Optimisation conjointe source/canal d'une transmission vidéo H.264/AVC sur un lien sans fil

Cyril Bergeron

Thèse

présentée pour obtenir le grade de docteur

de l'Ecole nationale supérieure
des télécommunications

Spécialité : Traitement du Signal et de l'image

Cyril Bergeron

Optimisation conjointe source/canal d'une transmission vidéo H.264/AVC sur un lien sans fil

présentée le 24 janvier 2007 devant le jury composé de

Pierre Duhamel	Président du jury
Christine Guillemot	Rapporteur
Michel Kieffer	
Marco Chiani	Examinateur
Catherine Lamy-Bergot	Directrice de thèse
Béatrice Pesquet-Popescu	

Ecole Nationale Supérieure des Télécommunications

A ma famille,

A toutes celles et ceux qui ont beaucoup compté pour moi,

A stéphane, "parce que c'était lui, parce que c'était moi."
(Montaigne - Essais)

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude envers les membres du jury : Pierre Duhamel, qui m'a fait l'honneur de bien vouloir présider, Christine Guillemot et Michel Kieffer, qui ont accepté la lourde tâche de rapporteurs, et Marco Chiani pour avoir accepté d'examiner cette thèse. J'en profite également pour remercier Maria G. Martini, Matteo Mazzotti, Peter Amon, Jyrki Huusko, Janne Vehkaperä, Johannes Peltola, Soon Xin Ng, Gábor Jeney, Gábor Feher, Gianmarco Panza, et tous les autres partenaires du projet européen PHOENIX, qui ont permis que ce projet puisse aboutir, et sans lesquels je n'aurai pu obtenir tous ces résultats expérimentaux.

Je souhaite exprimer ma profonde reconnaissance à Béatrice Pesquet-Popescu pour son écoute et sa patience, ainsi que tout le laboratoire TSI de l'ENST pour leur accueil chaleureux durant ces trois années et qui m'ont permis de rencontrer des gens formidables comme Grégoire Pau, Christophe Tillier, Maria Trocan, Teodora Petrisor, Lionel Gueguen, Slim Essid, Cléo Baras, et surtout George Abitbol, dont la classe n'a d'équivalent dans ce bas monde.

Je tiens aussi à remercier chaleureusement l'ensemble des laboratoires WFD, MMP et SBP de THALES Communications, avec qui j'ai apprécié sincèrement de travailler pendant plus de trois ans. J'y ai rencontré des gens formidables et attachants, qui ont su m'apporter leurs expériences et leur soutien tant sur le plan scientifique que sur le plan humain, avec en particulier : Dominique Mérel et Jean-Jacques Monot qui ont accepté que j'effectue cette thèse au sein du laboratoire WFD, Didier Nicholson, et Cedric Le Barz, pour leurs expertises du traitement de l'image, Martine Le Forban, pour sa gentillesse infinie, Bruno Sourdillat, à qui je souhaite une bonne continuation dans ses nouvelles fonctions, Bertrand Ravera, pour nos discussions passionnante sur la musique, Sami Moussa, Rachid Elmostadi et Gwenael Guilmin, pour leur jovialité, Erwann Renan, qui devient désormais le grand expert au sein du service de la norme H.264, Marc Leny, et Audrey Blin, qui ont choisi comme moi d'effectuer un doctorat au sein de ce service, et à qui je souhaite le meilleur pour la suite, Pierre André Laurent, pour toutes ses réponses à toutes ces questions plus ou moins stupides énoncées par moi même lors de nos regrettées pauses-café, Jacques Eudes, lui aussi pour toutes ses réponses et ses certitudes d'experts, Jean François Delaune, qui, avec M. Laurent, partage la même passion de la pomme croquée auquel il a tenté à de nombreuses reprises de me convertir, François Van de Wiele, Laurent Belmon, David Depierre et Grégoire Hourlier, pour leurs enrichissantes discussions sur la loi des séries, Joel Thibault, pour son humour et son éternel jeunesse, Nathalie Carves-Bideaux, pour ses indispensables connaissances sur Omega, Marc Chenu

pour notre passion commune et nos courses poursuites, Pierre Hammes et Florent Barthe, pour leur porte toujours ouverte, Benjamin Boissy et Alexandre Preti, pour m'avoir supporté pendant cette interminable rédaction de manuscrit, Damien Gillette, qui, j'en suis certain, continuera avec bonheur mes travaux, et enfin Grégoire Guibé, Christophe Le Martret, Isabelle Icart, Claude Lemenager, Bertrand Mercier, Pascal Chevalier, Frédérique Sainte Agathe, Nabil Abdelli, Pascal Burlot, Vincent Azan, Séverin Collet et tous les stagiaires qui se sont succédé ([NDLR : sans 's']), et que je ne peux oublier.

Je ne peux oublier non plus celle qui m'a offert l'opportunité de faire cette thèse, Catherine Lamy-Bergot. Je ne trouverai ni les mots ni le temps pour exprimer ma profonde gratitude de m'avoir donner cette chance et sa confiance. J'espère ne pas l'avoir trop déçue, hormis peut être pour mes goûts musicaux qui n'englobent pas la discographie de Blondie, Cher, la grande Annie Cordy ou Catherine Ribeiro, pour mes faibles connaissances dans la politique économique de Georges Pompidou que je n'ai pas connu, pour mes talents d'orateur à faire dormir un chef de laboratoire sous Guronsan, pour l'image des français que j'ai laissée aux autres partenaires du projet européen, et enfin pour ma passion inaltérable pour le Fernet Branca. Néanmoins, je souhaite qu'elle gardera tout comme moi, un bon souvenir de nos années passées dans le même bureau.

Je tiens aussi à remercier aussi tous ceux qui m'ont précédé et qui m'ont donné envie de faire une thèse : Raquel Urtasun , Ana Andres Del Valle (merci d'être venue), Christophe Parisot (ce puits de sciences), Stephane Tramini, Lionel Brunel (mon Rootboy, futur maire de Carcès), Karim Ben Chehida (ce grand docteur), Anwar Al Hamra (le libanais fou), Mamadou Ndiaye (et sa passion des Bonobos), Anne-Laure Deleuze (coucou !) et Adrien Renoult qui m'ont précédé à THALES.

Je ne saurais oublier Franck Lanoizelé et Guillaume Crozet, pour nos souvenirs d'enfance à Charolles, Ibrahima Ly, pour sa disponibilité même à 4h du matin, Valeria Nuzzo et Christophe Dalys, pour notre passion commune pour le football, Houcine et Hannane Zbeir, pour la petite Yasmine, David Teixier et Zina Gilbert, pour avoir décidé de faire le grand saut, Jean Michel Forcheron (Filip'), Steve Giraud (Franck), David Navoret (bouba), Gregory Cavaller et sa petite famille, Simon Gallard et tous les anciens du Lycée Lalande, Sandrine Izraël, Najia Tamda, Hervé Detour, Saloua Hebchane, Michel Rihani, Félicie Cazes, Vincent Gardais, Malika Nambatingué, et tout le Foyer des Jeunes Travailleurs de Garbejaire, Zinédine Zidane parti sur un coup de tête, ainsi que toute la bande des 3 frères : Xavier Roumegue (binome de feu), Virginie Hirvois (binomette), Marc Caterini (Marco le rigolo), Virgile Speich (le grand blond), Robert yeung (master Bob), Laurent Papadopoulos (dernier fan de l'OM), Rémi Payan (fan d'opéra de bizet), et Mick Cornut.

Je ne peux pas écrire de remerciements sans remercier chaleureusement toute ma famille (d'Oyonnax et d'ailleurs) et ceux sans qui je ne serai pas là : mes parents. Je les remercie infiniment pour leur soutien, leur sacrifice, leur éducation et pour m'avoir permis, pendant ces longues vacances scolaires offertes par le ministère de l'éducation nationale, de passer d'inoubliables moments au fond d'une vallée des monts du Jura, au près de mes grands parents (comme Proust, j'avais moi aussi ma madeleine). Merci, enfin, à celle qui m'a

supporté au plus près tout au long de ma thèse, et qui m'a offert la joie immense d'être le papa d'une petite julie : ma chère et tendre, Liset.

Table des matières

Table des figures

Liste des tableaux

Liste des notations

AES	Advanced Encryption Scheme
ASO	Arbitrary Slice Ordering
AVC	Advance Video Coding
AWGN	Addititive White Gaussian Noise
BCJR	Bahl Cocke Jelinek Raviv (MAP [7])
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CIF	Common Intermediate Format
DCT	Discrete Cosinus Transform
DP	Data Partitionning
DPB	Decoded Picture Buffer
DVB-C	Digital Video Broadcasting-Cable
DVB-S(2)	Digital Video Broadcasting-Satellite
DVB-T	Digital Video Broadcasting-Terrestrial
DVD	Digital Versatil Disc
EEP	Equal Error Protection
FEC	Forward Error Correction
FMO	Flexible Macroblock Ordering
GOP	Group Of Pictures
HD	High Definition (Haute définition)
IDR	Instantaneous Decoding Refresh
IP	Internet Protocol
ISO	International Standardization organization
ITU	International Telecommunication Union
JM	Joint Model

JSCC	Joint Source Chanel Coding
JSCD	Joint Source Chanel Decoding
JVT	Joint Video Team
MAC	Medium Access Control
MAP	Maximum A Posteriori
MB	Macro Block
MDC	Multi-Description Coding
MPEG	Moving Pictures Expert Group
MPEG2-TS	MPEG2-Transport Stream
MSE	Mean Square Error
NAL	Network Abstraction Layer
NIST	National Institute of Standards and technology
OSOVA	Optimal Soft-Output Viterbi Algorithm
POC	Picture Order Count
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter Common Intermediate Format
QP	Quantization parameter
RCPC	Rate Compatible Punctured Codes
RCPT	Rate Compatible Punctured Turbocode
ROHC	RObust Header Compression
RTP	Real-Time Transfert Protocol
SNR	Signal to Noise Ratio
SISO	Soft Input Soft Output
SOVA	Soft Output Viterbi Algorithm
SVC	Scalable Video Coding
TNT	Télévision Numérique Terrestre
UEP	Unequal Error Protection
UMTS	Universal Mobile Telecommunication System
UVLC	Universal Variable Length Coding
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VLC	Variable Length Code

Introduction

Chaîne de communication considérée

Depuis la fin des années 1990, le téléphone mobile, l'Internet mobile, et le multimédia font partie du quotidien de la plupart des personnes. De nouveaux systèmes de transmission numérique à haut débit permettent ou permettront d'offrir des solutions de transmissions de données sans fils plus souples, plus configurables, plus fiables, plus intelligentes et plus sûres. En particulier, on voit se développer avec l'avènement de la troisième génération de téléphones mobiles des services de transmission de données multimédias, qu'il s'agisse de flux vidéos pré-enregistrés ou de premières tentatives de visioconférences. Cependant, la transmission de ces données vidéo est encore loin d'être totalement satisfaisante, en particulier car elles nécessitent à la fois un débit disponible sur le canal sans fil bien supérieur et un taux d'erreur résiduel au niveau applicatif bien inférieur à ceux classiquement offerts pour des services de voix. Pour offrir de débit suffisant et la bonne robustesse face aux erreurs ou aux évanouissements inhérents à un canal sans fils, un monde de recherche et d'amélioration des solutions de transmission s'est donc ouvert ces dernières années, dans lequel s'insère le travail présenté dans cette thèse.

Afin de mieux décrire les aspects sur lesquels nous allons intervenir, nous présentons tout d'abord le système de transmission de bout en bout, c'est-à-dire de la source à l'émission à l'utilisateur en réception, que nous considérons. La figure 1 représente le schéma global d'une transmission multimédia dans lequel nous nous plaçons. On y trouve les différents blocs suivants :

• **Codage Source** ("Source Coding") : l'étape de codage de source, qui s'insère dans la couche application de notre chaîne de transmission, permet de compresser les données à transmettre en éliminant la plus grande partie de la redondance (spatiale et temporelle) du flux vidéo. Elle a donc pour objet et intérêt de réduire le débit utile nécessaire à la transmission.

• **Réseau** ("Network") : cette étape correspond à la mise en réseau des données, selon les impératifs de la pile protocolaire standard considérée. Cette couche réseau permet en particulier de prendre en compte l'impact d'éventuelles congestions réseaux et, dans un système de type UMTS, 4G, celui d'une commutation de paquets impliquant perte(s) de

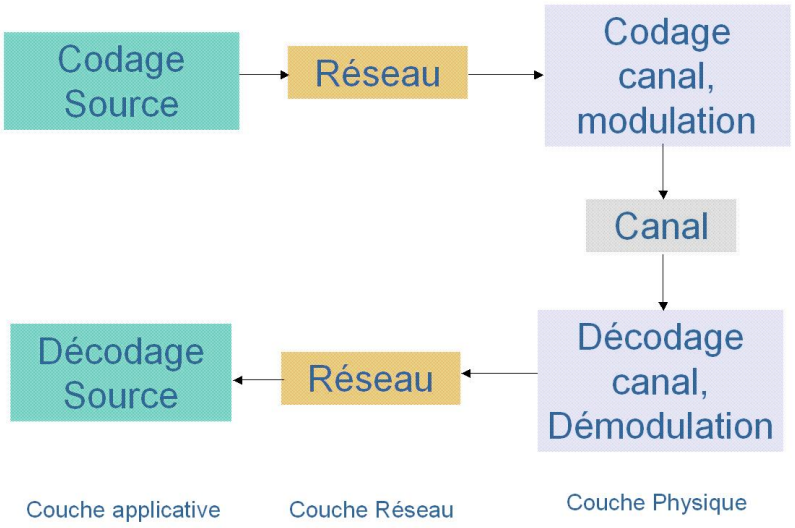

FIGURE 1 – Chaîne de transmission considérée.

paquets et/ou retard(s).

• **Codage Canal et Modulation** ("Channel Coding") : l'étape de codage de canal et modulation s'intègre dans ce que l'on appelle parfois la couche accès radio d'un système de télécommunications. En pratique, le codeur de canal rajoute une redondance structurée à l'information transmise par le codeur source, afin de rendre robuste l'information transmise face aux erreurs et aux évanouissements introduits par le canal physique. Cette redondance est rajoutée de façon contrôlée afin de rester dans les limites de débit imposées par le canal. Suit alors l'étape de modulation qui permet de transformer en signal physique les données numériques issues du codage de canal.

• **Canal Physique** ("Channel") : le canal physique correspond à la transmission sur un vrai médium sans fil. Cette étape regroupe donc toutes les composantes qui perturbent la transmission des informations dans la chaîne de base, donc en particulier le bruit ambiant en réception, les interférences multi-utilisateurs et les multi-trajets.

• **Décodage Canal et Démodulation** ("Channel Decoding") : pendant de l'étape de codage de canal et modulation, on trouve en réception l'étape de démodulation et le décodage de canal. Son objectif est de récupérer le signal en sortie du canal de transmission, de le mettre en forme et aussi de corriger autant que possible, grâce à la redondance introduite au niveau de l'émission, les erreurs introduites par le canal physique.

• **Décodage Source** ("Source Decoding") : l'étape de décodage de source est le pendant de celle de codage de source et permet de reconstruire, par la connaissance du processus de compression employé à l'émission, les données transmises en les interprétant pour

générer un flux vidéo interprétable. Dans les solutions les plus évoluées, des techniques permettant de pallier aux erreurs résiduelles présentes dans le flux remonté au niveau applicatif peuvent être utilisées

L'emploi traditionnel du codage canal indépendamment du codage source est justifié par le théorème de séparation source/canal énoncé par Shannon. Selon ce théorème et sous certaines conditions, des performances optimales peuvent ainsi être atteintes en optimisant indépendamment les deux modules [47]. Cependant, certaines limitations pratiques telles que la complexité du codeur/décodeur, ainsi que la difficulté pratique à atteindre les limites citées par le théorème (comme par exemple l'emploi de mots de codes de taille très grande), font que les hypothèses de celui-ci sont rarement vérifiées dans le cadre d'une transmission multimédia en temps réel. Ceci a donc amené le développement de techniques d'optimisations conjointes de codage/décodage source et codage/décodage canal, qui peuvent prendre en compte les caractéristiques du canal physique, de l'étape de mise en réseau ... Ces différentes approches sont regroupées traditionnellement sous le vocable de techniques de codage/décodage conjoint source/canal (JSCC/D).

En pratique, ces techniques s'emploient à développer des algorithmes suffisamment puissants pour rendre les systèmes de transmission à la fois efficaces et simples à utiliser, prenant en compte les différents éléments que sont une gestion souple des ressources (radio et matérielle), une qualité de service minimale nécessaire ou souhaitée pour l'application cible, et une optimisation de la bande passante disponible, ressource souvent rare et donc chère. Les systèmes de télécommunications mobiles au-delà de la troisième génération, ou systèmes de quatrième génération (4G) sont classiquement perçus comme devant apporter une réponse à ces problèmes, et fournir des solutions de communication globales, en interconnectant de manière transparente une multitude de réseaux et de systèmes hétérogènes. Le principal défi à relever dans l'établissement de tels systèmes sera celui de la définition d'une architecture permettant simultanément l'optimisation de la bande passante et une gestion efficace de la qualité de service (QoS management).

Notre approche, qui suit celle du codage source/canal conjoint, a pour objectif de développer des stratégies où le codage de source et le codage canal sont déterminés conjointement tout en prenant en compte les paramètres du réseau présent dans la chaine de communication et d'éventuelles contraintes utilisateurs telles que l'introduction de sécurité (par exemple par un chiffrement des données transmises) afin d'atteindre la meilleure performance possible de bout en bout. Cet axe de travail s'insère donc naturellement dans la direction générale retenue pour l'établissement des nouveaux systèmes sans fil, puisque notre approche JSCC/D offre la possibilité de faire converser le monde de l'application (codage source, chiffre) et le monde des transmissions (codage canal) afin qu'ils optimisent conjointement l'usage du lien de communications sans fil de bout en bout.

La figure 2 montre le schéma complet d'une telle transmission, telle que nous l'avons considéré dans le cadre de cette thèse, et dans lequel les éléments clefs de la chaîne sont déployés pour permettre les stratégies d'optimisation, en particulier au moyen d'outils de coordination représentés par les contrôleurs conjoints, au sein desquels sont implémentés

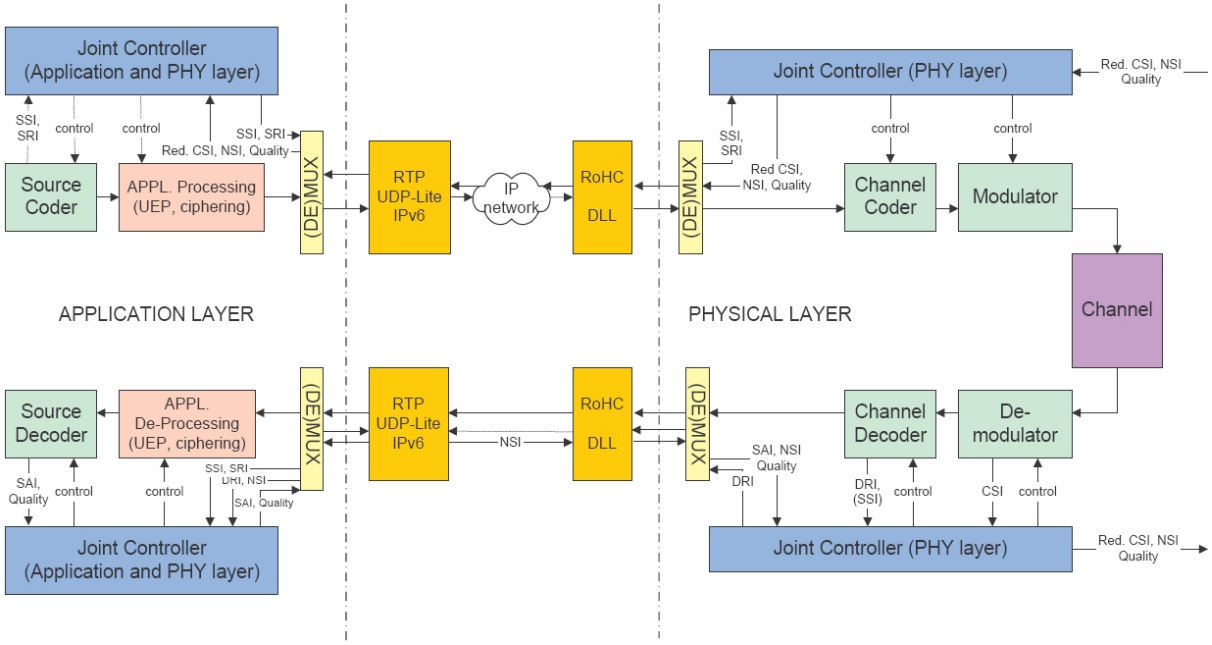

FIGURE 2 – Chaîne de communication complète considérée.

les stratégies de contrôle et d'optimisation conjointe. La tâche de ces contrôlleurs est de gérer l'ensemble de la chaîne de communication afin d'optimiser le rendu en réception pour l'utilisateur, notamment en apportant aux données transmises une protection sélective selon leur importance. Les flèches représentées sur le schéma indiquent les flux de données et de contrôle qui transitent dans le système.

Dans le cadre de cette thèse, l'étape de codage de source a été menée en considérant le codeur source vidéo H.264/AVC, communément appelé H.264. La norme H.264 offre pour une bande de transmission limitée le rapport de compression le plus élevé du moment (comme son extension scalable SVC). De plus, ce standard, élaboré conjointement par les groupes ITU et ISO, est d'ores et déjà déployé ou en cours de déploiement pour des services commerciaux, comme par exemple pour la Télévision Numérique Terrestre (TNT) et satellite (DVB-S2), le format PodCast, et dans certains formats de DVD nouvelle génération (HD-DVD et Blu-Ray).

Organisation de la thèse

Ce document est divisé en quatre chapitres et deux annexes. Certains développements analytiques ont en effet été placés en annexe afin de simplifier au maximum la lecture des idées exposées dans le corps du document.

Le chapitre 1 de ce document débute par une introduction au codage de source et plus particulièrement à la norme H.264, avec un rapide panorama des aspects techniques du standard dans lesquels vont s'intégrer les améliorations et modifications que nous proposons. Suit alors un introduction au codage de canal, avec un rappel du principe du codage correcteur d'erreurs et notamment des codes convolutifs à rendements compatibles (RCPC) qui seront utilisés par la suite. Enfin, ce chapitre se termine par un panorama rapide de l'emploi des techniques de (dé)codage conjoint source/canal.

Dans le chapitre 2, nous proposons deux solutions d'utilisation de la redondance résiduelle présente dans un flux binaire en sortie d'un codeur source H.264. La première solution, qui s'inscrit dans la lignée des décodeurs conjoint source canal, présente l'adaptation d'une méthode de décodage pondéré (ou souple) dans le cadre du standard H.264. La seconde solution, qui permet d'introduire de la confidentialité dans le système, décrit l'intégration d'un chiffre compatible du standard directement au sein du codeur/décodeur source par chiffrement sélectif des données de flux binaire H.264.

Dans le chapitre 3 est introduite une méthode s'inspirant des bancs de filtres pour introduire une propriété de scalabilité (ou mise à l'échelle) temporelle compatible du standard H.264. Cette méthode, qui repose sur le mélange des trames de la séquence vidéo (ce qui nous a amené à lui donner l'appelation *'frame shuffle'* en anglais) permet de fournir des capacités de scalabilité et de meilleure robustesse face à la propagation d'erreurs, notamment en présence d'un canal de propagation bruité, et ceci sans perte en terme de compression.

Le chapitre 4 présente une méthode d'optimisation conjointe de la répartition de débit entre le codeur de source et le codeur de canal au moyen d'un contrôleur applicatif calculant la distorsion globale introduite par les étapes de compression et de protection. Reposant sur le calcul de la sensibilité au bruit des différentes parties du flux binaire, ce modèle original et simple de calcul de la distorsion de bout en bout a été établi dans le cadre de la norme H.264. L'étude de sensibilité est donc déclinée pour différents types de flux H.264 codés selon des modes standards (normal, partition de données, ou *Data Partitionning* en anglais) mais aussi pour le mode mélange de trame introduit au chapitre précédent. Les résultats de simulation fournis dans le cadre de la transmission sur différents canaux (canal à bruit additif blanc gaussien, canal de type WIFI, ...) mais aussi avec ou sans codage correcteur d'erreur montrent la validité du modèle et de l'approche conjointe.

En conclusion, nous fournissons dans la dernière partie un résumé des travaux présentés dans ce document ainsi que quelques perspectives de recherches ultérieures sur le sujet traité.

Chapitre 1

État de l'art

1.1 Le codage vidéo

1.1.1 Historique

La compression vidéo fait partie intégrante de nombreux applications multimédia disponibles aujourd'hui. Pour certaines applications et certains équipements, par exemple les lecteurs DVD, la transmission de la télévision numérique, la vidéo conférence, la sécurité vidéo, et les caméras numériques, une bande de transmission limitée ou une capacité mémoire limitée contribuent à une demande de rapports de compression toujours plus élevés. Pour répondre à ces différents scénarios, plusieurs standards de codage vidéo ont vu le jour et se sont succédés au cours de ces dernières décennies comme le montre la figure 1.1.

La norme vidéo du codage MPEG-2 [1], qui a été développée il y a environ 15 ans, est une technologie utilisée aujourd'hui dans le monde entier dans tous les systèmes de télévision numériques. Elle permet une transmission efficace des signaux de télévision par satellite (DVB-S), par câble (DVB-C) ou terrestre (DVB-T). De plus, elle permet aussi la transmission d'autres médias sur l'Internet haut débit (xDSL). Mais même sur le système DVB-T, la bande passante disponible n'est pas toujours suffisante. Par conséquent, le nombre de programmes devient limité, indiquant ainsi un besoin de compression vidéo à améliorer encore.

En 1998, Le *Video Coding Experts Group* (VCEG, ITU-T SG16 Q.6) a débuté un projet appelé H.26L ayant comme objectif de doubler l'efficacité de compression en comparaison avec toutes les autres normes vidéo existantes. En décembre 2001, devant les premiers résultats de ce projet, le VCEG et le *Moving Pictures Expert Group* (MPEG. ISO/IEC

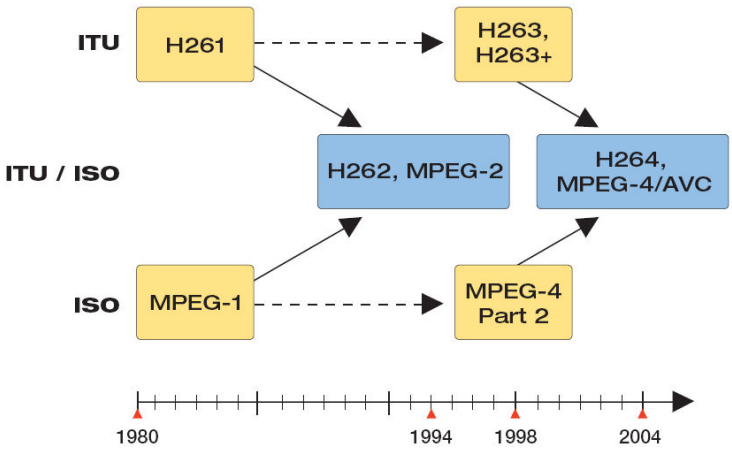

FIGURE 1.1 – Historique des différents standards ISO et ITU

JTC) ont formé une équipe de travail conjointe, le *Joint Video Team* (JVT) pour mener à terme le nouveau standard de compression vidéo H.264/AVC [2].

1.1.2 Généralités

Luminance-Chrominance

Il existe de nombreuses manières de représenter une image. Dans le cas général d'une image en couleurs, chaque pixel (élément de base, point) de l'image peut être représenté par un triplet :

- soit (R,G,B), où R désigne la quantité de rouge, G celle de vert et B celle de bleu dans la composition de la couleur du pixel ;
- soit (Y,U,V), où Y désigne la luminance, et U et V contiennent des informations de chrominance, c'est-à-dire la différence entre la couleur du pixel et un gris de même luminance.

Hormis certains formats professionnels, qui utilisent le format RGB et stockent sur 10 ou 12 bits chaque valeur des composantes du pixel, le monde de l'image diffuse et échange ses contenus vidéo couleur dans la base YUV, où chaque composante est représentée sur 8 bits. L'oeil humain étant moins sensible aux variations de chrominance qu'aux variations de luminance Y, les chrominances orthogonales U et V sont sous-échantillonnées horizontalement et verticalement. La figure 1.2 représente le format d'échantillonnage de chrominance le plus utilisé par les divers codeurs vidéo dit format 4 : 2 : 0. Dans ce format, il y a un élément luminance (Y) pour chaque pixel, et les éléments de chrominance (U et V) un pixel sur deux, et une ligne sur deux en alternance.

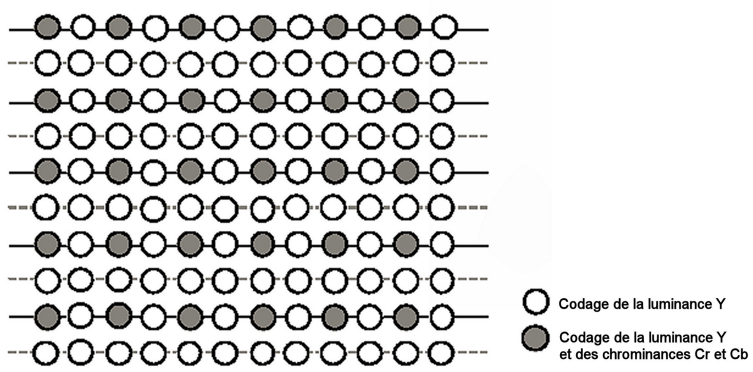

Figure 1.2 – Le codage des pixels pour chaque composante dans la norme 4 :2 :0

Évaluation de la qualité

L'expression et la mesure de la qualité vidéo sont une donnée d'études en soi, aucune solution objective ne satisfaisant parfaitement au critère de bonne représentation des résultats qui seraient obtenues par une évaluation subjective par des observateurs humains. Comme la très grande majorité de la communauté scintifique, nous ferons donc ici appel à des méthodes permettant d'évaluer automatiquement la qualité intrinsèque d'une image comme le PSNR (*Peak Signal to Noise Ratio*), qui correspond au rapport signal à bruit crête-à-crête, et le MSE (*Mean Square Error*) qui équivaut à l'erreur quadratique moyenne.

Ces deux critères permettent de mesurer la distorsion entre l'image reconstruite et l'image originale. Le MSE est défini pour deux images I_0 et I_1 de taille $M \times Q$ comme suit :

$$MSE = \frac{1}{M.Q} \sum_{i=1}^{M} \sum_{j=1}^{Q} (I_0(i,j) - I_1(i,j))^2 \tag{1.1}$$

et le PSNR s'écrit alors (dans le cas standard d'une image où un pixel d'une composante est codé sur 8 bits) :

$$PSNR = 10\, log_{10} \left(\frac{255^2}{MSE} \right) \tag{1.2}$$

Les valeurs typiques de PSNR obtenues pour des images de bonnes qualité sont supérieures à 30 dB.

1.2 La norme H.264

La structure définie par H.264/AVC regroupe une couche vidéo (*Video Coding Layer* ou VCL), qui représente le contenu vidéo, et une couche réseau (*Network Abstraction Layer*

ou NAL), qui comprend les en-têtes appropriées pour le transport de l'information par des couches transports ou des supports de stockages particulier. La conception du VCL, comme dans toute norme antérieure (de l'ITU-T et d'ISO/IEC JTC1 depuis H.261), suit l'approche de codage vidéo par blocs.

L'algorithme de base du codage source est constitué d'une prédiction d'images (*Inter-picture prediction*) pour exploiter la corrélation statistique temporelle, et d'une transformation orthogonale permettant la décorrélation des dépendances statistiques spatiales. Etonnament, H.264 réussit à obtenir des gains d'efficacité de compression par rapport aux normes vidéo antérieurs sans changer notablement le schéma de codage prédictif, ou introduire d'éléments réellement innovant, comme le montre le schéma de codage du standard H.264 (voir figure 1.3).

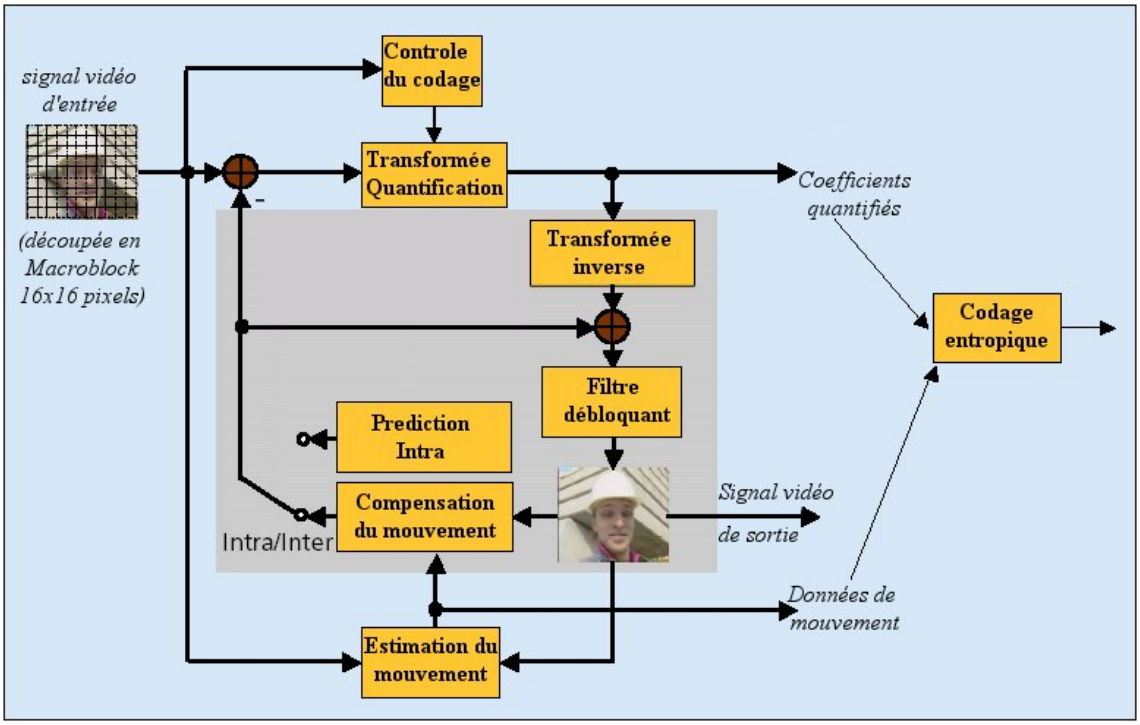

FIGURE 1.3 – Schéma de codage principal du standard H.264

1.2.1 Vue d'ensemble du standard H.264

La couche réseau-NAL

La VCL, qui est décrite dans la section suivante, a été créé pour contenir l'information relative aux signaux vidéo. Quant à la NAL, elle a été élaborée pour organiser ces données et pour fournir des informations d'en-têtes adaptées au transport par les couches de

transmission ou par les supports de stockage. Toutes les données sont contenues dans les unités de NAL, dont chacune contient un nombre entier d'octets. Une unité de NAL est élaborée dans un format générique commun aux systèmes de transmission par flux binaire complet (*bitstream* ou *codestream*) ou par paquets.

