecteurs de mouvement sélectionnés par notre approche et en mesurant les performances de notre approche par rapport à un transcodage en cascade, qui est connu pour obtenir une efficacité de codage élevée. Par rapport aux méthodes de raffinement de mouvement présentées dans la littérature, l'approche proposée a l'avantage d'effectuer les calculs complexes au niveau des CTUs, plutôt qu'au niveau des PUs, et d'éliminer les calculs redondants en employant une liste des vecteurs de mouvement candidats unique pour l'ensemble d'une CTU. Les résultats ont montré que notre approche est plus rapide que les approches proposées dans la littérature, en plus de préserver l'efficacité de codage. Dans le prochain chapitre, nous montrerons que

notre approche a aussi pour avantage de produire des données (la liste des vecteurs de mouvement candidats et les erreurs précalculées) pouvant être exploitées par d'autres modules du transcodeur.

CHAPITRE 4

TRANSCODAGE RAPIDE BASÉ SUR UN PARCOURS POSTFIXE DE L'UNITÉ DE CODAGE ARBORESCENT

Dans ce chapitre, nous présentons une approche pour déterminer rapidement le mode HEVC à encoder pour une CTU donnée. L'approche que nous proposons repose sur un parcours postfixe de la CTU traitée. Cette approche se distingue donc des autres approches proposées dans la littérature, qui reposent plutôt sur un parcours préfixe (voir section 1.4.4). Notre approche est constituée de différentes méthodes qui ont pour objectif de réduire la complexité de l'évaluation des modes d'une CTU. Nous proposons notamment des méthodes pour réduire le nombre de modes à évaluer. De plus, nous proposons une méthode pour arrêter prématurément le traitement d'un mode lorsque celui-ci est jugé peu prometteur. Cette méthode est particulièrement efficace lorsqu'elle est appliquée sur un parcours postfixe de la CTU, comme c'est le cas pour l'approche de transcodage que nous proposons. Enfin, nous proposons une méthode qui détermine la structure de partitionnement initiale de la CTU à l'aide de l'information de mouvement (la liste des vecteurs de mouvement candidats et les erreurs de prédiction précalculées) générée par notre approche de propagation de mouvement, présentée au chapitre précédent.

Le reste de ce chapitre est structuré de la manière suivante. D'abord, la section 4.1 identifie et décrit certains problèmes associés à un parcours préfixe de la CTU traitée. Par la suite, la section 4.2 présente l'approche de transcodage proposée. Plus spécifiquement, elle illustre l'architecture de transcodage proposée et décrit ses principales méthodes. Enfin, la section 4.3 présente les résultats expérimentaux de notre approche pour différentes configurations. Ces résultats sont comparés aux résultats obtenus par un transcodeur en cascade et par notre transcodeur basé sur la propagation du mouvement présenté au chapitre précédent.

4.1 Problèmes d'un parcours préfixe d'une unité de codage arborescent

Comme mentionné au chapitre 2, plusieurs chercheurs ont proposé des approches pour réduire la complexité de la sélection du mode HEVC à encoder dans un contexte de transcodage H.264 à HEVC. Ces approches réutilisent l'information extraite de la séquence H.264 pour créer un sous-ensemble de modes HEVC à évaluer. Autrement dit, elles éliminent des modes jugés peu prometteurs avant de débuter l'évaluation des modes HEVC. En plus de ces techniques, certaines approches réutilisent de l'information HEVC générée (au fur et à mesure) durant le traitement d'une CTU, afin d'arrêter prématurément le traitement de certains modes HEVC lorsque certaines conditions sont satisfaites. Par exemple, Peixoto *et al.* (2014b) proposent d'arrêter prématurément le traitement d'une CU (c'est-à-dire, l'évaluation des PUs restantes et des sous-CUs) lorsque le meilleur mode HEVC actuel a un cout débit-distorsion en dessous d'un certain seuil (voir section 2.2.1.2.4).

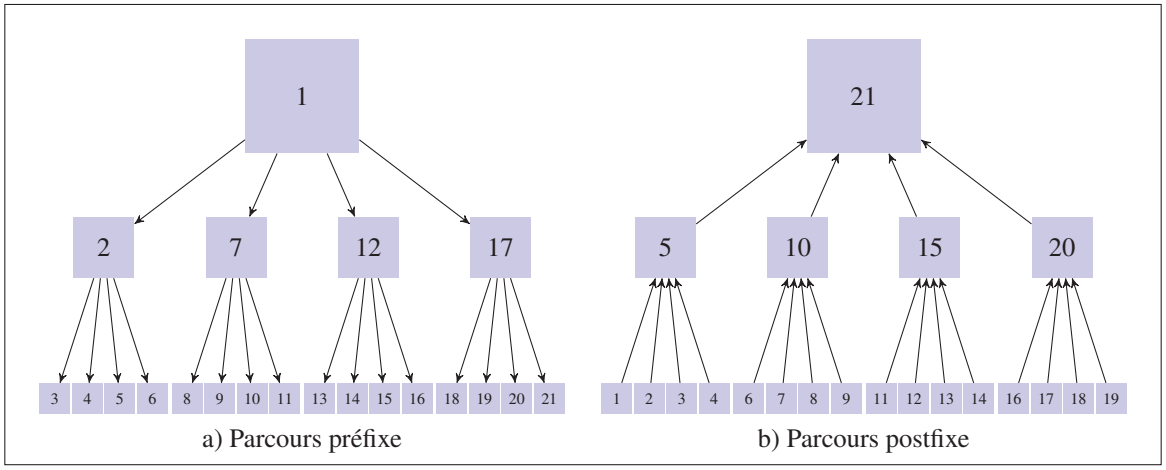


Figure 4.1 Exemples de parcours en préfixe et postfixe d'une unité de codage arborescent

À la limite de nos connaissances, toutes les approches proposées dans la littérature sont basées sur un parcours préfixe de la CTU traitée. La figure 4.1.a illustre un tel parcours ; où chaque index représente l'ordre de traitement d'une CU. Ainsi, le parcours traite d'abord la CU #1 en évaluant les PUs qui lui sont associées. Par la suite, le parcours se poursuit afin d'évaluer

récursivement les descendants de la CU #1. Pour réduire le nombre de CUs à traiter dans ce type de parcours, les transcodeurs vidéos doivent prendre jusqu'à deux types de décision : le premier type de décision consiste à déterminer si la CU courante doit être traitée ou, au contraire, si le parcours doit sauter directement au traitement des descendants. Cette décision se prend habituellement sur la base des informations H.264 disponibles. Le deuxième type de décision consiste à déterminer, après avoir traité une CU, si les descendants doivent être traités à leur tour. Cette décision s'effectue généralement sur la base de l'information HEVC disponible, comme discuté au paragraphe précédent. Dans les deux cas, les décisions sont complexes à prendre puisque plusieurs combinaisons de sous-CUs et PUs sont possibles, et n'importe quelle de ces combinaisons est susceptible d'obtenir un meilleur cout débit-distorsion que le meilleur cout débit-distorsion possible pour une CU donnée. Par exemple, plusieurs combinaisons de sous-CUs et de PUs sont possibles par rapport à la CU #1. Il est donc difficile de prédire si le traitement de cette CU doit être sauté et, de même, si ses descendants doivent être visités après l'avoir traité, étant donné le nombre de combinaisons possibles.

En plus de ces problèmes, le parcours préfixe ne permet pas de comparer le cout d'une CU donnée au cout d'une CU enfant, puisque les deux CUs ont des tailles différentes et ne sont donc pas directement en compétition. Par exemple, sur la figure 4.1.a le cout de la CU #2 ne peut pas être comparé directement avec le cout de la CU #3 pour déterminer le meilleur mode. On doit d'abord déterminer les couts des sous-CUs #3, #4, #5 et #6 ; puis comparer la sommation de ces couts avec le cout de la CU #2, qui a été obtenu précédemment. Ce type de parcours a donc pour inconvénient de traiter de manière indépendante des CUs ayant des niveaux de profondeurs différents. Autrement dit, lorsqu'un transcodeur décide de traiter les enfants d'une CU, le cout de la CU ne peut pas être exploité efficacement par les enfants (du moins, aucune solution proposée dans la littérature exploite cette information).

4.2 Approche proposée

Pour résoudre les différents problèmes reliés à un parcours préfixe, nous avons opté pour une approche de transcodage effectuant un parcours postfixe de la CTU traitée durant l'évaluation

des modes, comme le montre la figure 4.1.b. Ce type de parcours inverse les deux types de décisions prises pour un parcours préfixe. Ainsi, le problème qui consiste à déterminer si le traitement de la CU courante doit être sauté se transforme en un problème qui consiste à déterminer si le traitement des CU enfants doit être sauté. Autrement dit, le problème d'éliminer des partitions grossières peu prometteuses (généralement, situées à des niveaux peu profonds de l'arbre) selon l'information H.264, devient un problème qui consiste à éliminer des partitions fines peu prometteuses (généralement, situées à des niveaux profonds de l'arbre). Nous croyons que ce dernier problème est plus facile à traiter, puisque les partitions les plus fines de HEVC ont généralement un équivalent dans H.264, ce qui n'est pas le cas pour les partitions HEVC plus grandes que 16×16 pixels.

Pour sa part, le problème du parcours préfixe qui consiste à déterminer si les enfants d'une CU doivent être traités en connaissant l'information HEVC de la CU devient, pour un parcours en postfixe, un problème qui consiste à déterminer si une CU doit être traitée en connaissant l'information HEVC de ses enfants (ses sous-CUs). Par exemple, ce dernier problème peut consister à déterminer si la CU #5 de la figure 4.1.b doit être traitée en connaissant les résultats des CUs enfants (les CUs #1, #2, #3 et #4). C'est un problème plus simple que son équivalent dans un parcours préfixe pour deux raisons. Premièrement, le meilleur mode actuel est en compétition avec un seul mode à la fois (une PU de la CU parent). Deuxièmement, seuls des modes de mêmes tailles sont comparés, puisque les meilleurs couts des sous-CUs (les CUs #1, #2, #3 et #4) peuvent être combinés en un seul cout qui, par la suite, peut être comparé au cout de la CU parent (la CU #5).

Pour déterminer la structure de partitionnement initiale de la CTU, nous proposons deux méthodes alternatives qui déterminent si les enfants de la CU courante doivent être visités ou non. La première est une méthode structurelle qui utilise les modes H.264 pour déterminer si la CU courante doit être divisée ou non. Cette méthode préserve l'efficacité de codage, mais a pour inconvénient de surestimer la profondeur de l'arbre, ce qui a pour conséquence de limiter les gains de vitesse (par rapport à un transcodeur en cascade). La seconde méthode réutilise la liste des vecteurs de mouvement candidats et les erreurs de prédiction précalculées, générées

par notre approche de propagation du mouvement durant l'initialisation de la CTU traitée, pour calculer des bornes inférieures du cout du mouvement. Pour chaque CU, deux bornes sont calculées : l'une qui considère la CU comme étant non-divisée, et l'autre, la considérant comme divisée. Ces deux bornes sont par la suite utilisées pour déterminer si les enfants de la CU courante doivent être traités, comme nous le verrons plus loin. Cette méthode, basée sur l'analyse du mouvement, affecte peu l'efficacité de codage et a pour avantage d'éliminer davantage de modes HEVC que l'autre méthode proposée.

Pour réduire davantage le nombre de modes HEVC traités, l'approche que nous proposons utilise une méthode peu complexe pour pré-évaluer l'efficacité de codage d'un mode. Si le mode est jugé peu prometteur, son traitement est alors arrêté prématurément. Au contraire, si le mode est jugé prometteur, son traitement se poursuit afin d'évaluer son cout débit-distorsion, une étape très complexe en calculs. Enfin, quelques heuristiques simples sont utilisées pour éliminer des modes selon l'information H.264. Les prochaines sections décrivent plus en détail les différents aspects de notre approche de transcodage.

4.2.1 Architecture de transcodage proposée

La figure 4.2 montre une vue sommaire de l'architecture de transcodage proposée. Cette architecture combine les méthodes que nous proposons pour accélérer l'estimation de mouvement (notre approche de propagation du mouvement) ainsi que les méthodes que nous proposons pour accélérer le traitement des modes. Le transcodeur proposé décode d'abord la trame H.264 courante afin d'y extraire les modes, les vecteurs de mouvement, les données résiduelles et les pixels de la trame reconstruite. Ces informations sont par la suite transférées au système de transcodage.