La couche vidéo-VCL

Le codage de la couche vidéo de H.264/AVC est réalisé dans le même esprit que les précédentes normes comme le standard MPEG-2. Elle consiste principalement en une double prédiction temporelle et spatiale de la séquence. H.264 s'inspire de la prédiction classique de type MPEG d'une séquence, ce qui amène différents types d'images codées :

• **Image Intra** : la première image d'une séquence est une image *Intra* (ou I), c'est-à-dire une image qui est codée sans utiliser d'information autre que celle contenue dans l'image même. Pour exploiter la redondance spatiale d'une *Intra*, l'image est découpée en plusieurs blocs de 4x4 pixels (voir figure 1.4) eux même regroupés en macroblocs (16x16 pixels) traités non plus indépendamment comme les versions précédentes de MPEG, mais selon le contexte (le codage intrinsèque du macrobloc dépendra de sa place dans l'image, et de ses macroblocs adjacents comme détailé au paragraphe 1.2.2). Chaque bloc subit une transformation orthogonale (permettant de décorréler le signal vidéo), une quantification et un codage entropique.

• **Image Inter** : Les autres images, appelées images *Inter*, sont prédites à partir de l'image Intra précédente ou d'autres références, grâce à l'étape de compensation de mouvement des différents macroblocs. Il existe ainsi deux types d'images *Inter* :

- les images de type P (Prédites) : ce sont des images prédites en fonction de l'image Intra précédente ainsi que d'autres images de type P ;
- les images de type B (Bidirectionnelles) : ce sont des images prédites en fonction de l'image Intra et/ou des images de type P, qui précèdent et qui suivent déjà décodés.

Le schéma 1.5 explicite la prédiction des images P et B dans une séquence type d'images.

Dans la norme H.264/AVC, il existe un type particulier d'Intra nommée "Instantaneous Decoding Refresh" ou (IDR). Ce type de trame est une trame codée en Intra comme n'importe quelle autre trame mais elle a pour but d'effacer de la mémoire du codeur (et du décodeur) toutes les autres trames précédemment codées (et décodées), et sur lesquels les images Inter suivantes peuvent se référencer. Ce type de trame permet par exemple de se resynchroniser dans le cas d'une transmission avec pertes.

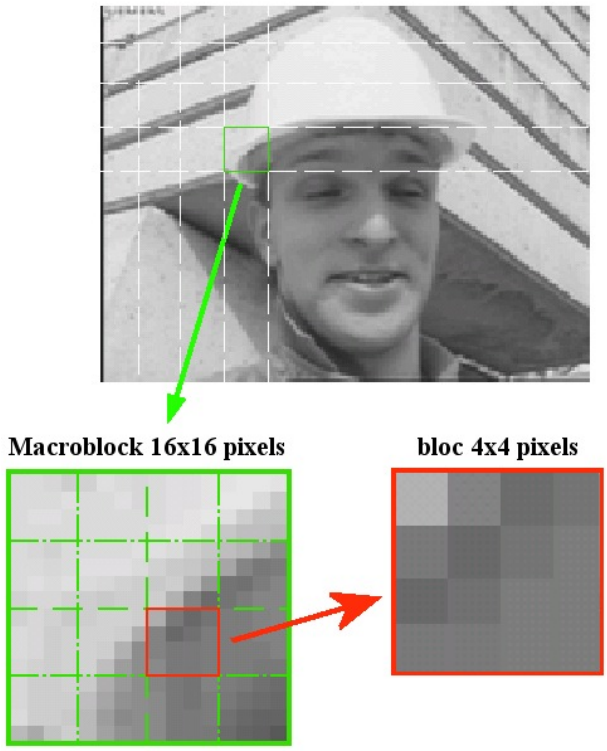

Figure 1.4 – Subdivision d'une image en macroblocs de 16x16 pixels et en blocs de 4x4 pixels

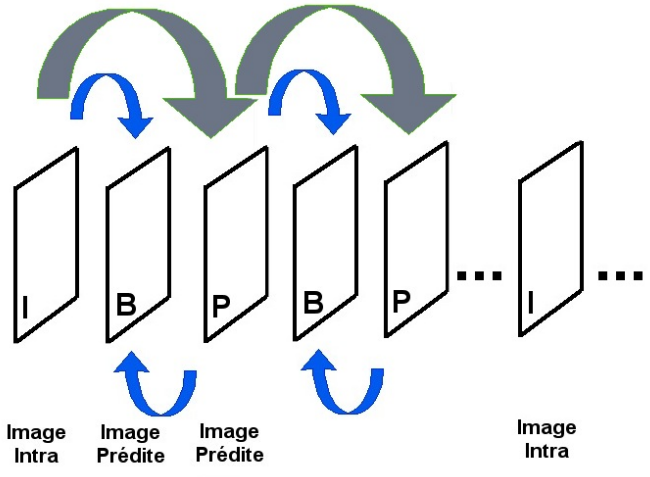

Figure 1.5 – Séquence type d'images et illustration des différentes dépendances possibles

1.2.2 Techniques propres du standard H.264

Découpage de l'image en Slice (imagettes)

H.264 peut gérer un système de portions d'images indépendantes, nommées *slices*. Elles subdivisent une image et sont codées indépendamment. Ces slices peuvent avoir des tailles et des formes différentes comme illustrée sur la figure 1.6. L'intérêt d'utiliser un tel découpage est premièrement de limiter la propagation d'erreurs résiduelles, et deuxièmement de pouvoir définir des zones d'intérêt dans l'image considérée.

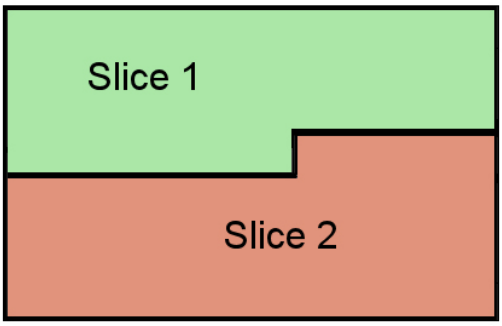

Figure 1.6 – Exemple d'image divisée en deux slices

Prédiction Intra

Lorsque l'on considère des images *Intra*, les macroblocs sont d'abord prédits par rapport aux blocs et aux macroblocs précédemment codés dans la même slice. Cette prédiction est ensuite soustraite au bloc courant avant son codage. Dans le cas des échantillons de luminance (*luma*), on peut réaliser la prédiction soit bloc par bloc, soit macrobloc par macrobloc : il y a ainsi au total 9 prédictions possibles pour chaque bloc, comme illustré dans la figure 1.7 (où les lettres A,B,C,...,M, représentent les pixels des blocs adjacents), et 4 prédictions possibles pour chaque macrobloc, comme illustré dans la figure 1.8.

Toutefois, il est à noter qu'il existe un autre mode de prédiction basé sur des blocs de taille 8 par 8 pixels (8x8), identique au mode de prédiction 4x4 mais utilisé uniquement pour le profil "haute définition" (voir partie 1.2.4).

Quant aux échantillons de chrominance (ou *chroma*) bleu et rouge (respectivement, C_B et C_R), il y a 4 possibilités de prédiction pour chaque macrobloc dans le mode 8x8 pixels, similaires aux modes de prédiction 16x16 des composantes *luma* représentés en figure 1.8.

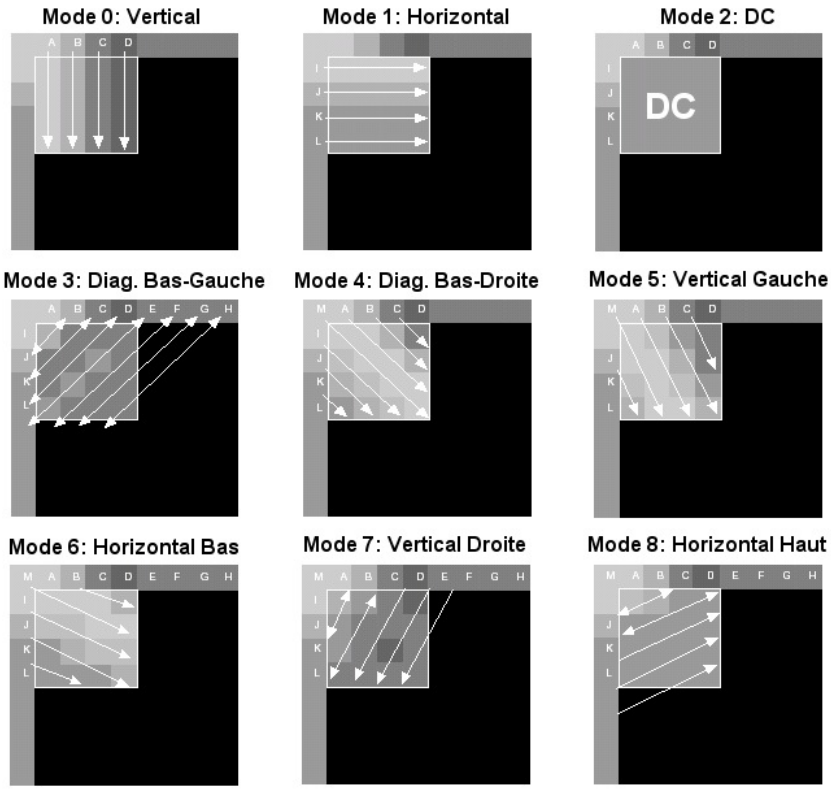

Figure 1.7 – Les modes de prédiction 4x4 (luma)

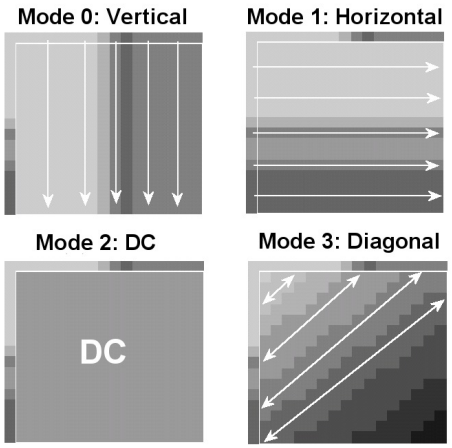

Figure 1.8 – Les modes de prédiction 16x16 (luma)

Prédiction de mouvement des images Inter

La compensation de mouvement permet d'introduire un autre axe de prédiction, en autorisant l'exploitation de la redondance temporelle entre des images chronologiquement

proches. Cela consiste à ne coder que la différence entre l'image N à coder et une image prédite à partir de l'image N-1. L'image prédite est construite par estimation et compensation de mouvement en utilisant des macroblocs de l'image N-1 les plus ressemblants à ceux de l'image N. Le mouvement apparent des objets entre les deux images, supposé être une translation, est représenté par des vecteurs mouvements de macroblocs.

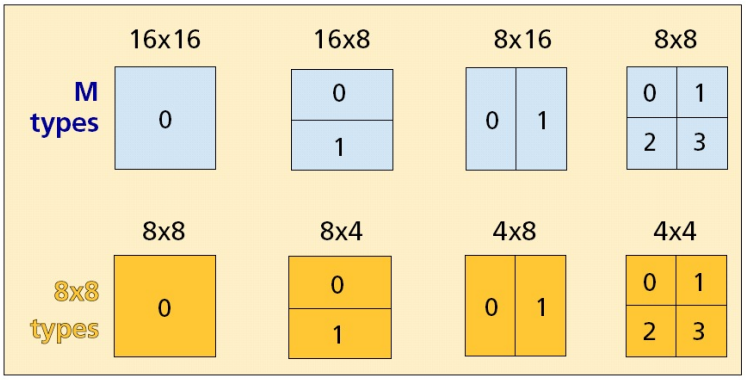

Figure 1.9 – Les différentes configurations des subdivisions des macroblocs pour l'estimation de mouvement

Les différentes tailles de blocs prédits Ce codage par prédiction rend les images interdépendantes, ce qui a pour effet de limiter la capacité d'accès aléatoire à une image quelconque, puisqu'il faut alors décoder auparavant toutes les images dont elle dépend. Afin de limiter cet effet, les images sont regroupées en GOP (*Group of Pictures*), composés classiquement dans les formats MPEG et ITU des trois types d'images vus précédemment (I, P et B). Il est à noter que cette organisation peut également permettre de modifier rapidement le débit du flux vidéo en modifiant la structure du GOP : une réduction du débit est par exemple obtenue en réduisant le nombre d'images I au profit des images P et B, qui nécessitent moins de ressources, du fait de l'efficacité plus grande du codage par prédiction de mouvement.

Pour la prédiction de mouvement, comme indiqué dans la figure 1.9, chaque macrobloc dans une image *Inter* peut être divisé en blocs de taille variable : 16x16, 16x8, 8x16 ou 8x8. Mais à l'intérieur de ces divisions, on peut les subdiviser de nouveau par des blocs de tailles variables : 8x8, 4x8, 8x4, ou 4x4. A la différence des standards vidéo précédents, le standard H.264/AVC supporte donc des tailles de blocs variables indépendants comme l'illustre la figure 1.10, où l'on voit une subdivision plus fine sur les zones les plus fortes en détails : ces configurations permettent de mieux s'adapter à la résolution de l'image, et d'être plus précis dans la compensation de mouvement.

Les vecteurs de mouvement La prédiction de mouvement repose sur le calcul d'un vecteur de mouvement, donnant la référence du bloc de pixels précédent le plus

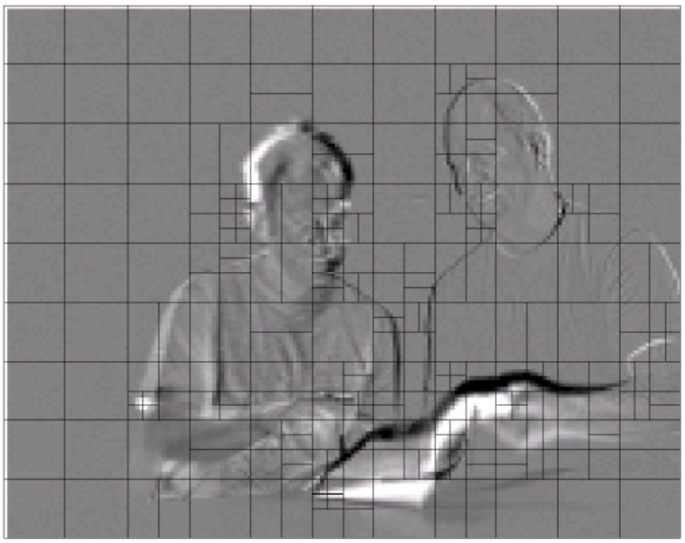

Figure 1.10 – Exemple de partitions de macroblocs pour l'estimation de mouvement

proche, auquel s'ajoutera une correction nommée "résidu". Pour une meilleure précision des vecteurs de mouvement, les prédictions peuvent être calculées au 1/4 ou au 1/8 de pixel, comme indiqué dans la figure 1.11, où l'on montre l'exemple d'une prédiction de précision entière (au centre) et d'une prédiction "quart-pixelique" (à droite). Les résolutions en fractions de pixels sont obtenues par interpolation sur les pixels existants. Elles permettent

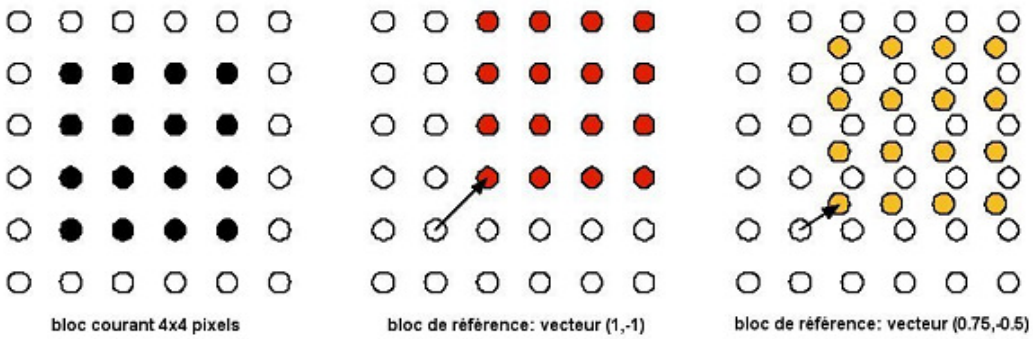

Figure 1.11 – Exemples de différentes prédictions de mouvement

une amélioration conséquente de la précision de calcul du mouvement, d'où une bonne optimisation du procédé.

Prédiction multi-images La norme H.264 offre la possibilité de ne pas utiliser une seule référence mais plusieurs. Ainsi la prédiction de mouvement peut utiliser jusqu'à 16 différentes images de références selon le profil utilisé (voir paragraphe 1.2.4). Cette possibilité est illustrée sur la figure 1.12, où l'on voit qu'une trame peut se référer à des

images différentes. Des tests effectués ont montré qu'elle permettrait de gagner un facteur de compression d'environ 10% par rapport à une prediction mono-référence.

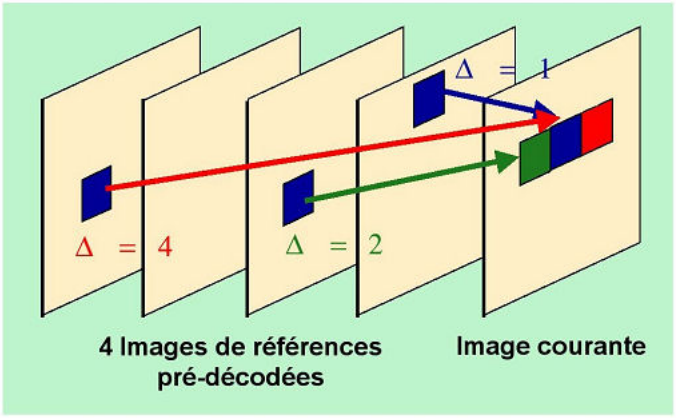

Figure 1.12 – Exemples de prédiction multi-références

Figure 1.13 – Exemples illustrant l'efficacité du filtre reconstruction (à gauche : sans filtre ; à droite : avec filtre)

Filtre de reconstruction Le standard H.264 propose également une option qui permet l'usage d'un filtre de lissage (*«deblocking»*) ou de reconstruction, qui s'opère sur les contours verticaux et horizontaux de chaque macrobloc, avec 4 niveaux de lissage. Ce filtre, dont le rôle est d'éviter les artefacts dus au codage par blocs, réalise un lissage des zones voulues aux bords des macroblocs, comme illustré par le figure 1.13. Ce filtre n'amène pas véritablement de gain en compression, mais apporte un gain en qualité « subjective » non négligeable en reconstruction. Néanmoins, on notera que ce n'est pas une innovation à proprement parler car de nombreux décodeurs (MPEG-2, MPEG-4, H.263 ...) commerciaux proposent eux aussi ce type de filtrage, bien que leur norme ne l'impose pas.

Transformée et Quantification

Transformée entière À la différence des standards précédents (hormis MJPEG 2000), H.264 n'utilise pas la transformée en cosinus discrète (DCT). Elle utilise à la place une transformée entière sur des blocs de 4x4 pixels qui offre plusieurs avantages par rapport à la DCT classique :

• la transformée ne contient que des divisions ou des multiplications par deux : elle est donc moins complexe et plus rapide que la DCT, puisqu'elle n'utilise que des additions et des décalages (*shifts*). Cela permet d'effectuer les calculs de la transformée sur un calculateur 16 bits à virgule fixe, sans perte de précision.

• la transformée entière possède une transformée inverse exacte, à la différence de la DCT qui utilise des valeurs « flottantes », ceci évite les erreurs d'arrondi à la reconstruction, qui étaient systématiques dans les précédents standards.

• l'application de la transformée sur un bloc 4x4 pixels et non plus avec un bloc 8x8 pixels permet de réduire les "effets de blocs" traditionnellement rencontrés lors des forts taux de compression avec une DCT 8x8 (employée dans les précédents standards MPEG).

En pratique, les coefficients de la transformée entière sont une approximation de ceux de la DCT, qui reste parmi les transformées les plus efficaces connues actuellement. Ils possèdent donc les mêmes propriétés d'orthogonalité, et quasiment les même propriétés de décorrélation [61].

La transformée entière définie par H.264 d'une matrice X est donc donnée par :

$$Y = HXH^T, \text{ avec } H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}$$

Cependant, dans le cas d'une image Intra, une option permet d'aller plus loin et de regrouper les 16 coefficients DC (composante continue) de chaque bloc d'un macrobloc de luma et les 8 coefficients DC des composantes chromatiques associées (comme illustré par la figure 1.14), afin de calculer une nouvelle transformée sur ces blocs. C'est une transformée d'Hadamard qui est alors utilisée. La transformée de Hadamard d'une matrice W est donnée par :

$$Y = H_D W H_D^T, \text{ avec dans le cas Luma } H_D = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}$$

et dans le cas chroma, pour chaque composante C_B et C_R,

$$H_D = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

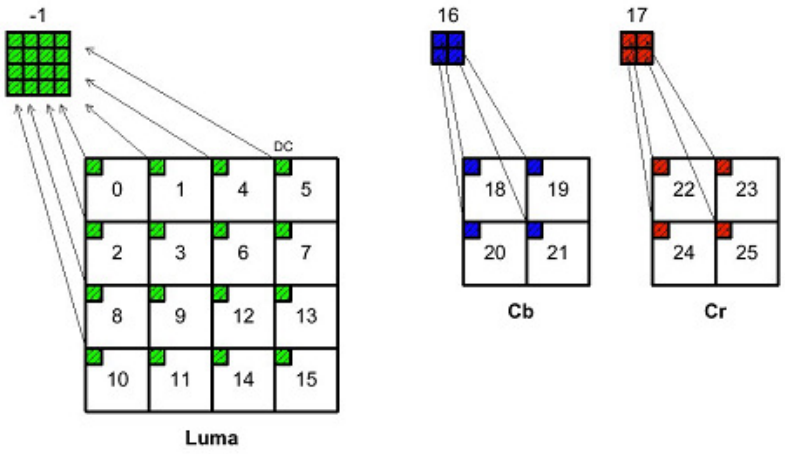

Figure 1.14 – Regroupement des coefficients DC de tout le macrobloc, après transformée entière

Quantification Tout d'abord, à la différence du standard précédent (MPEG-4 part.2), la norme H.264/AVC (ou MPEG-4 part.10) ne supporte pas l'option « dead zone », pour laquelle le pas de quantification centrée sur la valeur zéro est plus large que celle des autres intervalles. À la place, elle utilise une quantification scalaire uniforme. Le tableau 1.1 ré-

QP	0	1	2	3	4	5	6	7	8	9	10	···
Q_{Step}	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	···
QP	···	18	···	24	···	30	···	36	···	42	···	51
Q_{Step}	···	5	···	10	···	20	···	40	···	80	···	224

Table 1.1 – Tableau d'équivalence entre le paramètre QP et le pas de quantification

capitule l'équivalence entre le paramètre QP (le paramètre de quantification de la norme) et la valeur du pas de quantification à utiliser pour quantifier les coefficients issus de la transformée entière qui suit une évolution logarithmique : lorsque on augmente de 6 la valeur de QP, on double le pas de quantification. La norme autorise donc seulement 52 valeurs pour QP (*Quantization Parameter*).

Codage entropique

La norme autorise deux types de codages entropiques, le CABAC (*Context-Based Adaptive Binary Arithmetic Coding*), et le CAVLC (*Context-Based Adaptive Variable-Length Coding*). Le premier est un codage arithmétique, le deuxième utilise divers codes VLC (*Variable Length Code*) dont principalement celui de type Exponentiel-Golomb. Nous nous sommes efforcés d'appliquer nos solutions uniquement au mode CAVLC puisqu'il est moins complexe et plus adapté aux applications choisies dans cette thèse, que le co-

deur arithmétique. De plus, à la différence du CABAC, le CAVLC fait partie de tous les profils prévus par la norme H.264 (voir paragraphe 1.2.4). C'est pourquoi nous ne présenterons principalement que ce mode de codage. Cependant, sauf mention explicite, les algorithmes proposées dans cette thèse peuvent être adaptés au cas de la compression arithmétique. Le codage CAVLC code les coefficients quantifiés des transformées entières, les vecteurs de compensation de mouvement, et les autres données utiles (type de macro-bloc, paramètre de quantification, ...) avec le même type de code à taille variable, à savoir le code VLC Exponentiel-Golomb (appelé aussi UVLC : *Universal Variable Length Coding*) détaillé ci-après. En ce qui concerne les autres données, il existe plusieurs tableaux de correspondance dans la norme que nous ne récapitulerons pas ici.

Codage Exponentiel-Golomb Ce codage est construit régulièrement comme indiqué dans le tableau 1.2 :

Numéro de code	Mots de code
0	1
1	01 0
2	01 1
3	001 00
4	001 01
5	001 10
6	001 11
7	0001 000
8	0001 001
...	... ...
	bits de préfixes de suffixes

Table 1.2 – Tableau d'équivalence des mots de codes Exponentiel-Golomb

Les symboles de forte probabilité (i.e. faibles valeurs) ont donc les tailles les plus petites. En effet, les petits vecteurs mouvement sont les plus fréquents, ce qui amène à les représenter par un symbole court pour obtenir un codage entropique efficace. De plus, avant d'être codé par VLC, il ne faut pas oublier que les vecteurs de mouvement font tout d'abord l'objet d'une approximation contextuelle (c'est-à-dire par rapport aux blocs adjacents codés précédemment), ce qui amène à coder une différence qui est forcément assez faible dès que la prédiction est efficace.

Codage VLC adaptatif-CAVLC Ce codage sert uniquement à coder les données résiduelles issues de la transformée entière. Il a été élaboré pour tirer profit de plusieurs caractéristiques de la quantification de blocs 4x4 pixels. Explicitons ces caractéristiques pour bien comprendre le principe de codage. Après prédiction, transformation et quantification, les blocs contiennent généralement beaucoup de zéros. Le CAVLC utilise un

codage de type « run-level », c'est-à-dire qu'il procède en donnant un couple de chiffres : le nombre de zéros précédant le coefficient non nul, puis la valeur du coefficient. Par exemple, la séquence {0,0,0,1,0,0,-3,5}, sera codée par la séquence de couples suivante : {(3,1),(2,-3),(0,5)}. De même, les coefficients non nuls des « hautes-fréquences » après un scanning « zig-zag » (figure 1.15) sont souvent une suite de +/-1 appelés *Trailing ones* pour "1 de fin de transformée". Le CAVLC tire profit de cette propriété en codant unique-

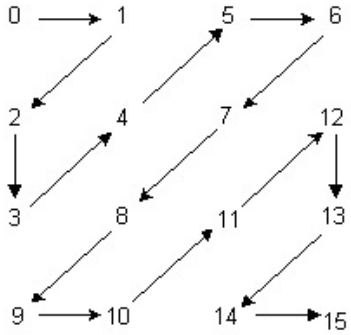

Figure 1.15 – Scanning Zig-Zag

ment le nombre de « +/-1 ». Il a été remarqué également que le nombre de coefficients non nuls est corrélé avec celui des blocs adjacents. Le CAVLC utilise donc une table spécifique en fonction du nombre des coefficients non nuls des blocs voisins.

Les valeurs des coefficients des « basses fréquences » sont souvent plus grandes que celle des « hautes-fréquences ». En scannant tous les coefficients en « zig-zag inverse», c'est-à-dire en débutant par les « hautes-fréquences », on procède astucieusement en utilisant différentes tables en fonction des coefficients déjà codés dans le bloc : plus on code des fortes valeurs (et plus on se rapproche de la composante continue) et plus on espère obtenir de valeurs encore plus grandes, donc on s'adapte en changeant la table pour le symbole suivant.

Le codage CAVLC suit donc ces 5 étapes suivantes :

1 - Codage du nombre de coefficients et des trailing-ones (Coeff_token) Le premier code VLC représente donc le couple "nombre de coefficients non nuls" (Total_Coeff) et le "nombre de trailing-ones". Total_coeff est compris entre 0 et 15, quant à Trailing-ones, il sera compris entre 0 et 3. Et s'il y a plus de 3 coefficients +/-1, alors ils seront codés comme les autres coefficients non nuls. Le choix de la table dépend de la position du bloc dans l'image : il dépend du nombre de coefficients non nuls (Total_Coeff), des blocs se trouvant à gauche (N_L) et au-dessus (N_U) précédemment codés, comme nous le montre la figure 1.16.

Pour savoir quelle table utiliser, on calcule le paramètre N de la manière suivante :

- Si les deux blocs adjacents sont disponibles (i.e. on ne se trouve pas aux bords de l'image ou d'une slice), alors le paramètre $N=(N_U+N_L)/2$.

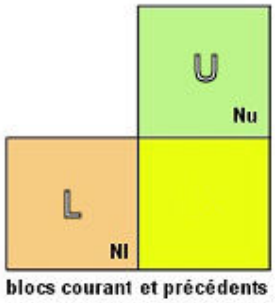

Figure 1.16 – Blocs courant, adjacents (U et L)

- Si un seul bloc adjacent n'est disponible, alors le paramètre N est égal aux nombres de coefficients non nuls du bloc disponible.
- Si aucun n'est disponible (i.e. on se trouve a l'extrémité en haut à gauche de l'image ou de la slice), N=-1.

On choisira en fonction de la valeur de N la table à utiliser parmi les cinq disponibles.

2 - Codage des Trailing-ones Pour chaque Trailing-ones (suite de +/-1) obtenu par Coeff_token, un seul bit est codé pour préciser le signe (0=+, 1=-).

3 - Codage des coefficients non nuls restants Les valeurs des coefficients non nuls restants sont encodées en sens inverse (les « hautes-fréquences » en premier). Le choix de la table VLC pour encoder chaque coefficient dépend de la succession des valeurs encodées. Il y a 7 tables différentes (VLC_0, VLC_1,$\cdots$ et VLC_6) qui sont basées sur un modèle d'Exponentiel-Golomb biaisé. Les conditions de passage étant très compliquées, puisqu'il existe énormément d'exceptions contextuelles, nous n'étudierons que le principe général. Pour simplifier, le premier coefficient est codé par VLC_0, et à chaque fois que la valeur codée dépasse un certain seuil, on change de table (on incrémente la table : VLC_0 -> VLC_1). Pour chaque table, le seuil n'est pas le même.

4 - Codage du nombre total de zéros "Totalzeros" est la somme de zéros précédant le dernier coefficient non nul (i.e. le dernier des « hautes-fréquences »). Cette valeur est codée par une table VLC qui dépend directement du nombre de coefficients non nuls. En effet, le nombre Total_Coeff ajouté à Totalzeros ne peut pas dépasser 16. Il y a donc 15 tables différentes.

5 - Codage des plages de zéros Les plages de zéros précédant chaque coefficient non nul (run_before), sont codées seulement s'il reste des plages de zéros non vides à coder. Il n'est pas nécessaire non plus de coder la plage de zéros précédant le premier coefficient non nul (i.e. premier coefficient « basse-fréquence »). Pour chaque valeur de run_before à coder, on utilise évidemment une table VLC dépendant du nombre de zéros restant à coder.

1.2.3 Options du standard H.264

H.264/AVC comprend aussi de nombreuses options qui lui donnent plus de flexibilité, et qui le rendent plus "efficace" en terme de compression et plus "robuste" aux erreurs. Voici une liste non exhaustive des différentes options décrites dans la norme :

Notions de slices indépendants

L'ordonnancement flexible des macroblocs (FMO : *Flexible macrobloc ordering*) (figure 1.17) et l'ordonnancement arbitraire des slices (ASO : *Arbitrary slice ordering*) sont des techniques de restructuration des régions fondamentales de l'image (subdivisée en macroblocs). Typiquement utilisés pour améliorer la résistance aux erreurs et aux pertes, ces techniques peuvent être utilisées aussi pour définir des régions d'intérêt.

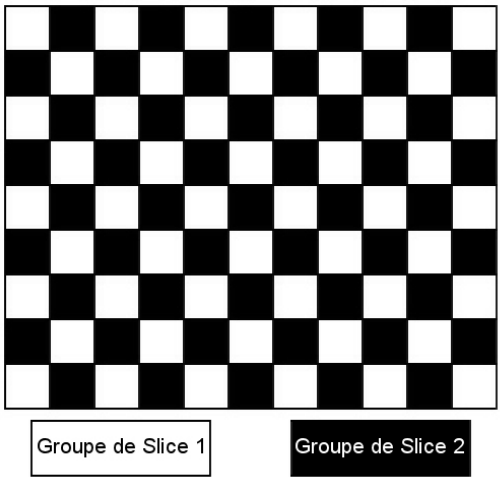

Figure 1.17 – Exemple de 2 slices codées en FMO en mode dispersé

Partitionnement des données

Le partitionnement des données (DP : *Data partitionning*) peut être utile dans toute situation où des erreurs de transmission peuvent survenir, comme les environnements de diffusion (*streaming* ou *broadcast*). Le partitionnement des données consiste à organiser en une manière différente les données dans le flux. Chaque macrobloc inclut habituellement un vecteur de mouvement (prédiction) et une information sur les coefficients de la transformée entière (le résidu ou *residual*).

Dans la norme, on réorganise ainsi dans trois NAL différentes les informations des macroblocs contenues dans chaque slice :

- les données d'en-têtes ainsi que les vecteurs de mouvement ou le mode de prédiction Intra,
- les coefficients de la transformée pour les macroblocs codés en mode *Intra* (i.e. macrobloc codé sans prédiction de mouvement),
- les coefficients de la transformée résiduelle résultants de la prédiction de mouvement (macroblocs codés en mode *inter*)

Cela permet au flux d'être plus résistant à la transmission d'erreurs. En effet, dans cette modalité, le vecteur de mouvement et la texture sont séparés (donc non entrelacés dans le codage de chaque macrobloc) et groupés dans des paquets vidéo. Chaque paquet vidéo est une entité indépendante dans le flux et peut être décodé séparément des autres. L'utilisation du partitionnement de données peut aussi permettre l'activation d'une série d'outils prévus dans le standard pour permettre la récupération d'erreur et la resynchronisation des paquets.

Ordonnancement flexible des images

Le comptage de l'ordre des images permet de conserver l'ordre des images décodées indépendamment des informations de minutage contenues, par exemple, dans les couches réseaux ou protocolaires de télédiffusion. H.264/AVC possède deux niveaux d'ordonnancement, le premier nommé "*Frame_number*" donne l'ordre de codage/décodage des trames, le deuxième nommé "*Picture Order Count*" (POC) donne l'ordre d'apparition de l'image. Cette option permet ainsi la création de sous-séquences par l'inclusion optionnelle d'images supplémentaires entre d'autres images, comme par exemple les trames B, comme on le verra plus en détail dans le chapitre 3.

1.2.4 Profils et niveaux

Les profils et les niveaux définissent les points de conformité auxquels un décodeur doit se conformer pour être compatible avec la norme H.264. Ces derniers sont conçus pour faciliter l'interopérabilité entre plusieurs applications reposant sur la norme H.264 et qui requièrent donc des configurations semblables. Un profil définit un ensemble d'outils de décodage ou d'algorithmes qui peuvent être utilisés pour décoder un flux compatible tandis qu'un niveau impose des contraintes à certains paramètres clés du flux, comme notamment la taille maximale de la zone tampon de décodage (DPB : *Decode picture Buffer*)

La norme H.264 définit quatre profils, chacun ciblant une classe d'applications précise :

• **Le profil de base (*baseline profile*)** est principalement recommandé pour les applications bas-coût (mobiles, visio-conférence,$\cdots$) qui utilisent peu de ressources.

• **Le profil principal (*main profile*)** est prévu pour les applications grand public

de diffusion et de stockage.

• **Le profil étendu (X ou *extended profile*)** a été conçu pour la diffusion en flux (*streaming*) des vidéos et pour la transmission sans fil.

• **Le profil haute définition (*High profile*)** créé plus récemment a été introduit pour les applications "haute définition" (*HD*).

Tous les décodeurs conformes à un profil spécifique doivent prendre en charge toutes les fonctionnalités appartenant à ce profil. Les décodeurs sont tenus d'être compatibles avec un ensemble particulier de fonctionnalités prises en charge dans un profil. Ainsi, le codage CABAC n'existe que dans les profils principal et haute définition, alors que le codage CAVLC est commun à tous les profils. Le mode "data-partitionning" , quant à lui, n'est utilisable que dans le profil étendu. De part leur domaine d'application souhaité, c'est-à-dire soit le plus large possible, soit pour des applications de diffusion sans fil, nous ne décrivons et n'utilisons dans cette thèse que les options issues des profils de base et étendu.

Numéro de niveau	Macroblocs par seconde maximum	Nombre maximum de macroblocs	Débit en Kbit maximum	Taille maximum du DPB en Mo	Exemple de résolution et cadence (Fps)
1	1485	99	64 Kbit/s	148,5	176x144/15.0
1b	1485	99	128 Kbit/s	148,5	176x144/15.0
1.1	3000	396	192 Kbit/s	337,5	176x144/30.0
1.2	6000	396	384 Kbit/s	891	352x288/15.0
1.3	11880	396	768 Kbit/s	891	352x288/30.0
2	11880	396	2 Mbit/s	891	352x288/30.0
2.1	19800	792	4 Mbit/s	1782	352x576/25.0
2.2	20250	1620	4 Mbit/s	3037,5	720x480/15.0
3	40500	1620	10 Mbit/s	3037,5	720x576/25.0
3.1	108000	3600	14 Mbit/s	6750	1280x720/30.0
3.2	216000	5120	20 Mbit/s	7680	1280x720/60.0
4	245760	8192	20 Mbit/s	12288	2048x1024/30.0
4.1	245760	8192	50 Mbit/s	12288	1920x1088/30.0
4.2	522240	8704	50 Mbit/s	13056	1920x1088/64.0
5	589824	22080	135 Mbit/s	41400	2560x1920/30.0
5.1	983040	36864	240 Mbit/s	69120	4096x2048/30.0

Table 1.3 – Les différents niveaux (levels) et leurs limitations de ressource mémoire

La norme H.264/AVC utilise le même jeu de définitions de niveau pour tous les profils, mais des applications individuelles peuvent prendre en charge un niveau différent pour chaque profil compatible. Onze niveaux sont définis, spécifiant les limites supérieures de la

taille d'images (dans des macroblocs), la vitesse de traitement du décodeur (en macroblocs par seconde), la taille des mémoires multi-images, le débit binaire vidéo et la taille de la mémoire vidéo (voir tableau 1.3).

1.2.5 Gains du Standard H.264

La figure 1.18 montre donc le gain obtenu grâce à ces multitudes d'améliorations par rapport aux versions précédentes des standards ISO (MPEG-2, MPEG-4 part.2) et de l'ITU (H.263). On constate que l'objectif d'un gain d'un facteur 2 est largement atteint par rapport à MPEG-2.

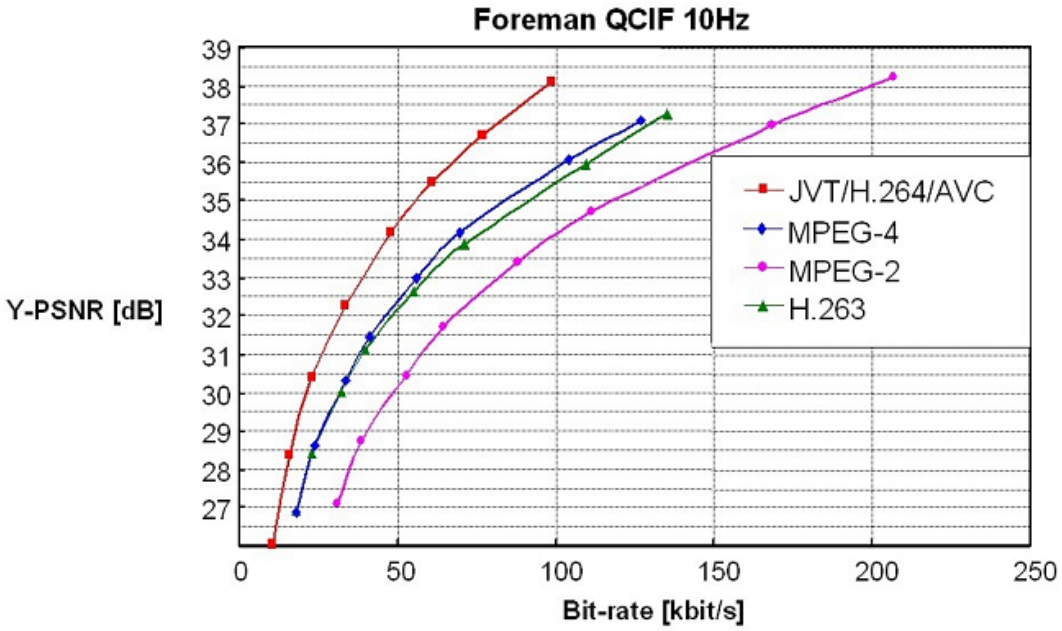

Figure 1.18 – PNSR de la composante *Luma* de la séquence Foreman en fonction du débit moyen

1.3 Codage correcteur d'erreurs

Les symboles transmis sont souvent perturbés par du bruit provenant de plusieurs sources. Ceci peut entraîner une décision incorrecte sur les symboles reçus et par conséquent, dégrader la fiabilité du message reçu. Un moyen efficace de détecter et corriger les symboles erronés est d'utiliser une technique de codage correcteur d'erreurs.

1.3.1 Codes correcteurs

Dans un système de communication numérique (comme illustré par la figure 1.19), trois opérations et leurs inverses sont à effectuer respectivement à l'émission et à la réception :

- **le codeur source** permet de minimiser la quantité d'information nécessaire, et génère un message U ;
- **le codeur canal** (ou codage correcteur d'erreurs) sert à lutter contre les perturbations apportées par le support de la transmission en remplaçant le message U par une séquence X (redondante). Du fait de l'adjonction d'une redondance, le message effectivement transmis est plus long. Un code correcteur d'erreurs se caractérise alors par son rendement R. Si le codeur génère N bits pour un total de K bits d'informations, le rendement R vaut K/N ;
- **la modulation** transforme la séquence X en signal physique.

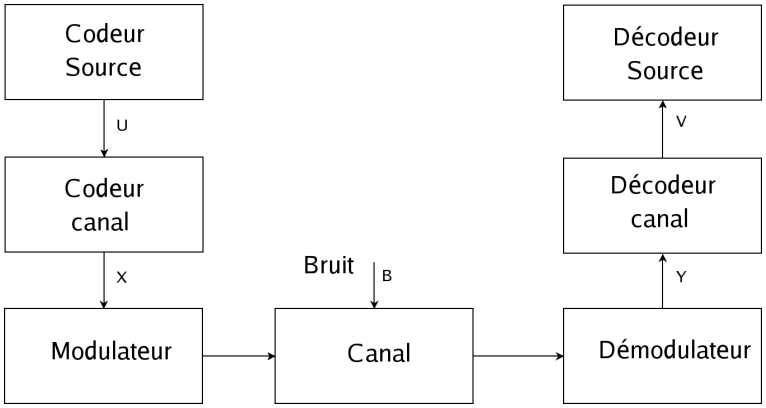

Figure 1.19 – Système de communication numérique

Le canal définit le support de transmission. Le bruit B sur le canal provoque l'apparition d'erreurs sur la séquence Y transmise après démodulation. Le décodeur canal (ou décodage correcteur d'erreurs) a pour tâche d'estimer à partir de la séquence transmise Y, le message V le plus vraisemblable.

Il existe ainsi deux familles de codes correcteurs d'erreurs :

- **les codes en bloc** : le message décomposé en blocs de K bits, est remplacé par un bloc de N bits comprenant directement les K bits d'information et $N - K$ dans le cas où le code est "systématique". Le codage d'un bloc se fait indépendamment des précédents.
- **les codes convolutionnels** : à K bits, le codeur associe N bits codés, mais contrairement au cas précédent, le codage d'un bloc de K bits dépend non seulement de ce bloc mais également des blocs précédents.

Nous avons utilisé cette dernière famille de codage, puisque celle-ci s'adapte parfaitement aux systèmes de communications sans fil considérés, et permet, grâce aux codes convolutionnels perforés, de pouvoir faire varier le rendement de façon quasi continue.

1.3.2 Codes perforés

Les codes perforés ont d'abord été introduits par Cain, Clark et Geist ([15]), puis adaptés par Hagenauer [24] pour créer les codes RCPC, qui offrent une propriété de compatibilité sur les tables de perforation de rendements différents : tout élément perforé (i.e. non transmis) de la table d'un code l'est aussi pour un codage de supérieur et réciproquement tout élément transmis l'est aussi pour un codage de rendement inférieur.

Cette capacité à changer dynamiquement de motif de perforation permet d'obtenir un rendement effectif inférieur et ainsi de réduire progressivement et inégalement le débit, tout en gardant algorithme de codage/décodage unique (un code convolutif de rendement mère 1/3 par exemple). Avec une telle méthode on arrive à atteindre de manière quasi-continue ($R = P/P+l$ avec $l = 1 \rightarrow N(P-1)$) des rendements variants entre le code mère (1/3 par exemple) et presque l'entier (8/9 dans notre cas d'usage). Cette possibilité de variation du niveau de protection pour un codeur unique offre une souplesse au niveau système et une faible complexité. La figure 1.20 illustre classiquement qu'à chaque rendement on peut associer une fonction de qualité de service, c'est-à-dire ici un certain taux d'erreurs binaire pour une qualité donnée d'un canal AWGN en BPSK. Il a été établi ([24]) que les performances de ces codes sont presque identiques aux meilleurs code de même rendement sans perforation.

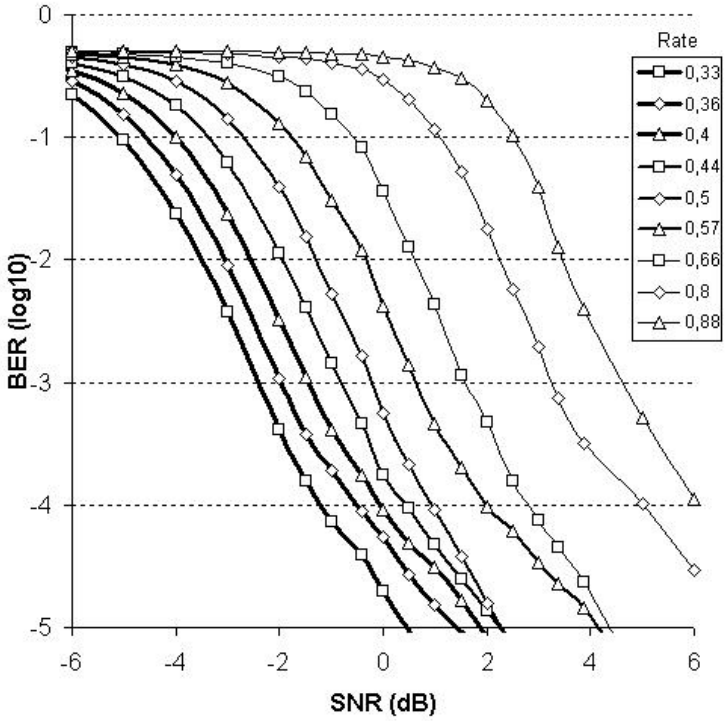

Figure 1.20 – Qualité de Service des codes RCPC utilisés en fonction du rapport signal à bruit dans un canal bruité de type Gaussien

Le code convolutif utilisé est le meilleur code convolutif pour un rendement 1/3 et de taille mémoire 6 (taille de contrainte K=7) : soit 554-744-624 avec la notation en octale de Lin & Costello [31], c'est-à-dire 133-145-175 avec celle de Proakis [45]. Sa distance libre est 15.

Les tables de perforation de la table 1.4 sont issues de [24], on peut basculer d'une table à l'autre de manière dynamique.Le décodage utilisé est, sauf précision contraire, l'algorithme de maximum a posteriori (MAP [7]). D'autres codeurs canal ont également été évalués au sein de ce système comme le SOVA [23] ou l'OSOVA [37] afin de limiter la complexité du décodage utilisé.

8/9	8/10	8/12	8/14	8/16
∼0.88	∼0.80	∼0.66	∼0.57	∼0.50
1111 0111	1111 1111	1111 1111	1111 1111	1111 1111
1000 1000	1000 1000	1010 1010	1110 1110	1111 1111
0000 0000	0000 0000	0000 0000	0000 0000	0000 0000

8/18	8/20	8/22	8/24
∼0.44	∼0.40	∼0.36	∼0.33
1111 1111	1111 1111	1111 1111	1111 1111
1111 1111	1111 1111	1111 1111	1111 1111
1000 1000	1100 1100	1110 1110	1111 1111

Table 1.4 – Tables de perforation utilisées pour atteindre des rendements supérieurs à 1/3 avec le RCPC (m=6, R=1/3)

Au décodage, l'opération inverse de la perforation consiste à insérer des symboles nuls (effacés c'est-à-dire sans a priori) dans le flux de symboles reçus selon le même motif qu'à l'émission. On a donc toujours une même taille de données en entrée du décodeur quelque soit la table de perforation choisie, et c'est uniquement la taille sur le canal qui varie.

On note que les effets de transition (changements de rendement) peuvent être "pénalisants" en terme de performances de décodage en incluant des effets de bord localisés : ainsi au niveau de la transition (quelques bits avant et après en fonction de la taille mémoire du code convolutif utilisé) la protection réelle évolue progressivement et non immédiatement.

Ce phénomène n'est ainsi sensible que pour des changements de rendements fréquents. Dans notre cas d'application nous ne pouvons envisager d'utiliser ces codes pour de trop petites tailles. De plus, pour décoder les codes RCPC, il faut nécessairement connaître les rendements et la disposition des éventuels changements de rendements. Ceci implique alors la transmission d'informations supplémentaires et un réel surcoût de débit dans le cas de changement de rendements trop nombreux. C'est pour cela que toutes les solutions envisagées dans notre cas n'utiliseront des changements de rendements que NAL par NAL, pour limiter le nombre de transitions.

1.4 JSCC : codage conjoint source/canal

La séparation des systèmes de transmission en deux modules distincts, respectivement le codage de source et le codage de canal, est justifiée par le théorème de séparation [47]. Cependant, les hypothèses de ce théorème ne sont pas nécessairement vérifiées en pratique. Dans une première partie, elles sont rappelées ainsi que leurs limitations. Nous présenterons ensuite dans une deuxième partie, les techniques efficaces de codage et de décodage conjoint source/canal en fonction du contexte dans lequel elles peuvent être utilisées, et en positionnant les techniques proposées dans cette thèse parmi celles-ci.

1.4.1 Principales limitations du théorème de séparation

Le théorème de séparation est un résultat asymptotique : il suppose que le codeur source "parfait" décorrèle totalement l'information (c'est-à-dire que la redondance résiduelle en sortie du codeur source est nulle), et que le codeur canal peut ramener la probabilité d'erreur à zéro pour le canal considéré. Pour cela, il fait l'hypothèse que les séquences sont de longueur arbitrairement grande et que la capacité de calcul disponible n'est pas bornée. De plus, les performances d'un système de transmission sont mesurées par la probabilité de récupérer parfaitement un signal, c'est-à-dire pour le critère du taux d'erreurs séquence. Enfin, il suppose qu'il est possible de modéliser parfaitement les propriétés du canal et que celles-ci sont invariables au cours du temps.

Cependant, en pratique, ces hypothèses ne s'accordent avec les contraintes de délai et de complexité dans le cas d'une transmission sans fil. Reprenons un à un les quatre principaux points :

• **Temps infini** : Lorsqu'une contrainte stricte de délai existe, ou si le volume de données est fortement borné, le théorème de séparation n'est plus valable. Or, dans les protocoles de transmission des réseaux de communications classiques manipulent des paquets de tailles relativement petite (188 octets pour une trame MPEG2-TS, et 1400 octets pour une trame internet de type UDP). Cette limitation de taille ne permet pas d'utiliser des codes correcteurs longs, qui sont les seuls à se rapprocher des limites asymptotiques.

• **Taux d'erreurs** : La mesure du taux d'erreurs paquets n'est pas significative dans le cas de transmissions multimédias. Dans ce cas particulier, le signal ne nécessite pas d'être parfaitement reconstitué pour être considéré comme représentant fidèlement le contenu multimédia original : les erreurs n'ont pas toujours le même impact visuel. On parle alors de "qualité de reconstruction", qui peut-être une mesure subjective humaine, ou par une mesure objective (PSNR ou MSE, comme défini au paragraphe 1.1.2) sensée représenter fidèlement celle subjective.

• **Codeur entropique parfait** : Il n'existe pas de codeur entropique parfait pour la

vidéo (ni d'ailleurs pour l'image et pour le son). Dans un cadre pratique, on se réfère alors à la théorie débit-qualité, qui suppose là aussi que l'on puisse quantifier le plus fidèlement possible la subjectivité du rendu visuel, pour un débit donné.

• **Stationnarité des caractéristiques du canal** : Les modèles de canaux standards (AWGN, Rayleigh, Rice,...) ne reflètent pas parfaitement les canaux réels. Ainsi, les canaux sans fil (*wireless*) de type Wifi ou Internet ne sont pas des canaux stationnaires. En fait, on peut les considérer comme stationnaires uniquement si l'on considère un temps suffisamment grand. Il suppose alors que les codes utilisés soient robustes pour une très large classe de canaux, ou que le système possède un canal de retour, permettant au codeur de s'adapter de manière optimale au canal.

1.4.2 Solutions de codage et de décodage conjoint source/canal

Le paragraphe précédent a donc présenté les différents inconvénients liés à la minimisation du débit global de transmission selon la méthode définie par le théorème de séparation. Pour pallier à cet inconvénient, beaucoup de techniques ont été explorées depuis une quinzaine d'année [58] [52] [6]. Elles sont généralement qualifiées de techniques de codage conjoint source-canal. Ces techniques tirent profit principalement des défauts du codeur source, qui ne permet pas d'éliminer la totalité de la redondance.

Nous répertorions ici dans une liste non exhaustive les principales techniques de codage et décodage conjoint source/canal. Nous ne décrirons pas les techniques populaires de retransmisson, car cette thèse ne prend en compte que les flux multimédias à faible latence, comme par exemple la visiophonie ou la diffusion (*broadcasting*). L'objectif de cette thèse est de rendre robuste des flux compressés transmis sur des canaux avec erreurs.

Codage par descriptions multiples

Le codage par descriptions multiples (*Multiple Description Coding* ou MDC) [56] introduit de la redondance au niveau de la source pour gérer les erreurs ou les pertes de paquets introduites par le canal de transmission. Cette technique vise la construction des descriptions corrélées de coefficients à transmettre sur des canaux indépendants. Cette technique suppose utiliser plusieurs canaux de débit équivalent. Dans le cas d'échec de transmission sur certains canaux, des décodeurs dits "latéraux" doivent être capables de reconstruire la source avec une qualité acceptable. En revanche, la réception de toutes les descriptions doit permettre une reconstruction de grande qualité, qui est obtenue en sortie du décodeur dit "central".

RVLC

Les codes à longueurs variables (VLC) et les codes arithmétiques sont utilisés très largement dans les standards pour leur efficacité de compression lors du codage entropique. Mais ces codes ont le défaut d'être très sensibles aux erreurs bits. Une simple erreur bit peut affecter non seulement la reconstruction du symbole source émise mais aussi les symboles codés suivants. On parle alors de "désynchronisation".

Certains codes ont été proposés pour permettre de garantir une resynchronisation du code [54] en diminuant quelque peu les performances. Ces codes ne sont pas traités dans cette thèse, puisque les solutions que nous voulons proposer ici ont pour vocation de s'adapter à un codeur source existant et plus particulièrement au cas du standard H.264. Le standard H.264 possède déjà ses propres codeurs entropiques ($CAVLC$ et $CABAC$), et il n'a donc pas été pas question dans ce travail de proposer d'autres codes VLC plus "robustes".

Pour éviter la propagation d'erreur, il a été aussi proposé dans la littérature (comme options dans la norme) de rajouter des marqueurs de resynchronisation comme le partitionnement de données présenté précédemment (cf 1.2.3). Dans cette thèse, nous avons proposés des méthodes utilisant cette option (voir le chapitre 4)

Protection inégales aux erreurs

Comme nous l'avons vu précédemment, le décodeur canal ne peut corriger toutes les erreurs. Or, les signaux multimédias contiennent des informations dont l'importance pour la reconstruction est très inégale. Les techniques qui exploitent la non uniformité de l'importance de l'information source sont dites de "protection inégale" (UEP : *Unequal Error Protection*). Afin d'atténuer les dommages causés par les erreurs, les techniques de protection inégale contre les erreurs séparent l'information en différentes sous-séquences à la sortie du codeur source. À chacune d'elles est associée une sensibilité aux erreurs différente des autres sous-séquences. Chacune est donc protégée par un sous-codeur de canal différent, adapté à sa sensibilité aux erreurs.

Dans le chapitre 4, nous proposons un système dont le but est d'estimer la sensibilité de chaque partie de l'information, pour ensuite appliquer de manière optimale une protection inégale sur chaque entité.

Décodage souple

Sachant qu'il n'existe pas de codeur entropique parfait, tout codeur à base de VLC ou arithmétique possède donc une redondance résiduelle exploitable pour améliorer la recons-

truction en présence de bruit. Ainsi, plusieurs auteurs ([16], [38], [37] et [20]) ont proposé d'utiliser les sorties souples du décodeur canal. Certains même [22] proposent d'utiliser un code correcteur d'erreurs convolutif et le codeur à longueur variable, comme si ce dernier était un codeur convolutif, et d'appliquer un décodage itératif. Dans le chapitre 2, nous tirons donc profit de ce principe pour proposer une méthode de décodage souple adaptée à la norme H.264, à base d'un décodeur souple de type MAP.

Chapitre 2

Exploitation de la redondance résiduelle du flux binaire H.264 *

Les petits ruisseaux font les grandes rivières
Proverbe paysan

Si l'objectif de tout codeur de source efficace est de réduire la redondance présente dans le flux d'informations à transmettre, il est néanmoins classiquement reconnu qu'aucun codeur entropique standard ne sait parfaitement éliminer cette redondance. Alors que chaque nouvelle génération de codeur de source, qu'il s'agisse de codeur de parole, de musique, d'image ou de vidéo, progresse toujours vers une meilleure efficacité de compression, une part de redondance résiduelle reste présente, partiellement inhérente à la volonté de définir des standards généraux applicables à de nombreux types de contenus. C'est à cette redondance résiduelle que nous nous proposons de nous intéresser dans cette première partie, afin de l'exploiter pour améliorer les performances ou les services fournis dans le cadre d'une transmission multimédia. Deux axes seront donc considérés : le premier, qui s'inscrit dans la lignée des solutions de décodage conjoint source-canal, présente l'adaptation d'une méthode de décodage pondéré (ou souple) dans le cadre du standard H.264/AVC, et montre qu'il est possible d'adapter un décodage de complexité réduite au cas de ce standard qui repose non plus sur un unique code à longueur variable mais une série de codes différents et reliés contextuellement, ce qui nécessite d'introduire un module de gestion du contexte. Le second axe considéré s'intéresse à l'introduction de confidentialité dans le flux d'une manière compatible du standard, c'est-à-dire en s'imposant la contrainte de ne pas générer des flux non interprétables par un décodeur classique, et non attaquables par cryptanalyse au moyen de l'emploi de la redondance résiduelle présente dans le codeur, c'est-à-dire notamment au moyen d'un décodeur souple. On verra dans cette deuxième partie que l'interprétation du mode de fonctionnement du décodeur souple introduit dans la première partie du chapitre nous permet d'isoler les bits que l'on pourra chiffrer.

∗. Certaines parties de ce chapitre ont été publiées dans les conférences *MMSP'04* [64] et *MMSP'05* [65].

Lors de notre survol de la norme H.264 au chapitre 1, nous avons vu que deux modes de codage entropiques étaient possibles, avec des restrictions éventuelles selon le profil considéré. Comme nous nous souhaitons être compatibles de tous les profils du standard, et notamment du mode étendu (X), dédié aux transmissions sans fil, c'est tout naturellement le mode CAVLC (*Context Adaptive Variable Length Coding*) que nous allons considérer. Ce mode, contrairement au mode CABAC qui repose sur un codage arithmétique, réalise une compression en codant chaque symbole du flux vidéo par un mot d'un code à longueur variable. Chacun de ces mots est de taille variable provenant d'une table de mots VLC donnée, éventuellement dépendante des précédents symboles codés, mais toujours représenté par un nombre entier de bits.

2.1 Décodage Souple

Le processus de décodage appliqué par un décodeur vidéo classique consiste à entrer les observations dures obtenues en sortie du canal ou du décodeur de canal et à les interpréter sans plus d'intelligence pour fournir un flux décodé interprétable pouvant être visionné. Ce type de décodage suppose en effet en général qu'aucune erreur résiduelle n'est présente dans le flux et donc que le processus de décodage n'est que l'opération inverse de celui du codage, à d'éventuels masquages, liés à des pertes de parties du flux, près. En pratique cependant, lorsque la transmission comporte un canal sans fil, les erreurs binaires peuvent être fréquentes et amener à la fourniture d'un flux binaire partiellement erroné au décodeur de source. Dans ce cas, il sera intéressant d'estimer le train de bits le plus vraisemblable en sortie du codeur, sachant d'une part les observations en sortie du canal mais aussi sachant les diverses informations a priori disponibles telles les probabilités respectives d'apparition de bits ou symboles, la structure du codeur source, des contraintes sémantiques .... Ces différentes informations a priori font que l'on considérera la métrique Maximum A Posteriori (MAP) et non seulement la métrique du Maximum de Vraisemblance (ML) qui se révélerait sous-optimale [41]. Différentes approches ont été proposées pour réaliser ce décodage à entrées (et éventuellement sorties souples) de codes à longueur variable depuis la fin des années 1990, qui comportent des approximations plus ou moins efficaces [8][13][42][60][41]. Ces solutions reposent sur la présence, mentionnée ou non, d'un treillis grâce auquel des techniques classiques de décodage correcteur d'erreur peuvent être employées. Le principe de ces approches est d'appliquer des solutions de type décodage de Viterbi [19], SOVA [23] ou BCJR [7] sur un treillis pour lequel il n'y a plus un rapport constant entre le nombre de bits et de symboles, mais un rapport variable, comme dans le code VLC considéré. En effet, pour un décodage de type treillis classique, à chaque étape seul l'un des chemins entrant dans un état est considéré (nommé le chemin survivant) alors que les autres, avec de moins bonnes métriques, sont élagués car il a été montré que cet élagage n'affectait pas l'optimalité du processus de décodage. Cependant, dans le cas où l'on considère des codes à longueur variable, les différents chemins arrivant à un même état ne sont pas tous comparables, puisque l'on peut avoir un nombre de bits différent pour un même nombre de symboles ou inversement un nombre de symboles diffé-

rent pour un même nombre de bits. Ce sont donc des treillis modifiés qui sont considérés, soit orientés symboles [13], soit orientés bits [41], soit à trois dimensions, qui incluent ces deux premiers axes [9]. Les performances obtenues sont toujours nettement supérieures à celles atteintes par des solutions classiques de décodage dur, mais l'inconvénient principal et récurrent de ces approches est le grand nombre d'états dans le treillis lorsque le code considéré comporte de nombreux mots, nombre d'état devenant rapidement prohibitif, et rendant les algorithmes inapplicables en pratique. Ceci a amené différentes approches de réduction de la taille du treillis, par élimination de branches en fonction de la métrique, du nombre de symboles, du nombre de bits[41][29], à considérer des formes de treillis plus compactes [12][43] ou utiliser des solutions moins complexes que le décodage complet du treillis, de type décodage séquentiel comme l'algorithme stack ou le M-algorithme [31], comme les travaux proposés dans [11][40].

Notre objectif étant ici de montrer la faisabilité de l'application de ce type de décodage non plus dans le cadre d'un code à longueur variable donné, pour lequel les informations a priori du nombre de bits ou du nombre de symboles exacts émis est en général supposé disponible mais dans le contexte du standard H.264/AVC, sans aucune modification de ce standard, faisant suite aux travaux proposés dans le cadre de MPEG-4 [40] ou H.263+[30]. La présence encore plus importante de mots VLC issus de différents codes, avec inversement une diminution du nombre de mots de resynchronisation rend cet objectif à la fois intéressant et ambitieux. Nous intéressant ici à un décodage à entrées souples et le souhaitant le moins complexe possible, c'est l'approche de simplification suivant les travaux de Buttigieg [11], avec le calcul de la métrique directement sur l'arbre du code VLC proposé dans [40] qui a été testée ici, puisqu'elle offre des performances semblables à celles des solutions avec treillis pour une moindre complexité. Néanmoins, on notera que l'adaptation proposée, avec l'introduction du module contextuel de gestion des tables, ainsi que les conditions d'arrêt propres au codage H.264/AVC sans ajout d'information supplémentaire du type nombre de bits ou de symboles utilisés contrairement aux approches classiques [9] ou plus récentes [59], peuvent également être employées avec une autre métrique.

Commençons donc tout d'abord par présenter la méthode de décodage MAP de codes VLC à entrées souples sur l'arbre du code [40] retenue, avant de passer à l'application de cette méthode au cas d'un flux codé avec le standard H.264/AVC.

2.1.1 Décodage souple des codes VLC

Le point important du décodage souple réside dans l'exploitation de la redondance résiduelle présente dans tous codeurs multimédias, qui peut être considéré comme une implicite protection contre les effets du canal de transmission, et donc a fortiori comme une capacité à corriger des erreurs.

Le décodeur à entrées souples de codes à longueur variable que nous allons présenter est composé d'un estimateur sélectionnant la séquence originellement transmise grâce au

critère du *Maximum A Posteriori* (MAP), c'est-à-dire qui utilise les informations *a priori* sur les statistiques de la source pour maximiser la probabilité *a posteriori* de la séquence décodée et les informations souples des valeurs échantillonnées.

Notations et définitions

Le modèle de chaîne de transmission considérée dans notre cas d'étude est présenté par la figure 2.1. Il consiste en un codeur VLC, un canal de transmission, que nous prendrons par souci de simplification à bruit additif Gaussien et un décodeur VLC.

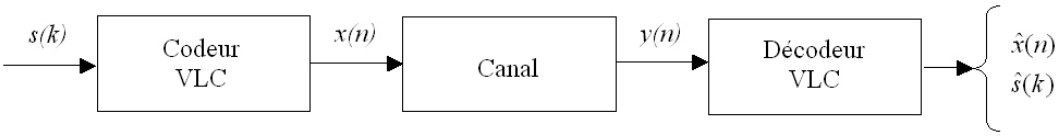

Figure 2.1 – Modèle considéré

Considérons un alphabet de K symboles $S_1, S_2, S_3...S_K$ émis par une source discrète Markovienne spécifiée par les probabilités marginales $P(S_i)$ avec

$$\sum_{i=1}^{K} P(S_i) = 1.$$

Le canal de transmission ajoute un bruit blanc Gaussien de variance σ^2. Soit T la longueur de la séquence de bits transmise par le codeur, et $x(n)$ le $n^{ième}$ bit de la séquence. Nous utiliserons les notations suivantes :

$$x[n : n+m] = [x(n), x(n+1), ..., x(n+m)] \quad \text{avec} \quad x = x[1 : T]$$

Décodeur à entrée souple MAP

L'objectif du décodeur est donc de déterminer la meilleure séquence à la sortie du décodeur source à décision MAP (*Maximum a posteriori*), qui peut tenir compte d'une information probabiliste *a posteriori* disponible sur les symboles émis, c'est-à-dire de trouver la séquence telle que :

$$\hat{x} = arg\ max_x\ P(x|y).$$

Les décodeurs classiques actuels prennent des décisions sur les bits, soit 0 ou 1, après le détecteur à seuil. C'est ce qu'on appelle un décodage à entrée dure (*hard decoding*).

Cependant, il est possible d'obtenir en sortie du canal, puis du décodeur de canal une sortie pondérée ou *souple*.