Durant l'initialisation d'une CTU, on crée la liste des vecteurs de mouvement candidats et on précalcule les erreurs de prédiction comme décrit au chapitre précédent. Par la suite, ces données sont réutilisées par le module de décision rapide du mode HEVC à encoder afin de calculer des bornes inférieures pour le cout. Ces bornes sont utilisées pour déterminer la

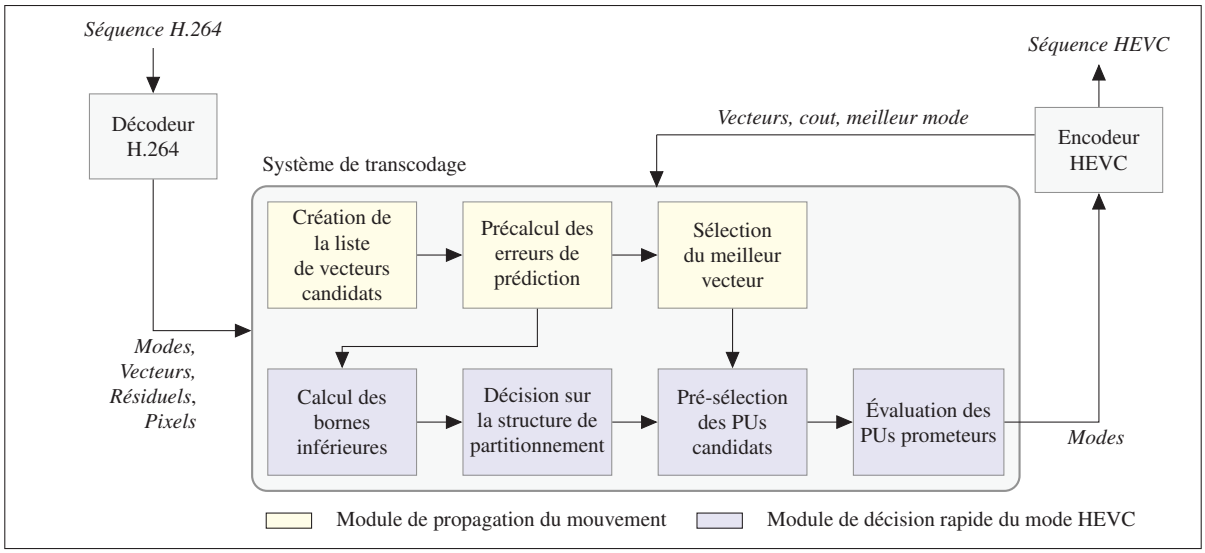


Figure 4.2 Architecture de transcodage proposée

structure de partitionnement initiale de la CTU traitée. Par la suite, le module de décision présélectionne, pour chaque CU traitée, une liste de PUs candidats. Les candidats jugés prometteurs, par rapport au meilleur mode HEVC actuel, sont évalués entièrement afin d'obtenir leur cout débit-distorsion.

4.2.2 Description de l'approche proposée

La fonction TRAITER-CU (Algorithme 4.1) constitue la base de l'approche proposée. Nous présentons ici une vue sommaire de cet algorithme et décrivons ses principaux aspects plus en détail dans les prochaines sous-sections.

La première partie (lignes 3 à 10) de l'algorithme traite de manière récursive l'ensemble des descendants de la CU courante, notée $C_{i,j}$ (voir section 1.3.1.1), selon un parcours postfixe. La fonction DIVISER-CU (appelée à la ligne 4) détermine si les enfants du CU doivent être visités, comme décrit par la section 4.2.4. Cette fonction permet d'éliminer les structures de partitionnement jugées trop fines pour produire un codage efficace.

La seconde partie (lignes 11 à 18) de l'algorithme traite pour sa part la CU courante. La fonction MODES-CANDIDATS (appelée à la ligne 12) retourne une liste de modes PU à évaluer, comme

Algorithme 4.1 Traitement des modes d'une CU

```
1  procedure TRAITER-CU(C_{i,j})
2      J_rd[C_{i,j}] ← J_motion[C_{i,j}] ← ∞
3      ▷ Traiter récursivement les sous-CUs
4      si DIVISER-CU(C_{i,j}) alors
5          J_rd[C_{i,j}] ← λ_mode · R_Divisé
6          J_pm[C_{i,j}] ← λ_pred · R_Divisé
7          pour k ← 0 à 3 faire
8              TRAITER-CU(C_{i+1,4j+k})
9              J_rd[C_{i,j}] ← J_rd[C_{i,j}] + J_rd[C_{i+1,4j+k}]
10             J_pm[C_{i,j}] ← J_pm[C_{i,j}] + J_pm[C_{i+1,4j+k}]
11     ▷ Traiter les PUs inter de la CU
12     pour chaque m dans MODES-CANDIDATS(C_{i,j}) faire
13         J_pm[m] ← PU-COUT-PM(C_{i,j},m)
14         si (J_pm[m] < (J_pm[C_{i,j}] + T)  alors
15             J_rd[m] ← PU-COUT-RD(C_{i,j},m)
16             si J_rd[m] < J_rd[C_{i,j}] alors
17                 J_rd[C_{i,j}] ← J_rd[m]
18                 J_pm[C_{i,j}] ← J_pm[m]
```

discuté à la section 4.2.5. Chaque mode PU est évalué en deux étapes, comme décrit à la section 4.2.3. La première étape (ligne 13) estime le cout du mode à l'aide d'une fonction cout peu complexe, notée J_{pm} et calculée par la fonction PU-COUT-PM. Si le mode candidat est jugé prometteur, la seconde étape (ligne 15) calcule le cout débit-distorsion, noté J_{rd} et retourné par la fonction PU-COUT-RD.

Dans le parcours postfixe proposé, la condition de la ligne 14 est très efficace pour réduire la complexité relative à l'évaluation des modes, puisque le cout J_{pm} de la meilleure combinaison de sous-CUs (obtenue durant la première partie de l'algorithme (ligne 3 à 10)) peut être comparé avec le cout J_{pm} d'une PU appartenant à la CU courante.

4.2.3 Calcul du cout débit-distorsion

L'algorithme de propagation de mouvement présenté au chapitre 3 sélectionne le meilleur vecteur de mouvement d'une partition PU inter à l'aide d'une fonction cout peu complexe basée sur l'erreur de prédiction (voir les équations 3.3 et 3.5). Cependant, le traitement d'un mode

inter reste complexe dans le HM, puisque le calcul du cout débit-distorsion, noté J_{rd}, nécessite l'exécution d'opérations très complexes, tel qu'un traitement récursif des TUs impliquant notamment un codage entropique complet (Bossen *et al.* (2012)).

Pour une PU intra, une partie importante de la complexité est aussi due au calcul du cout débit-distorsion. Cependant, l'algorithme implémenté dans (la version originale de) le HM réduit cette complexité en présélectionnant un sous-ensemble de modes de prédiction à évaluer par la fonction cout débit-distorsion. Par exemple, pour un bloc de 8×8 pixels, le HM sélectionne parmi 35 candidats, les 3 candidats obtenant le plus faible cout définit par l'équation suivante :

$$J_{\text{pm}} = (\text{SATD}_{\text{luma}}) + \lambda_{\text{pred}} \cdot R_{\text{pred}}\,, \tag{4.1}$$

où SATD est la somme des différences absolues transformées des erreurs de prédiction, λ_{pred} est égal à $\sqrt{\lambda_{\text{mode}}}$ et R_{pred} est le nombre de bits nécessaires au codage de l'information de prédiction (le mode de prédiction). À noter que la somme des différences absolues transformées est calculée de la même manière que décrite à la section 1.4.1, c'est-à-dire, en partitionnant d'abord la région traitée en blocs de 8×8 pixels (ou 4×4 pour les régions plus petites), puis en appliquant la transformée de Hadamard sur chacun de ces blocs.

Le HM est donc en mesure de réduire le nombre de candidats à évaluer par la fonction cout débit-distorsion. Cependant, elle présente deux limitations importantes. Premièrement, les modes de prédiction intra à évaluer sont déterminés uniquement sur la base de l'information de la PU traitée. Ainsi, l'information des autres modes testés jusqu'à maintenant n'est pas considérée par le HM. Deuxièmement, seuls les modes intra exploitent la fonction J_{pm} pour réduire le nombre de modes à évaluer par la fonction cout débit-distorsion. Autrement dit, le HM n'exploite pas le J_{pm} pour réduire le nombre de modes inter à tester.

Pour résoudre ces deux problèmes, la méthode que nous proposons divise le traitement d'un mode en deux parties, quel que soit le type de prédiction. Ces deux parties sont représentées respectivement par les fonctions PU-COUT-PM (ligne 13 de l'Algorithme 4.1) et PU-COUT-

RD (ligne 15 de l'Algorithme 4.1). La première partie calcule le cout J_{pm}, peu complexe à obtenir, alors que la seconde partie calcule le cout J_{RD}, nettement plus complexe à calculer, pour un sous-ensemble de candidats prometteurs. Pour un mode inter, le J_{pm} correspond au cout retourné par l'algorithme de propagation du mouvement pour une partition donnée. Pour un mode intra, le J_{pm} correspond au meilleur cout calculé durant la présélection des modes de prédiction intra à tester.

4.2.3.1 Arrêt prématuré du traitement d'un mode

Une fois le cout $J_{\text{pm}}^{\text{Courant}}$ obtenu, la fonction PU-COUT-RD est seulement appelée lorsque le critère suivant est vrai (sinon, l'évaluation du mode est arrêtée prématurément) :

$$J_{\text{pm}}^{\text{Courant}} < (J_{\text{pm}}^{\text{Meilleur}} + T), \text{où } T \geq 0, \tag{4.2}$$

où $J_{\text{pm}}^{\text{Courant}}$ est le cout de faible complexité du mode traité et $J_{\text{pm}}^{\text{Meilleur}}$ est le cout de faible complexité du meilleur mode actuel. Cette fonction de décision est basée sur l'idée que le mode obtenant le meilleur cout J_{pm}, parmi deux modes concurrents, est généralement aussi le mode obtenant le meilleur cout J_{RD}. Le seuil T est utilisé pour contrôler le compromis entre l'efficacité de codage et la complexité de calcul du transcodeur. Ce paramètre doit toujours être positif ou nul. Son augmentation permet de calculer le cout débit-distorsion pour davantage de modes, au prix d'une complexité plus grande.

La figure 4.3 montre de quelle manière le seuil T influence : 1) le ratio du nombre de modes pour lesquels le cout J_{RD} est évalué par rapport au nombre total de modes ; 2) l'augmentation de l'erreur débit-distorsion lorsqu'un mode n'est pas testé, mais a un meilleur cout J_{RD}. À titre d'information, nous avons fixé ce paramètre à $3\times\lambda_{\text{pred}}$. Cette valeur a été déterminée de manière empirique.

L'encodeur x265 exploite aussi la fonction J_{RD} pour éliminer le calcul débit-distorsion de plusieurs modes (MulticoreWare (2016)). Notamment, l'un de ses algorithmes calcule le cout

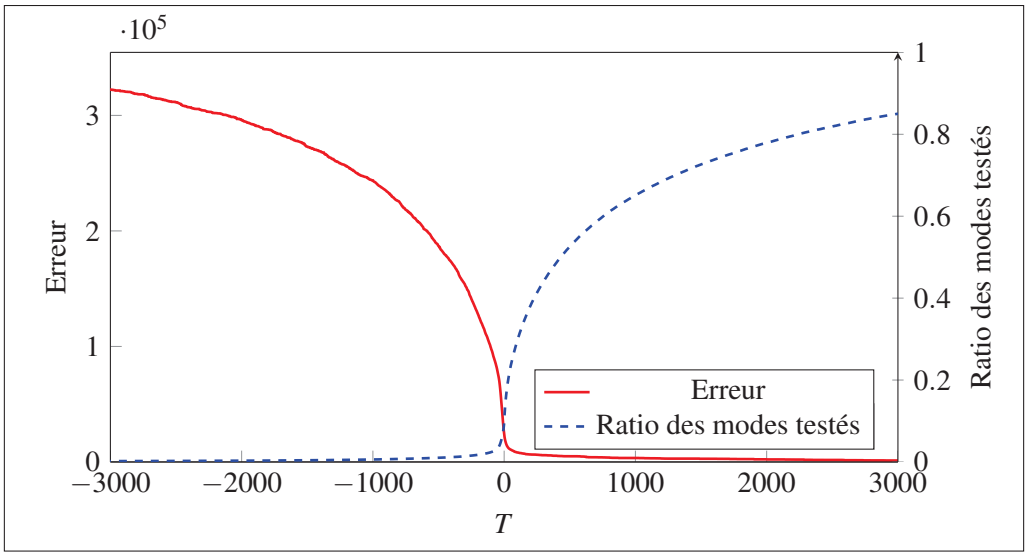


Figure 4.3 Impact du seuil T sur le nombre de modes testés et l'efficacité de codage : résultats pour la séquence Racehorse encodée, puis transcodée avec QP = 22

J_{pm} de plusieurs PUs inter associées à la CU courante, afin de déterminer la PUs inter qui sera évaluée par la fonction cout débit-distorsion. Cette stratégie permet de réduire la complexité de l'encodage par rapport au HM. Cependant, comme ce dernier, elle ne permet pas d'exploiter l'information des autres CUs pour éliminer davantage de modes.