Une information souple signifie que chaque symbole 0 ou 1 est individuellement accompagné d'une information de fiabilité. Cette information peut alors être prise en compte pour améliorer les performances du décodeur de source. Bien évidemment, dans un systèmr réel, ceci nécessitera le passage des valeurs (quantifiées, mais non binaires) des échantillons au décodeur de source [28].

L'algorithme présenté dans [40] est un algorithme MAP qui repose sur la recherche d'une séquence $\hat{x}$ minimisant la métrique M(x,y), suivante :

$$M(x|y) = -log\big(P(x|y)\big).$$

En appliquant la formule de Bayes :

$$P(x|y) = \frac{P(y|x).P(x)}{P(y)},$$

on obtient alors :

$$M(x,y) = -log\big(P(y|x)\big) - log\big(P(x)\big) + log\big(P(y)\big).$$

La séquence optimale au sens du critère de MAP peut donc être théoriquement trouvée en essayant toutes les possibilités de vecteurs y reçus et de vecteurs x émis, et en les comparant toutes. Mais ceci est beaucoup trop complexe, et plusieurs auteurs (dont Fano et Massey [17]) ont démontré qu'on peut résoudre ce problème en modifiant ou en faisant quelques approximations, pour utiliser les méthodes de décodage bien connues utilisées pour le décodage de code canal.

Pour toutes ces solutions on calcule la métrique suivante appliquée aux différentes suites possibles de mots de codes ou séquences, et on les compare entre elles :

$$M\big(x[s:t], y[s:t]\big) = -log\Big(P\big(y[s:t]\big|x[s:t]\big)\Big) - log\Big(P\big(x[s:t]\big)\Big) + log\Big(P\big(y[s:t]\big)\Big)$$

Le calcul du terme $log\Big(P\big(y[s:t]\big)\Big)$ est complexe et nécessite beaucoup trop de temps de calculs, pour être réellement implémenté. Au lieu de le négliger comme certains auteurs [17], on approximera ce terme par :

$$P_o\big(y[s:t]\big) = \prod_{i=s}^{t} \sum_{x(i)} P\big(x(i)\big).P\big(y(i)\big|x(i)\big)$$

On obtient ainsi la métrique de Fano-Massey [32] :

$$M_2\big(x[s:t], y[s:t]\big) = -log\Big(P\big(y[s:t]\big|x[s:t]\big)\Big) - log\Big(P\big(x[s:t]\big)\Big) + log\Big(P_o\big(y[s:t]\big)\Big).$$

Il est à noter que dans la plupart des cas, $P_o\big(y[s:t]\big) \neq P\big(y[s:t]\big)$, et donc que la métrique M_2 de Fano-Massey est une approximation de la métrique au sens du critère MAP.

Calcul de la métrique

Soit la séquence $\tilde{x}$ représentant une séquence de R mots de codes VLC indexés par τ, tels que :

$$\tilde{x} = [C_{\tau(1)}, \cdots, C_{\tau(R)}]$$

où C_i le mot de code associé au symbole S_i, pour $i = 1, ..., K$.

En reformulant la métrique associée symbole par symbole, on obtient la formulation récursive suivante :

$$M_2(\tilde{x}, y) = \sum_{i=1}^{R} M_2\left(C_{\tau(i)}, y\left[\sum_{j=1}^{i-1} l_{\tau(j)} : \sum_{j=1}^{i} l_{\tau(j)} - 1\right]\right)$$

où l_i représente la longueur du $i^{ème}$ mot de code, pour $i = 1, ..., K$.

On peut ainsi définir une métrique du mot de code C_i à un temps donné t, de la manière suivante :

$$\begin{aligned} M_2\big(C_i, y[t:t+l_i-1]\big) \;=\; & \underbrace{-\,log\Big(P\big(y[t:t+l_i-1]\big|C_i\big)\Big)}_{T_1} \underbrace{-\,log\Big(P\big(C_i\big)\Big)}_{T_2} \\ & \underbrace{+\,log\Big(P_o\big(y[t:t+l_i-1]\big)\Big)}_{T_3}, \end{aligned} \tag{2.1}$$

où les 3 termes T_1,T_2 et T_3 peuvent être calculés comme suit :

- dans les cas d'un bruit Gaussien sur le canal de transmission, le terme T_1 peut être calculé de la façon suivante :

$$T_1 = \sum_{n=t}^{t+l_i-1} -log\Big(P\big(y[t+l_i-1]\big|C_i(n)\big)\Big)$$

avec

$$-log\Big(P\big(y[t:t+l_i-1]\big|C_i(n)\big)\Big) = \frac{\big\|y[t+n] - C_i\big\|^2}{2\sigma^2} + Q$$

avec dans cette équation, Q représente une constante qui peut donc être négligée dans les calculs ;
- le terme T_2 dépend des probabilités des occurrences des symboles VLC (nous supposerons que le décodeur connaît celles-ci) ;
- le terme T_3 est, quant à lui, calculé par la définition de P_o donnée précédemment.

2.1.2 Application à un arbre VLC

La méthode que nous proposons d'utiliser exploite la représentation en arbre binaire des symboles de code VLC pour calculer les termes de (2.1). Un arbre binaire est composé de nœuds et de branches : dans la figure 2.2, nous avons représenté un arbre binaire associé au code VLC de la table 2.1.

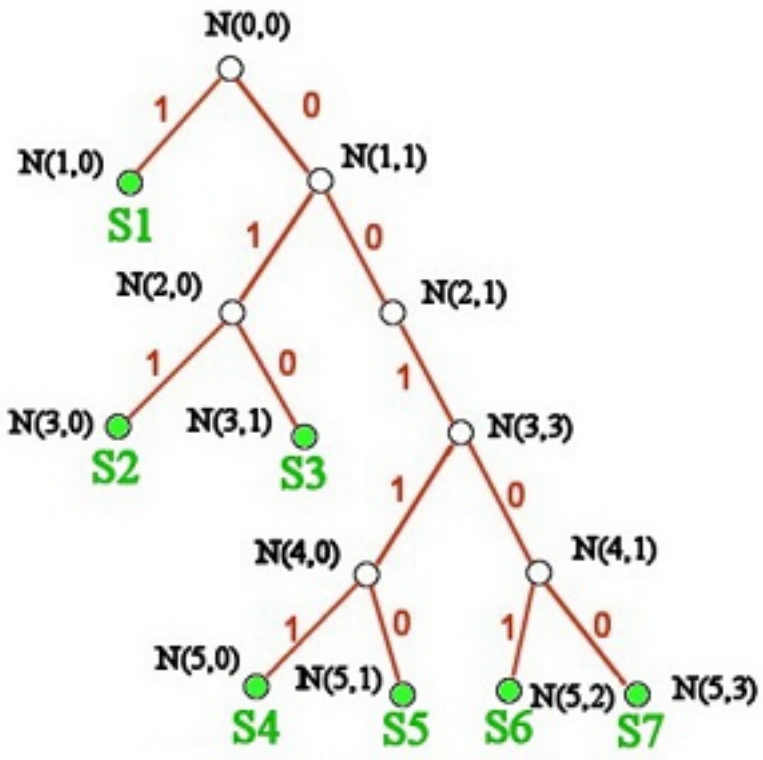

Figure 2.2 – Représentation en arbre binaire d'une table VLC

A chaque branche de l'arbre est alors associée une métrique correspondant à l'observation et la valeur du bit. L'algorithme développé cherchera à chaque instant le plus intéressant des chemins dans l'arbre pour déterminer le mot de code à la plus faible métrique.

Pour définir la métrique d'une branche, nous avons besoin de la probabilité de chaque mot

Mot de code	Symbole
1	S_1
010	S_2
011	S_3
00100	S_4
00101	S_5
00110	S_6
00111	S_7

Table 2.1 – Tableau de correspondance des mots de code d'une table VLC

de code C_i. Pour tout instant t, on peut écrire cette probabilité de la manière suivante :

$$\begin{aligned} P(C_i) &= P\big(x[t:t+l_i-1] = C_i[0:l_i-1]\big) \\ &= P\big(x(t) = C_i(0)\big) \times P\big(x(t+1) = C_i(1) | x(t) = C_i(0)\big) \times \cdots \\ \cdots &\times P\big(x(t+k) = C_i(k) | x(t) = C_i(0), \cdots, x(t+k-1) = C_i(k-1)\big) \times \cdots \\ \cdots &\times P\big(x(t+l_i-1) = C_i(l_i-1) | x(t) = C_i(0), \cdots, x(t+l_i-2) = C_i(l_i-2)\big) \end{aligned} \tag{2.2}$$

Donc :

$$P(C_i) = P\big(x(t) = C_i(0)\big) \times \prod_{k=1}^{l_i-1} P\big(x(t+k) = C_i(k) | x[t:t+k]\big) \tag{2.3}$$

avec :

$$P\Big(x(t+k) = C_i(k)\Big|x[t:t+k]\Big) = P\Big(x(t+k) = C_i(k)\Big|x(t) = C_i(0), \cdot, x(t+k-1) = C_i(k-1)\Big) \tag{2.4}$$

Il est à noter que le terme $P\big(x(t+k) = C_i(k)|x[t:t+k]\big)$ est supposé être connu par le décodeur, puisqu'il s'agit des probabilités des mots de codes *a priori*.

En regroupant les différentes formules, on obtient une nouvelle formule de la métrique d'un mot de code C_i :

$$M_2\big(C_i, y[t:t+l_i]\big) = \sum_{k=0}^{l_i-1} m_2\big(C_i(k), y(t+k)\big)$$

$$\begin{aligned} avec: \ m_2 &= \big(C_i(k), y(t+k)\big) = log\Big(P\big(y(t+k) = C_i(k)\big)\Big) \\ &\quad -log\Big(P\big(x(t+k) = C_i(k)|x[t:t+k-1]\big)\Big) + log\Big(P_o\big(y(t+k)\big)\Big) \end{aligned} \tag{2.5}$$

qui est la métrique associée à la $n^{ième}$ branche du mot de code C_i.

Si on fait l'hypothèse que le décodeur connaît N (le nombre de bits transmis) et K (le nombre de symboles), on peut construire un treillis contenant toutes les séquences de symboles possibles s'adaptant à ses deux contraintes : on obtient alors un arbre d'arbres binaires, puisque chaque symbole est un arbre linéaire.

Mais cette méthode est beaucoup trop fastidieuse pour être réellement implémentée. La solution retenue pour calculer nos métriques est d'employer une pile (ou *stack*) virtuelle contenant à chaque instant les X chemins possédant la métrique la plus faible [40]. On se déplace alors sur l'arbre en choisissant à chaque instant la métrique la plus faible et en stockant dans la pile les nœuds de l'arbre qui possèdent la plus faible métrique. A chaque étape, on repart depuis le nœud possédant la métrique la plus faible et on avance dans l'arbre en calculant les nouvelles métriques.

Commme le décodage stack des codes convolutifs par comparaison à un décodage en treillis, cette méthode ne nous assure donc pas d'obtenir la solution optimale, mais elle permet d'obtenir un gain énorme en complexité et de limiter à notre volonté cette complexité en jouant sur la taille de la pile. La figure 2.3 montre l'efficacité de cet algorithme sur une séquence de mots de codes VLC bruités.

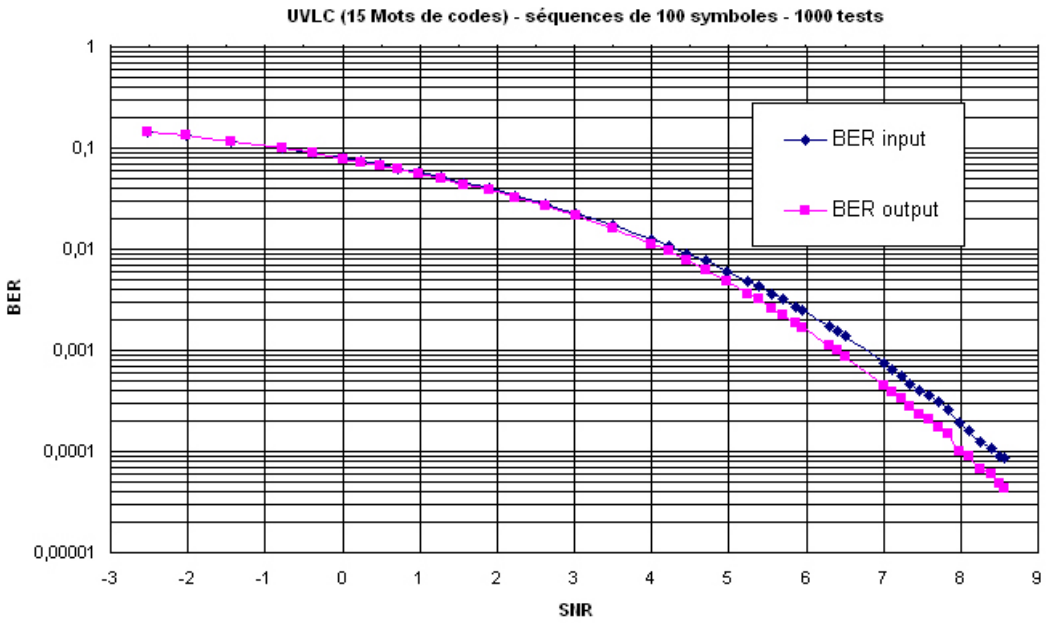

Figure 2.3 – Taux d'erreurs bits avant (BER input) et après (BER output) décodage en fonction du rapport signal à bruit (SNR)

On rappellera ici au lecteur que cette figure illustre une capacité de correction venant de la seule redondance résiduelle d'un code de compression de type VLC. Dans notre exemple, nous avons considéré une séquence de 100 symboles d'une table VLC qui compte

15 éléments (voir table 2.2) modulé en BPSK (*Binary Phase Shift Keying*) et bruités à différents niveaux de bruits de type Gaussien.

Mot de code	Symbole
1	S_1
010	S_2
011	S_3
00100	S_4
00101	S_5
00110	S_6
00111	S_7
0001000	S_8
0001001	S_9
0001010	S_{10}
0001011	S_{11}
0001100	S_{12}
0001101	S_{13}
0001110	S_{14}
0001111	S_{15}

Table 2.2 – Tableau de correspondance des mots de code d'une table VLC

Cet algorithme offre donc des performances de décodage intéressantes quand on sait qu'on ne rajoute aucune information supplémentaire dans le flux binaire. On s'est contenté de connaître le nombre de symboles et le nombre de bits transmis. On peut remarquer sur ces courbes que pour des faibles valeurs de bruits (donc un SNR élevé), la différence entre le taux d'erreurs bits d'entrée (*BER input*), et celui de sortie (*BER output*) devient constante. Ceci s'explique par le fait que certaines erreurs sur le mot de code sont "invisibles" pour le décodeur.

En effet, dans le cas où deux mots $\{b_1, b_2, \ldots b_i, 0\}$, et $\{b_1, b_2, \ldots b_i, 1\}$ sont possibles et équiprobables, si l'on transmet un des deux mots, et que l'on bruite le dernier bit, le décodeur le considérera toujours comme valide ; il laissera donc passer cette erreur.

Dans le cas des mots de codes de type de Exponentiel-Golomb, ce sont les bits de suffixes (voir table 1.1) qui ne pourront jamais être corrigés. En effet, étant donné que les codes Exponentiel-Golomb sont symétriques (c'est-à-dire le nombre de zéros précédant le premier est égal à celui des bits de suffixes), le décodeur ne peut corriger que la moitié des erreurs. C'est pour cela que pour de faibles bruits, le taux d'erreur bit conserve une distance constante : il ne divise que par deux le BER input, et ne peut donc pas corriger la totalité des erreurs. Ce cas montre bien les limites du décodage souple, puisque rien ne garantit qu'on obtiendra la vraie " séquence", mais seulement la plus vraisemblable en fonction des informations dont nous disposons (observations *a priori*, $\cdots$).

Il est à noter qu'il existe aussi la possibilité de "resynchronisation" à travers plusieurs symboles VLC. Ainsi, si on utilise la table VLC Exponentiel-Golomb à trois symboles : $\{1, 010, 011\}$ et que l'on transmet la séquence :

$$\underbrace{0\,1\,0}_{S_2}\ \underbrace{1}_{S_1}\ \underbrace{1}_{S_1}$$

En cas d'erreur sur le premier bit, la séquence sera décodée de la manière suivante :

$$\underbrace{\mathbf{1}}_{S_1}\ \underbrace{1}_{S_1}\ \underbrace{0\,1\,1}_{S_3}$$

Aucune erreur ne sera alors détectée : on obtient bien une "resynchronisation" totale puisque on décode le même nombre de symboles pour le même nombre de bits transmis initialement. Néanmoins, en pratique, ce cas se relève très rare puisqu'il dépend de la succession dans un ordre particulier de plusieurs symboles. Grâce à ses informations *a priori* et aux observations souples, le décodage souple permet néanmoins de résoudre ce type d'erreurs "transparent" par rapport à un décodage de type "*hard*".

2.1.3 Applications à H.264/AVC

Comme nous l'avons vu dans la partie précédente, H.264/AVC utilise des mots de code VLC. Mais il utilise des tables différentes à chaque niveau du codage. Ainsi dans la figure 2.4, nous avons représenté le processus de décodage d'un macrobloc. Chaque entité représente une variable à coder et correspond à une table VLC particulière. On voit ainsi que le décodage d'un flux H.264/AVC, ne correspond pas à la simple succession de mots de codes d'une unique table VLC, mais que chaque étape amène à choisir la table contextuellement en fonction des valeurs prises par les précédentes variables codées.

On notera que l'influence très forte du contexte augmentera les performances du décodeur souple, car la détection d'une impossibilité au $n^{ième}$ symbole décodé pourra permettre de revenir sur le $m^{ième}$ symbole (avec $n > m$) qui était erroné, tant que la taille de la pile est suffisamment grande pour que ce retour en arrière s'effectue. Mais plus la taille de la pile est importante et plus la complexité l'est aussi : ceci implique donc un compromis efficacité/complexité crucial.

Nous avons donc ajouté à notre décodeur MAP, un module "contextuel", illustré en figure 2.5, qui fournira à chaque instant t au décodeur MAP la bonne table VLC à utiliser, et permettra de gérer les contraintes d'arrêt du décodeur (nombre de symboles, mots interdits et nombre de bits).

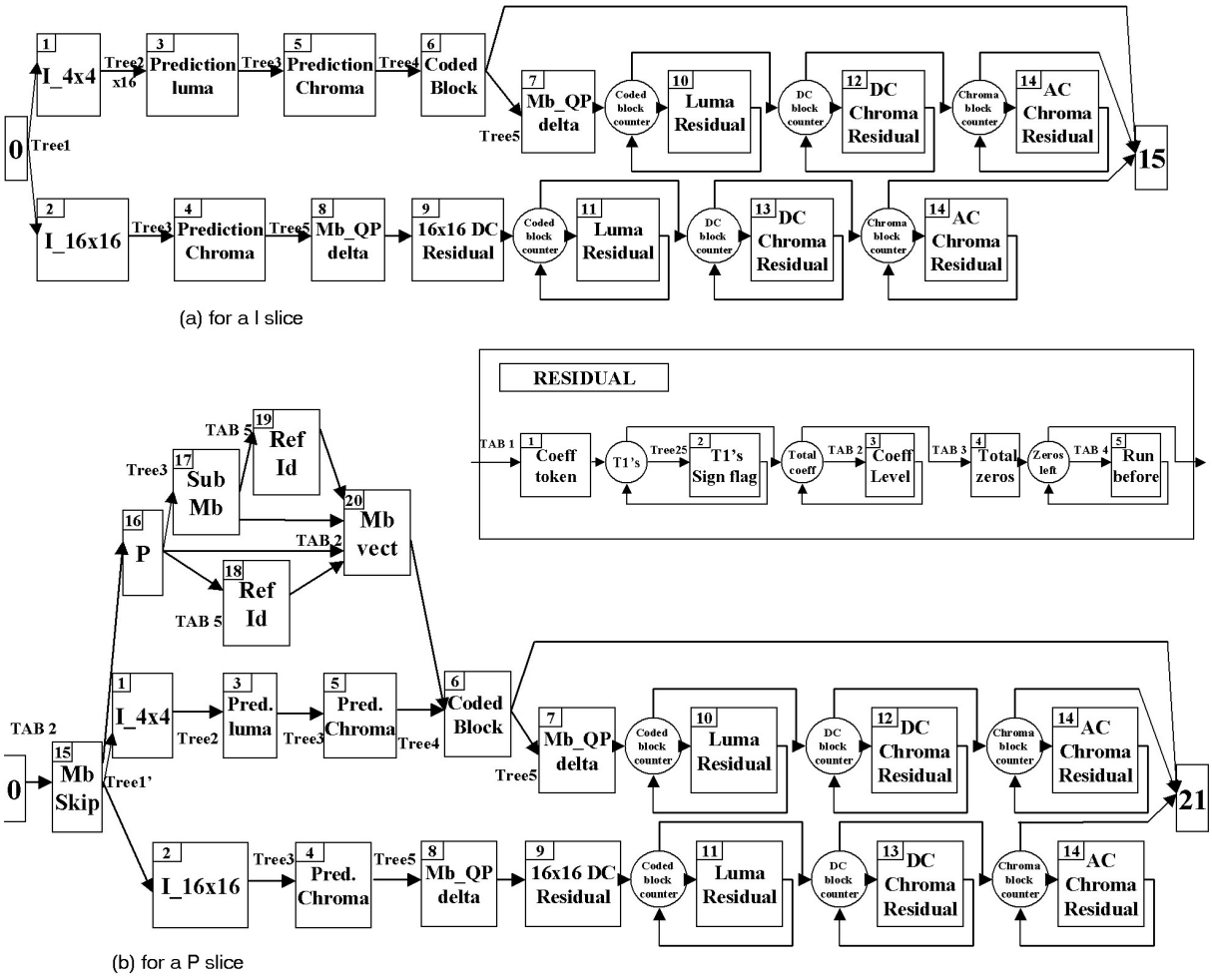

Figure 2.4 – Processus de décodage d'un macrobloc d'une trame I (a) et d'une trame P (b)

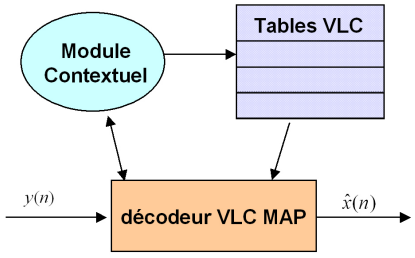

Figure 2.5 – Schéma global du décodage souple adapté à H.264/AVC

2.1.4 Simulations et résultats expérimentaux

Pour analyser les performances de notre décodeur souple, nous nous sommes placés dans le cas d'une transmission modulée en BPSK (*Binary Phase Shift Keying*). Nous avons ensuite bruité à différents rapports signal-à-bruit (*Signal Noise Ratio*) dans un canal à bruit additif de type Gaussien (AWGN ou *Additive White Gaussian Noise*), le flux binaire H.264/AVC. Il est à noter que les en-têtes NAL, qui permettent la synchronisation des différentes parties du flux, ont été supposées non bruitées. Cette hypothèse se justifie par le fait que dans notre chaîne de transmission que chaque NAL est encapsulée dans un paquet réseau contenant diverses informations, comme le rendement particulier du code RCPC utilisé (voir paragraphe 1.3), et permettant aussi une meilleure resynchronisation dans le cas de transmission erronée.

On supposera dans un premier temps que la taille de la pile utilisée par le décodeur souple est de 128, et que les valeurs souples en sortie, démodulée de la BPSK, ne sont pas quantifiées. De plus, dans toutes les expériences menées, le flux H.264/AVC est limité à un seul groupe d'image (GOP), représenté par une trame Intra et 14 trames prédites au format QCIF à 15 images par secondes. Appliqué à la séquence de référence 'Foreman' pour un débit cible de 96 kbits/s, cela donne une Intra de 22000 bits soit 23% du GOP.

Pour estimer les performances du décodeur souple, nous avons comparé d'une part le taux d'erreurs bits en sortie avec celui en entrée comme illustré dans la figure 2.6, et le PSNR résultant, qui équivaut à la qualité de reconstitution de l'image transmise (1.2), du décodeur souple et du décodeur à entrée binaire "classique" (dit "dur" ou "hard") comme montré par les figures 2.7 et 2.8.

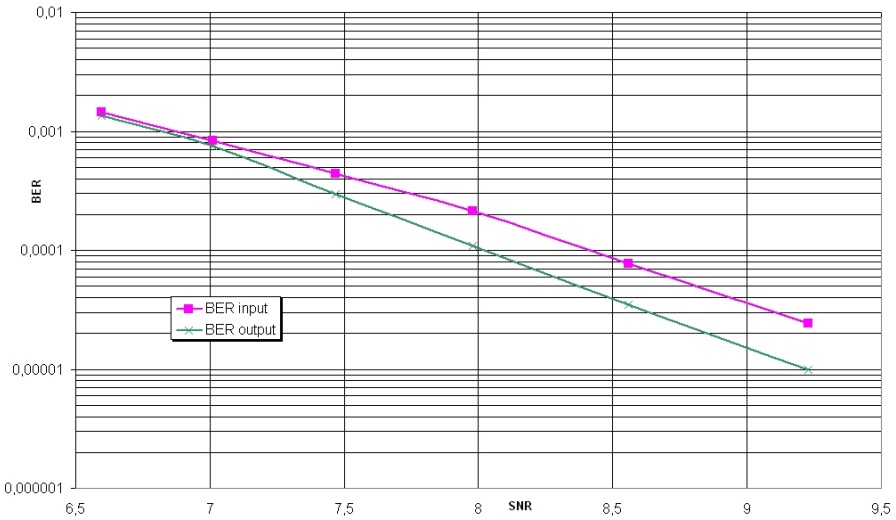

Figure 2.6 – Taux d'erreurs bits du décodeur souple.

Le décodeur “hard” source n’a pas vocation à corriger des erreurs bits. Il considère que le flux est compatible et ne cherche pas à changer le flux pour le rendre “correct”, à la différence du décodeur souple. C’est pour cela que le taux d’erreurs bits en entrée est le même que le taux d’erreurs en sortie du décodeur “hard”. Le décodeur souple, quant à lui, a vocation à corriger les erreurs bits grâce à son calcul de métrique.

Ainsi, dans la figure 2.6, on remarque le gain apporté par le décodage souple en fonction du niveau de bruit. Comme dans le cas d’application en mode UVLC, ce gain peut paraitre relativement faible par rapport à un décodeur correcteur d’erreurs canal classique, mais il s’agit toujours que de tirer profit de la redondance résiduelle du codeur entropique de H.264/AVC, sans ajout de redondance autre.

On peut remarquer également que le gain est limité même à fort SNR, puisque, comme nous l’avons expliqué dans la partie 2.1.2, certaines erreurs sont indétectables pour le décodeur. Ces erreurs bits sont situées sur des parties de mots de code qui “transforment” le flux binaire H.264/AVC original en un autre flux binaire compatible H.264/AVC ; elles ne sont donc pas détectées et corrigées par le décodeur souple. Dans ce cas, seul un ajout de redondance de type codage correcteur d’erreur peut permettre de “récupérer” cette information.

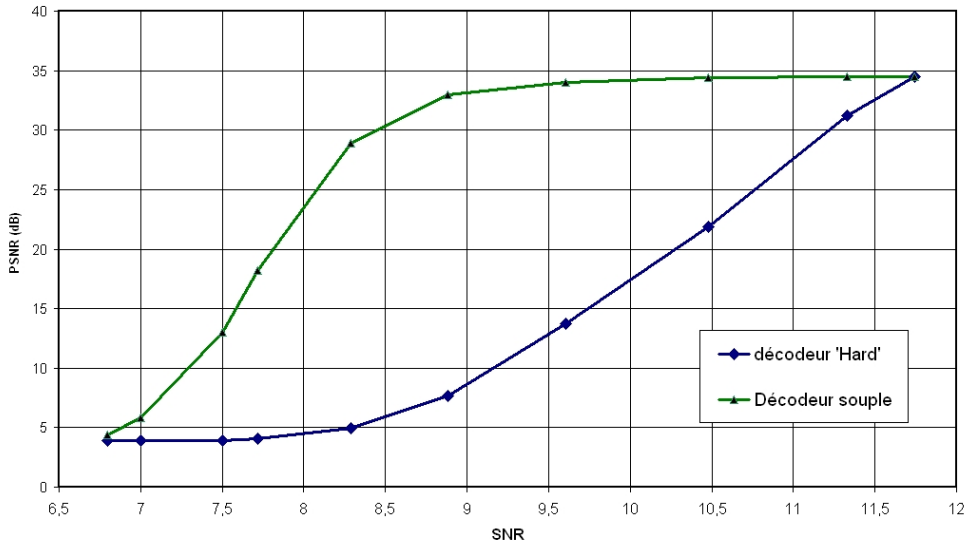

Figure 2.7 – Résultat en PSNR des performances du décodeur souple sur une trame codée en Intra. (Séquence ’Foreman’ QCIF, première trame)

Considérons à présent les résultats en terme de fidélité de reconstruction, avec les mesures de PSNR. Dans nos expériences nous supposons que lorsque le flux H.264/AVC est corrompu, le décodeur source utilise une méthode de masquage d’erreurs simple : dans le cas d’une trame Intra erronée, le décodeur source reconstituera une image complètement noir, et dans le cas d’une trame prédite erronée, le décodeur dupliquera l’image précédente. Les figures 2.7 et 2.8 illustrent les performances obtenues par le décodeur souple

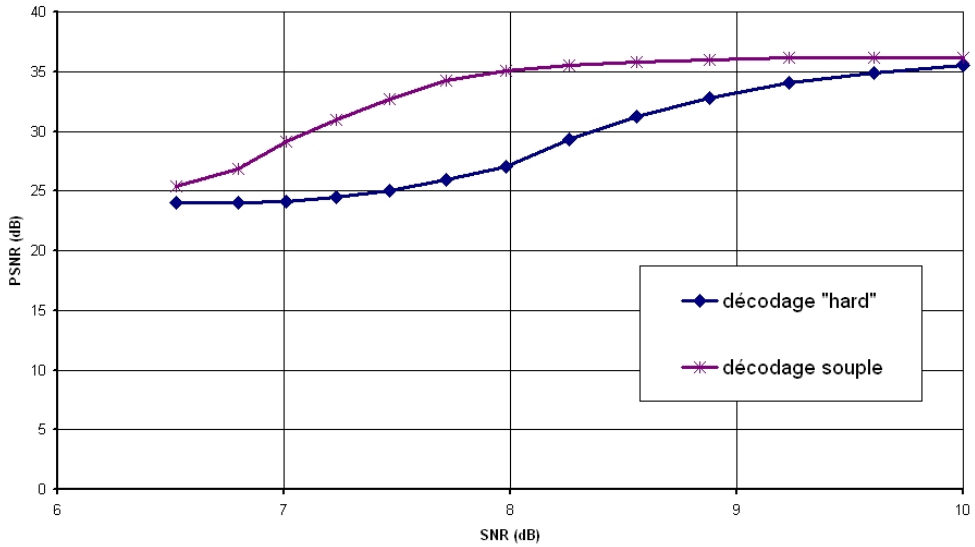

Figure 2.8 – Résultat en PSNR des performances du décodeur souple sur une trame Prédite. (Séquence 'Foreman' QCIF, deuxième trame)

(pour une taille de pile de 128) par comparaison avec le décodeur "hard" qui n'est autre que le décodeur souple avec une taille de pile égale à 1. A qualité équivalente, on obtient un gain en SNR jusqu'à 2 dB entre notre décodeur souple et le décodeur "hard" dans le cas d'une trame codée en Intra. En terme de PSNR, calculé en moyenne de MSE, on observe un gain d'environ 15 dB dans le cas de la trame Intra.

Les résultats concernant le décodage souple des trames P (*Prédites*) ne montrent pas un aussi fort gain que celui pour une trame de type I (*Intra*). C'est pour cela que toutes les prochaines expériences ne seront appliquées que sur la même trame de type I.

Influence de la quantification des données souples

Il est évident qu'on ne peut considérer comme réaliste la précision "flottante" des valeurs souples. La quantification de ces valeurs souples parait donc indispensable. Pour cela, il faut adapter la quantification à la densité de probabilité des valeurs souples. Si on suppose que l'on reste dans une modulation de type BPSK dans une canal AWGN, chaque symbole d'entrée 1 et 0, transmis respectivement avec une énergie $\pm\varepsilon$, possède une probabilité de transition $P(y|j)$ où $y \in \mathbb{R}$, et $j = 0, 1$. Cette probabilité peut donc s'écrire dans le cas AWGN, comme :

$$p(y|j) = \frac{1}{\sqrt{2}\pi\sigma} e^{-\frac{(y-(1-2j)\sqrt{\varepsilon})^2}{2\sigma^2}} \tag{2.6}$$

où, σ^2 est la variance du canal AWGN.

Les valeurs souples prises en sortie du démodulateur sont donc comprises entre $-\infty$ et $+\infty$. Il faut donc avant de quantifier définir une région dans laquelle on appliquera la quantification : on parle alors de "saturation". Soit δ nommé "le pas de saturation", il définit les deux régions $]-\infty, -\sqrt{\varepsilon}-\delta]$ et $[\sqrt{\varepsilon}+\delta, +\infty[$. Dans ces régions, les valeurs souples obtenues seront considérées comme correctes, c'est-à-dire qu'elles ne sont pas modifiables pour le décodeur souple.

Dans la figure 2.9, nous représentons les densités de probabilités des symboles 0 et 1 dans le cas BPSK, ainsi que les régions de "saturation" considérées.

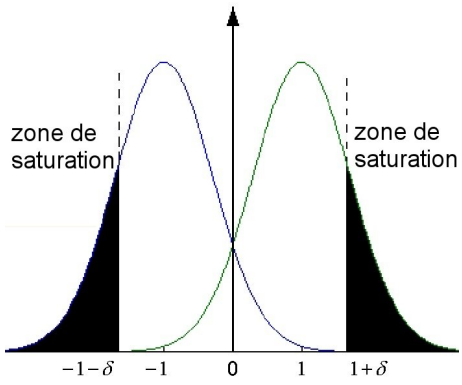

Figure 2.9 – Densités de probabilités des symboles BPSK, et positionnement des régions "saturées"

Le choix du paramètre δ dépend naturellement des caractéristiques du canal. Pour calculer l'impact de la saturation, calculons la probabilité d'erreur induite :

$$P_{err} = \int_{-\infty}^{-\sqrt{\varepsilon}-\delta} P_1.p(y|1)dy + \int_{\sqrt{\varepsilon}+\delta}^{\infty} P_0.p(y|0)dy \tag{2.7}$$

avec P_0 et P_1, respectivement les probabilités d'apparition des symboles 0 et 1.