Notre critère d'arrêt prématuré (ligne 14 de l'Algorithme 4.1) a l'avantage d'être applicable lorsque les modes comparés ont la même taille. Dans un parcours préfixe (le type de parcours implémenté par le HM), l'utilisation de ce critère est limitée puisqu'il est impossible de comparer une CU avec une sous-CU. Cependant, le parcours postfixe que nous proposons permet d'appliquer ce critère en combinant le cout de 4 sous-CUs (le J_{pm} obtenu par le traitement récursif de l'Algorithme 4.1) et le candidat PU à traiter (le J_{pm} calculé à la ligne 13 de l'Algorithme 4.1), ce qui constitue un avantage important par rapport à la méthode implémentée dans l'encodeur x265.

En plus de ce critère, nous proposons d'arrêter prématurément le traitement d'un mode PU lorsque ce dernier contient deux partitions ayant le même vecteur de mouvement. Cette condition d'arrêt suppose que la PU inter couvrant cette région, mais constitué d'une seule

partition, va encoder ce même vecteur de mouvement en nécessitant un cout plus petit (puisque sa structure de partitionnement et de prédiction est plus simple pour un même modèle de prédiction).

4.2.4 Partitionnement initial d'une unité de codage arborescent

La première étape (ligne 4) de la fonction TRAITER-CU (Algorithme 4.1) consiste à déterminer si la CU traitée doit être initialement divisée, c'est-à-dire, si ses enfants (les sous-CUs) doivent être visités. La fonction DIVISER-CU (voir Algorithme 4.2) a donc pour rôle de déterminer si la division de la CU est susceptible d'améliorer l'efficacité de codage (par rapport à une non-division). Si c'est le cas, elle retourne vrai. Inversement, si elle juge que la division de la CU en sous-CUs n'est pas susceptible d'améliorer l'efficacité de codage, elle retourne faux. En appliquant la fonction DIVISER-CU de manière récursive sur la CTU traitée, on obtient un arbre élagué constituant la structure de partitionnement initiale. Ce partitionnement représente le partitionnement le plus fin qui peut être atteint pour la CTU traitée selon les décisions retournées par la fonction DIVISER-CU. En pratique, le partitionnement final (celui qui sera encodé) est rarement aussi fin, puisque le parcours de l'arbre a souvent pour effet d'éliminer des divisions (des sous-CUs) qui réduisent l'efficacité de codage.

L'algorithme de la fonction DIVISER-CU est constitué de deux méthodes de partitionnement alternatives : une méthode de partitionnement quaternaire structurel (PQS) (lignes 8-12) et une méthode de partitionnement quaternaire basé le mouvement (PQM) (lignes 14-20). La méthode de PQS est une méthode simple qui effectue une mise en correspondance entre la structure de partitionnement H.264 et la décision de division, comme décrit à la section 4.2.4.1. Pour sa part, la méthode de PQM, décrite à la section 4.2.4.2, utilise deux bornes inférieures du cout du mouvement pour déterminer si la CU doit être divisée ou pas. Comme décrit à la section 4.2.4.3, ces bornes sont calculées en réutilisant l'information de mouvement générée par notre approche de propagation de mouvement durant l'initialisation d'une CTU (voir Chapitre 3). Elles sont calculées en considérant la CU traitée respectivement comme étant non divisée et divisée. Cette dernière méthode élague généralement davantage l'arbre de partitionnement initial, surtout

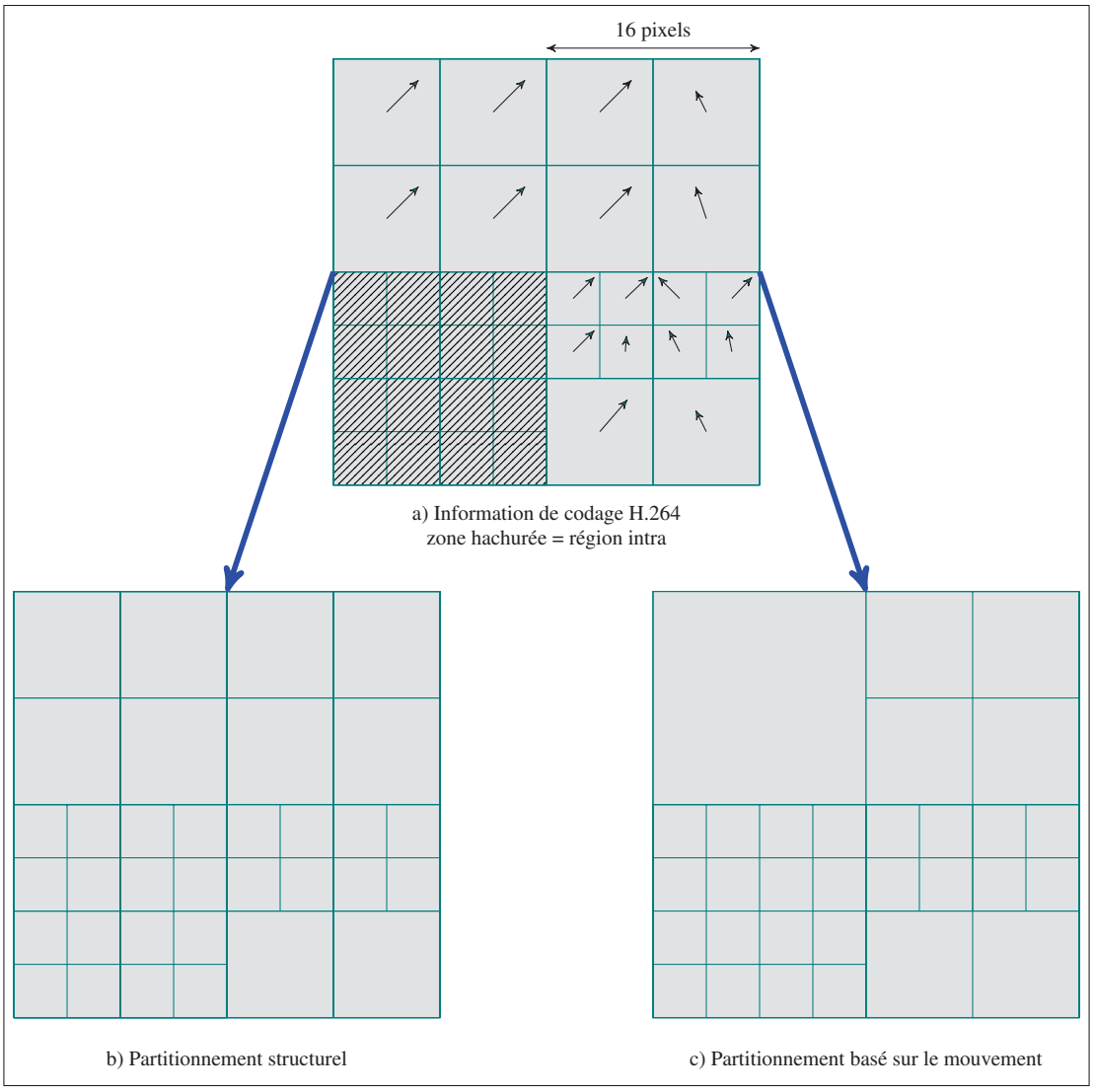


Figure 4.4 Exemple de partitionnement maximal d'une unité de codage arborescent de 32×32 pixels

dans les régions où le mouvement H.264 est continu. À titre d'exemple, la figure 4.4 montre la structure de partitionnement maximale (le partitionnement qui correspond à l'arbre élagué) obtenue respectivement par les méthodes de PQS et PQM. Comme nous le verrons plus loin, la méthode PQS est automatiquement utilisée lorsque la région H.264 correspondante est une région intra, même lorsque la méthode PQM est activée.

Algorithme 4.2 Division d'une CU

```
1  fonction DIVISER-CU(C_{i,j})
2      taille ← TAILLE-CU(C_{i,j})
3      PS ← Un paramètre défini dans le fichier de configuration
4      intra ← H.264-CONTIENTPREDICTIONINTRA(C_{i,j})
5      si taille = 8 alors                          ▷ Si la profondeur maximale est atteinte
6          retourner FAUX
7      si intra = VRAI ou PS = VRAI ou taille = 16 alors
8          ▷ Décision de partitionnement structurel
9          si size = 16 et (PROFONDEUR-H.264(C_{i,j})=0) alors
10             retourner FAUX
11         sinon
12             retourner VRAI
13     sinon
14         ▷ Décision de partitionnement basée sur le mouvement
15         Ĵ_NonDivisé ← COUT-MIN-CU(C_{i,j},i,i,i)
16         Ĵ_Divisé ← COUT-MIN-CU(C_{i,j},i+1,3,i)
17         si Ĵ_NonDivisé < Ĵ_Divisé alors
18             retourner FAUX
19         sinon
20             retourner VRAI
```

4.2.4.1 Méthode de partitionnement structurel

La méthode PQS (lignes 8-12) crée une mise en correspondance directe entre la structure de partitionnement H.264 et la décision de diviser ou pas la CU traitée. Cette méthode est basée sur l'hypothèse que le partitionnement HEVC est généralement similaire au partitionnement H.264, mais rarement plus fin que ce dernier. Ainsi, les CUs 64×64 et 32×32 sont automatiquement divisées par cette méthode puisque les plus grandes partitions de H.264 couvrent des régions de 16×16 pixels. Pour les CUs 16×16, la méthode analyse le partitionnement H.264 à l'aide de la fonction PROFONDEUR-H.264. Cette dernière retourne 1 lorsque la plus petite partition H.264 est égale ou inférieure à 8×8 pixels. Sinon, la fonction retourne 0 (lorsque la plus petite partition est égale à 16×16, 16×8 ou 8×16). Ainsi, une CU 16×16 n'est pas divisée lorsque la fonction PROFONDEUR-H.264 retourne 0, puisque la CU est alors en mesure de reproduire la même structure de partitionnement que celle de H.264.

4.2.4.2 Méthode de partitionnement basée le mouvement

Comme nous le verrons plus loin dans les expériences que nous avons menées, la méthode de PQS précédemment décrite est en mesure d'atteindre une haute efficacité de codage. Cependant, cette méthode simple tend à surestimer la profondeur de partitionnement HEVC, puisqu'elle divise toujours les CUs 64×64 et 32×32. Pour résoudre ce problème, nous avons développé la méthode de PQM, une méthode basée sur l'information de mouvement générée par notre approche de propagation de mouvement.

L'idée principale de cette méthode est de déterminer si la division CU en sous-CUs est susceptible d'améliorer le modèle de prédiction du mouvement (de réduire le cout J_{pm}). À ce stade, les vecteurs de mouvement prédits (les AMVPs) sont inconnus puisque l'encodeur HEVC n'a pris aucune décision sur les modes HEVC à encoder, il est donc impossible de connaitre le cout du mouvement exact. Pour contourner ce problème, nous calculons deux bornes inférieures du cout du mouvement : la borne $\widehat{J}_{\text{NonDivisé}}$ (ligne 15), qui considère l'option de ne pas diviser la CU traitée en sous-CUs ; et la borne $\widehat{J}_{\text{Divisé}}$, qui considère l'option de diviser la CU traitée. Ces deux bornes estiment le cout réel du mouvement. La différence entre ces deux bornes ($\widehat{J}_{\text{NonDivisé}} - \widehat{J}_{\text{Divisé}}$) est notée $\widehat{J}_{\text{Diff}}$.

Comme le montre la figure 4.5, la différence $\widehat{J}_{\text{Diff}}$ est fortement corrélée avec la différence entre les vrais couts du mouvement (ceux obtenus après l'encodage complet d'une CTU), notée J_{Diff}. Les observations en rouge montrent les cas pour lesquelles la division de la CTU traitée en sous-CUs permet de réduire le cout J_{pm}, donc d'améliorer la prédiction. Inversement, les observations en bleu illustrent les cas où il est plus avantageux de ne pas diviser la CU traitée en sous CUs. On remarque sur la figure que la différence $\widehat{J}_{\text{Diff}}$ est généralement inférieure à la différence J_{Diff} (les observations sont généralement situées en dessous de la ligne noire pointillée). Notre méthode de prédiction a donc tendance à sous-estimer J_{Diff}. Sur la base de ce fait, la méthode PQM bloque la division de la CU traitée en sous-CUs lorsque $\widehat{J}_{\text{Diff}} < 0$. Cette méthode de classification binaire peut résulter en 4 cas différents :

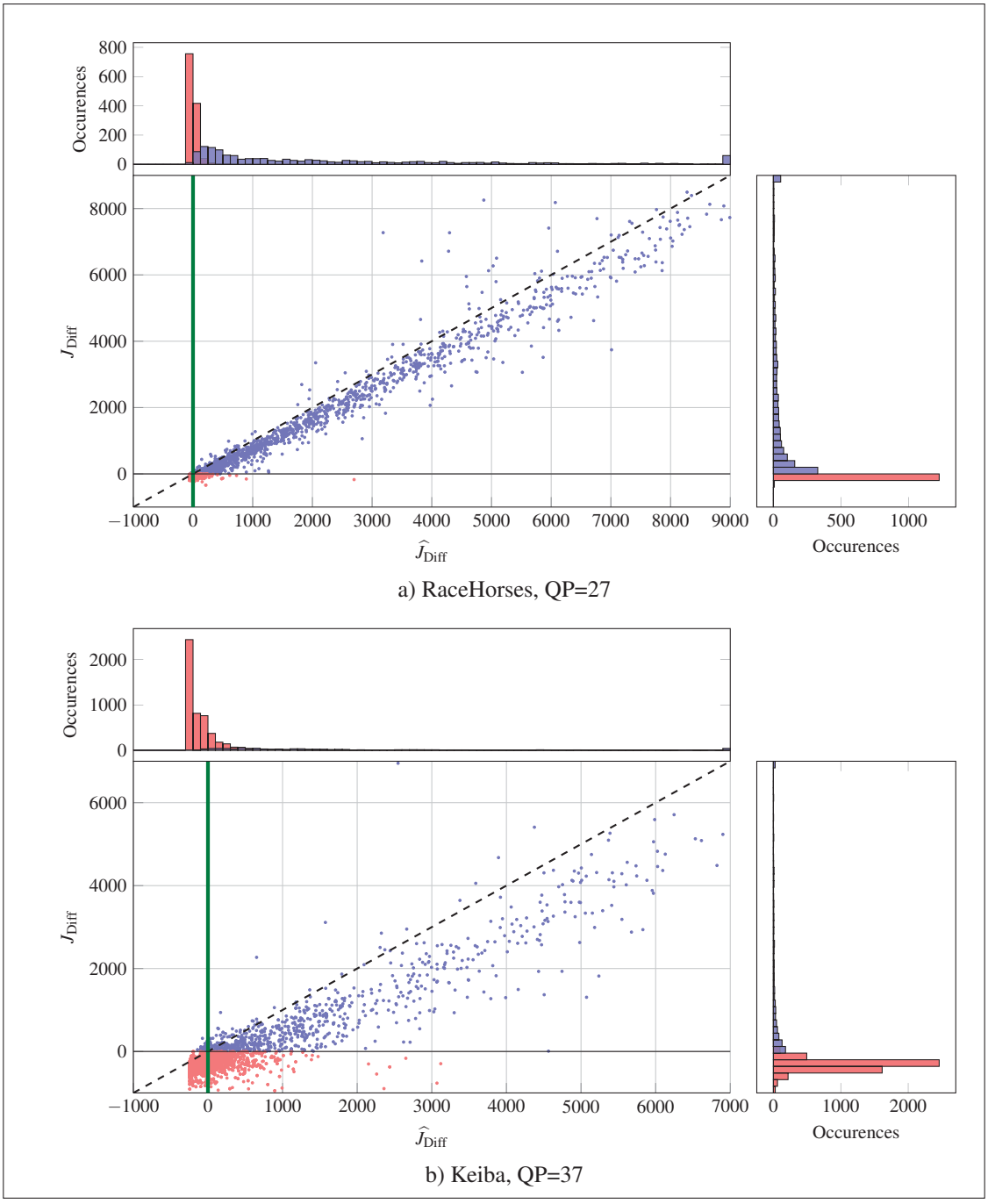


Figure 4.5 Relation entre le cout du mouvement réel et le cout estimé : exemples de relations linéaires entre la différence $\widehat{J}_{\text{Diff}}$, obtenue à l'aide de la méthode que nous proposons, et la différence J_{Diff}, la différence réelle obtenue durant le transcodage ; la ligne verte illustre la frontière de décision

- Vrai positif ($J_{\text{Diff}} > 0, \widehat{J}_{\text{Diff}} > 0$) : la méthode prédit que la CTU doit être divisée et c'est effectivement l'option qui permet d'obtenir le plus petit J_{pm}.
- Faux positif ($J_{\text{Diff}} < 0, \widehat{J}_{\text{Diff}} > 0$) : la méthode prédit que la CTU doit être divisée, cependant le plus petit J_{pm} est obtenu lorsque la CTU n'est pas divisée. Cette erreur a pour effet d'augmenter la complexité de l'approche, sans affecter l'efficacité de codage, puisque notre approche traite tout de même le cas où la CU n'est pas divisée.
- Vrai négatif ($J_{\text{Diff}} < 0, \widehat{J}_{\text{Diff}} < 0$) : la méthode prédit que la CTU ne doit pas être divisée, et c'est effectivement le cas.
- Faux négatif ($J_{\text{Diff}} > 0, \widehat{J}_{\text{Diff}} < 0$) : la méthode prédit que la CTU ne doit pas être divisée, cependant le plus petit J_{pm} est obtenu lorsque la CTU est divisée. Cette erreur a pour effet de réduire l'efficacité de codage du transcodeur, puisque notre approche ne teste pas la division CTU dans un tel cas. On remarque cependant sur les nuages de points que la perte d'efficacité de codage (l'augmentation du J_{pm}) est généralement faible.

À noter qu'il est envisageable de déplacer la frontière de décision afin de gérer le compromis entre l'efficacité de codage et la complexité de l'approche de transcodage. Ainsi, un déplacement vers la droite de cette frontière augmenterait l'efficacité de codage au prix d'une complexité plus grande.

4.2.4.3 Calcul des bornes inférieures du cout du mouvement

Les bornes inférieures du cout du mouvement sont calculées par la fonction récursive COUT-MIN-CU (Algorithme 4.3). Les paramètres d'entrée de cette fonction sont : la CU traitée, notée $C_{i,j}$; le plus petit et le plus grand niveau de profondeur supportés, respectivement définis par l et u ; et le niveau de profondeur actuel, noté c. La ligne 15 de la fonction DIVISER-CU (Algorithme 4.2) calcule la borne inférieure de $C_{i,j}$, en considérant l'option de ne pas diviser la CU traitée en sous-CUs (la borne inférieure pour le niveau de profondeur i). Tandis que la ligne 16 calcule la borne inférieure en considérant l'option de diviser la CU traitée en sous-CUs (la borne inférieure pour les niveaux de profondeur allant de $(i+1)$ à u). Dans nos expériences,

Algorithme 4.3 Calcul de la borne inférieure d'une CU

```
1  fonction COUT-MIN-CU(C_{i,j}, l, u, c)
2      J_min ← ∞
3      si c ≥ l alors
4          ▷ Traiter les PUs inter de la CU C_{i,j}
5          pour chaque mode dans INTER-MODES(C_{i,j}) faire
6              J ← λ_pred · mode.R_min
7              pour chaque partition dans mode faire
8                  (x,y) ← partition.position
9                  (M,N) ← partition.taille
10                 J ← J + min(e^{4N×4M}_{x,y,k} | k = 1..K)
11             si J < J_min alors
12                 J_min ← J
13     si c < u alors
14         ▷ Traiter récursivement les sous-CUs
15         c ← c + 1
16         J ← λ_pred · mode.R_Divisé
17         pour k ← 0 à 3 faire
18             J ← J + CU-COUT-MIN(C_{i+1,4j+k}, l, u, c)
19         si J < J_min alors
20             J_min ← J
21     retourner J_min
```

u est égal à 3. À ce niveau de profondeur, les CUs couvrent des régions de 8×8 pixels, ce qui correspond à la plus petite taille de CU supportée par HEVC.

La première partie (lignes 4 à 12) de la fonction COUT-MIN-CU calcule la borne inférieure pour chaque PU inter de la CU traitée, puis sélectionne la plus petite borne comme J_{min}. La seconde partie (lignes 14 à 20) calcule récursivement les bornes inférieures pour les sous-CUs et met à jour la borne J_{min} au besoin. Le calcul d'une borne est effectué en réutilisant les erreurs de prédiction qui ont été précalculées par notre algorithme de propagation de mouvement durant l'initialisation d'une CTU (voir Chapitre 3). Étant donné que les CUs sont partiellement ou totalement inconnues à ce stade, le calcul d'une borne suppose que les prédicteurs du mouvement sont inconnus et, par conséquent, le cout du mouvement n'est pas considéré dans le calcul. Cependant, une pénalité est ajoutée à la borne en fonction de la structure de partitionnement. Ainsi, plus le partitionnement est fin et plus la pénalité est

élevée. Cette pénalité permet de tenir compte du fait que le cout associé à la signalisation du mouvement et du partitionnement augmente selon la complexité du partitionnement (augmente selon le nombre de partitions).

La pénalité correspond à une approximation du nombre de bits minimal nécessaire à l'encodage de l'information de prédiction (en supposant un cout nul pour l'information du mouvement). Ainsi, la borne inférieure d'une PU (lignes 6 à 12) est calculée en sommant les minimisations de l'équation 3.2 (ligne 10) pour chacune des partitions, en plus d'ajouter des bits de pénalité, notés $mode.R_{min}$, multipliés par λ_{pred} (ligne 6). La borne inférieure des sous-CUs est calculée d'une manière similaire en sommant les bornes inférieures de 4 sous-CUs et en ajoutant une pénalité pour la division de la CU en sous-CUs.

Nous avons déterminé les pénalités de manière expérimentale avec l'objectif de préserver l'efficacité de codage, c'est-à-dire, en sélectionnant des pénalités qui surestiment très rarement le cout du mouvement réel, afin de ne pas rejeter une structure de partitionnement susceptible d'être la meilleure. Ainsi, 3 bits de pénalité sont utilisés pour une PU ayant une seule partition ; alors que 8 bits de pénalité sont utilisés pour une PU ayant deux partitions. Enfin, une pénalité de 1 bit, notée $mode.R_{\text{Divisé}}$, est utilisée lorsque la CU est sous-divisée en sous-CUs.

Enfin, la complexité de l'algorithme proposé (la fonction COUT-MIN-CU) est faible puisque le calcul de J_{min} réutilise la liste des vecteurs de mouvement candidats et les erreurs de prédiction qui ont été précalculées par l'algorithme de propagation de mouvement durant l'initialisation de la CTU traitée.

4.2.4.4 Sélection de la méthode de partitionnement

La sélection de la méthode de décision à appliquer dépend de l'information H.264 et de la taille du CU traitée. Basée sur nos résultats expérimentaux, la méthode de PQS est plus efficace, et doit être activée, lorsqu'une des conditions suivantes est vraie (ligne 7 de l'Algorithme 4.2) :

- au moins un mode intra est présent dans la région H.264 correspondante ;
- la CU a une taille de 16×16.

Pour la première condition, la méthode de PQM est inefficace parce qu'elle considère seulement l'information de mouvement ; elle peut donc difficilement prédire le partitionnement, puisque la région traitée est susceptible d'être encodée à l'aide d'au moins un mode intra. Pour le second cas, la méthode de PQS est légèrement plus efficace parce que la structure de partitionnement HEVC est alors fortement corrélée au mode H.264 qui lui est associé. Enfin, la méthode PQS est aussi activée lorsque le drapeau *PS* (pour partitionnement structurel) est fixé à vraie (c'est le cas lorsqu'on veut évaluer les performances de cette méthode uniquement). Pour les autres cas, la méthode PQM est activée.

4.2.5 Sélection des modes candidats

La fonction MODES-CANDIDATS appelée par la fonction TRAITER-CU retourne une liste de modes PU à tester pour la CU traitée. Ces modes sont retournés dans l'ordre suivant : modes inter ; des partitions les plus fines aux partitions les plus grossières ; les modes merge/skip, triés dans un ordre ascendant selon leur cout J_{pm} ; et les modes intra, des partitions les plus grossières aux plus fines. Tous les modes inter sont désactivés lorsque la région H.264 correspondante est seulement constituée de modes intra. De manière similaire, les partitions HEVC plus petites que la partition H.264 correspondante sont désactivés lorsque la région H.264 ne contient aucune donnée résiduelle. En effet, les données résiduelles constituent un bon indice sur l'efficacité de la prédiction. Lorsqu'aucune donnée n'a été encodée, la prédiction est considérée comme bonne.

Finalement, nous définissions un drapeau, appelé décision accélérée du mode (DAM). Lorsque ce dernier est activé, d'autres modes peuvent être retirés selon les règles suivantes :

- les modes intra sont désactivés lorsque la région H.264 correspondante n'a aucun macrobloc intra, tel que proposé par d'autres auteurs (Peixoto et Izquierdo (2012); Shen *et al.* (2013)) ;

- les partitions asymétriques sont désactivées puisqu'elles sont très complexes, tout en ayant une faible influence sur l'efficacité de codage, comme l'ont observé d'autres auteurs (Shen *et al.* (2013); Luo *et al.* (2014); Xing *et al.* (2013)).

L'utilisation de cette méthode DAM est uniquement recommandée lorsqu'une accélération supplémentaire, en addition à celle obtenue par les autres méthodes que nous proposons, est nécessaire. Cette méthode a pour effet de réduire l'efficacité de codage.

4.3 Expérimentation

Dans cette section, nous présentons les expériences qui ont été menées afin d'évaluer les performances des approches et des méthodes de transcodage que nous avons proposées dans cette thèse. À l'exception des combinaisons de méthodes testées, la méthodologie employée est la même que celle décrite au chapitre précédent, à la section 3.4.1. La section 4.3.1 compare les performances de notre approche de transcodage, pour différentes combinaisons de méthodes, avec une approche de transcodage en cascade (un transcodage complet). La section 4.3.2 présente pour sa part une comparaison de notre approche avec les approches de l'état de l'art.