Nous supposerons que les symboles 0 et 1 sont équiprobables ($P_0 = P_1 = 1/2$) et que leurs densités de probabilité $p(y|j)$ ont les mêmes caractéristiques ($\sigma_0 = \sigma_1$). De plus, pour simplifier les calculs on considérera que $\varepsilon = 1$. La probabilité d'erreur est alors :

$$P_{err} = \int_{-\infty}^{-1-\delta} p(y|1)dy = \int_{-\infty}^{-1-\delta} \frac{1}{\sqrt{2}\pi\sigma} e^{-\frac{(y-1)^2}{2\sigma^2}} dy$$

$$P_{err} = \frac{1}{2} erfc\left(\frac{\delta+2}{\sqrt{2}\sigma}\right), \text{ où } erfc \text{ est la fonction d'erreur complémentaire.} \tag{2.8}$$

Pour une erreur de saturation d'environ 10^{-9}, l'inversion de la fonction *erfc* donne la contrainte $\frac{2+\delta}{\sigma\sqrt{2}} \simeq 3.\sqrt{2}$, ce qui permet de déterminer le δ tel que $P_{err} < 10^{-9}$:

$$\delta \simeq 6.\sigma - 2$$

La figure 2.10 présente les performances du décodage souple avec et sans saturation, avec le rappel des résultats "hard". Cette méthode permet donc de diminuer le nombre d'itérations sans pour autant de diminuer significativement les performances obtenues précédemment. Toutes les simulations suivantes utiliseront donc cette zone de saturation.

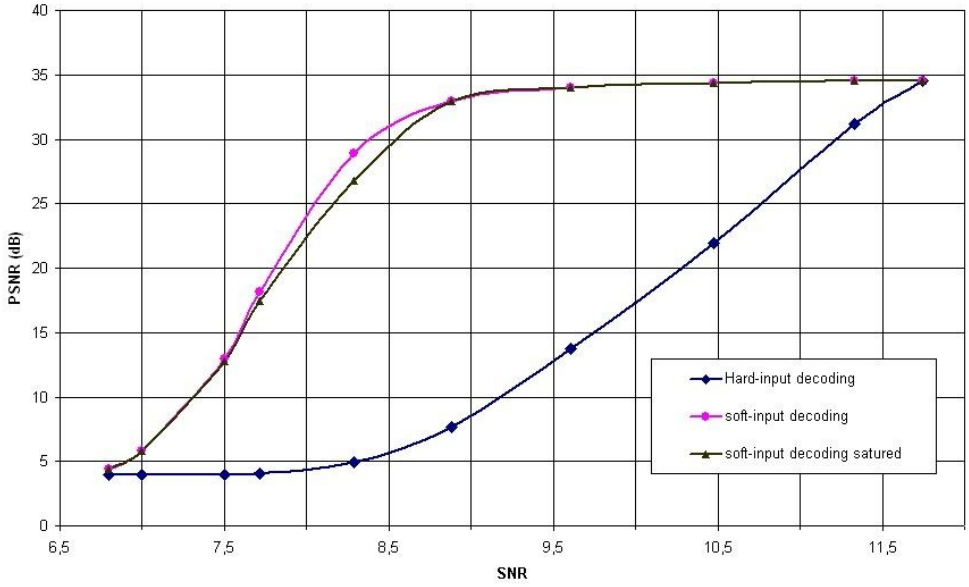

Figure 2.10 – Influence de l'étape de saturation sur les performances du décodeur souple en PSNR pour une trame codée en Intra. (Séquence 'Foreman' QCIF, première trame)

Le reste de la région "non-saturée" est quantifiée uniformément, c'est-à-dire avec un pas de quantification Δ est constant. Ainsi, pour chaque intervalle on définit un représentant $\tilde{x_k}$ du segment $[(k-1).\Delta\ ,\ k.\Delta]$, où $\Delta = \frac{-2+2\delta}{2^{N_{bits}-1}}$ avec N_{bits} nombre de bits alloués par le quantificateur.

Par conséquent, la distorsion résultante de la quantification peut ainsi s'écrire de la manière suivante :

$$D = \sum_{k=1}^{2^{N_{bits}-1}} \int_{(k-1)\Delta}^{k\Delta} f(\tilde{x_k} - x)p(x)dx$$

avec $p(x)$ densité de probabilité des symboles BPSK dans le cas d'une transmission sur un canal AWGN. Pour minimiser cette distorsion, $\tilde{x_k}$ est choisi par la méthode du centroïde pour chaque intervalle, grâce à l'équation suivante :

$$\tilde{x_k} = \frac{\int_{(k-1).\Delta}^{k.\Delta} x.p(x)dx}{\int_{(k-1).\Delta}^{k.\Delta} p(x)dx} \tag{2.9}$$

La figure 2.11 illustre les résultats obtenus pour différentes quantifications (2, 3 et 4 bits).

On remarque que la quantification sur 4 bits des valeurs souples suffit à se ramener aux performances du décodeur souple sans quantification. C'est pourquoi les expériences et les résultats suivants utiliseront sauf mention contraire une quantification de 4 bits.

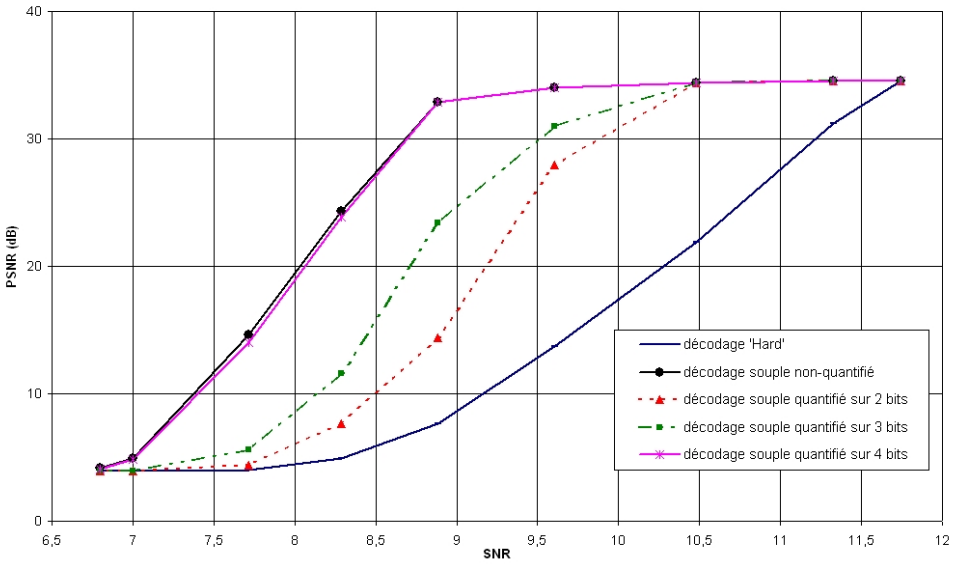

Figure 2.11 – Influence de l'étape de quantification sur les performances du décodeur souple en PSNR pour une trame codée en Intra. (Séquence 'Foreman' QCIF, première trame)

Influence des paramètres de la taille de la pile du décodeur souple

Comme décrit précédemment (sections 2.1.1 et 2.1.2), le décodeur souple repose sur un décodeur MAP qui utilise une pile (*stack*) qui conserve à chaque instant les N chemins possédant les métriques les plus faibles. Plus la taille de la pile est importante, et plus le nombre d'itérations est important, et plus le décodage est performant. Réciproquement, plus la pile est importante, plus la complexité de l'algorithme grandit. Dans toutes les simulations précédentes la taille de la pile avait été fixée à 128 (c'est-à-dire qu'on ne gardait que les 128 meilleurs métriques à chaque instant). Pour mieux estimer l'influence de cette taille, nous avons étudié pour différentes tailles les performances du décodeur, comme illustré en figure 2.12.

On note en particulier que même pour une pile de taille 4 le gain du décodeur souple par rapport au décodage de type "hard" reste supérieure à $1,5\ dB$ en terme de SNR.

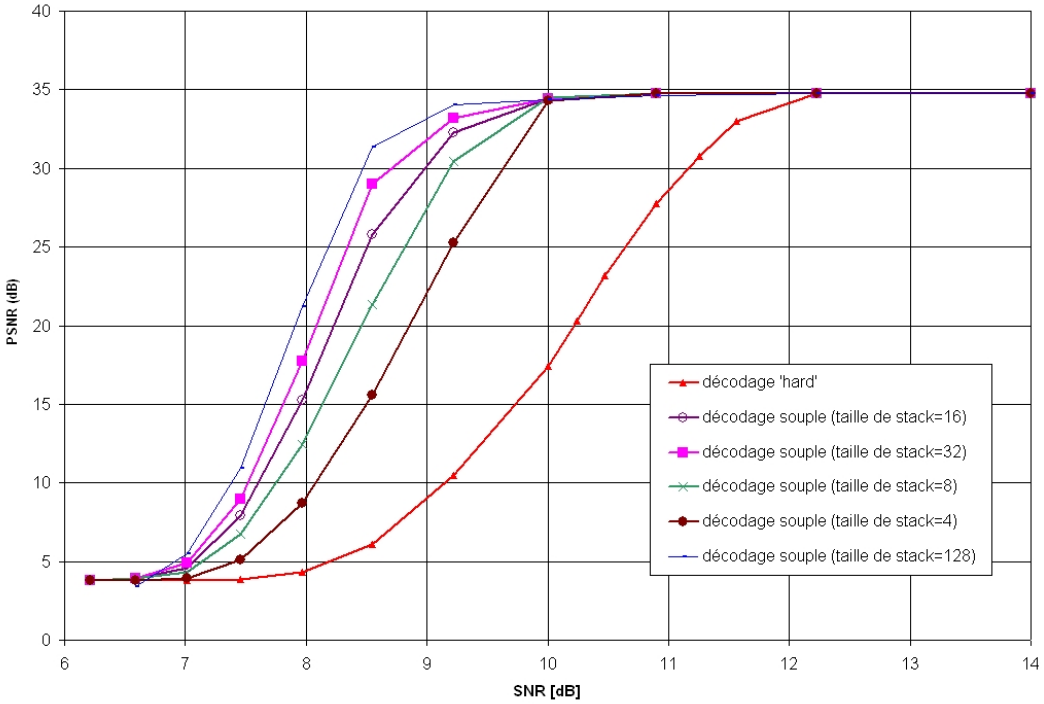

Figure 2.12 – Influence de la taille de la pile sur les performances du décodeur souple. (Séquence 'Foreman' QCIF, première trame)

Influence du choix des probabilités statistiques des VLC du décodeur souple

Comme nous l'avons vu dans la partie 2.1.2, la formule (2.6) utilise les valeurs de probabilités d'apparition des mots de codes (2.4). Ces probabilités sont des informations *a priori* estimées grâce aux statistiques des mots de codes VLC utilisés par la norme H.264/AVC. Les simulations et figures précédentes ont été obtenues en calculant les *a priori* en supposant que ces statistiques suivaient les propriétés d'un codeur entropique parfait. Ainsi on estime pour chaque mot de code VLC S_k de taille l_k, une probabilité d'apparition suivante :

$$P(S_k) = 2^{-l_k} \tag{2.10}$$

Afin d'estimer l'influence de cette hypothèse, nous avons testé notre décodeur souple en modifiant les probabilités des VLC, en supposant cette fois-ci qu'elles sont équiprobables. De fait, la métrique calculée n'utilise alors que les données souples en entrée pour chercher à obtenir le flux H.264/AVC le plus vraisemblable, puisque tous les mots de code VLC sont supposés équiprobables. Les résultats obtenus, comme le montre la figure 2.13, sont impressionnants. On remarque d'ailleurs que même en variant la taille de la pile, nous n'obtenons pas du tout les mêmes gains que précédemment. Cette dernière expérience nous permet de conclure sur l'importance de l'expression de la probabilité d'apparition

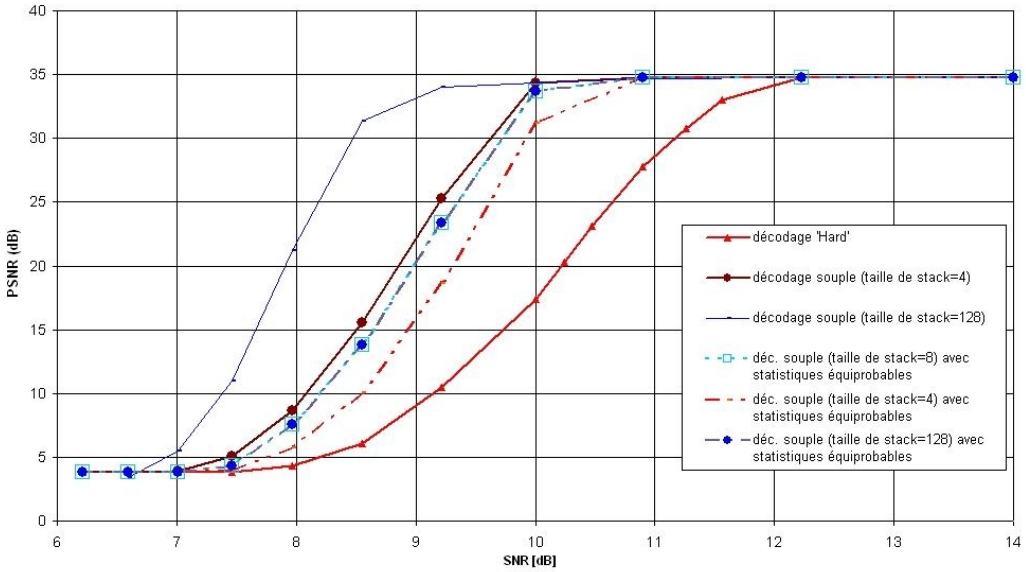

Figure 2.13 – Influence de la statistique utilisée sur les performances du décodeur souple en PSNR

$P(C_i)$ dans les performances du décodeur souple [26].

Il est évident que les symboles VLC ne suivent pas une loi de probabilité figée, en fonction des différentes sources choisies et en fonction du débit source. On pourra donc imaginer d'optimiser le décodeur souple en fonction des statistiques des symboles VLC, en s'adaptant avec une statistique correspondante à une taille ou un type de flux. Cette approche nous amènerait donc à avoir une conscience précise à tout instant des statistiques des mots, avec communication de l'évolution de ses statistiques vers le décodeur source, ce qui nous rapproche des certains travaux sur les décodeurs souples arithmétiques [21].

2.2 Chiffrement sélectif compatible pour flux vidéo H.264

2.2.1 Présentation de la méthode

Dans la plupart des systèmes de chiffrement, la donnée vidéo compressée est traitée comme une donnée binaire quelconque par le mécanisme de chiffrement placé après le processus de codage vidéo. Du côté récepteur, le déchiffrement a lieu avant le processus de décodage vidéo, comme l'illustre la figure 2.14. Un tel schéma ajoute un temps de latence et implique plus de calculs, puisque soit on chiffre la totalité du flux vidéo codé, soit on doit le segmenter en plusieurs flux qui seront traités séparément et ensuite rassemblés du côté du décodeur. C'est pourquoi d'autres solutions ont été introduites qui conjuguent intimement

les procédés de codage et de compression. Les solutions de chiffrement mises en œuvre avant le mécanisme de compression conduisent toutefois à des procédés de chiffrement moins efficaces. Par exemple, il a été montré que le fait de permuter aléatoirement les coefficients transformés, a le défaut de "déformer" la distribution de la probabilité de ces coefficients, rendant le codage entropique moins efficace pour le processus de compression [55] et [48].

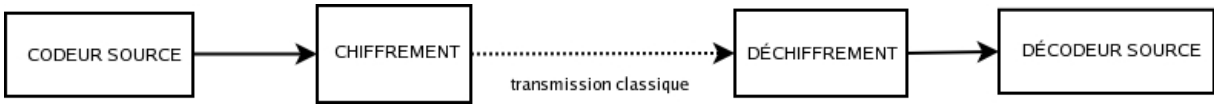

Figure 2.14 – Schéma de chiffrement classique d'un flux multimédia

L'idée que nous proposons dans ce paragraphe garde l'idée mais propose d'enfouir le mécanisme de chiffrement à l'intérieur du mécanisme de compression, et d'effectuer le chiffrement après que le codage entropique soit effectué, de façons à ne pas modifier l'efficacité de la compression. L'objectif était d'obtenir un chiffrement compatible de la norme vidéo, il ne s'agit pas d'effectuer un chiffrement complet, qui résulterait en un brouillage des données à transmettre, mais de sélectionner dans le flux des bits que l'on peut brouiller tout en respectant la norme, et ce de façon suffisante pour ressentir l'efficacité du chiffrement.

2.2.2 Application de la méthode à la norme H.264/AVC

La sélection des bits à chiffrer consiste typiquement à déterminer les parties du flux de données qui n'ont pas d'influence dans le procédé de décodage. On peut l'étendre également aux bits qui ne modifient que de manière négligeable les contextes du procédé de décodage, dans le sens où la modification due au chiffrement ne génère pas de désynchronisation ou ne conduit pas à des flux de bits non compatibles. Cette approche découle directement de l'observation des limitations du décodeur souple présentées au paragraphe 2.1. En effet, certaines erreurs (bits erronés lors de la transmission du flux H.264/AVC) ne sont ni détectées ni corrigées. Le décodeur souple est en fait composé d'un estimateur sélectionnant la séquence originellement transmise grâce au critère du *Maximum A Posteriori* (MAP), c'est-à-dire qui utilise les informations *a priori* sur la statistique de la source pour maximiser la probabilité *a posteriori* de la séquence décodée et les informations souples des valeurs échantillonnées.

Si le décodeur n'arrive pas à détecter et a fortiori à corriger ces "erreurs", c'est parce que celles-ci n'en sont pas au sens de la norme H.264/AVC. Malgré ces "erreurs", le flux H.264/AVC résultant reste en effet compatible avec la norme, les bits erronés étant "transparents" : ils n'influent pas le procédé de décodage, ils n'ont qu'un impact visuel.

La méthode de chiffrement que nous proposons consiste à recenser ces bits dits "transpa-

rents" et à les chiffrer afin de rendre inaccessible le flux vidéo tout en restant compatible avec le standard vidéo H.264/AVC, au sens où un codeur standard H.264/AVC peut décoder en totalité le flux de données chiffrées sans avoir besoin de prévoir des mécanismes de resynchronisation particuliers ou de masquage d'erreur. Ainsi, grâce à ces bits "transparents" on transforme un flux compatible H.264/AVC en un autre flux compatible H.264/AVC de même taille, la différence étant uniquement visuelle, comme l'illustre la figure 2.15.

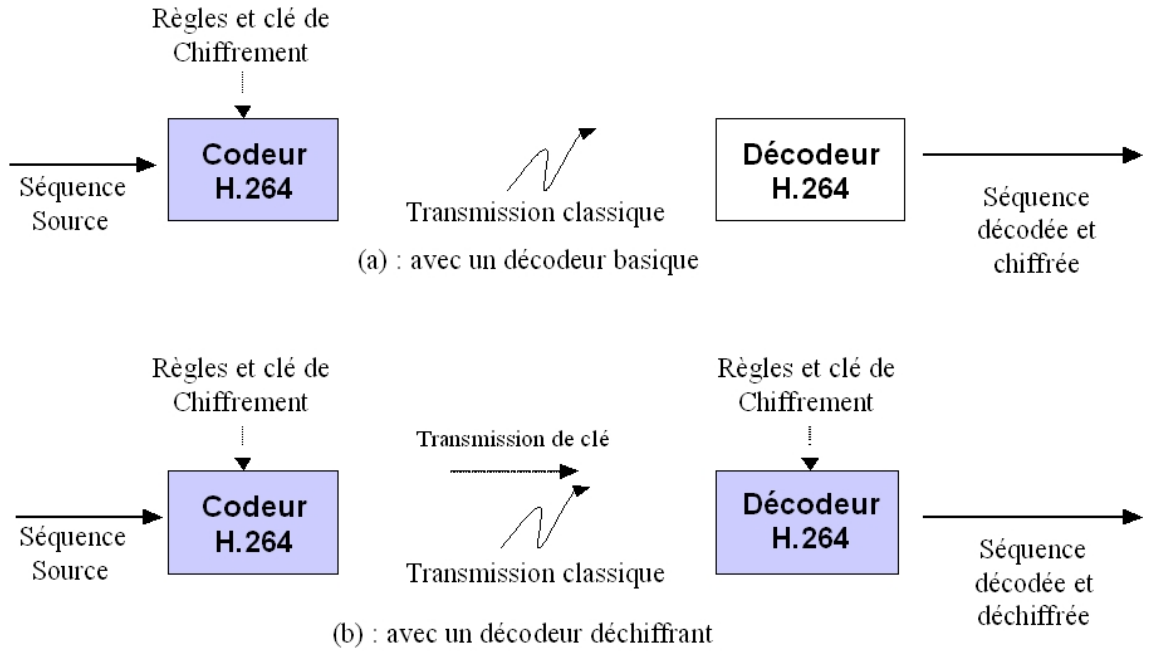

Figure 2.15 – Schéma de chiffrement sélectif proposée pour le codeur H.264/AVC. (Les modifications introduites sont représentées en grisé)

Si la méthode est transposable aisément dans le principe à d'autres codeurs que le codeur H.264/AVC, la première étape qui consiste à répertorier l'ensemble (ou un nombre suffisant) de bits "transparents" modifiables dans le sens où le chiffrement ne génère pas de désynchronisation de décodage est propre à chaque standard. Nous avons placé en annexe A une liste quasi-complète de tous ces bits "transparents", pour lesquels il a été vérifié qu'aucune information supplémentaire n'était nécessaire, ni aucune hypothèse à faire afin de décoder le flux, tant avec un décodeur déchiffrant qu'avec un décodeur standard.

Parmi les critères du processus de sélection des bits, on impose par exemple que les probabilités des mots restent inchangées, c'est-à-dire que le processus de chiffrement transforme toujours un mot de code en un autre mot de code de même taille et de même probabilité d'apparition. Au vu des résultats obtenus avec un décodage souple par des statistiques fausses, cette condition semble en effet nécessaire pour éviter l'introduction d'un biais important.

Comme pour le reste des travaux présentés précedemment, le codeur H.264/AVC considéré a été testé dans le mode *CAVLC*. Cette méthode peut néanmoins être adaptée au mode *CABAC* en choisissant les bits à chiffrer selon le principe énoncée ci-dessus. Dans le cas du codeur entropique *CAVLC*, décrit dans le paragraphe 1.2.2, les tables VLC sont adaptées à chaque information ou coefficient codé, comme illustré sur la figure 2.4. Le procédé est donc de sélectionner les mots de code dans le flux H.264/AVC, qui une fois chiffrés ou partiellement chiffrés n'altéreront pas le décodage du reste du flux, car le mot chiffré est de même taille et de même contexte que l'original.

Pour bien comprendre le principe, détaillons un exemple de type de mots de code sur lesquels on peut appliquer cette méthode. Dans le flux H.264/AVC, avant de coder dans chaque macrobloc les coefficients de la transformée entière, on transmet le paramètre "Mb_QP_Δ", qui permet de changer la valeur du pas de quantification pour le macrobloc considéré. Ce mot de code utilise une table VLC de type Exponentiel-Golomb, comme illustré par le tableau 2.3.

Index	Mot de code	Mb_QP_Δ
1	1	0
2	01*0*	+1
3	01*1*	−1
4	001*00*	+2
5	001*01*	−2
6	001*10*	+3
7	001*11*	−3
8	0001*000*	+4
9	0001*001*	−4
10	0001*010*	+5
11	0001*011*	−5
12	0001*100*	+6
13	0001*101*	−6
14	0001*110*	+7
15	0001*111*	−7
...	...	...

Table 2.3 – Tableau de correspondance des mots de code Mb_QP_Δ des 15 premiers symboles

Sachant que le changement de valeur du symbole Mb_QP_Δ n'a pas d'influence sur la manière de décoder le reste du flux, seul le risque de désynchronisation nous empêche de chiffrer le mot de code entièrement. Pour ne pas créer de désynchronisations, nous restreignons donc le chiffrement aux bits (en italique dans le tableau 2.3) qui ne modifient pas la taille du mot de code s'ils sont chiffrés.

On notera dans la recherche de bits transparents que certains bits, de prime abord ré-

pertoriés comme transparents, ne le sont pas car des contraintes spécifiques leurs sont applicables : c'est notamment le cas des bits appartenant aux macroblocs de bord de slice pour lesquels les limitations de types de prédiction par exemple, nous ont amené à exclure le chiffrement en pratique.

Empiriquement, nous avons remarqué à travers les expériences menées sur différentes séquences en CIF et QCIF, que ces bits "chiffrables" correspondent à environ 25 % du flux pour une trame Intra, et à environ 15 % pour une trame de type P.

2.2.3 Introduction du chiffrement

Procédé de chiffrement

En cryptographie, le chiffrement est le procédé grâce auquel on peut rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (dé)chiffrement. Il s'agit de considérer n'importe quel flux binaire comme une succession de chiffres, pour ensuite faire des calculs sur ces chiffres, de manière à, d'une part, les modifier de telle façon à les rendre incompréhensibles, d'autre part de faire en sorte que le destinataire saura les déchiffrer.

Le chiffrement se fait généralement à l'aide d'une *clef de chiffrement*, le déchiffrement nécessite quant à lui une *clef de déchiffrement*. On distingue deux types de clefs :

- les clefs symétriques : lorqu'on utilise la même clef pour chiffrer et déchiffrer.

- les clefs asymétriques : au cas où l'on utilise des clefs différentes. Il s'agit typiquement d'une paire composée d'une clef publique, servant au chiffrement, et d'une clef privée, servant à déchiffrer. Le point fondamental soutenant cette décomposition publique/privée est l'impossibilité calculatoire de déduire la clef privée de la clef publique.

Le résultat de cette modification (le message chiffré) est appelé "cryptogramme" (en anglais *ciphertext*) par opposition au message initial, appelé "message en clair" (en anglais *plaintext*). Les méthodes les plus connues sont le RSA pour la cryptographie asymétrique, et l'AES (*Advanced Encryption Standard*) pour la cryptographie symétrique [3] que nous utiliserons dans nos expériences.

Adaptation du chiffrement avec AES

Proposé par Daemen et Rijmen [46], l'AES est un algorithme de chiffrement adopté par le NIST (*National Institute of Standards and Technology*). Nous utiliserons ce procédé de chiffrement avec le mode dit "compteur" (*CTR*) pour générer notre cryptogramme.

Le principe est basé sur une opération de "Ou-exlusif" (en anglais *X-OR*) effectué sur le message initial pour obtenir le cryptogramme, et inversement pour le décodeur. Cependant, cette opération de "Ou-exclusif" génère un cryptogramme en sortie pouvant prendre toutes les valeurs possibles, comme l'indique la figure 2.16 - (b), ce qui peut engendrer des configurations impossibles selon le standard H.264/AVC.

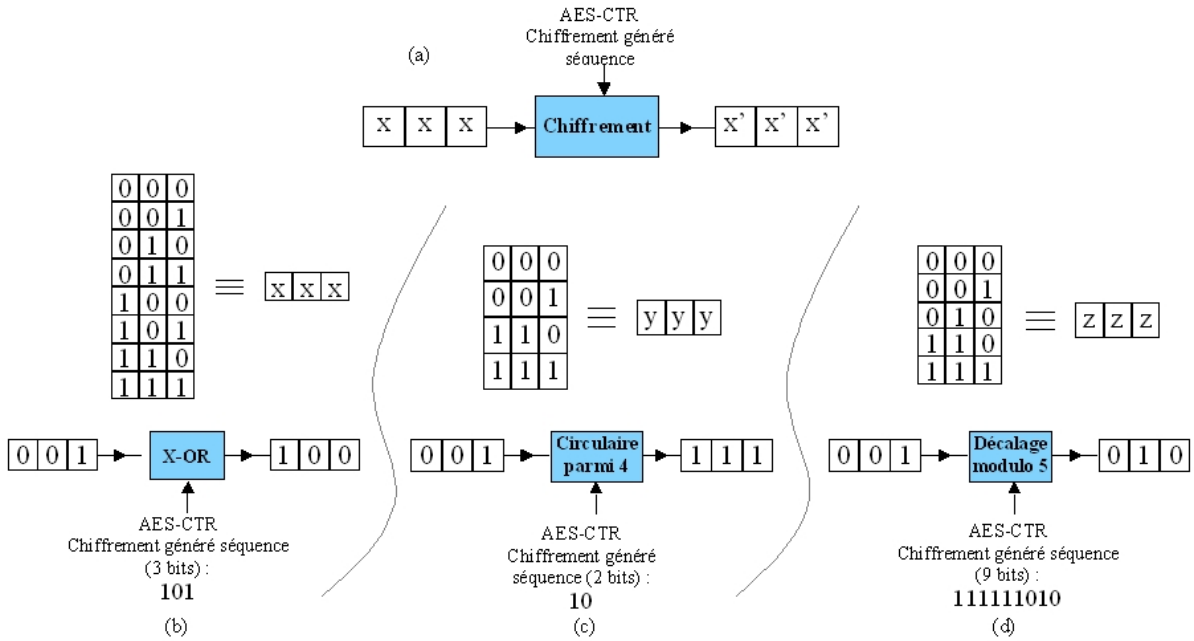

Figure 2.16 – Procédé de chiffrement AES (a) et exemples de ces applications (b)-(d)

Nous ne pouvons directement appliquer le procédé d'AES-CTR pour chiffrer les symboles, sans avoir auparavant sélectionne parmi toutes les configurations possibles fournies par le standard, celles qui seront utilisés comme cryptogramme. Ces configurations possibles peuvent être répertoriées dans une table, avec des positions allant de 0 à $n-1$. Deux cas de figure sont alors à prendre en compte, si le nombre de configurations possibles est une puissance de 2, ou non :

• Avec 2^k configurations possibles, on peut utiliser k bits pour réaliser le chiffrement. Une façon de procéder est d'utiliser ces k bits pour sélectionner de manière circulaire à partir d'une position i du message initial, la configuration choisie comme cryptogramme. En générant des blocs de sortie équiprobables, l'AES-CTR offre, grâce à ce chiffrement circulaire, de bonnes propriétés de résistance face à une cryptanalyse. Ce schéma est illustré par la figure 2.16 - (c), pour $k = 2$ et $i = 1$.

• Le problème est plus complexe lorsque le nombre de possibilités n'est pas une puissance de deux. Une solution est d'utiliser une répartition asymétrique des permutations. Si nous considérons une clé de k bits, correspondant à 2^k blocs de sortie possibles, et n configurations possibles par le standard, on peut passer d'une $i^{ème}$ configuration à une

autre en choisissant parmi les 2^k prochaines configuration *modulo* n (la $(i+2^k[n])^{ème}$ configuration). Une configuration peut être alors utilisée en tant que cryptogramme entre $\lceil \frac{2^k}{n} \rceil$ et $\lfloor \frac{2^k}{n} \rfloor$. Ceci laisse donc un biais dans la probabilité de distribution, égale à la différence maximum entre la distribution considérée et celle supposée équiprobable des mots de code de même longueur :

$$\alpha = max\left(\left| \frac{\lfloor \frac{2^k}{n} \rfloor + 1}{2^k} - \frac{1}{n} \right|, \left| \frac{\lfloor \frac{2^k}{n} \rfloor}{2^k} - \frac{1}{n} \right| \right) \tag{2.11}$$

La figure 2.16 - (d) illustre cette solution de chiffrement pour $n = 5$, $i = 1$, et $k = 9$, engendrant un biais $\alpha \simeq 0,001302$.

Il est à noter que dans le reste du chapitre nous supposerons que la transmission des clefs de chiffrement est réalisée sans erreurs.

D'un point de vue cryptanalyse, il semble difficile de pouvoir attaquer cette méthode de chiffrement, puisque celle-ci n'engendre pas d'incompatibilité avec le standard H.264/AVC, et évite donc la possibilité d'utiliser une attaque par force brute, c'est-à-dire en essayant toutes les solutions possibles. C'est seulement l'étude précise des probabilités d'apparition des symboles, qui pourrait devenir, un angle d'attaque utile pour la cryptanalyse. Or le nombre relativement faible de bits "'chiffrables"' et la multitude de table VLC considérées par le chiffrement (voir annexe A), ne semblent pas pouvoir fournir une indication statistique suffisante à la cryptanalise pour retrouver toute l'information originale.

2.2.4 Résultats expérimentaux

La méthode proposée a été testée dans le cadre du codec de référence du standard H.264/AVC (*Joint Model* version 10.3 : JM 10.3 [4]).

Divers tests ont été menés, avec les séquences de référence 'Foreman', 'Children' et 'Stefan' codées au format QCIF (CIF pour 'Stefan') à 15 *trames/sec* que nous avons ensuite décodées de deux manières différentes :

- avec une version du JM 10.3 dans lequel nous avions rajouté un module décryptant les bits "chiffrables" et ayant accès à la même clef de cryptage que le codeur.
- avec une version du JM 10.3 standard (non modifiée).

La séquence déchiffrée donnée à gauche des figures 2.17, 2.18, 2.19, et 2.20, et la partie chiffrée donnée à droite de ces mêmes figures permet d'apprécier le résultat de décodage obtenu avec un décodeur standard ne connaissant pas l'opération de chiffrement, et permet donc de montrer d'une part la compatibilité effective de la méthode de chiffrement partiel appliquée et de donner une illustration du résultat obtenu pour un décodeur ne connaissant pas la clef de déchiffrement.