4.3.1 Comparaison avec un transcodage en cascade

Les tableaux 4.1 à 4.3 présentent les résultats expérimentaux du transcodeur proposé, relatives à un transcodage en cascade, respectivement pour des résolutions de 1920×1080, 832×480 et 416×240 pixels. La configuration que nous privilégions, pour le compromis qu'elle offre entre l'efficacité de codage et le temps de transcodage, est mise en gras dans les tableaux. Les deux transcodeurs comparés utilisent une seule trame de référence. Des résultats expérimentaux supplémentaires sont présentés à l'Annexe II, d'une part, pour comparer l'approche proposée, dont l'implémentation supporte une seule trame de référence, avec l'approche en cascade configurée pour supporter quatre trames de référence et d'autre part, pour analyser l'impact de la version parallèle de notre approche sur les accélérations obtenues.

Tableau 4.1 Performances du transcodeur proposé pour des séquences 1920×1080 (classe B) et quatre combinaisons de méthodes de transcodage pour une configuration avec une seule trame de référence

Séquence	Configuration	Accélération (par QP) 22	27	32	37	Moy.	BD-Rate(%)
BasketballDrive	PML seulement	1.40	1.48	1.52	1.54	1.49	-1.57
	PQS	4.92	5.12	5.09	5.04	5.04	3.02
	PQM	**5.02**	**5.44**	**5.70**	**5.93**	**5.52**	**2.99**
	PQM-DAM	5.66	6.15	6.39	6.42	6.16	5.80
BQTerrace	PML seulement	1.30	1.38	1.47	1.48	1.41	-0.87
	PQS	7.42	7.49	5.92	4.94	6.44	-0.37
	PQM	**7.57**	**8.19**	**7.02**	**6.64**	**7.36**	**0.42**
	PQM-DAM	8.96	9.86	8.06	7.47	8.59	0.82
Cactus	PML seulement	1.27	1.33	1.39	1.43	1.36	0.15
	PQS	6.24	5.87	5.33	4.93	5.59	1.47
	PQM	**6.45**	**6.88**	**6.67**	**6.78**	**6.70**	**2.20**
	PQM-DAM	7.80	8.07	7.75	7.70	7.83	3.57
Kimono	PML seulement	1.32	1.33	1.39	1.43	1.37	0.96
	PQS	5.71	5.63	5.41	5.08	5.46	4.70
	PQM	**5.78**	**5.79**	**5.83**	**5.86**	**5.82**	**4.64**
	PQM-DAM	7.03	7.08	6.94	6.68	6.93	7.03
ParkScene	PML seulement	1.30	1.37	1.44	1.46	1.39	0.33
	PQS	7.55	6.31	5.39	4.74	6.00	0.37
	PQM	**7.89**	**7.06**	**6.45**	**6.53**	**6.98**	**0.88**
	PQM-DAM	9.55	8.41	7.56	7.44	8.24	1.87
Moyenne	PML seulement	1.32	1.38	1.44	1.47	1.40	-0.20
	PQS	6.37	6.08	5.43	4.95	5.71	1.84
	PQM	**6.54**	**6.67**	**6.33**	**6.35**	**6.47**	**2.23**
	PQM-DAM	7.80	7.91	7.34	7.14	7.55	3.82

Pour chaque ensemble d'expériences, quatre combinaisons de méthodes du transcodeur proposé ont été évaluées :

- Le transcodeur avec la méthode *PML* uniquement : qui correspond au transcodeur basé sur la propagation du mouvement tel que proposé au chapitre précédent.
- Le transcodeur avec les méthodes *PML* et *PQS* (noté PQS dans le tableau) : qui correspond au transcodeur proposé complet, paramétré pour utiliser la méthode de partitionnement structurel, c'est-à-dire, avec le paramètre *PS* fixé à vrai.
- Le transcodeur avec les méthodes *PML* et *PQM* (noté PQM dans le tableau) : qui correspond au transcodeur proposé complet, paramétré pour utiliser la méthode de partitionnement basé sur le mouvement, lorsque les conditions le permettent (avec le paramètre *PS* fixé à faux).

Tableau 4.2 Performances du transcodeur proposé pour des séquences 832×480 (classe C) et quatre combinaisons de méthodes de transcodage pour une configuration avec une seule trame de référence

Séquence	Configuration	Accélération (par QP)					BD-Rate(%)
		22	27	32	37	Moy.	
BasketballDrive	PML seulement	1.31	1.35	1.42	1.46	1.39	-0.66
	PQS	7.98	6.88	6.55	6.50	6.98	2.12
	PQM	**8.73**	**8.05**	**8.05**	**8.59**	**8.36**	**3.34**
	PQM-DAM	10.83	9.91	9.81	10.30	10.21	6.33
BQMall	PML seulement	1.30	1.36	1.40	1.46	1.38	-0.34
	PQS	9.06	8.19	7.18	6.56	7.75	1.96
	PQM	**9.57**	**9.19**	**8.80**	**8.88**	**9.11**	**2.89**
	PQM-DAM	12.21	11.45	10.77	10.66	11.27	5.55
PartyScene	PML seulement	1.27	1.30	1.35	1.42	1.34	-0.15
	PQS	9.96	8.98	7.47	6.41	8.21	1.33
	PQM	**10.26**	**9.37**	**8.05**	**7.63**	**8.83**	**2.00**
	PQM-DAM	12.82	11.60	9.87	9.36	10.91	3.27
RaceHorses	PML seulement	1.30	1.33	1.41	1.49	1.38	-0.99
	PQS	7.43	7.34	6.87	6.53	7.04	0.62
	PQM	**7.47**	**7.52**	**7.28**	**7.37**	**7.41**	**1.00**
	PQM-DAM	9.23	9.35	8.94	8.97	9.12	2.94
Moyenne	PML seulement	1.30	1.34	1.40	1.46	1.37	-0.54
	PQS	8.61	7.85	7.02	6.50	7.49	1.51
	PQM	**9.01**	**8.53**	**8.05**	**8.12**	**8.43**	**2.31**
	PQM-DAM	11.27	10.58	9.85	9.82	10.38	4.52

- Le transcodeur avec les méthodes *PML*, *PQM* et *DAM* (noté PQM-DAM dans le tableau) : qui correspond au transcodeur proposé complet, paramétré pour utiliser la méthode de partitionnement basé sur le mouvement et la méthode de décision accélérée du mode HEVC à encoder.

4.3.1.1 Performances de l'algorithme de propagation du mouvement

Comme discuté au chapitre précédent, notre approche de transcodage basée exclusivement sur notre méthode de propagation du mouvement préserve généralement l'efficacité de codage tout en réduisant la complexité par rapport à un transcodeur en cascade. L'amélioration de l'efficacité de codage pour certains cas de transcodage s'explique en partie par l'utilisation des données chromatiques, en supplément aux données de luminance utilisées par la HM, durant l'évaluation de l'erreur de prédiction (voir section 3.2.3.1).

Tableau 4.3 Performances du transcodeur proposé pour des séquences 416×240 (classe D) et et quatre combinaisons de méthodes de transcodage pour une configuration avec une seule trame de référence

Séquence	Configuration	Accélération (par QP) 22	27	32	37	Moy.	BD-Rate(%)
BasketballPass	PML seulement	1.29	1.30	1.37	1.43	1.35	0.09
	PQS	7.80	6.97	6.33	5.96	6.77	2.15
	PQM	**9.20**	**8.56**	**8.07**	**7.94**	**8.44**	**2.33**
	PQM-DAM	10.16	9.52	8.93	8.89	9.38	4.96
BQSquare	PML seulement	1.26	1.29	1.36	1.48	1.35	0.17
	PQS	11.55	10.82	8.62	6.84	9.46	0.35
	PQM	**12.34**	**11.76**	**10.18**	**8.90**	**10.80**	**0.78**
	PQM-DAM	15.47	14.51	11.71	10.42	13.03	1.60
BlowingBubbles	PML seulement	1.27	1.31	1.40	1.42	1.35	0.38
	PQS	9.88	8.40	6.59	5.68	7.64	1.63
	PQM	**10.97**	**9.49**	**8.06**	**7.91**	**9.11**	**2.17**
	PQM-DAM	12.98	11.05	9.19	8.73	10.49	3.89
RaceHorses	PML seulement	1.27	1.28	1.36	1.42	1.33	-0.87
	PQS	8.32	7.45	6.68	6.19	7.16	0.86
	PQM	**9.42**	**8.72**	**7.89**	**7.87**	**8.48**	**0.87**
	PQM-DAM	10.46	9.56	8.22	8.74	9.25	3.18
Moyenne	PML seulement	1.27	1.30	1.37	1.44	1.34	-0.06
	PQS	9.39	8.41	7.06	6.17	7.76	1.25
	PQM	**10.48**	**9.63**	**8.55**	**8.16**	**9.21**	**1.54**
	PQM-DAM	12.27	11.16	9.51	9.20	10.53	3.41

Enfin, comme discuté au chapitre précédent, notre algorithme de propagation du mouvement a tendance à obtenir des accélérations plus élevées pour des QPs élevés, puisque la proportion du temps consommé par l'estimation du mouvement augmente en fonction du QP. En effet, comme nous l'avons vu au chapitre précédent, l'augmentation du QP simplifie l'encodage des données résiduelles, ce qui a pour conséquence d'augmenter la proportion du temps passé dans le module d'estimation du mouvement.

4.3.1.2 Comparaison des configurations de transcodage

Comme le montrent les trois tableaux, le transcodeur complet (les configurations *PQS*, *PQM* et *PQM-DAM*) augmente significativement l'accélération et affecte généralement très peu l'efficacité de codage. De plus, les tableaux montrent que notre approche de partitionnement basée sur le mouvement est nettement plus efficace que notre approche de partitionnement

structurel, en particulier pour des séquences et des QPs élevés, puisque la méthode basée sur le mouvement est en mesure d'arrêter la division des CUs 64×64 et 32×32, contrairement à la méthode structurelle. Enfin, l'activation de l'option DAM permet d'augmenter l'accélération au prix d'une efficacité de codage plus faible, par exemple, dans certains cas l'activation de l'option a pour effet d'augmenter le BD-Rate d'environ 2%.

Les tableaux II-2 à II-3 montrent que l'ajout de trames de référence supplémentaires à l'approche de transcodage en cascade augmente, en général, légèrement son efficacité de codage par rapport à notre approche, qui supporte une seule trame de référence. Cependant, dans certains cas rares, l'ajout de ces trames permet au transcodeur en cascade d'améliorer l'efficacité de codage de manière significative par rapport à notre approche. C'est notamment le cas pour les séquences *BQSquare* et *BasketballDrive*, où notre approche augmente le BD-Rate jusqu'à 14% environ. D'un autre côté, l'ajout de trames de référence supplémentaires à l'approche en cascade améliore détériore la vitesse de l'approche en cascade et, par conséquent, augmente significativement les accélérations obtenues par notre approche de transcodage pour l'ensemble des cas de transcodage simulés. Par exemple, l'accélération moyenne de notre approche pour les séquences de 832×480 pixels testées passe de 8.43 à 15.85x lorsque notre approche est comparée avec l'approche en cascade configurée respectivement avec une et 4 trames de référence.

4.3.2 Comparaison avec les approches proposées dans la littérature

Par rapport aux autres approches proposées dans la littérature, notre approche obtient des accélérations supérieures tout en ayant un impact peu significatif sur l'efficacité de codage. Les résultats de la configuration *PQM* obtiennent en moyenne une accélération de 7.81x, pour une augmentation moyenne du BD-Rate de 2.05%, sur l'ensemble des séquences testées. En comparaison, l'approche de Jiang *et al.* (2013) obtient une accélération moyenne de 2.0x pour un BD-Rate moyen de 1.73%. Les auteurs ont obtenu ces résultats avec une configuration faible délai P (*low delay P*) et quatre trames de référence. À titre de comparaison, notre approche obtient en moyenne un BD-Rate de 4.73% et une accélération de 14.09x, pour notre

configuration *PQM*, lorsqu'elle est comparée avec un transcodeur en cascade utilisant 4 trames de référence. Enfin, l'approche de Peixoto *et al.* (2014b) obtient pour sa part une accélération de 3.83x pour un BD-Rate moyen de 4% pour une configuration faible délai P (*low delay P*) et utilisant une seule trame de référence.

Dans un autre ordre d'idées, Shen *et al.* (2013) ont proposé un transcodeur parallèle avec des accélérations de type instruction unique sur des données multiples (*Single instruction multiple data*, SIMD). Les auteurs annoncent des accélérations allant jusqu'à 70x. Cependant, les accélérations obtenues se situent entre 2 et 10x lorsque les accélérations SIMD et le traitement parallèle sont désactivés. De plus, l'approche proposée par ces auteurs affecte significativement l'efficacité de codage (dans certaines cas, le BD-Rate augmente de plus de 10%).

4.4 Résumé

Dans ce chapitre, nous avons proposé une approche pour réduire le nombre de modes HEVC à évaluer et, plus généralement, pour réduire la complexité du module qui a pour rôle de déterminer le mode HEVC à encoder pour la CTU traitée. Nous avons vu que l'approche proposée se distingue des approches existantes, notamment en appliquant un parcours postfixe sur la CTUs traitée, plutôt qu'un parcours préfixe. Sur la base de ce parcours en postfixe, nous avons proposé plusieurs méthodes pour réduire la complexité du choix du mode HEVC à encoder. Nous avons notamment présenté deux méthodes pour déterminer le partitionnement initial de la CTU traitée : une méthode basée sur la structure de partitionnement H.264 ; et l'autre réutilisant l'information de mouvement (la liste des vecteurs de mouvement candidats et les erreurs de prédiction précalculées) générée par notre approche de propagation de mouvement, présentée au chapitre précédent. Ces méthodes éliminent les modes de partitionnement jugés trop complexes pour produire un codage efficace. Nous avons aussi présenté une méthode pour arrêter prématurément le traitement d'un mode, lorsque son cout J_{pm} est jugé peu prometteur par rapport au cout J_{pm} du meilleur mode actuel. Enfin, les résultats ont montré que l'approche de transcodage proposée dans son ensemble est plus rapide que les approches proposées dans la littérature, tout en affectant peu l'efficacité de codage.

CONCLUSION

Comme nous l'avons vu dans cette thèse, le standard de compression vidéo H.264, publié en 2003, a obtenu un succès phénoménal pour différents types d'applications, dont le stockage sur des disques Blu-Ray, la télévision numérique et la diffusion de vidéo sur Internet. De nos jours, ce standard est encore employé couramment par plusieurs applications vidéo. Cependant, il est maintenant en compétition avec plusieurs autres formats vidéos plus récents et plus efficaces, dont son successeur HEVC, qui a été publié en 2013.

Pour bénéficier de l'efficacité de codage de HEVC, et pour assurer l'interopérabilité entre les différents systèmes et appareils, plusieurs séquences vidéos H.264, existantes ou futures, devront être transcodées en séquences HEVC. Comme nous l'avons mentionné précédemment, l'approche la plus simple pour effectuer une telle conversion consiste à décoder entièrement la séquence d'origine et, par la suite, à l'encoder entièrement selon le format et les paramètres cibles. Cette approche produit un codage efficace, mais a l'inconvénient d'être très complexe en calculs. D'autres solutions, basées sur la réutilisation de l'information de codage H.264, ont été proposées dans la littérature pour réduire cette complexité. Cependant, les accélérations obtenues par ces approches dépassent rarement les 4x.

L'objectif principal de nos travaux était de développer une approche de transcodage H.264 à HEVC plus rapide que les approches existantes, tout en affectant peu l'efficacité de codage, par rapport à un transcodage complet. Dans le second chapitre, nous avons présenté et analysé les approches de transcodage H.264 et HEVC existantes en les divisant en deux catégories de méthodes : les méthodes rapides d'estimation du mouvement et les méthodes rapides de sélection du mode HEVC à encoder.

En ce qui concerne l'estimation de mouvement rapide, nous avons vu que les approches proposées dans la littérature reposent sur l'idée que l'information de mouvement H.264 constitue un bon point de départ pour déterminer le vecteur de mouvement HEVC à encoder, mais que cette information doit être raffinée, généralement en utilisant une petite fenêtre de recherche. De notre côté, nous avons montré que l'information de mouvement H.264

est suffisamment précise pour être réutilisée sans raffinement, mais que cette information se propage de manière différente dans HEVC. Nous avons notamment montré que le meilleur vecteur de mouvement pour la région HEVC traitée se situe parfois à l'extérieur de la région H.264 correspondante. De ce fait, nous avons proposé un algorithme d'estimation de mouvement rapide basé sur la propagation du mouvement. Cette approche crée une liste des vecteurs de mouvement candidats au niveau de la CTU traitée, puis détermine le meilleur candidat pour chacune des partitions traitées au niveau des PUs. Par rapport aux approches de raffinement de mouvement présentées dans la littérature, l'approche que nous proposons a pour avantage d'éliminer les calculs redondants reliés aux erreurs de prédiction. De plus, les résultats expérimentaux ont montré qu'elle est plus rapide que les approches présentées dans l'état de l'art, en plus d'affecter peu l'efficacité de codage par rapport à un transcodage en cascade (complet).

En ce qui concerne la sélection rapide du mode HEVC à encoder, nous avons vu que plusieurs approches proposées dans la littérature réutilisent l'information extraite de la séquence H.264 pour éliminer des modes HEVC à tester. En addition à cette information H.264, quelques autres méthodes considèrent aussi l'information de codage HEVC pour éliminer davantage de modes. Tel que discuté précédemment, ces méthodes sont efficaces pour réduire la complexité des régions simples, puisqu'elles emploient un parcours préfixe sur la CTU traitée. Elles sont cependant moins efficaces pour les régions complexes. De notre côté, nous avons proposé une approche de transcodage basée sur un parcours postfixe de la CTU. Cette approche est constituée de plusieurs méthodes qui ont pour objectif de réduire la complexité de l'évaluation des modes. Nous avons notamment proposé une méthode pour déterminer la structure de partitionnement maximale d'une CTU donnée selon l'information de codage H.264 et l'information de mouvement générée par notre approche de propagation du mouvement. De plus, nous avons proposé une méthode pour arrêter prématurément le traitement d'un mode lorsque son cout J_{motion} est jugé peu prometteur par rapport au cout J_{motion} du meilleur mode actuel. Par rapport à l'état de l'art, l'approche de transcodage proposée, incluant l'algorithme

de propagation du mouvement, obtient des accélérations supérieures, tout en affectant peu l'efficacité de codage.

Notre approche a été développée et testée uniquement pour de la prédiction temporelle unidirectionnelle (pour des trames P) utilisant une seule trame de référence. Cependant, nous croyons qu'elle peut être adaptée, afin de supporter aussi la prédiction temporelle bidirectionnelle (les trames B) ainsi que plusieurs trames de référence. Pour ce faire, il est important que le transcodeur utilise les mêmes trames de référence que celles utilisées durant l'encodage de la séquence H.264 ainsi que les mêmes types de prédiction temporelle. En effet, l'approche est basée sur l'idée que l'information de mouvement H.264 est suffisamment précise pour être réutilisée dans HEVC, à la condition, bien attendu, d'utiliser des configurations similaires pour estimer le mouvement.

Enfin, ces travaux ont fait l'objet d'un article de conférence (Franche et Coulombe (2015)), d'un article de revue soumis (Franche et Coulombe (2016a)) et d'une demande de brevet provisionnelle (Franche et Coulombe (2016b)).

ANNEXE I

SÉQUENCES TEST

Le tableau I-1 présente les séquences utilisées par nos expériences. Pour chacune de ces séquences, nous indiquons le nombre de trames, les résolutions supportées, la fréquence d'images par seconde. De plus, nous décrivons qualitativement le contenu selon quatre aspects : le mouvement des objets, le mouvement de la caméra, les occlusions et la variation de l'éclairage. Ces descriptions sont particulièrement utiles pour analyser les résultats. De plus, elles permettent de constater qu'on couvre plusieurs types de caractéristiques qu'on retrouve couramment dans des séquences vidéos : mouvement rapide ou lent ; peu ou beaucoup d'objets en mouvement ; éclairage stable ou variable ; déplacement de la caméra simple ou complexe ; occlusions à l'avant-plan ou à l'arrière plan, à l'intérieur ou l'extérieur du cadre ; changements de scène.

Tableau-A I-1 Séquences vidéos utilisées durant les expériences

Séquence	Trames à 0, 2, 4, 6, 8 et 10 sec.
RaceHorses (300 trames) • résolutions : 416 × 240 et 832 × 480 • i/s : 30 • mouvement : rapide et peu d'objets • caméra : déplacements translationnels variés • occlusions : occasionnelles, cadre et objets • éclairage : stable	
BasketballPass (500 trames) • résolution : 416 × 240 • i/s : 50 • mouvement : rapide et complexe • caméra : déplacements translationnels variés • occlusions : occasionnelles, cadre et objets • éclairage : stable	

suite à la page suivante

Tableau-A I-1 – suite de la page précédente

Séquence	Trames à 0, 2, 4, 6, 8 et 10 sec.
BlowingBubbles (500 trames) • résolution : 416 × 240 • i/s : 50 • mouvement : moyen, plusieurs petits objets • caméra : zoom arrière lent • occlusions : fréquentes • éclairage : variation de la chrominance	
BQSquare (600 trames) • résolution : 416 × 240 • i/s : 60 • mouvement : moyen • caméra : zoom arrière lent • occlusions : rares • éclairage : stable	
BQMall (600 trames) • résolution : 832 × 480 • i/s : 60 • mouvement : moyen • caméra : déplace de la droite vers la gauche • occlusions : occasionnelles • éclairage : stable	
BasketballDrill (500 trames) • résolution : 832 × 480 • i/s : 50 • mouvement : rapide, flou cinétique • caméra : faible, arrière-plan quasi-stationnaire • occlusions : rares, cadre • éclairage : stable	

suite à la page suivante

Tableau-A I-1 – suite de la page précédente

Séquence	Trames à 0, 2, 4, 6, 8 et 10 sec.
PartyScene (500 trames) • résolution : 832 × 480 • i/s : 50 • mouvement : rapide et complexe • caméra : zoom avant lent • occlusions : occasionnelles, cadre et objets • éclairage : stable	
BasketballDrive (500 trames) • résolution : 1920 × 1080 • i/s : 50 • mouvement : rapide et complexe • caméra : Déplacements variés • occlusions : occasionnelles, cadre et objets • éclairage : stable	
BQTerrace (600 trames) • résolution : 1920 × 1080 • i/s : 60 • mouvement : simple et lent • caméra : déplacement simple • occlusions : rares, cadre • éclairage : stable	
Cactus (500 trames) • résolution : 1920 × 1080 • i/s : 50 • mouvement : simple et lent • caméra : déplacement simple • occlusions : rares, cadre • éclairage : stable	

suite à la page suivante

Tableau-A I-1 – suite de la page précédente

Séquence	Trames à 0, 2, 4, 6, 8 et 10 sec.
Kimono (240 trames) • résolution : 1920 × 1080 • i/s : 24 • mouvement : simple et lent • caméra : déplacement simple • occlusions : rares • éclairage : stable	
ParkScene (240 trames) • résolution : 1920 × 1080 • i/s : 24 • mouvement : simple et lent • caméra : Déplacement simple • occlusions : rares • éclairage : stable	

ANNEXE II

RÉSULTATS SUPPLÉMENTAIRES

Cette section présente des résultats expérimentaux complémentaires aux résultats présentés à la section 4.3.1. En premier lieu, les tableaux II-2 à II-3 présentent les performances de notre approche de transcodage relatives aux performances d'un transcodage en cascade. Notre approche de transcodage utilise une seule trame de référence, tandis que l'approche de transcodage en cascade en emploie quatre, contrairement aux expériences présentées à la section 4.3.1, où cette dernière utilise une seule trame de référence. Les résultats montrent que l'ajout de trames de référence supplémentaires permet au transcodeur en cascade d'améliorer légèrement son efficacité de codage dans plusieurs cas, ou significativement dans quelques cas plus rares. Dans tout les cas, l'ajout de trames de référence supplémentaires augmente significativement le temps d'exécution du transcodage.

Pour leurs parts, les tableaux II-4 à II-6 présentent les performances de la version parallèle de notre approche de transcodage relatives aux performances d'un transcodeur en cascade. Les deux approches comparées utilisent une seule trame de référence, comme c'est le cas pour les expériences présentées à la section 4.3.1. Les résultats montrent que la version parallèle de notre approche améliore l'accélération, sans affecter l'efficacité de codage, comparativement aux résultats montrés à la section 4.3.1.