Empiriquement on constate que dans chacun des cas présentés, qu'il s'agisse du format CIF ou du format QCIF, qu'il s'agisse de trames I ou P, qu'il s'agisse d'image de haute

Figure 2.17 – Résultats visuels décryptés et cryptés de la 1^{re} image de la séquence 'Foreman' (QCIF, QP=30)

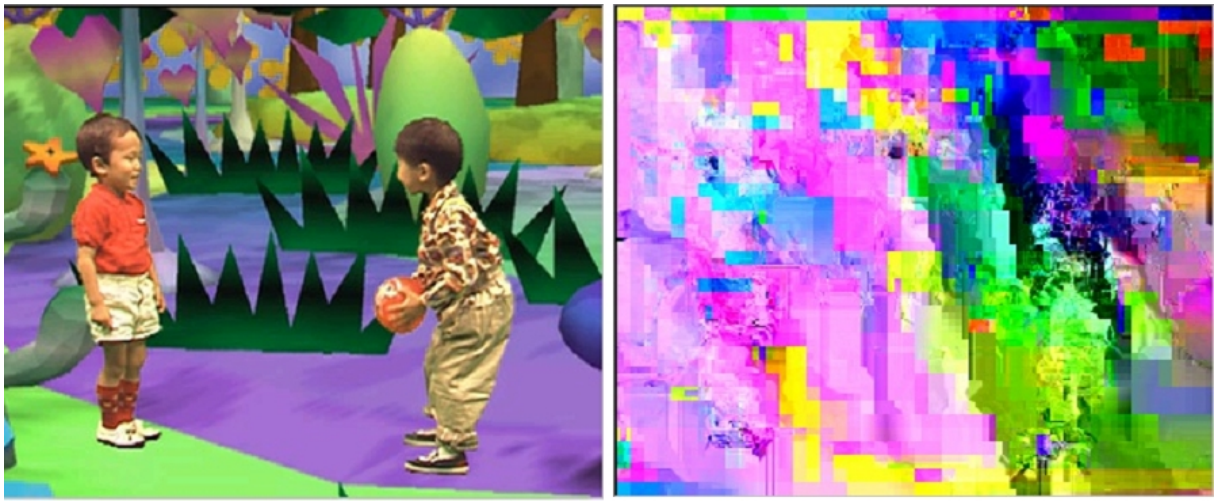

Figure 2.18 – Résultats visuels décryptés et cryptés de la 1^{re} image de la séquence 'Children' (QCIF, QP=15)

qualité (faible valeur de QP) ou de plus basse qualité (grande valeur de QP), et quelque soit le type de séquence, la méthode proposée présente un niveau de chiffrement visuellement très satisfaisant.

Ceci est confirmé plus objectivement par l'évolution du rapport signal à bruit (PSNR) donné en figure 2.21, avec la séquence 'Foreman' au format QCIF, avec QP=30, et un débit de rafraîchissement d'une Intra chaque 15 images (IP=14) pour une longueur de séquence totale de 255 trames. Une dégradation d'environ 25 à 30 dB est observée pour la composante Y, composante de luminance qui est la plus importante dans le rendu visuel ;

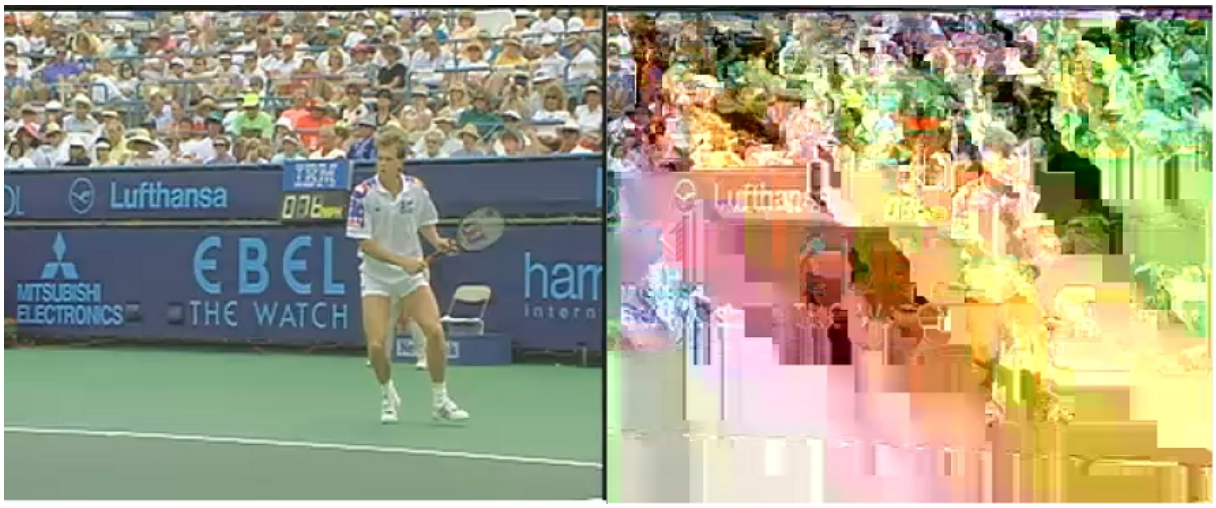

Figure 2.19 – Résultats visuels décryptés et cryptés de la 1^{re} image de la séquence 'Stefan' (CIF, QP=30)

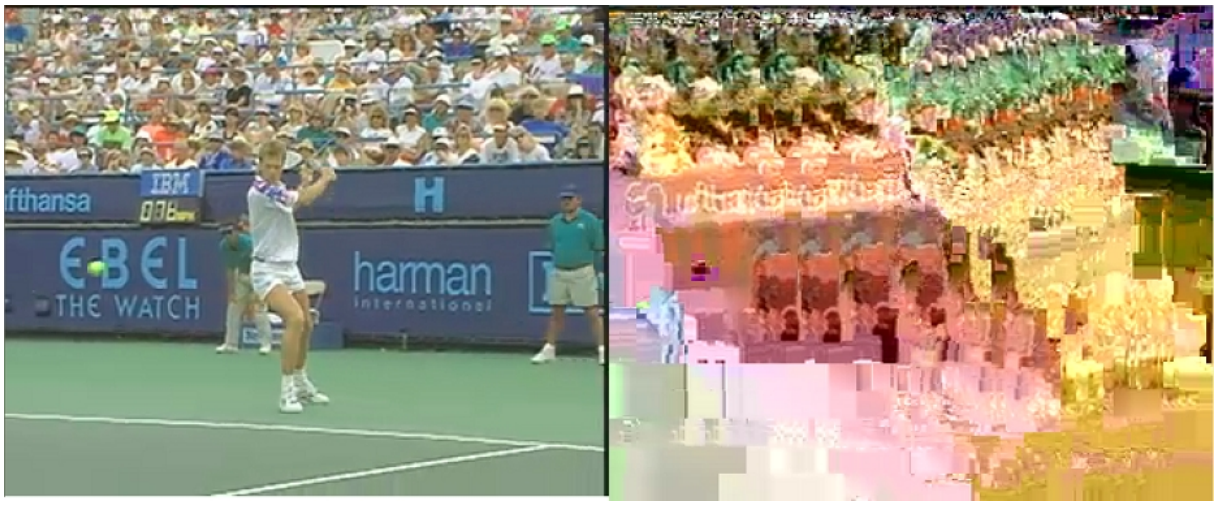

Figure 2.20 – Résultats visuels décryptés et cryptés de la $15^{ème}$ image de la séquence 'Stefan' (CIF, QP=30)

mais aussi d'environ 10 dB pour les composantes chrominances U et V.

Cette différence est due au fait que les composantes chroma sont moins chiffrées dans notre méthode, puisqu'elles ont moins d'importance dans la restitution visuelle, et donc au niveau de la compression.

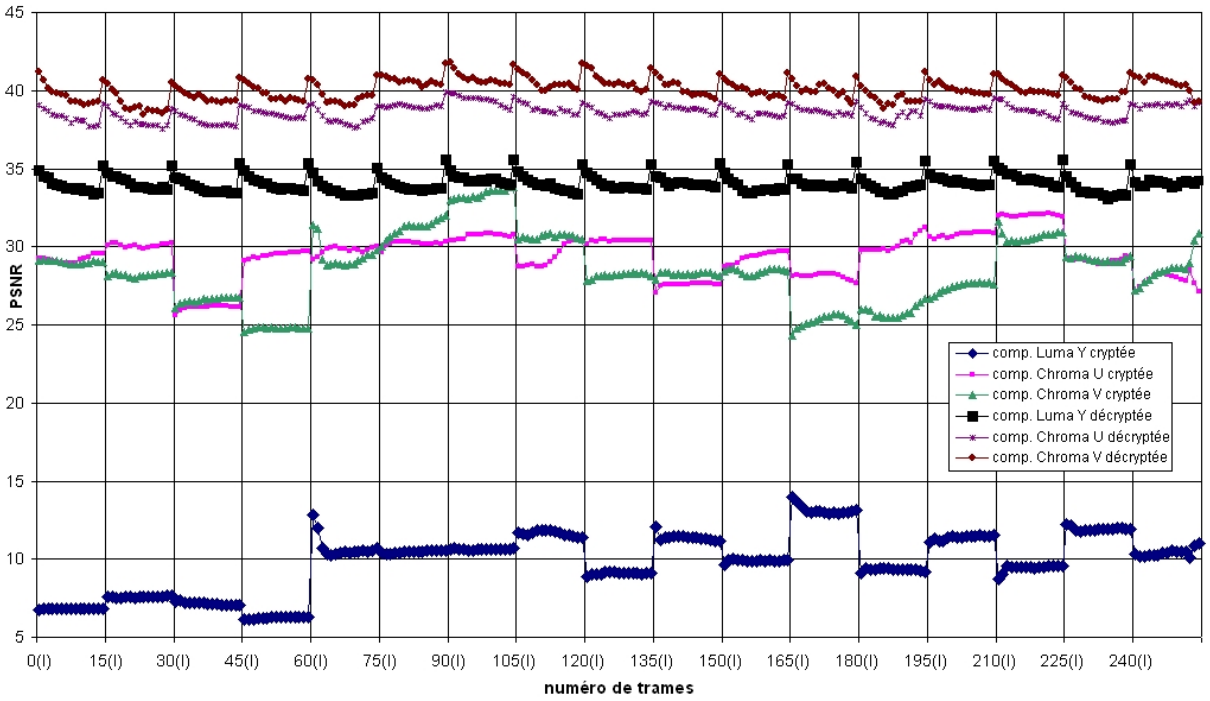

Figure 2.21 – Evolution du PSNR des composantes Y,U et V cryptées et décryptées, pour la séquence 'Foreman' (QCIF, QP=30, 15 *trames/sec*)

2.2.5 Conclusions

Nous avons donc montré qu'il était possible de proposer une méthode de chiffrement partiel visuellement satisfaisant et compatible avec le standard H.264/AVC. Sans avoir réalisé une cryptanalyse complète de ce procédé de chiffrement, l'estimation du faible nombre de bits chiffrés dans le flux, et la limitation du biais engendré dans le cas d'une répartition asymétrique nous permettent de penser qu'il est théoriquement difficile de décrypter aisément ce type de chiffrement.

Chapitre 3

Mélange de trames et scalabilité temporelle *

Je mélange,... je mélange ! et toc, ça marche !
Garcimore, *prestidigitateur prestigieux du vingtième siècle (France)*

Lorsque ITU-T et ISO ont établi leur spécification commune, H.264/AVC [2], ce standard devait s'appliquer à des systèmes aussi différents que les communications sans fil ou filaires, et ses applications cibles devaient être aussi diverses que la visiophonie basse qualité que les services vidéo haute définition. Cependant, H.264 dans sa version de base, présente un inconvénient majeur : il ne supporte pas la scalabilité. Des solutions ont été actuellement proposées dans la littérature en utilisant des trames de type B (bidirectionnelles) qui n'offrent que peu de scalabilité ([10], [49]). ITU-T et ISO ont d'ailleurs défini un nouveau standard, baptisé H.264/SVC (pour *Scalable Video Coding*), qui présenterait la particularité, à partir d'un seul flux (et donc d'une phase unique de codage), de pouvoir fournir des flux variables déduis des schémas de distribution comportant un grand nombre de configurations possibles.

En parallèle, d'autres solutions reposant sur des décompositions en sous-bande compensées en mouvement (ou *Motion compensated : MC*) ont été proposées [39], [44], [14] et [62]. Ces solutions ne sont malheureusement pas compatibles avec la norme H.264/AVC, et introduisent souvent un haut niveau de complexité, qui ne pourrait être acceptable pour une utilisation dans un équipement mobile léger. L'objectif de ce chapitre est de présenter une nouvelle méthode basée sur la théorie des bancs de filtres permettant d'obtenir des propriétés de scalabilité temporelle.

Dans une première partie, nous rappelons donc la théorie et les principes des bancs de filtres. L'application de cette théorie au codage vidéo prédictif, par notre méthode de "mélange de trames" est ensuite développée en seconde partie, avec l'application à la

∗. Certaines parties de ce chapitre ont été publiées dans la conférence *ICASSP'05* [66] et le journal *EURASIP Journal on Applied Signal Processing (2006)* [67].

norme H.264/AVC. Nous montrerons que le mélange de trames est compatible avec ce standard. Enfin, dans une troisième partie, nous montrerons comment utiliser ces mélanges de trames pour fournir un flux scalable temporellement.

3.1 Principe des bancs de filtres

3.1.1 Présentation théorique des bancs de filtres

Les bancs de filtres, figure 3.1, sont un outil de décomposition du signal, où un signal d'entrée $x[n]$ est filtré par M filtres $G_i[n]$, et où les sorties de ces filtres peuvent être sous-échantillonnées par le facteur M. Les sorties des sous-échantillonneurs sont notées $h_i[m]$. D'un côté, le banc d'«analyse» est l'étape de décomposition proprement dite du signal $x[n]$ en sous-bandes $h_i[m]$. De l'autre côté, le banc de «synthèse» est celui qui reconstitue le signal à partir de ses sous-bandes.

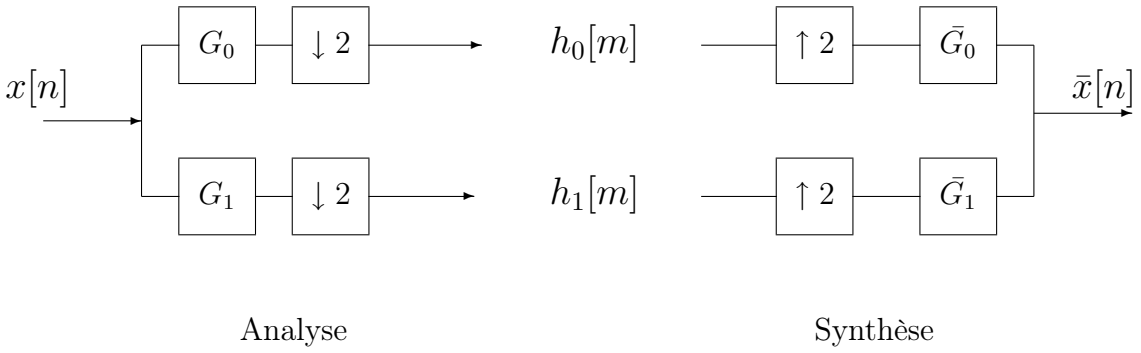

Figure 3.1 – Banc de filtres d'analyse-synthèse avec M=2

Pour que ces filtres assurent que le signal reconstruit $\hat{x}[n]$ soit strictement égal au signal d'entrée $x[n]$, il faut que les filtres $G_i[n]$ (respectivement $\bar{G}_i[n]$) respectent certaines conditions, dites de "reconstruction parfaite" [57].

3.1.2 Application des bancs de filtre au codage vidéo par blocs

Dans le cas des codeurs vidéo par blocs, le principe fondamental de la compression

s'effectue à travers d'une part des trames codées par elles mêmes, appelées Intra (I ou IDR), et d'autre part des trames prédites, appelées Inter qui peuvent être prédites (P) ou bidirectionnelles (B), qui utilisent les images précédemment codées.

Nous expliciterons ici le principe uniquement pour les trames prédites P, puisque celles-ci sont les plus communes et présentes dans toutes les configurations des codeurs vidéo par prédiction, mais le principe est applicable aux trames bidirectionnelles.

Considérons un groupe d'images (ou GOP pour *Group Of Pictures*) de N trames, numérotées par leur ordre temporel : $\{0, 1, 2, \cdots, N-1\}$. Dans une prédiction classique, la première image est codée en mode Intra, et les suivantes sont codées en mode Inter dans le même ordre que celui d'apparition. Cette méthode classique et simple a pour but de tirer avantage de la redondance temporelle qui existe dans une séquence vidéo entre deux images successives. Ainsi, plus l'image est "proche" temporellement et plus la compression sera efficace. Cette décomposition classique que nous appellerons "normale", est illustrée en figure 3.2 Les dépendances entre les trames sont illustrées par des flèches sur la figure. Ainsi dans l'exemple, la trame 1 dépend de la trame 0, et la trame 2 dépend de la trame 1 et par conséquent de la trame 0, et ainsi de suite...

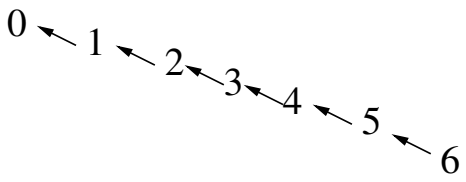

Figure 3.2 – Configuration "normale", GOP de taille 7

En retranscrivant ce type de configuration en banc de filtres d'analyse, on obtient la figure 3.3, où $\bar{x}_{7t}$ représente la trame codée en Intra, et $\bar{h}_{7t+i}$ avec $i \in \{1, .., 6\}$ représente la $i^{ème}$ trame codée en Inter (trames P) du $t^{ième}$ GOP. Le module "P" représente le procédé de prédiction par compensation de mouvement faisant appel à une ou plusieurs références : la deuxième trame codée $\bar{h}_{7t+1}$ a été prédite à partir de la référence $\bar{x}_{7t}$, alors que la dernière trame codée, $\bar{h}_{7t+6}$, a été prédite à partir de toutes les trames précédemment codées ($\bar{x}_{7t}$, $\bar{h}_{7t+1}$,..., $\bar{h}_{7t+5}$).

Ce banc de filtres a donc autant de niveaux de décomposition que d'images contenues dans le GOP. La première trame codée devient alors la plus importante, puisque toutes les trames codées suivantes sont susceptibles de la prendre comme référence directement ou indirectement. Si cette première trame décodée est erronée ou manquante, alors l'erreur se propagera sur le reste du GOP. Dans la configuration "normale", chaque trame codée a donc un niveau d'importance différent, et plus on avance dans le GOP, plus le niveau d'importance décroît.

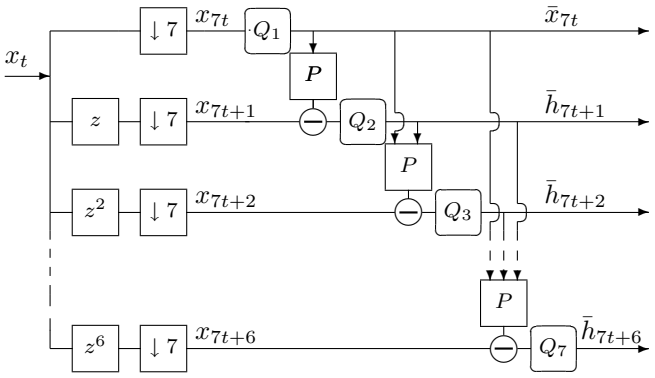

Figure 3.3 – Schéma d'une décomposition avec une prédiction en boucle fermée d'une configuration normale, GOP de taille 7

3.2 Mélange de trames ou "frame shuffle"

3.2.1 Principe du mélange de trames

Exemples de configurations : "sapin" et "miroir"

Le principe du "mélange de trame" (ou *"frame shuffle"* en anglais) consiste à ne plus considérer l'ordre de codage comme étant nécessairement l'ordre causal des images de la séquence. Un groupe d'images de N trames peuvent alors être numérotées par leur ordre d'apparition (temporel) : $\{0, 1, 2, \cdots, N-1\}$, et dénotées par leur ordre de (dé)codage : $\{A, B, \cdots, G\}$. Le mode de configuration "sapin", présenté sur la figure 3.4, illustre cette double notation : la première trame codée sera la quatrième trame dans l'ordre d'apparition dans la séquence (3_A), et la deuxième trame codée sera la troisième trame dans l'ordre temporel (2_B), et ainsi de suite ...

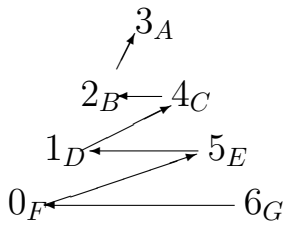

Figure 3.4 – Configuration en "sapin", GOP de taille 7

De même, si l'on considère la configuration en "miroir", présentée sur la figure 3.5, on voit que la première trame codée sera la quatrième trame dans l'ordre d'apparition dans la séquence (3_A), la deuxième trame codée sera la troisième trame dans l'ordre temporel (2_B), et ainsi de suite, jusqu'à atteindre la cinquième trame codée (4_E), à laquelle il n'est

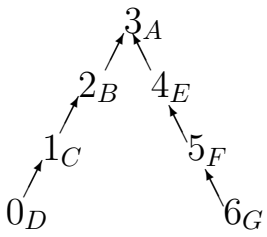

Figure 3.5 – Configuration en “miroir”, GOP de taille 7

permis que de se référer à la première trame codée (3_A). On limite ainsi les trames dénotées E, F et G à se référer seulement aux trames dénotées A, E et F.

En retranscrivant en banc de filtres les configurations “sapin” et “miroir”, nous obtenons respectivement les figures 3.6 et 3.7. Considérons le cas du “sapin” : nous obtenons un banc de filtres comportant autant de niveaux de décomposition que d'images contenues dans le GOP, comme pour la configuration dite “normale”. En revanche, dans le cas de la configuration en “miroir” nous obtenons deux séries de quatre niveaux de décomposition.

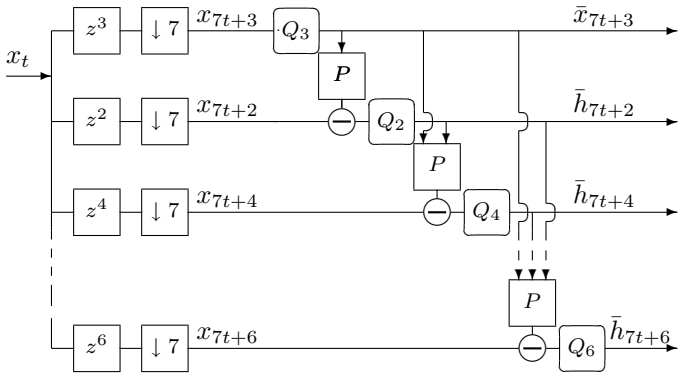

Figure 3.6 – Schéma d'une décomposition avec une prédiction en boucle fermée d'une configuration “sapin”, GOP de taille 7

Dans la figure 3.6, on peut remarquer que toutes les trames précédemment codées peuvent servir de références aux trames suivantes : la première trame codée est donc la plus importante puisqu'elle est nécessaire à toute les autres. De même, chaque trame codée devient importante pour toutes celles qui suivent, puisqu'elle est susceptible de leur servir de référence. En revanche, dans la figure 3.7, les trames ne dépendent que de certaines trames codées précédemment. Le niveau d'importance sera alors approximativement lié au nombre de trames que pourrait utiliser la trame d'intérêt comme référence. Ainsi en mode “miroir”, la deuxième et la cinquième trame codée sont de même importance.

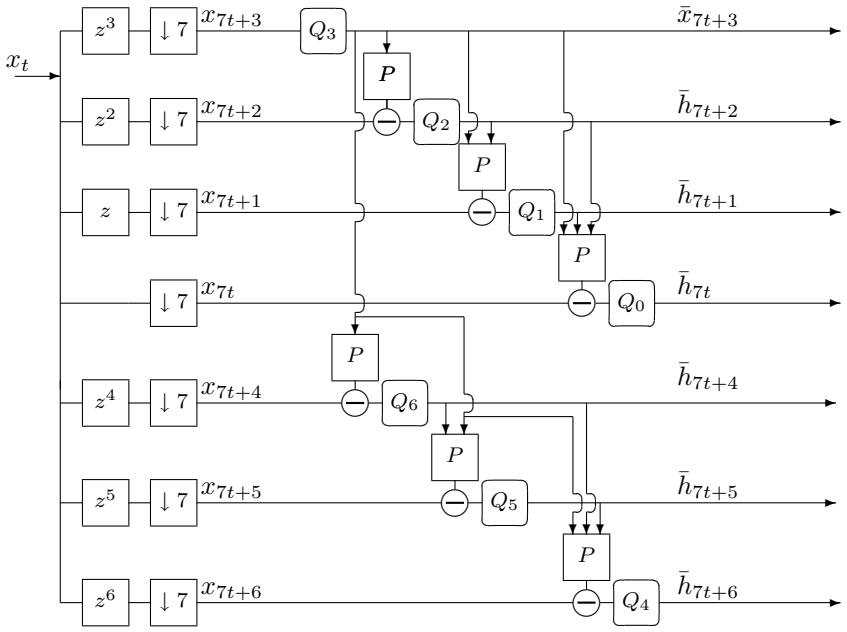

Figure 3.7 – Schéma d'une décomposition avec une prédiction en boucle fermée d'une configuration "miroir", GOP de taille 7

3.2.2 Application du "mélange de trames" à la norme H.264/AVC

Au sein de la norme H.264/AVC comme introduit au paragraphe 1.2.3, il existe deux systèmes de numérotation des trames vidéo. La première solution, qui repose sur des "numéros de trame" ou "frame_num" correspond à l'ordre de décodage des unités d'accès dans le flux. Ce paramètre est décodé à partir de chaque en-tête d'un segment d'image et augmente dans l'ordre de décodage des unités d'accès (NAL). Cependant, ce numéro de trame n'indique pas nécessairement l'ordre d'affichage final que le décodeur utilisera. Cette dernière valeur correspond à un autre champ de l'en-tête, désigné habituellement par le terme «*Picture Order Count*», ou *POC* pour numéro d'apparition de l'image. Ce POC correspond à l'ordre d'affichage des trames décodées qui sera utilisé par le décodeur.

Il est à noter qu'une image Intra de type IDR (comme définie au paragraphe 1.2.1), a pour "frame_num" et POC une valeur nulle. Ceci explique que contrairement au "frame_num", les POC peuvent prendre des valeurs négatives, ce qui arrivera notamment lorsque la trame IDR se trouve au centre du GOP, comme en mode miroir par exemple.

Un exemple de résultat obtenu par ré-arrangement du POC est donné dans la table 3.1.

Cette double numérotation va nous permettre d'appliquer le principe du mélange de trames à la norme H.264/AVC de façon transparente pour le décodeur. Ainsi, pour implémenter ce principe au sein de notre codeur H.264, nous rajoutons un entrelaceur, présenté

Numéro de trame	Type	Frame_num	POC	ordre d'apparition
0	IDR	0	0	2
1	P	1	-2	0
2	P	2	-1	1
3	P	3	1	3
4	P	4	2	4
5	IDR	0	0	7
6	P	1	-2	5
7	P	2	-1	6
8	P	3	1	8
9	P	4	2	9

Table 3.1 – Exemple d'arrangement pour l'ordre d'affichage.

dans la figure 3.8, qui mélange les trames de la séquence originale, avant le processus normal de codage. Pour rester compatible au niveau du décodeur, nous modifions ensuite, grâce à la table d'équivalence, les valeurs des POC en fonction de leur valeur de "frame_num". Le décodeur JM 10.3 [4] est alors utilisé sans modifications et conformément au standard, puisque le flux obtenu est complètement compatible. Le seul inconvénient notable de cette méthode est qu'elle induit un délai dû à la mise en mémoire de ces trames dans une mémoire tampon lors du mélange des trames.

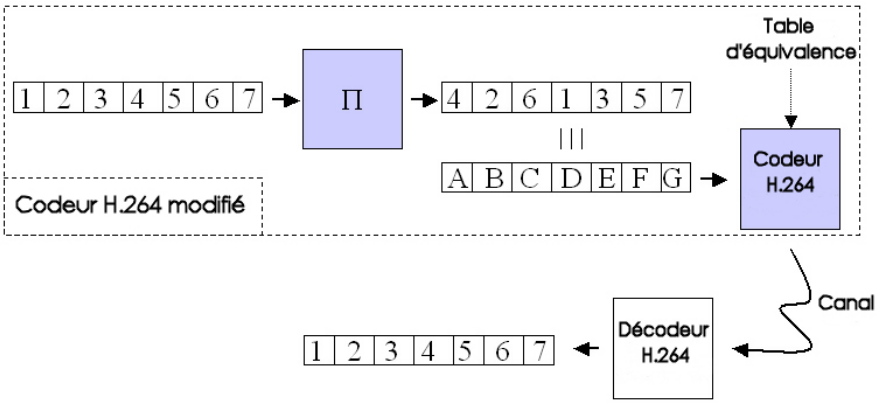

Figure 3.8 – Chaîne de simulation contenant le module de mélange de trames (les modifications introduites sont représentées en grisé)

Décrivons de manière plus précise le fonctionnement du codeur H.264/AVC et du référencement de trames afin de comprendre pourquoi le mélange de trames est effectivement efficace et possible. Contrairement aux standards précédents de codage vidéo, qui étaient utilisés en mode simple référence, c'est-à-dire où la prédiction d'apparition est faite uniquement en utilisant une image précédente donnée, le standard H.264 permet d'utiliser jusqu'à 16 trames différentes comme références pour chaque segment P (ou en anglo-saxon P-slice) et jusqu'à 32 trames différentes pour chaque segment B (ou B-slice).

Les images qui sont codées et décodées, et ensuite sélectionnées pour servir de références sont mémorisées dans une mémoire contenant les images décodées nommée DPB (*decoded picture buffer*). Une distinction existe dans cette mémoire entre, d'une part, les images qui ne serviront de références que pour celles très proches, qui sont appelées en anglais *short term reference picture* et qui sont alors référencés par leur POC, et d'autre part, les images qui serviront de références pour des images éloignées, qui sont référencées par un compteur spécifique nommé *LongTermPicNum* (ou *long term reference picture*). Lorsque la mémoire DPB est pleine, seule l'image référencée comme *short term reference picture* la plus ancienne est retirée de la mémoire. Les références "long term" ne sont pas éliminées, excepté par une commande explicite dans le flux de bits. On peut ainsi de cette manière limiter le nombre d'images sur lesquelles une trame peut se référer, et garantir que pour des groupes d'images importants, l'image codée Intra puisse continuer à être une référence utilisable. Les résultats de simulation présentés dans le reste de ce chapitre ont été obtenus en intégrant ces commandes (*Memory Management Control Operation : MMCO*) au niveau du codeur modifié pour réaliser le mélange.

Typiquement, le cas de la configuration "miroir" (voir figure 3.5), au moment de coder la cinquième trame (notée E), on force le codeur H.264 à n'utiliser que la première trame codée (notée A) comme référence. On restreint les trames suivantes à ne prendre comme références que la première trame et celles codées juste après la quatrième, comme schématisé en figure 3.7 par la représentation équivalente en banc de filtres de ce "miroir". Les résultats sont nécessaires pour garantir les non-dépendances recherchées, et donc définir les niveaux d'importance et de rafinement respectifs des différentes trames.

Performance du "mélange de trames" en terme de PSNR

Après avoir modifié le logiciel de référence JM 10.3 pour introduire notre fonctionnalité de mélange de trames comme illustré en 3.8, nous avons testé les deux configurations "sapin" et "miroir" en les comparant aux moyennes de PSNR à même débit avec la configuration "normale". Ces résultats, obtenus avec les trois séquences de référence 'Foreman', 'Akiyo' et 'Mobile' en QCIF, sont donnés dans la table 3.2.

Ces résultats montrent que le "mélange de trames" avec les configurations "sapin" et "miroir", loin de dégrader l'efficacité du codeur H.264, permet au contraire de gagner environ de 0.20 dB à 0.30 dB de PSNR à débit équivalent. Ce gain vient du choix de placer la trame codée en Intra au milieu du GOP. Les calculs présentés en annexe B démontrent dans un cas simplifié l'intérêt du positionnement de la trame Intra au centre des trames codées, et plus généralement du placement d'une référence mieux quantifiée entre deux trames prédites plutôt qu'en tête de GOP.

Séquence	Débit	Configuration	PSNR moyen
Akiyo	64 kbit/s	normale	40.19 dB
Akiyo	64 kbit/s	sapin	40.33 dB
Akiyo	64 kbit/s	miroir	40.41 dB
Foreman	64 kbit/s	normale	32.19 dB
Foreman	64 kbit/s	sapin	32.36 dB
Foreman	64 kbit/s	miroir	32.54 dB
Mobile	128 kbit/s	normale	27.94 dB
Mobile	128 kbit/s	sapin	28.26 dB
Mobile	128 kbit/s	miroir	28.32 dB

Table 3.2 – Comparaison débit-distorsion pour différentes configurations avec une taille de GOP de 7 trames

Comportement du "mélange de trames" dans un milieu bruité

Plus encore que le gain absolu en compression, la principale propriété du "mélange de trame" est d'offrir un comportement différent dans le cas d'une transmission sur un canal bruité ou à effacements. Ainsi dans le cas où une trame est affectée ou effacée, les trames décodées qui suivront celle-ci seront erronées et l'impact visuel de ces erreurs sur la séquence décodée sera différent dans chaque configuration de celle de la configuration dite "normale".

Considérons, par exemple, une transmission avec pertes et que l'on perd la troisième trame codée (trame notée C) dans chacune des configurations proposées. On peut comprendre que les trames susceptibles d'utiliser celle-ci comme référence (les trames dénotées D, E, F et G) sont placées différemment dans chaque configuration :

- dans la configuration "normale", les trames sont codées dans le même ordre que l'ordre temporel. Ce seront donc les trames numérotées 2, 3, 4, 5, et 6 qui seront mal reconstituées comme l'illustre la figure 3.9.
- dans la configuration "sapin", ce sont les trames numérotées 0, 1, 4, 5 et 6 (c'est-à-dire, les images se trouvant au bord du GOP) qui seront corrompues comme l'illustre la figure 3.10.
- dans la configuration "miroir", seules les trames numérotées 0 et 1 seront affectées, comme l'illustre la figure 3.11.

La limitation de la propagation d'erreurs, variable naturellement selon la matrice de mélange utilisée, permet de prédire l'impact visuel des erreurs sur celles qui suivent la trame erronée ou perdue. Le "frame shuffle" et ses configurations "sapin" et surtout "miroir" présentent donc, sans dégradation du rapport débit-distorsion, de meilleurs comportements face aux transmissions erronées.