Tableau-A II-1 Performances du transcodeur proposé, relatives à un transcodeur en cascade utilisant 4 trames de référence, pour des séquences 1920×1080

Séquence	Configuration	Accélération (par QP)					BD-Rate(%)
		22	27	32	37	Moy.	
BasketballDrive	PML seulement	2.51	2.89	3.13	3.23	2.94	-1.17
	PQS	8.80	10.03	10.46	10.57	9.97	3.43
	PQM	**8.98**	**10.64**	**11.72**	**12.45**	**10.95**	**3.41**
	PQM-DAM	10.13	12.04	13.14	13.47	12.20	5.73
BQTerrace	PML seulement	1.71	2.00	2.35	2.50	2.14	0.83
	PQS	9.72	10.89	9.49	8.34	9.61	1.32
	PQM	**9.91**	**11.90**	**11.24**	**11.15**	**11.05**	**2.13**
	PQM-DAM	11.72	14.34	12.91	12.61	12.90	2.55
Cactus	PML seulement	1.75	2.14	2.40	2.58	2.22	0.53
	PQS	8.62	9.46	9.20	8.90	9.05	1.86
	PQM	**8.90**	**11.09**	**11.52**	**12.24**	**10.94**	**2.59**
	PQM-DAM	10.77	13.01	13.38	13.90	12.77	3.97
Kimono	PML seulement	2.32	2.53	2.80	2.99	2.66	0.66
	PQS	10.07	10.66	10.90	10.63	10.57	4.40
	PQM	**10.18**	**10.97**	**11.74**	**12.27**	**11.29**	**4.34**
	PQM-DAM	12.13	13.40	13.97	14.00	13.38	5.96
ParkScene	PML seulement	1.87	2.20	2.52	2.65	2.31	0.80
	PQS	10.88	10.17	9.41	8.58	9.76	0.84
	PQM	**11.38**	**11.31**	**11.27**	**11.83**	**11.45**	**1.36**
	PQM-DAM	13.76	13.47	13.20	13.48	13.48	2.33
Moyenne	PML seulement	2.03	2.35	2.64	2.79	2.45	0.33
	PQS	9.62	10.24	9.89	9.40	9.79	2.37
	PQM	**9.87**	**11.18**	**11.50**	**11.99**	**11.13**	**2.77**
	PQM-DAM	11.70	13.25	13.32	13.49	12.94	4.11

Tableau-A II-2 Performances du transcodeur proposé, relatives à un transcodeur en cascade utilisant 4 trames de référence, pour des séquences 832×480

Séquence	Configuration	Accélération (par QP)					BD-Rate(%)
		22	27	32	37	Moy.	
BasketballDrive	PML seulement	2.17	2.56	2.94	3.20	2.72	3.34
	PQS	13.28	13.06	13.62	14.26	13.56	6.22
	PQM	**14.53**	**15.28**	**16.74**	**18.83**	**16.35**	**7.50**
	PQM-DAM	18.08	18.79	20.39	22.58	19.96	10.59
BQMall	PML seulement	2.12	2.47	2.79	3.07	2.61	2.26
	PQS	14.74	14.89	14.30	13.80	14.43	4.62
	PQM	**15.56**	**16.70**	**17.54**	**18.70**	**17.13**	**5.58**
	PQM-DAM	19.85	20.80	21.45	22.45	21.14	8.30
PartyScene	PML seulement	1.73	1.92	2.30	2.67	2.16	5.57
	PQS	13.58	13.27	12.74	12.03	12.91	7.10
	PQM	**14.00**	**13.84**	**13.73**	**14.32**	**13.97**	**7.81**
	PQM-DAM	17.48	14.14	16.83	17.56	16.50	9.12
RaceHorses	PML seulement	2.45	2.84	3.37	3.82	3.12	0.33
	PQS	14.07	15.63	16.42	16.73	15.71	1.95
	PQM	**14.16**	**16.00**	**17.40**	**18.86**	**16.61**	**2.34**
	PQM-DAM	17.48	19.90	21.35	22.97	20.43	4.30
Moyenne	PML seulement	2.12	2.45	2.85	3.19	2.65	2.88
	PQS	13.92	14.21	14.27	14.21	14.15	4.97
	PQM	**14.56**	**15.46**	**16.35**	**17.68**	**16.01**	**5.81**
	PQM-DAM	18.22	18.41	20.01	21.39	19.51	8.08

Tableau-A II-3 Performances du transcodeur proposé, relatives à un transcodeur en cascade utilisant 4 trames de référence, pour des séquences 416×240

Séquence	Configuration	Accélération (par QP)					BD-Rate(%)
		22	27	32	37	Moy.	
BasketballPass	PML seulement	2.20	2.44	2.82	3.10	2.64	0.84
	PQS	13.32	13.06	13.03	12.90	13.08	2.90
	PQM	**15.70**	**16.05**	**16.62**	**17.18**	**16.39**	**3.09**
	PQM-DAM	17.34	17.86	18.38	19.24	18.21	5.75
BQSquare	PML seulement	1.58	1.67	1.99	2.50	1.94	13.20
	PQS	14.43	14.00	12.56	11.59	13.15	13.41
	PQM	**15.42**	**15.22**	**14.82**	**15.07**	**15.13**	**13.88**
	PQM-DAM	15.47	19.34	18.77	17.04	17.65	14.81
BlowingBubbles	PML seulement	1.71	1.95	2.37	2.60	2.16	3.91
	PQS	13.26	12.56	11.14	10.39	11.84	5.20
	PQM	**14.72**	**14.18**	**13.63**	**13.91**	**14.11**	**5.76**
	PQM-DAM	17.42	16.51	15.55	15.96	16.36	7.52
RaceHorses	PML seulement	2.24	2.59	3.04	3.45	2.83	-0.01
	PQS	14.59	15.00	14.97	14.99	14.89	1.73
	PQM	**16.54**	**17.56**	**17.71**	**19.26**	**17.77**	**1.74**
	PQM-DAM	18.36	19.26	19.60	20.26	19.37	4.07
Moyenne	PML seulement	1.93	2.16	2.56	2.91	2.39	4.48
	PQS	13.90	13.66	12.93	12.47	13.24	5.81
	PQM	**15.60**	**15.75**	**15.70**	**16.36**	**15.85**	**6.12**
	PQM-DAM	17.15	18.24	18.08	18.13	17.90	8.04

Tableau-A II-4 Performances de la version parallèle du transcodeur proposé, relatives au transcodeur en cascade, pour des séquences 1920×1080

Séquence	Configuration	Accélération (par QP) 22	27	32	37	Moy.	BD-Rate(%)
BasketballDrive	PML seulement	1.43	1.52	1.55	1.57	1.52	-1.57
	PQS	5.20	5.41	5.35	5.25	5.30	3.02
	PQM	**5.29**	**5.72**	**6.02**	**6.25**	**5.82**	**2.99**
	PQM-DAM	6.08	6.45	6.81	7.10	6.61	5.80
BQTerrace	PML seulement	1.32	1.41	1.50	1.52	1.44	-0.87
	PQS	7.77	7.91	6.22	4.94	5.13	-0.37
	PQM	**7.99**	**8.72**	**7.47**	**6.99**	**7.79**	**0.42**
	PQM-DAM	9.53	10.61	8.58	7.98	9.18	0.82
Cactus	PML seulement	1.29	1.36	1.41	1.45	1.38	0.15
	PQS	6.70	6.29	5.50	5.20	5.92	1.47
	PQM	**6.89**	**7.45**	**7.16**	**7.24**	**7.19**	**2.20**
	PQM-DAM	8.50	8.88	8.38	8.30	8.52	3.57
Kimono	PML seulement	1.35	1.39	1.43	1.48	1.41	0.96
	PQS	6.40	6.29	5.97	5.48	6.04	4.70
	PQM	**6.45**	**6.49**	**6.52**	**6.35**	**6.45**	**4.64**
	PQM-DAM	8.02	8.18	7.88	7.54	7.91	6.27
ParkScene	PML seulement	1.32	1.40	1.47	1.49	1.42	0.33
	PQS	8.17	6.84	5.75	5.02	6.45	0.37
	PQM	**8.56**	**7.60**	**6.97**	**6.99**	**7.53**	**0.88**
	PQM-DAM	10.52	9.21	8.30	8.23	9.07	1.87
Moyenne	PML seulement	1.34	1.42	1.47	1.50	1.43	-0.20
	PQS	6.85	6.55	5.76	5.18	6.08	1.84
	PQM	**7.04**	**7.20**	**6.83**	**6.76**	**6.96**	**2.23**
	PQM-DAM	8.53	8.67	7.99	7.83	8.25	3.67

Tableau-A II-5 Performances de la version parallèle du transcodeur proposé, relatives au transcodeur en cascade, pour des séquences 832×480

Séquence	Configuration	Accélération (par QP)					BD-Rate(%)
		22	27	32	37	Moy.	
BasketballDrive	PML seulement	1.33	1.38	1.44	1.49	1.41	-0.66
	PQS	8.60	7.51	7.11	6.96	7.55	2.12
	PQM	**9.53**	**8.85**	**8.85**	**9.31**	**9.14**	**3.34**
	PQM-DAM	12.06	11.03	11.00	11.53	11.41	6.33
BQMall	PML seulement	1.33	1.39	1.45	1.50	1.42	-0.34
	PQS	9.98	9.00	7.87	7.01	8.47	1.96
	PQM	**10.64**	**10.27**	**9.79**	**9.79**	**10.12**	**2.89**
	PQM-DAM	13.85	13.18	12.27	12.24	12.89	5.55
PartyScene	PML seulement	1.28	1.33	1.40	1.46	1.37	-0.15
	PQS	10.55	9.65	8.13	6.94	8.82	1.33
	PQM	**10.92**	**10.08**	**8.78**	**8.36**	**9.54**	**2.00**
	PQM-DAM	13.83	12.78	10.97	10.51	12.02	3.27
RaceHorses	PML seulement	1.32	1.37	1.45	1.54	1.42	-0.99
	PQS	8.22	8.27	7.78	7.37	7.91	0.62
	PQM	**8.31**	**8.55**	**8.30**	**8.39**	**8.39**	**1.00**
	PQM-DAM	10.50	10.88	10.51	10.52	10.60	2.94
Moyenne	PML seulement	1.32	1.37	1.44	1.50	1.40	-0.54
	PQS	9.34	8.61	7.72	7.07	8.18	1.51
	PQM	**9.85**	**9.44**	**8.93**	**8.96**	**9.30**	**2.31**
	PQM-DAM	12.56	11.97	11.19	11.20	11.73	4.52

Tableau-A II-6 Performances de la version parallèle du transcodeur proposé, relatives au transcodeur en cascade, pour des séquences 416×240

Séquence	Configuration	Accélération (par QP)					BD-Rate(%)
		22	27	32	37	Moy.	
BasketballPass	PML seulement	1.32	1.34	1.39	1.48	1.38	0.09
	PQS	8.73	7.88	7.04	6.76	7.60	2.15
	PQM	**9.27**	**8.61**	**8.11**	**8.09**	**8.52**	**2.33**
	PQM-DAM	11.65	10.94	10.30	10.16	10.76	4.96
BQSquare	PML seulement	1.28	1.30	1.38	1.50	1.37	0.17
	PQS	12.40	11.24	9.13	7.34	10.03	0.35
	PQM	**12.34**	**12.50**	**12.02**	**10.09**	**9.08**	**0.78**
	PQM-DAM	16.61	15.87	12.80	11.22	14.13	1.60
BlowingBubbles	PML seulement	1.28	1.34	1.42	1.47	1.38	0.38
	PQS	10.83	9.06	7.20	6.08	8.29	1.63
	PQM	**11.05**	**9.42**	**8.00**	**7.63**	**9.03**	**2.17**
	PQM-DAM	13.85	12.00	10.15	9.49	11.37	3.89
RaceHorses	PML seulement	1.31	1.33	1.41	1.49	1.39	-0.87
	PQS	9.56	8.59	7.60	7.05	8.20	0.86
	PQM	**9.64**	**8.79**	**8.11**	**7.55**	**8.52**	**0.87**
	PQM-DAM	12.67	11.58	10.53	9.93	11.18	3.18
Moyenne	PML seulement	1.30	1.33	1.40	1.49	1.38	-0.06
	PQS	10.38	9.19	7.74	6.81	8.53	1.25
	PQM	**10.58**	**9.83**	**9.06**	**8.34**	**9.45**	**1.54**
	PQM-DAM	13.70	12.60	10.95	10.20	11.86	3.41

BIBLIOGRAPHIE

Aaron, A. et D. Ronca. 2015. « High Quality Video Encoding at Scale ». En ligne. < http://techblog.netflix.com/2015/12/high-quality-video-encoding-at-scale.html >. Consulté le 3 mars 2016.

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua, et S. Susstrunk. 2012. « SLIC superpixels compared to state-of-the-art superpixel methods ». *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, n° 11, p. 2274–2282.

Ahmad, I., X. Wei, Y. Sun, et Y.-Q. Zhang. 2005. « Video transcoding : an overview of various techniques and research issues ». *Multimedia, IEEE Transactions on*, vol. 7, n° 5, p. 793–804.

Amazon. 2016. « Amazon Fire TV ». En ligne. < http://www.amazon.com/Amazon-DV83YW-Fire-TV/dp/B00U3FPN4U >. Consulté le 3 mars 2016.

Bialkowski, J., M. Barkowsky, et A. Kaup. 2006. « Overview of low-complexity video transcoding from H.263 to H.264 ». Dans *Multimedia and Expo, 2006 IEEE International Conference on*. p. 49–52. IEEE.

Bjøntegaard, G. 2001. « Document VCEG-M33 : Calculation of average PSNR differences between RD-curves ». Dans *ITU-T VCEG Meeting, Austin, Texas, USA, Tech. Rep*. p. 2–4.

Björk, N. et C. Christopoulos. 1998. « Transcoder architectures for video coding ». *IEEE Transactions on Consumer Electronics*, vol. 44, n° 1, p. 88–98.

Bossen, F. 2012. « document JCTVC-J1100 : Common test conditions and software reference configurations ». Dans *JCT-VC Meeting, Stockholm, Sweden, Tech. Rep*.

Bossen, F., B. Bross, K. Suhring, et D. Flynn. 2012. « HEVC complexity and implementation analysis ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, n° 12, p. 1685–1696.

Chang, S.-F. et D. G. Messerschmitt. 1995. « Manipulation and compositing of MC-DCT compressed video ». *Selected Areas in Communications, IEEE Journal on*, vol. 13, n° 1, p. 1–11.

Chen, W.-N. et H.-M. Hang. 2008. « H.264/AVC motion estimation implmentation on compute unified device architecture (CUDA) ». Dans *Multimedia and Expo, 2008 IEEE International Conference on*. p. 697–700. IEEE.

Chen, Z.-Y., J.-T. Fang, T.-L. Liao, et P.-C. Chang. Apr 2014. « Efficient PU mode decision and motion estimation for H.264/AVC to HEVC transcoder ». *Signal & Image Processing : An International Journal (SIPIJ)*, vol. 5, n° 2, p. 81–93.

Chen, Z.-Y., C.-T. Tseng, et P.-C. Chang. 2013. « Fast inter prediction for H.264 to HEVC transcoding ». Dans *Proc. the 3rd International Conference on Multimedia Technology (ICMT), Guangzhou, China*. p. 1301–1308.

Choi, J.-A. et Y.-S. Ho. 2013. « Efficient residual data coding in CABAC for HEVC lossless video compression ». *Signal, Image and Video Processing*, p. 1–12.

Dogan, S., S. Eminsoy, A. H. Sadka, et A. M. Kondoz. 2004. « Video content adaptation using transcoding for enabling UMA over UMTS ». Dans *5th workshop on image analysis for multimedia interactive services, Lisbon*.

Eleftheriadis, A. et B. Anastassiou. 1995. « Constrained and general dynamic rate shaping of compressed digital video ». Dans *Image Processing, 1995. Proceedings., International Conference on*. p. 396–399. IEEE.

Fang, J.-T., Z.-Y. Chen, T.-L. Liao, et P.-C. Chang. 2014. « A Fast PU Mode Decision Algorithm for H.264/AVC to HEVC Transcoding ». *Computer Science & Information Technology*.

Franche, J.-F. et S. Coulombe. 2015. « Fast H.264 to HEVC transcoder based on post-order traversal of quadtree structure ». Dans *Image Processing (ICIP), 2015 IEEE International Conference on*. p. 477–481. IEEE.

Franche, J.-F. et S. Coulombe. 2016a. Fast H.264-to-HEVC transcoding based on motion propagation and post-order traversal of coding tree units. Soumis à IEEE Transactions on Circuits and Systems for Video Technology.

Franche, J.-F. et S. Coulombe. 2016b. « Method and apparatus for video intermodal transcoding ». demande de brevet PCT/CA2016/000027.

Furht, B., J. Greenberg, et R. Westwater, 2012. *Motion estimation algorithms for video compression*, volume 379.

Gao, Y. et J. Zhou. 2014. « Motion vector extrapolation for parallel motion estimation on GPU ». *Multimedia tools and applications*, vol. 68, n° 3, p. 701–715.

Gweon, R.-h. et L. Yung-Lyul. 2012. « Early termination of CU encoding to reduce HEVC complexity ». *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, n° 7, p. 1215–1218.

Hermann, V. 2015. « L'ultra HD Blu-ray est prêt, les premiers films 4K sur disques dans quelques mois ». En ligne. < http://www.nextinpact.com/news/94113-lultra-hd-blu-ray-est-pret-premiers-films-4k-sur-disques-dans-quelques-mois.htm >. Consulté le 3 mars 2016.

Hwang, J.-N. et T.-D. Wu. 1998. « Motion vector re-estimation and dynamic frame-skipping for video transcoding ». Dans *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers*. p. 1606–1610. IEEE.

ISO/IEC. 1994. « MPEG-2, ISO/IEC 13818 : Information technology - Generic coding of moving pictures and associated audio information ».

ITU-T. 1990. « H.261 Video codec for audiovisual services at p x 64 kbit/s ». *The International Telegraph and Telephone Consultative Committee*, vol. 115.

ITU-T. 2003. « ITU-T Recommendation H.264 : Advanced video coding for generic audiovisual services ».

ITU-T. 2013. « High Efficiency Video Coding ». *ISO/IEC JTC 1/SC 29/WG 11 et ITU-T VCEG, JVTG050*.

JCT-VC. 2013. « H.264/AVC Reference Software ». En ligne. < http://iphome.hhi.de/suehring/tml/download/old_jm/jm18.2.zip >. Consulté le 15 octobre 2014.

JCT-VC. 2014. « HEVC software repository ». En ligne. < https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-12.1/ >. Consulté le 15 octobre 2014.

Jiang, W. et Y. Chen. 2013. « Low-complexity transcoding from H.264 to HEVC based on motion vector clustering ». *Electronics Letters*, vol. 49, n° 19, p. 1224–1226.

Jiang, W., Y. Chen, et X. Tian. 2013. « Fast transcoding from H.264 to HEVC based on region feature analysis ». *Multimedia Tools and Applications*, vol. 73, n° 3, p. 2179–2200.

Joset, D. et S. Coulombe. 2013. « Visual Quality and File Size Prediction of H.264 Videos and Its Application to Video Transcoding for the Multimedia Messaging Service and Video on Demand ». Dans *Multimedia (ISM), 2013 IEEE International Symposium on*. p. 321–328. IEEE.

Kim, I.-K., S. Lee, M.-S. Cheon, T. Lee, et J. Park. 2012. « Coding efficiency improvement of HEVC using asymmetric motion partitioning ». Dans *Broadband Multimedia Systems and Broadcasting (BMSB), 2012 IEEE International Symposium on*. p. 1–4. IEEE.

Lee, Y.-K., S.-S. Lee, et Y.-L. Lee. 2006. « MPEG-4 to H. 264 transcoding using macroblock statistics ». Dans *Multimedia and Expo, 2006 IEEE International Conference on*. p. 57–60. IEEE.

Li, R., B. Zeng, et M. L. Liou. 1994. « A new three-step search algorithm for block motion estimation ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 4, n° 4, p. 438–442.

Li, Z.-N., M. S. Drew, et J. Liu, 2014. *Fundamentals of Multimedia*. New Jersey : Springer.

Luo, R., R. Xie, et L. Zhang. 2014. « Fast AVS to HEVC transcoding based on ROI detection using visual characteristics ». Dans *Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on*. p. 1–6. IEEE.

Matyunin, S. 2013. « Bjontegaard metric calculation (BD-PSNR) ». En ligne. < http://www.mathworks.com/matlabcentral/fileexchange/41749-bjontegaard-metric-calculation--bd-psnr- >. Consulté le 29 octobre 2014.

MulticoreWare. 2016. « x265 HEVC Upgrade ». En ligne. < https://x265.com/ >. Consulté le 3 mars 2016.

Nakajima, Y., H. Hori, et T. Kanoh. 1995. « Rate conversion of MPEG coded video by re-quantization process ». Dans *Image Processing, 1995. Proceedings., International Conference on*. p. 408–411. IEEE.

Negi, N. et J. Shukla. 2013. « Video Transcoding : An Overview of Various Architectures & Design Issues ». *International Journal of Emerging Trends & Technology in Computer Science (IJETICS)*, vol. 2, n° 2, p. 353–357.

Ohm, J., G. J. Sullivan, H. Schwarz, T. K. Tan, et T. Wiegand. 2012. « Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC) ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, n° 12, p. 1669–1684.

Ozer, J. 2015. « The State of Video Codecs 2015 ». En ligne. < http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/The-State-of-Video-Codecs-2015-102806.aspx >. Consulté le 3 mars 2016.

Peixoto, E., B. Macchiavello, E. Hung, et R. de Queiroz. 2014a. « A Fast HEVC Transcoder Based on Content Modeling and Early Termination ». *Image Processing (ICIP), 2014 21th IEEE International Conference on*, p. 2532-2536.

Peixoto, E. et E. Izquierdo. 2012. « A complexity-scalable transcoder from H. 264/AVC to the new HEVC codec ». Dans *Image Processing (ICIP), 2012 19th IEEE International Conference on*. p. 737–740. IEEE.

Peixoto, E., T. Shanableh, et E. Izquierdo. 2014b. « H.264/AVC to HEVC Video Transcoder based on Dynamic Thresholding and Content Modeling ». Dans *Transactions on Circuits and Systems for Video Technology*. p. 99-112. IEEE.

Qian, T., J. Sun, D. Li, X. Yang, et J. Wang. 2006. « Transform domain transcoding from MPEG-2 to H. 264 with interpolation drift-error compensation ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, n° 4, p. 523–534.

Rasmus, L. 2015. « Netflix talks next-gen video formats, 4K HDR and recommendations ». En ligne. < http://www.flatpanelshd.com/focus.php?subaction=showfull&id=1442583571 >. Consulté le 3 mars 2016.

Richardson, I. E., 2011. *The H.264 advanced video compression standard*. Great Britain : John Wiley & Sons.

Shanableh, T. et M. Ghanbari. 2000. « Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats ». *Multimedia, IEEE Transactions on*, vol. 2, n° 2, p. 101–110.

Shen, B., I. K. Sethi, et B. Vasudev. 1999. « Adaptive motion-vector resampling for compressed video downscaling ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 9, n° 6, p. 929–936.

Shen, T., Y. Lu, Z. Wen, L. Zou, Y. Chen, et J. Wen. 2013. « Ultra fast H. 264/AVC to HEVC transcoder ». Dans *Data Compression Conference (DCC), 2013*. p. 241–250. IEEE.

Shen, X. et L. Yu. 2013. « CU splitting early termination based on weighted SVM ». *EURASIP Journal on Image and Video Processing*, vol. 2013, n° 1, p. 1–11.

Stockhammer, T. 2011. « Dynamic adaptive streaming over HTTP– : standards and design principles ». Dans *Proceedings of the second annual ACM conference on Multimedia systems*. p. 133–144. ACM.

Su, Y., J. Xin, A. Vetro, et H. Sun. 2005. « Efficient MPEG-2 to H. 264/AVC intra transcoding in transform-domain ». Dans *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. p. 1234–1237. IEEE.

Sui, J. 2016. « HEVC Products Forecast Overview ». En ligne. < http://www.dtcreports.com/weeklyriff/2016/03/20/hevc-products-forecast-overview/ >. Consulté le 5 avril 2016.

Sullivan, G. J., J. Ohm, W.-J. Han, et T. Wiegand. 2012. « Overview of the high efficiency video coding (HEVC) standard ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, n° 12, p. 1649–1668.

Sun, H., W. Kwok, et J. W. Zdepski. 1996. « Architectures for MPEG compressed bitstream scaling ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, n° 2, p. 191–199.

Tourapis, A. M. 2002. « Enhanced predictive zonal search for single and multiple frame motion estimation ». Dans *Electronic Imaging 2002*. p. 1069–1079. International Society for Optics and Photonics.

WebM. 2016a. « WebM : an open web media project ». En ligne. < http://http://www.webmproject.org >. Consulté le 11 avril 2016.

WebM. 2016b. « VP9 Video Codec ». En ligne. < http://http://www.webmproject.org/vp9/ >. Consulté le 11 avril 2016.

Wien, M., 2015. *High Efficiency Video Coding*. Switzerland : Springer.

Xing, P., Y. Tian, X. Zhang, Y. Wang, et T. Huang. 2013. « A coding unit classification based AVC-to-HEVC transcoding with background modeling for surveillance videos ». Dans *Visual Communications and Image Processing (VCIP), 2013*. p. 1–6. IEEE.

Yoo, H.-M. et J.-W. Suh. 2013. « Fast coding unit decision algorithm based on inter and intra prediction unit termination for HEVC ». Dans *Consumer Electronics (ICCE), 2013 IEEE International Conference on*. p. 300–301. IEEE.

Youn, J., M.-T. Sun, et C.-W. Lin. 1999. « Motion vector refinement for high-performance transcoding ». *Multimedia, IEEE Transactions on*, vol. 1, n° 1, p. 30–40.

Zhang, D., B. Li, J. Xu, et H. Li. 2012. « Fast transcoding from H.264/AVC to high efficiency video coding ». Dans *Multimedia and Expo (ICME), 2012 IEEE International Conference on*. p. 651–656. IEEE.

Zhu, S. et K.-K. Ma. 2000. « A new diamond search algorithm for fast block-matching motion estimation ». *Image Processing, IEEE Transactions on*, vol. 9, n° 2, p. 287–290.

Zong-Yi, C., T. Chi-Teng, et C. Pao-Chi. 2013. « Fast Inter Prediction for H. 264 to HEVC Transcoding ». Dans *3rd International Conference on Multimedia Technology (ICMT-13)*. p. 1301–1308. Atlantis Press.