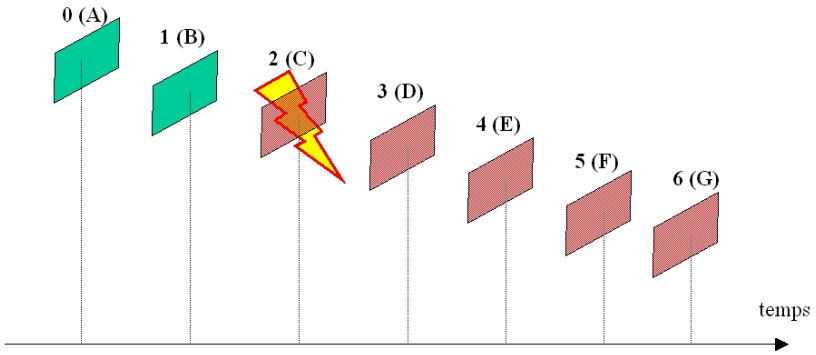

Figure 3.9 – Schéma représentant l'influence de la perte de la troisième trame dans un GOP de 7 trames pour la configuration "normale"

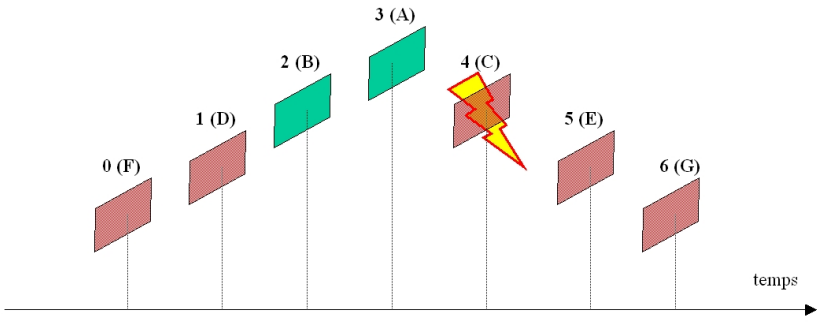

Figure 3.10 – Schéma représentant l'influence de la perte de la troisième trame dans un GOP de 7 trames pour la configuration "sapin"

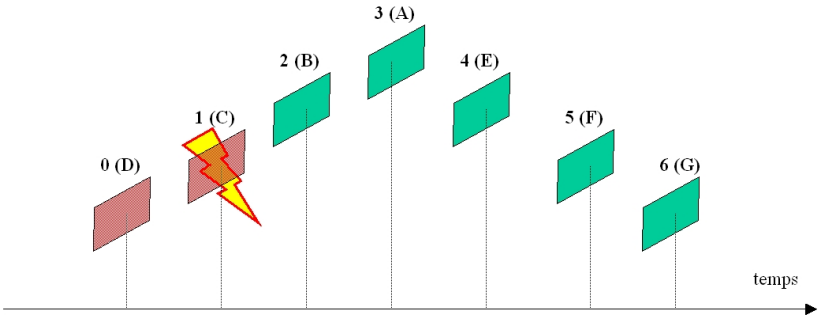

Figure 3.11 – Schéma représentant l'influence de la perte de la troisième trame dans un GOP de 7 trames pour la configuration "miroir"

3.3 Scalabilité temporelle

3.3.1 Définitions et propriétés

Par définition, la mise à l'échelle, granularité ou scalabilité ("scalability" en anglais) désigne la possibilité de transmettre et de fournir dans un même signal tous les éléments correspondant à différents niveaux de qualité (codage hiérarchique) qui pourront être exploités différemment en fonction des conditions de transmission ou du type de terminal utilisé par l'usager. Les informations peuvent ainsi être transmises selon un flux principal et des flux complémentaires qui ne sont exploités qu'en fonction de la performance du réseau ou du récepteur utilisé. On pense par exemple à la réception d'un même contenu sur un téléphone portable et sur un récepteur de télévision fixe ou mobile. Cette méthode contient plusieurs sous-ensembles imbriqués, qui peuvent représenter le contenu vidéo initial à différentes résolutions, et qui sont supposés correspondre à une compression efficace de l'information qu'elle contient.

Différentes méthodes sont alors envisageables et cumulables :

- "*Spatial scalability*" (scalabilité spatiale) : les données sont réparties en sous-ensembles représentant un contenu avec une résolution spatiale variable.
- "*Quality scalability*" (scalabilité qualitative) : les données sont réparties dans un certain nombre de couches qui permettent d'obtenir divers niveaux de qualité, correspondant à des débits différents.
- "*Temporal scalability*" (scalabilité temporelle) : le flux de données est réparti en différentes résolutions temporelles (par exemple par variations du nombre d'images par seconde).

La scalabilité permet donc d'échelonner l'information, en rendant le contenu accessible à différentes résolutions. Généralement ces résolutions sont des multiples de celle de base, et le rapport entre chaque niveau de raffinement est appelé facteur d'échelle. La figure 3.12 schématise un exemple de granularité temporelle de facteur d'échelle égale à deux, échelonné en trois niveaux, où le premier niveau de résolution correspond à un contenu vidéo à $7,5$ trames par secondes. Dans cet exemple, si l'on a accès aux sous ensembles de base et celui du premier niveau de raffinement, nous obtenons un contenu vidéo à 15 trames par seconde, et si l'on rajoute le dernier niveau de raffinement on obtient le contenu vidéo à la résolution temporelle originale (ici 30 trames par seconde).

Les intérêts de la granularité sont multiples. Avec elle, il est possible de répondre à différents besoins ou capacités sans avoir besoin d'une réévaluation des conditions de compression à chaque instant. En particulier, du point de vue du fournisseur de contenu ou de service, l'intérêt est que la vidéo peut être compressée une seule fois, pour être utilisée plus tard à différents débits, pour avoir la possibilité de commuter à un débit différent selon les capacités de largeur de bande du lien. Pour l'utilisateur, l'intérêt réside dans le fait que ce dernier peut facilement changer ses exigences et sa demande en temps réel pour l'adapter aux besoins courants, et en situation de mobilité, par exemple dans le cas

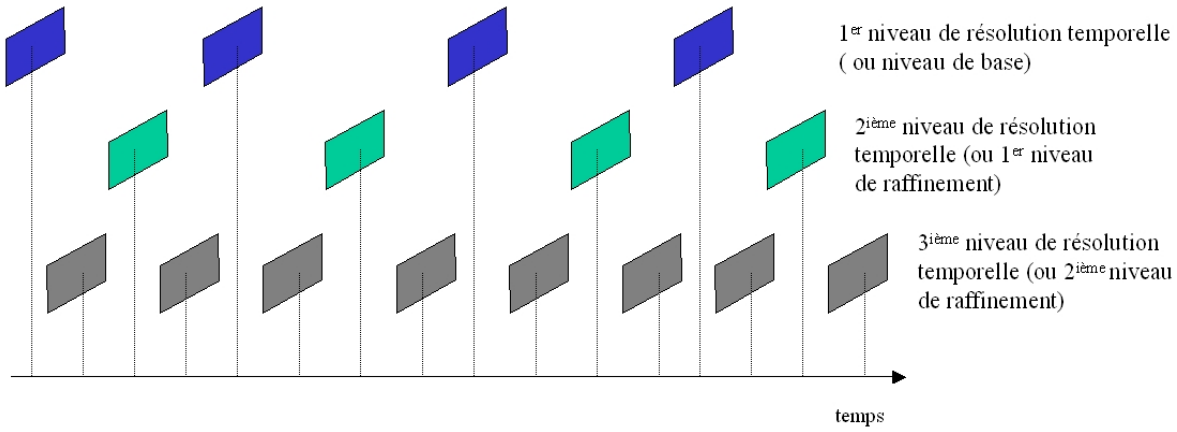

Figure 3.12 – Exemple de 3 sous ensembles décrivant chacun une résolution temporelle différente (ici par exemple : 30, 15 et 7, 5 trames/sec)

de changement de réseau mobile, le changement s'opère de façon totalement transparente.

Dans un contexte de communication sans fil, la granularité permet donc de s'adapter rapidement à la largeur de bande et au débit utile disponible, qui peuvent changer rapidement du fait des conditions de transmission de canal, du réseau existant pour la transmission et de la présence possible des autres utilisateurs et des interférences. Outre l'adaptation transparente du débit à la bande passante disponible, la scalabilité permet, dans le cas d'une transmission dans un canal bruité, d'être plus robuste. En effet, dans le cas où les données transmises sont en partie corrompues, les propriétés de scalabilité peuvent permettre d'obtenir un contenu de résolutions plus faibles mais complet et sans erreurs, à la différence d'un format classique.

Enfin, la granularité permet une hiérarchisation de l'information contenue dans les divers sous-ensembles. Pour accéder à un niveau de résolution donné, on doit alors pouvoir avoir accès au niveau de résolution précédent. Il faut donc nécessairement avoir accès au niveau de "base", pour pouvoir obtenir n'importe quelle résolution. Pour chaque couche de raffinement , on peut associer un niveau "d'importance". Dans le cas d'une transmission dans un canal bruitée, cette notion de hiérarchisation de l'information nous permettra d'appliquer une protection inégale efficace face aux erreurs (voir chapitre 4).

Il est à noter que notre méthode s'adapte aussi à la future norme SVC, puisque celle-ci reprend comme schéma de base celui de H.264/AVC (profile *baseline*). Nous n'avons utilisé dans notre méthode que des options accessibles à tous les profils de H.264/AVC, ce qui nous permet dès à présent de pouvoir proposer la scalabilité temporelle simple et efficace pour cette norme.

3.3.2 Implémentation de la scalabilité temporelle dans H.264 grâce au “mélange de trames”

Comme nous avons vu au paragraphe précédent, un flux scalable possède des sous-ensembles caractérisant chacun un niveau de raffinement particulier. Chaque niveau de résolution temporelle a besoin de tous les niveaux de raffinement précédents pour être reconstruit. Plus le niveau de résolution temporelle est donc faible et plus son contenu est important, puisqu'il est nécessaire à tous les autres niveaux de résolution supérieurs. Dans le modèle en banc de filtres, ceci revient à associer chaque niveau de raffinement à des niveaux de décomposition différents.

Décompositions symétriques

En suivant ce principe, nous avons construit deux configurations scalables nommées “zig-zag” et “arbre”, respectivement représentées dans les figures 3.13 et 3.14, qui sont retranscrites en bancs de filtres respectivement dans les figures 3.15 et 3.16.

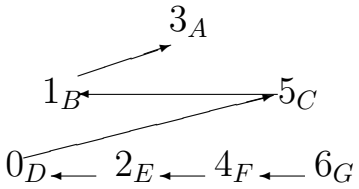

Figure 3.13 – Configuration en “zig-zag”, GOP de taille 7

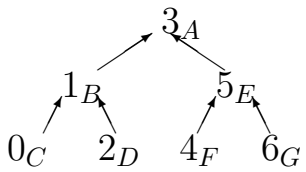

Figure 3.14 – Configuration en “arbre”, GOP de taille 7

Ces solutions proposent d'obtenir une scalabilité temporelle avec un facteur d'échelle égal à deux, et L niveaux de détails avec, dans l'exemple, $L = 3$. Ce sont des modèles réguliers, adaptés à des GOPs de taille $N_{GOP} = 2^L - 1$. Le premier niveau de résolution est constitué d'une trame Intra placée au milieu du GOP (dans l'ordre temporel). Cette trame Intra est codée en IDR, puisque nous supposons des GOPs “fermés”, c'est-à-dire indépendamment codés les uns des autres. Les autres niveaux de décomposition sont définis en sélectionnant dans chaque sous-GOP, les trames au milieu de ces sous ensembles restants.

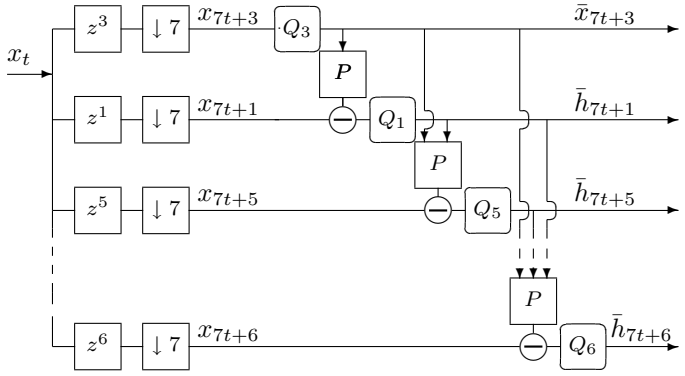

Figure 3.15 – Schéma d'une décomposition avec une prédiction en boucle fermée d'une configuration "zigzag", GOP de taille 7

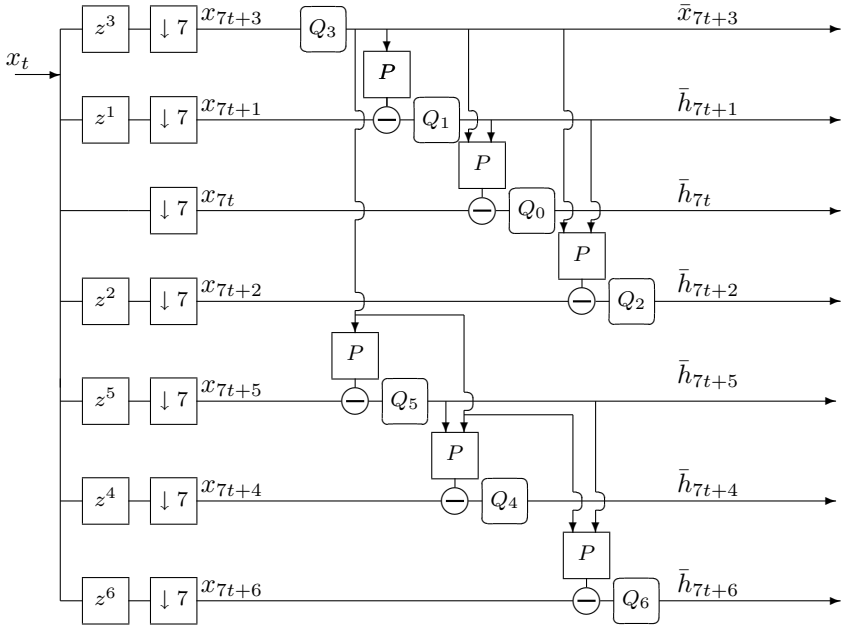

Figure 3.16 – Schéma d'une décomposition avec une prédiction en boucle fermée d'une configuration "arbre", GOP de taille 7

Mais ces constructions peuvent se généraliser à d'autres facteurs d'échelle (R), et à d'autres tailles de GOP ($N_{GOP} \neq 2^L - 1$), en définissant la décomposition de la sorte :

- On sélectionne $R-1$ trames placées à égales distances dans le GOP, c'est-à-dire celles numérotées par les indices $m_i = \lfloor i(N_{GOP}+1)/R \rfloor$ pour $i \in \{1, R-1\}$.
- On définit alors entre ces $R-1$ trames, R sous-GOPs. Ces $R-1$ trames appartiennent alors à un nouveau niveau de décomposition.
- On répète récursivement pour chaque sous-GOP les deux étapes précédentes, en considérant à chaque itération le sous-GOP comme un GOP.

Par exemple, on peut appliquer la configuration “zigzag” avec deux niveaux de détails et un facteur d'échelle de trois, pour un GOP de taille 9, comme illustré dans la figure 3.17. Dans ce cas, le premier niveau de détails ne contient plus seulement une IDR, mais une IDR (dénotée 3_A) et une trame prédite (dénotée 6_B).

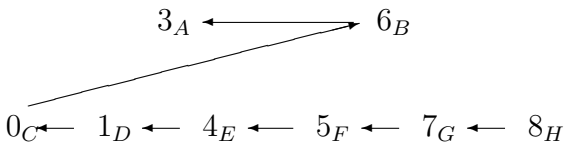

Figure 3.17 – Configuration en “zig-zag”, avec un facteur d'échelle de 3, et un GOP de taille 9

Performance des solutions scalables en terme de qualité visuelle

Nous avons donc testé ces deux nouvelles configurations scalables, “zigzag” et “arbre”, en les comparant aux moyennes de PSNR à même débit obtenues avec les configurations “normale”, “sapin” et “miroir”. Ces résultats, obtenus avec les trois séquences de référence 'Foreman', 'Akiyo' et 'Mobile' en QCIF, sont donnés dans la table 3.3 :

Séquence	Débit	Configuration	Moyenne de PSNR
Akiyo	64 kbit/s	normale	40.19 dB
Akiyo	64 kbit/s	sapin	40.33 dB
Akiyo	64 kbit/s	miroir	40.41 dB
Akiyo	64 kbit/s	zigzag	40.32 dB
Akiyo	64 kbit/s	arbre	40.32 dB
Foreman	64 kbit/s	normale	32.19 dB
Foreman	64 kbit/s	sapin	32.36 dB
Foreman	64 kbit/s	miroir	32.54 dB
Foreman	64 kbit/s	zigzag	32.28 dB
Foreman	64 kbit/s	arbre	32.48 dB
Mobile	128 kbit/s	normale	27.94 dB
Mobile	128 kbit/s	sapin	28.26 dB
Mobile	128 kbit/s	miroir	28.32 dB
Mobile	128 kbit/s	zigzag	28.27 dB
Mobile	128 kbit/s	arbre	28.30 dB

Table 3.3 – Comparaison débit-distorsion pour différentes configurations avec une taille de GOP de 7 trames.

Ces résultats montrent que les configurations scalables du “mélange de trames”, comme les configurations “miroir” et “sapin”, ne dégradent pas les performances du codeur. On peut même gagner jusqu'à environ 0.30 dB en qualité vidéo dans certains cas.

Similairement, les figures 3.18, 3.19 et 3.20, nous donnent les évolutions débit/distorsion pour les deux séquences de référence 'Foreman' au format QCIF et CIF, et 'Mobile' au format CIF.

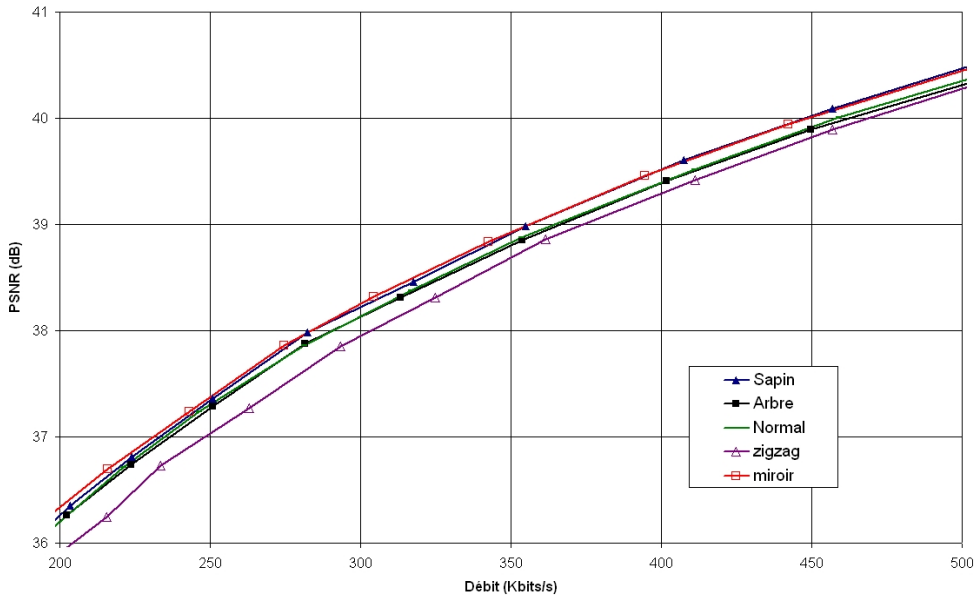

Figure 3.18 – Comparaison de l'efficacité de la compression avec différentes configurations de "frame shuffle", pour la séquence 'Foreman' (en CIF, à 30 trames/s, $I\ P_{14}$)

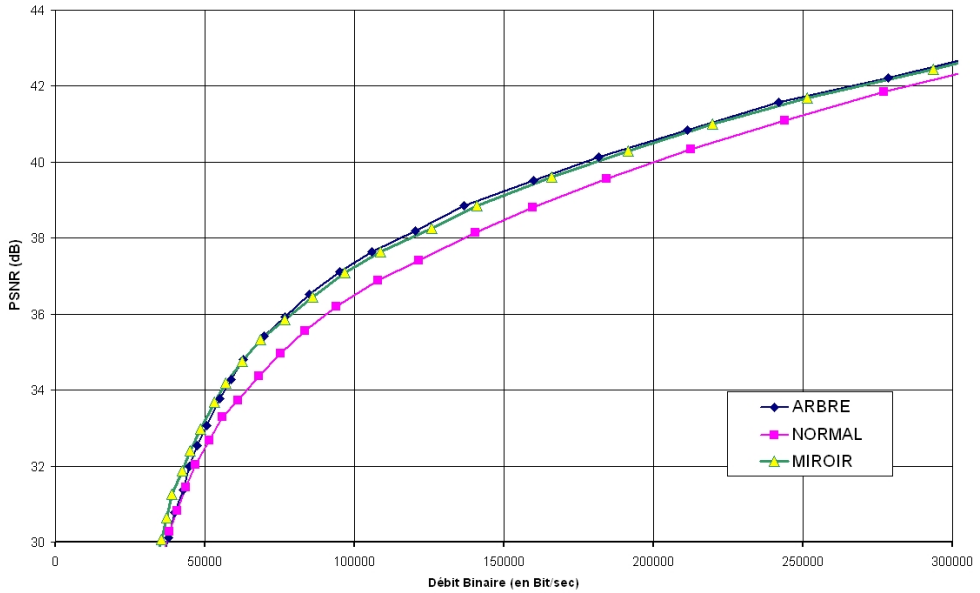

Figure 3.19 – Comparaison de l'efficacité de la compression avec différentes configurations de "frame shuffle", pour la séquence 'Foreman' (en QCIF, à 15 trames/s, $I\ P_{14}$)

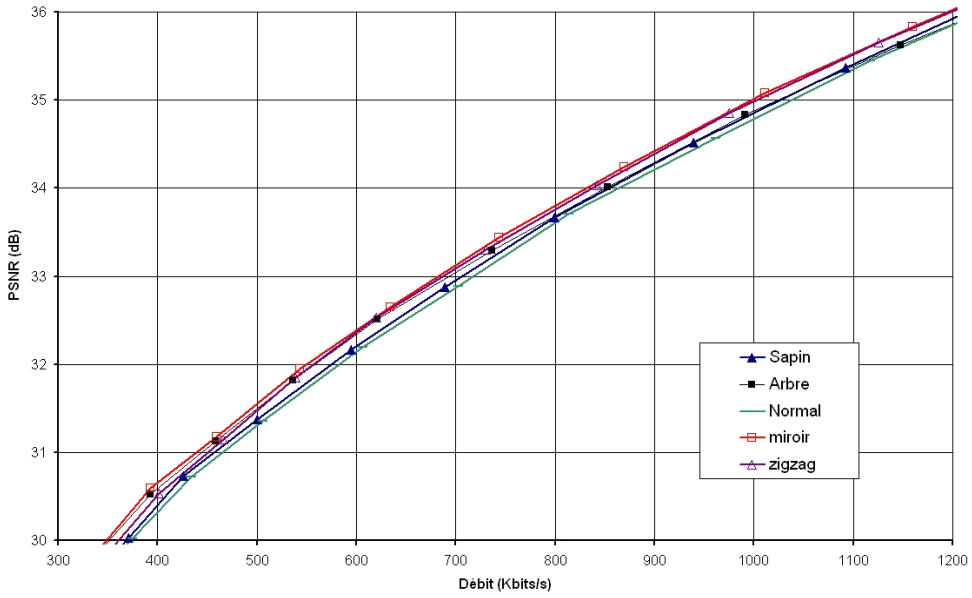

Figure 3.20 – Comparaison de l'efficacité de la compression avec différentes configurations de "frame shuffle", pour la séquence 'Mobile' (en CIF, à 30 trames/s, $I\ P_{14}$)

Ces courbes montrent le réel gain en PSNR offert par la méthode du "frame shuffle" pour n'importe quel débit cible. Il est à noter qu'à des débits très importants la différence entre les diverses configurations devient négligeable. Cela peut s'expliquer par le fait que pour ces forts débits la qualité de restitution demandée est tellement importante que chaque trame prédite ne peut plus tirer pleinement profit de la redondance temporelle.

On notera cependant que dans le cas du mélange de trames scalable, l'efficacité de compression est plus faible pour les premiers niveaux de décomposition, situés plus loin temporellement de leurs références que dans un schéma de codage classique. Nous donnons dans l'annexe B des éléments permettant de comprendre théoriquement ces résultats expérimentaux.

Comportement des solutions scalables dans un milieu bruité

Les configurations scalables possèdent pour chaque niveau de raffinement temporel, des niveaux de décompositions distincts, grâce à la contrainte imposant que chaque trame codée ne peut prendre comme référence que les trames codées d'une résolution inférieure. Dans la configuration en "arbre", on restreint de plus les trames codées à ne pas se référer à des trames de même niveau de résolution.

Cette contrainte supplémentaire permet de considérer que toutes les trames d'un même niveau de raffinement ont la même importance moyenne. Comme la configuration en

"miroir", la configuration "arbre" n'utilise donc pas toutes les trames précédemment codées comme référence. Ceci a aussi pour avantage d'éviter une trop grande propagation de l'erreur dans le cas d'une transmission avec erreurs, et de limiter la taille de la mémoire tampon DPB. Dans le cas "zigzag", la propagation pourra être plus importante, puisque chaque trame peut utiliser comme référence toutes les trames précédentes de niveaux d'importance égal ou supérieur.

Pour montrer les effets de propagation des erreurs, nous avons donc simulé le comportement d'une séquence codée par les types de configurations "normal", "zigzag" et "arbre" face à la perte d'une trame dans chaque GOP. Nous avons considéré un GOP de taille 15, pour mieux expliciter l'effet de la propagation d'erreurs au sein du GOP, correspondant à des configurations "zigzag" et "arbre" illustrées respectivement dans les figures 3.21 et 3.22.

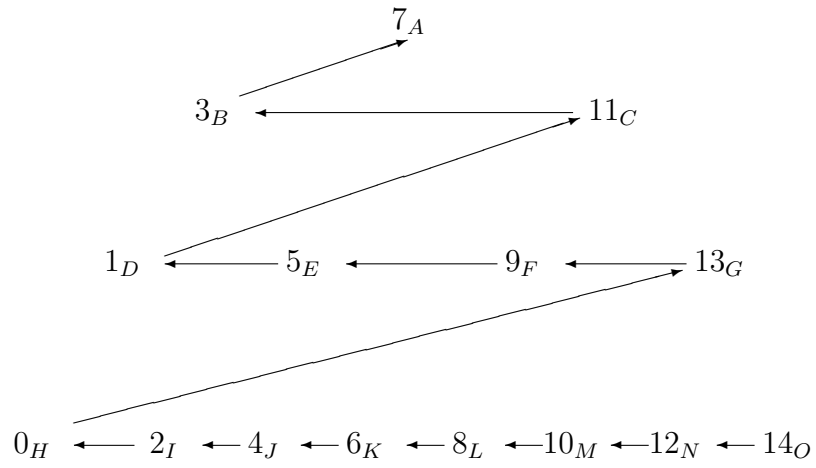

Figure 3.21 – Configuration en "zig-zag", GOP de taille 15

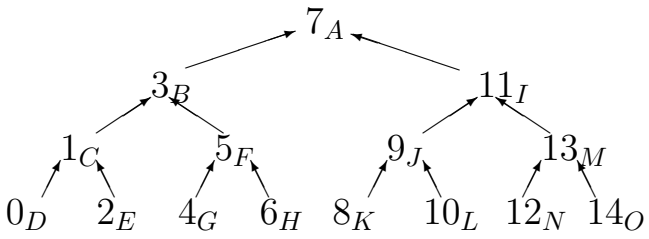

Figure 3.22 – Configuration en "arbre", GOP de taille 15

Les figures 3.23, 3.24 et 3.25 présentent donc l'évolution du PSNR de la séquence 'Foreman' en QCIF, à 15 trames/s, à 64 kbits/s, avec un GOP de taille 15. Dans chaque simulation, la 6^{eme} trame codée de chaque GOP (c'est-à-dire la trame notée F) est perdue, et est donc remplacée par le décodeur vidéo robuste par la trame décodée la plus proche temporellement afin de permettre aux trames suivantes de ne pas se référer à une trame inexistante.

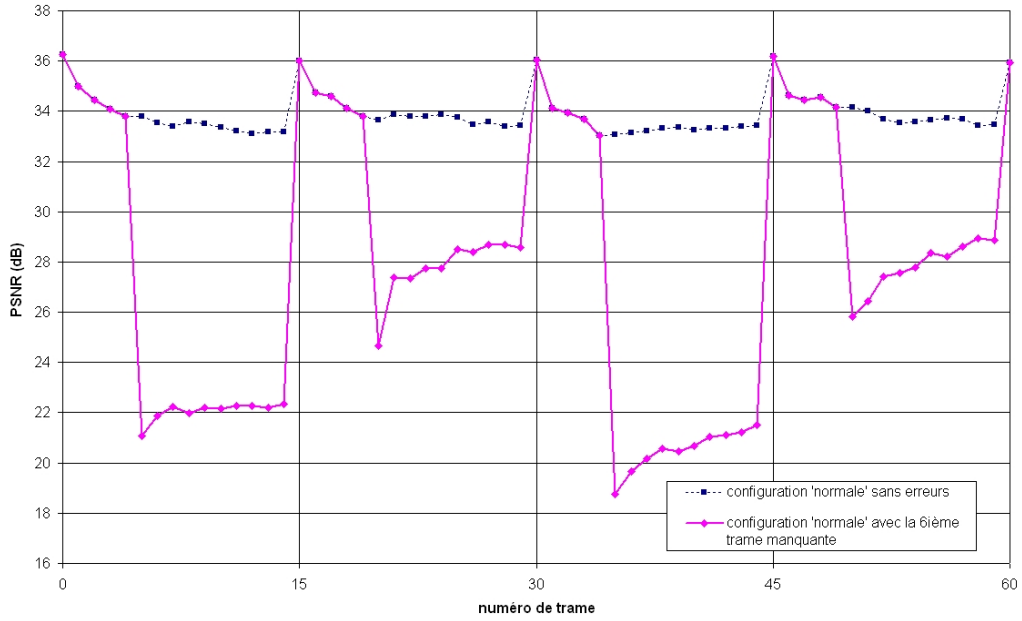

Figure 3.23 – Évolution du PSNR avec et sans perte de la 6ème trame codée de chaque GOP, pour la séquence 'Foreman' codée avec la configuration "normale" (QCIF, 15trames/s, 64 kbits/s)

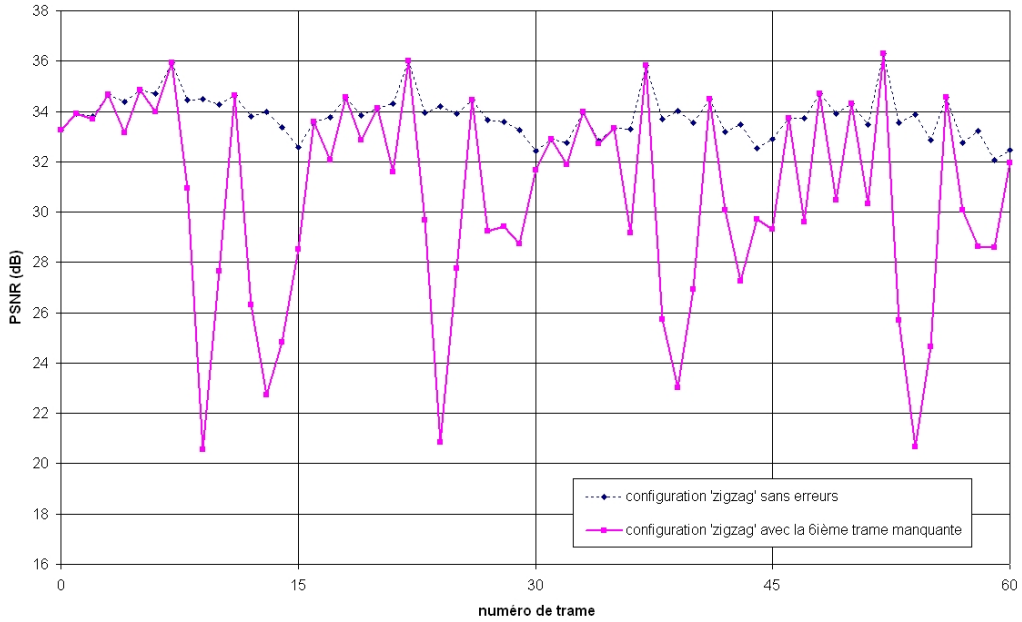

Figure 3.24 – Évolution du PSNR avec et sans perte de la 6ème trame codée de chaque GOP, pour la séquence 'Foreman' codée avec la configuration "zigzag" (QCIF, 15 trames/s, 64 kbits/s)

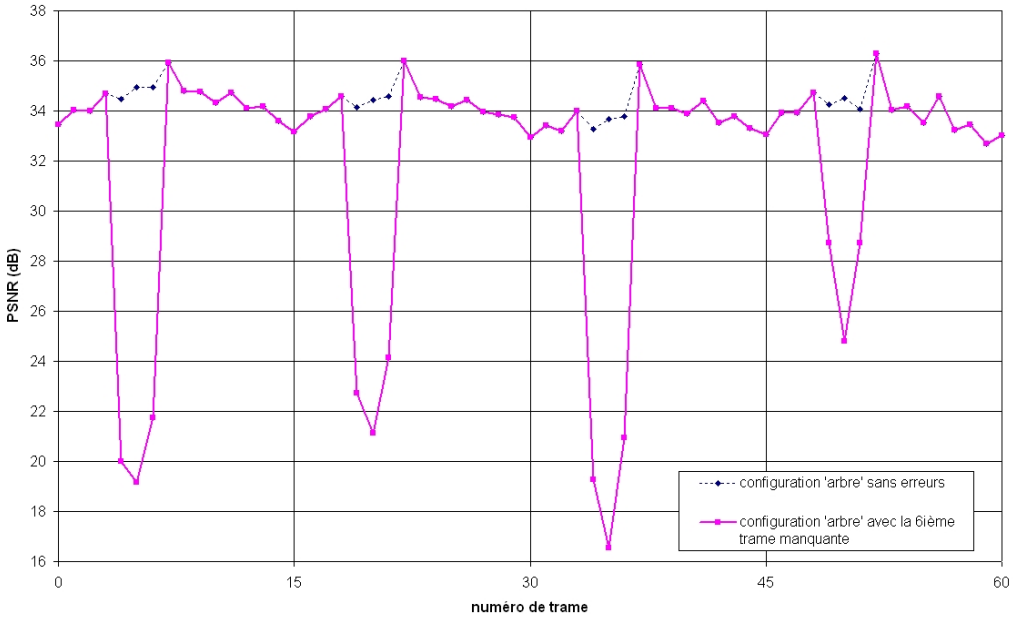

Figure 3.25 – Évolution du PSNR avec et sans perte de la 6ème trame codée de chaque GOP, pour la séquence 'Foreman' codée avec la configuration "arbre" (QCIF, 15 trames/s, 64 kbits/s)

Chaque configuration se comporte donc différemment en fonction du même type d'erreur.

- Dans la configuration "normale", ce sont toutes les trames de 6 à 15 qui sont mal reconstituées.
- Dans la configuration "zigzag", seules les trames 1, 3, 5, 7, et 11 ne sont pas corrompues.
- Dans la configuration "arbre", seules les trames numérotées 4, 5, et 6 sont affectées.

Les résultats visuels correspondants, obtenus pour un GOP entier, sont représentés en figure 3.26 pour la configuration "normale", en figure 3.27 pour celle en "zigzag", et en figure 3.28 pour celle en "arbre".

Cet exemple montre bien les avantages de la configuration en "arbre" puisque de par sa construction, elle permet d'éviter les propagations d'erreurs à l'intérieur du GOP, tout en apportant un gain de compression non négligeable par rapport à la méthode classique (comme montré au paragraphe 3.3).

Un point important à souligner, qui est fort important pour la réalisation pratique, est le problème de la gestion et de la capacité de la mémoire tampon (DPB). Ainsi, dans la configuration "zigzag" nous avons supposé que la capacité de la mémoire tampon gérant les trames de références était à son maximum (i.e. 16 trames de référence). Or, dans le paragraphe 1.2.4, on a vu que tous les niveaux dans les profils ne permettent pas d'allouer

Figure 3.26 – Résultats visuels sur le premier GOP de la séquence 'Foreman' pour la configuration "normale", lorsque la 6ème trame codée est absente (PSNR moyen = 26.28 dB)

Figure 3.27 – Résultats visuels sur le premier GOP de la séquence 'Foreman' pour la configuration "zigzag", lorsque la 6ème trame codée est absente (PSNR moyen = 30.74 dB)

Figure 3.28 – Résultats visuels sur le premier GOP de la séquence 'Foreman' pour la configuration "arbre", lorsque la 6ème trame codée est absente (PSNR moyen = 31.57 dB)

16 trames de référence dans son algorithme de prédiction. Il est clair que la méthode "zigzag" n'est donc pas implémentable si l'on considère des tailles de GOP importants.

En revanche, la configuration en "arbre" permet, quant à elle, de limiter l_{max}, le nombre de trames de référence, en fonction de la taille du GOP N_{GOP}, par la formule (3.1) ci-dessous :

$$\log_2(N_{GOP}+1) \leq l_{max} < \log_2(N_{GOP}+1)+1 \tag{3.1}$$

ce qui rend beaucoup plus facile à implémenter cette configuration.

Décompositions asymétriques

Dans les exemples précédents, nous avons considéré que les GOP sont "fermés", c'est-à-dire qu'ils sont codés indépendamment. Nous supposions donc que la trame Intra était une IDR, et donc que toutes les trames codées avant celle-ci étaient effacées de la mémoire tampon contenant toutes les trames de référence. Dans cette partie, nous nous proposons d'aborder la possibilité de GOPs "ouverts", où les trames Intra (hormis la première) ne seront pas codées en IDR. Chaque GOP contiendra alors 2 trames Intra, celle du GOP courant, et celle du GOP précédent. Cette décomposition asymétrique permet alors d'utiliser pleinement les trames Intra placées aux extrémités du GOP.

Pour illustrer ce type de décomposition, nous avons pris deux exemples : la configuration “dyadique”, et “dyadique limitée”. Elles sont respectivement représentées par les figures 3.29 et 3.30.

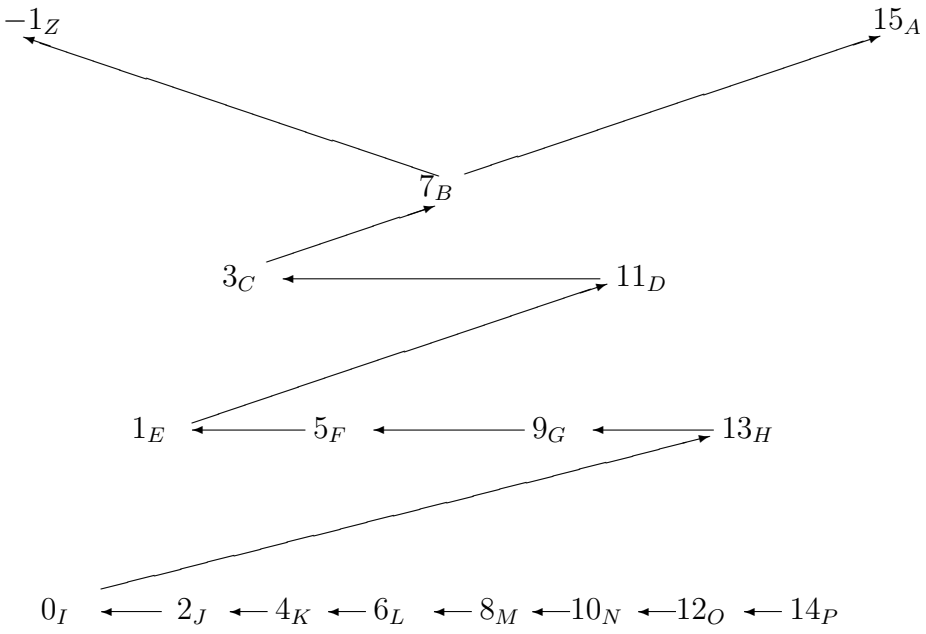

Figure 3.29 – Configuration “dyadique”, GOP ouvert de taille 16

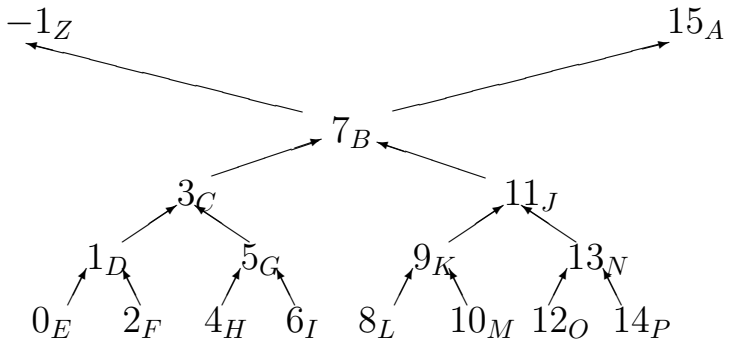

Figure 3.30 – Configuration “dyadique limité”, GOP ouvert de taille 16

La trame représentée par le numéro -1_Z correspond à la trame Intra codée dans le précédent GOP. Ces deux configurations reprennent respectivement les mêmes caractéristiques que “zigzag” et “arbre” dans leur gestion de trames de références. La décomposition “dyadique” peut donc se référer à des trames de niveaux de raffinements égales ou inférieures. Alors que la décomposition “dyadique limitée”, ne peut se référer qu’à des trames de résolutions strictement inférieures.

Ces solutions proposent donc d'obtenir la scalabilité temporelle avec un facteur d'échelle égal à deux, pour cinq niveaux de détails. Ce sont des modèles réguliers adaptés à des GOPs de taille $N_{GOP} = 2^L$. Cette décomposition peut aussi se généraliser à d'autres facteurs d'échelle et à d'autres tailles de GOP. À titre d'exemple, nous donnons dans les figures 3.31 et 3.32 une décomposition de ce modèle asymétrique pour une taille de GOP $N = 15$.

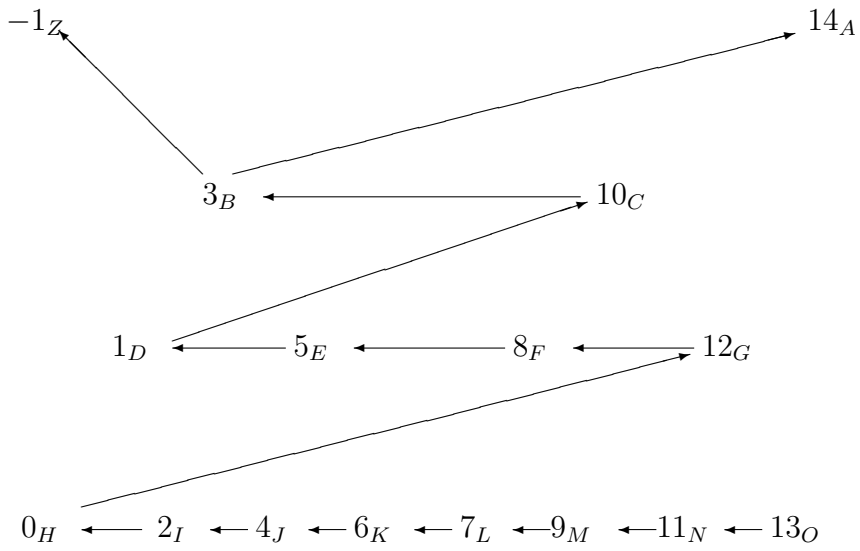

Figure 3.31 – Configuration généralisée "dyadique", GOP ouvert de taille 15

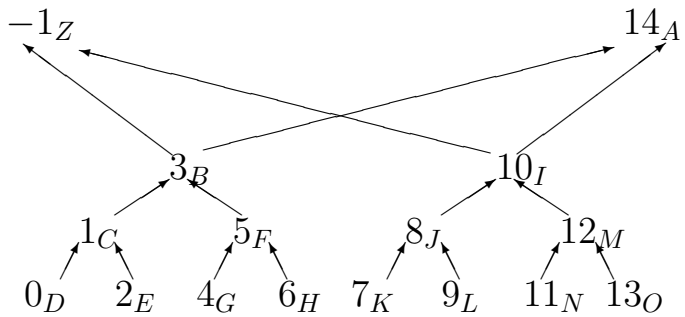

Figure 3.32 – Configuration généralisée "dyadique limité", GOP ouvert de taille 15

Nous avons donc testé ces deux nouvelles configurations scalables "dyadique" et "dyadique limité" en les comparant aux moyennes de PSNR à même débit avec les configuration "normale" pour les séquences de référence 'Foreman', 'Akiyo' et 'Mobile' en QCIF. Ces résultats sont donnés dans la table 3.4.

Séquence	Débit	Configuration	Moyenne de PSNR
Akiyo	64 kbit/s	normale	43.09 dB
Akiyo	64 kbit/s	dyadique	43.59 dB
Akiyo	64 kbit/s	dyadique limité	43.85 dB
Foreman	64 kbit/s	normale	33.35 dB
Foreman	64 kbit/s	dyadique	33.58 dB
Foreman	64 kbit/s	dyadique limité	33.79 dB
Mobile	128 kbit/s	normale	29.66 dB
Mobile	128 kbit/s	dyadique	30.21 dB
Mobile	128 kbit/s	dyadique limité	30.52 dB

Table 3.4 – Comparaison de rapport débit-distorsion pour différentes configurations avec une taille de GOP de 16 trames

On peut observer alors un gain d'environ 0.20 dB à 0.80 dB en fonction de la configuration et du type de séquence. Nous n'avons d'ailleurs pas présenté ici les simulations d'une transmission avec erreurs, puisque nous avions remarqué que les configurations "dyadique" et "dyadique limitée" ont un comportement similaire avec respectivement les configurations "zigzag" et "arbre". La différence principale qui existe entre les décompositions symétriques et asymétriques réside dans la dépendance des trames Intra.

Dans ces deux configurations asymétriques, la trame Intra est devenue encore plus importante, car une erreur sur celle-ci peut se propager sur les deux GOPs consécutifs. Inversement, le fait de pouvoir s'appuyer sur deux Intras offre l'opportunité de faire du masquage partiel et d'éviter des GOP perdus. En effet, si l'on considère une transmission bruitée, dans une configuration asymétrique, si une trame Intra est bruitée, alors ce sont toutes les trames codées sur 2 GOPs consécutifs qui sont susceptibles d'être corrompues par l'erreur. Dans le même cas de figure avec une configuration symétrique, la perte d'une Intra n'interfère qu'à l'intérieur du GOP "fermé". De plus, les méthodes asymétriques ont le défaut de devoir mettre en mémoire tout le GOP pour pouvoir commencer l'algorithme de compression.

La figure 3.33 présente donc les évolutions différentes du PSNR dans un milieu bruité de la séquence 'Foreman' en QCIF, à 15 trames/s, à 64 kbits/s, codée avec les configurations "dyadique limité" et "arbre". Dans cette simulation, nous avons considéré qu'une fois sur deux la trame Intra est perdue. Dans la configuration asymétrique, si une trame Intra est bruitée, alors ce sont toutes les trames codées sur 2 GOPs consécutifs qui sont susceptibles d'être corrompues suite aux erreurs présentes dans la trame Intra. Dans le même cas de figure avec une configuration symétrique, la perte d'une Intra n'interfère qu'à l'intérieur du GOP "fermé".

De plus, les méthodes asymétriques ont le défaut de devoir mettre en mémoire tout le GOP pour pouvoir commencer l'algorithme de compression. En revanche, ces dernières offrent une réelle meilleure compression pour des tailles de GOP équivalentes.

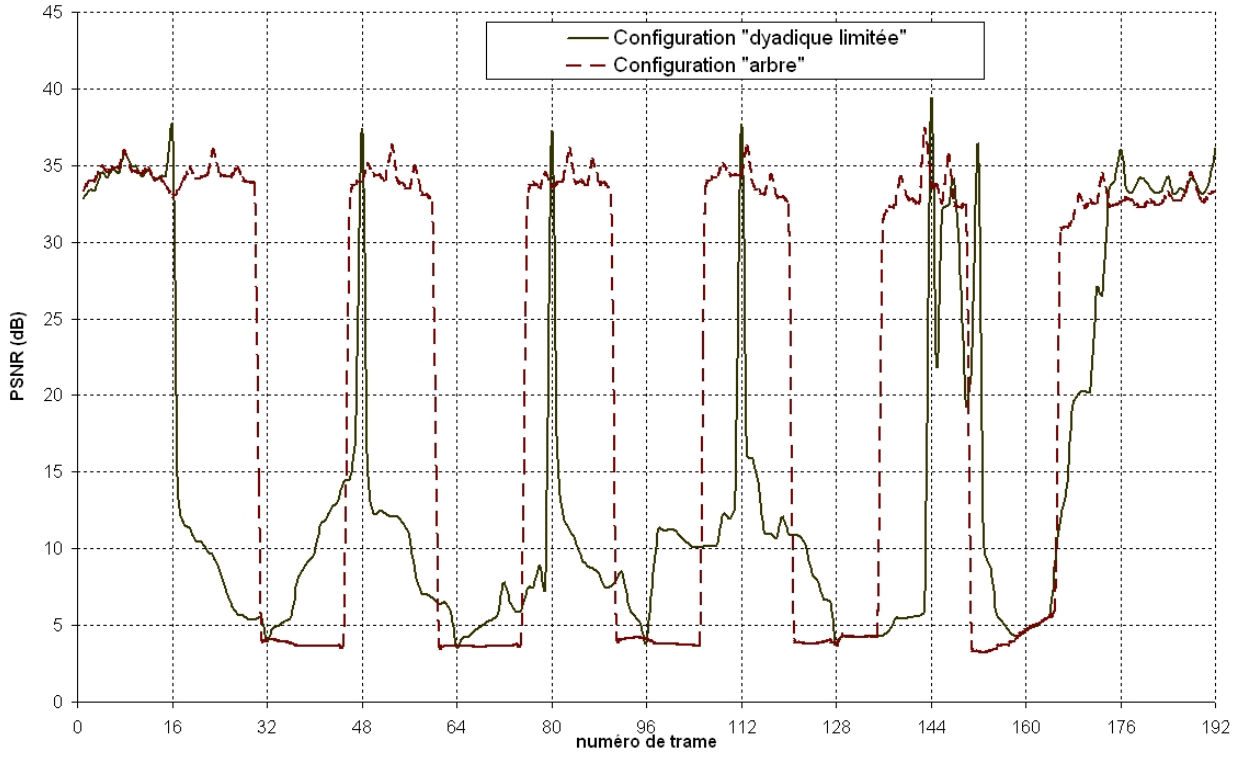

Figure 3.33 – Évolution du PSNR avec la perte régulière de la trame codée en Intra tous les deux GOPs, pour la séquence 'Foreman' codée avec la configuration "arbre" et "dyadique limité" (QCIF, 15 trames/s, 64 kbits/s)

3.4 Conclusions

Nous avons présenté une méthode qui permet d'introduire une scalabilité temporelle et qui sort des schémas classiques utilisés dans les codeurs vidéos prédictifs. Cette nouvelle approche offre un degré de liberté supplémentaire dans la compression et permet aussi d'augmenter la maîtrise de la propagation des erreurs de transmission.

La compatibilité avec la norme H.264/AVC permet de rendre vraiment intéressante cette solution, autant dans le domaine des transmissions sans fil, de part la limitation de la propagation d'erreurs, notamment avec le mode "arbre", que dans celles filaires, de part l'amélioration du niveau intrinsèquement bon de compression. De plus, notre méthode n'utilise pas obligatoirement des trames de type B, comme [50], [51] et [63], ce qui lui permet d'avoir pour avantage d'être compatible avec n'importe quel profil du standard H.264.

Enfin, nos simulations montrent bien que la scalabilité temporelle s'obtient sans perte en efficacité de compression tout en offrant les propriétés de robustesses inhérentes à la granularité temporelle. Néanmoins, il faut reconnaître un défaut au processus en cas de

communications limitées par les délais de codage et de transmission. La mise en mémoire des trames amène un délai de un demi à un GOP, en général, négligeable en diffusion, éventuellement plus gênant en modes conversationnel ou interactif.

Chapitre 4

Optimisation conjointe *

Sans maîtrise, la puissance n'est rien
Pirelli, *concurrent d'un bibendum*

Dans les transmissions de données multimédia, certaines solutions proposent de regrouper dans un seul module le codage source (compression) et le codage canal (protection) pour s'adapter aux caractéristiques des canaux bruités. Cela semble en effet l'approche la plus logique dans les conditions où le théorème de séparation ne s'applique pas : réaliser les opérations de compression et d'adaptation au canal conjointement en une unique opération. Mais si l'on veut rester compatible avec des standards existants et déployable sur des architectures existantes où des couches réseaux sont susceptibles d'exister entre le codeur source et canal, alors l'intégration de toutes ces solutions à un unique codeur conjoint n'est pas envisageable ici. Nous conserverons donc la séparation entre les deux codeurs, tout en organisant une interface entre eux, leur permettant de "coopérer" ensemble, afin de prendre en compte l'impact des erreurs, inévitables sur un canal sans fil, pour combiner efficacement le processus de compression et de protection, et pour optimiser ainsi la qualité du rendu visuel en fonction des caractéristiques propres du canal [58] [27].

Le principal défaut des algorithmes d'allocation débit-distorsion des codeurs sources, est leur non pris en compte générale des erreurs de transmission. Ces algorithmes font parfois la supposition irréaliste que toutes les données issues des méthodes de corrections d'erreurs ou FEC (*Forward Error Correction*) sont sans erreurs, souvent pour la bonne raison que ces codeurs visent également des applications filaires ou de stockage, où les erreurs sont quasi inexistantes. Ils négligent les effets potentiels de distorsion importante dus aux erreurs bits résiduelles dans les transmissions sans fil ou de télédiffusion (*broadcast*).

Ce chapitre propose donc une méthode permettant d'optimiser la qualité de reconstruction de la source dans les conditions d'un canal bruité ou avec pertes. Dans une première partie, nous définirons le critère de sensibilité et son calcul en fonction des types de trames codées.

∗. Certaines parties de ce chapitre ont été publiées dans les conférences *ICASSP'06* [68] et *MMSP'06* [69].

Ensuite, nous définirons une méthode d'allocation débit-distorsion prenant en compte les caractéristiques du canal. Enfin nous montrerons les résultats de simulations obtenus avec cette méthode sur un canal AWGN.

4.1 La sensibilité de trames vidéo en milieu imparfait

Dans la littérature, il existe diverses définitions de la sensibilité adaptées aux codeurs vidéo par blocs [33], [25] et [53]. Nous proposons ici une méthode semi-analytique simple permettant d'évaluer la distorsion de la vidéo reconstruite en fonction des différents types de trames ou de partitions. Cette méthode repose sur une estimation de la sensibilité aux erreurs bits et de la distorsion résultante de chaque trame dans tout le GOP.

4.1.1 Formulation théorique

Nous cherchons ici à estimer, pour une séquence vidéo, la moyenne de la distorsion totale de bout en bout $\hat{D}_{S+C}$ après codage source, codage canal et transmission sur un canal. Pour plus de simplicité, nous avons considéré qu'une trame était soit codée en une seule *Slice* (ou une seule NAL du standard H.264/AVC), soit codée avec le mode *Data partitioning* présenté dans la partie 1.2.3. La distorsion espérée $\hat{D}_{S+C}$ d'une trame transmise sur un canal bruité, peut être calculée en prenant en compte toutes les différentes distorsions D_i possibles associées à une probabilité d'apparition P_i ;

$$\hat{D}_{S+C} = \sum_{i \in \mathbb{N}} D_i.P_i$$

Théoriquement, chaque erreur bit possible, et leurs différentes combinaisons, engendrent un impact différent sur la reconstruction de l'image décodée (avec ou sans masquage d'erreur). Ne pouvant modéliser et calculer cette distorsion de manière exhaustive, on propose de grouper et de moyenner les différents cas d'erreurs de transmission en 3 cas, associés à une valeur de distorsion type :

- la trame est totalement corrompue ou perdue (D_{loss}),
- la trame est partiellement corrompue (D_{corr}),
- la trame est correctement transmise (D_o).

Pour une probabilité de recevoir correctement une NAL P_c (respectivement de la perdre complètement P_l), la distorsion conjointe source canal, ou sensibilité, peut s'écrire de la manière suivante :

$$\hat{D}_{S+C} = P_c.D_o + P_l.D_{loss} + (1 - P_c - P_l).D_{corr} \tag{4.1}$$

Cette distorsion peut alors s'écrire soit en moyenne de MSE (1.1), soit en moyenne de PSNR (1.2), dont on rappelle les formules données en 1.1.2 :

$$\widehat{MSE} = \sum_{i=1}^{M}\sum_{j=1}^{Q}\frac{(pl^*(i,j) - pl(i,j))^2}{M \times Q}$$

$$\widehat{PSNR} = 10\,log_{10}\Big(\frac{255^2}{\widehat{MSE}}\Big)$$

avec M, Q la hauteur et la largeur de l'image, et $pl(i,j)$,(respectivement pl^*) la valeur de la luminance du pixel original (respectivement reconstruit).

Si on exprime maintenant les probabilités en fonction du canal, en considérant un canal sans mémoire comme par exemple, un canal à bruit additif Gaussien ou binaire symétrique (BSC pour *Binary Symmetric Channel*), on obtient pour une probabilité d'apparition d'erreur P_e, la probabilité de recevoir correctement une NAL :

$$P_c = (1 - P_e)^n$$

où n est la taille de la trame en bits et $P_e = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_S}{N_0}}\right)$ pour un rapport signal-à-bruit $\frac{E_S}{N_0}$ dans le cas d'un canal AWGN sans codage canal.

Cependant nous avons observé dans le chapitre 2 que les erreurs bits n'engendraient pas forcément de désynchronisations du flux. Les trames obtenues avec quelques erreurs sont en effet interprétables par n'importe quel décodeur H.264/AVC, mais les images reconstruites possèdent des artefacts visuels. Nous avons aussi montré dans le paragraphe 2.2 qu'une fraction $1-\beta$ des bits d'une trame n'engendre pas de désynchronisation au sein de la NAL, et donc que cette NAL n'est pas à considérer comme totalement perdue. En prenant en compte cette probabilité $1-\beta$ pour exprimer la probabilité de perte complète P_l, on obtient alors :

$$P_l = 1 - (1 - P_e)^{(\beta)n}.$$

L'expression de la sensibilité peut donc s'écrire de la manière suivante :

$$\begin{aligned}\hat{D}_{S+C} &= (1-P_e)^n D_o + \big(1 - (1-P_e)^{(\beta)n}\big) D_{loss} \\ &\quad + \big((1-P_e)^{(\beta)n} - (1-P_e)^n\big) D_{corr}\end{aligned} \tag{4.2}$$

4.1.2 Sensibilité des trames Intra et des groupes d'images

Les observations empiriques faites pour le codeur H.264/AVC montrent que $MSE_{corr} \simeq MSE_o$ pour des trames Intra et Prédites codées, et que l'estimation de la fraction de bits n'engendrant pas de désynchronisation est égale à $1-\beta_0 \simeq 0.25$ pour les Intras et à $1-\beta_i \simeq 0.15$ pour la $i^{ème}$ trame prédite. Ces valeurs sont naturellement à rapprocher des chiffres obtenus en 2.2.2 pour le chiffrement partiel.

Trame Intra

Déclinons la formule (4.2) dans le cas d'une trame Intra à partir des observations ci-dessus, on obtient :

$$\hat{D}_{Intra} = (1-P_e)^{\beta_0 n} D_o + \left(1-(1-P_e)^{\beta_0 n}\right) D_{loss} \tag{4.3}$$

Ainsi, la sensibilité d'une trame codée par H.264/AVC peut se calculer en connaissant la taille n de la trame, et en estimant seulement la distorsion obtenue dans le cas où la trame est correctement transmise et dans le cas où elle est perdue. La distorsion D_o étant connue au moment du codage, il suffit d'estimer la distorsion D_{loss} obtenue par un décodeur de référence en simulant la perte de cette trame. Dans toutes nos simulations, le décodeur de référence utilisé suppose , pour les trames intras perdues, que l'image reconstitué est complètement "noire", et, pour les trames prédites, que l'image est obtenue par duplication de celle précédemment décodée. Nous n'avons pas utilisé de décodeur de référence offrant des techniques de masquage d'erreur évoluée puisque dans nos expériences nous utilisions seulement une version "robustifiée" du logiciel de référence JM 10.3.

Comme le montre la figure 4.1 la sensibilité calculée en moyenne de MSE (sur 6000 tirages) de la première image de la séquence de référence 'Foreman' codée en Intra à différents pas de quantification (QP), et respectivement de 'Football' est très proche de notre expression analytique $\hat{D}_{Intra}$. Dans la figure 4.2, nous montrons que la sensibilité calculée en moyenne de MSE et en moyenne de PSNR, suivent toutes les deux nos formules théoriques.

Trame Prédite

De la même manière, l'expression de la sensibilité de la $i^{ème}$ trame prédite P_i d'un GOP, s'obtient de l'équation (4.2) comme suit :

$$\hat{D}_{P_i} = (1-P_e)^{\beta_i n_i} D_{o_i} + \left(1-(1-P_e)^{\beta_i . n_i}\right) D_{loss_i} \tag{4.4}$$

avec $\beta_i = 0.85$, n_i la taille de la $i^{ème}$ trame P , et D_{o_i} (respectivement D_{loss_i}) la distorsion observée lorsque la trame décodée est correcte (respectivement perdue), et que les trames précédentes sont toutes correctement décodées.

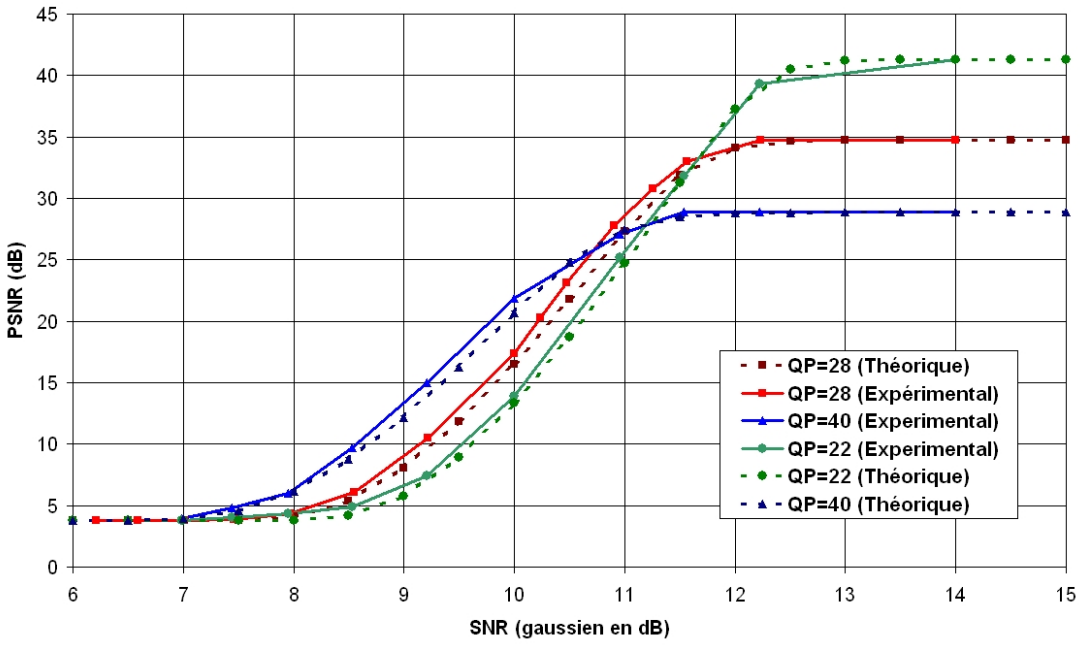

Figure 4.1 – Sensibilité en MSE de la première trame codée en Intra de la séquence 'Foreman' à différent pas de quantification (QP).

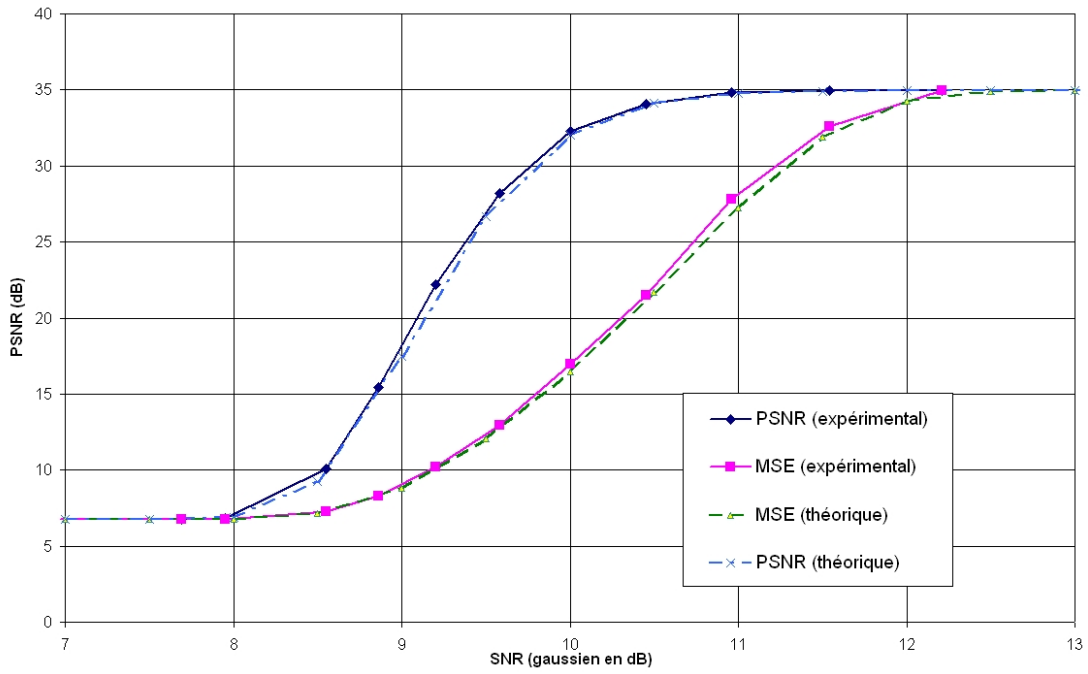

Figure 4.2 – Sensibilité en MSE et en PSNR de la première trame codée en Intra de la séquence 'Football'.

Groupe de trames Prédites

En pratique, les trames prédites et leurs sensibilités dépendent des trames précédentes, comme on le voit dans l'expression de l'équation (4.4). En effet, si on perd une trame P, même si les suivantes sont correctement transmises, elles ne seront pas correctement reconstruites. Par conséquent, nous considérerons que si nous perdons une trame, la contribution de distorsion des trames suivantes sera négligeable : l'impact des trames précédentes sera donc pris en compte en utilisant les probabilités conditionnelles des trames précédentes d'être correctes.

Soit ${P_c}^{(\beta_i)}$ la probabilité de recevoir correctement la $i^{ème}$ trame prédite, D_{loss_i} la distorsion moyenne sur tout le GOP observée lorsque seule la $i^{ème}$ trame prédite est erronée, et D_o la distorsion moyenne observée lorsque tout le GOP est correctement transmis. L'expression analytique devient :

$$\hat{D}_P = \prod_{i=1}^{N} {P_c}^{(\beta_i)} D_o + \sum_{i=1}^{N} \left[\prod_{j=0}^{i-1} {P_c}^{(\beta_j)} . (1 - {P_c}^{(\beta_i)}) D_{loss_i} \right]. \tag{4.5}$$

Si nous considérons le cas d'un canal erroné sans mémoire avec une probabilité d'événement d'erreur P_e, la probabilité d'obtenir une trame correcte est donnée par l'expression suivante,

$${P_c}^{(\beta_i)} = (1 - P_e)^{\beta_i . n_i}$$

La sensibilité des trames P devient alors :

$$\hat{D}_P = \prod_{i=1}^{N} (1 - P_e)^{\beta_i n_i} D_o + \sum_{i=1}^{N} \big[\prod_{j=1}^{i-1} (1 - P_e)^{\beta_j n_j} (1 - (1 - P_e)^{\beta_i n_i}) D_{loss_i} \big], \tag{4.6}$$

avec n_i la taille en bits de la $i^{ème}$ trame prédite.

La figure 4.3 illustre la grande proximité entre le modèle et les résultats expérimentaux pour une estimation de la sensibilité globale des 14 trames P du premier GOP de la séquence 'Foreman' (format QCIF, avec $QP_I = 28$ et $QP_P = 30$), les résultats pratiques ayant été moyennés sur 6000 tirages.

Tout comme les trames Intra, la sensibilité des trames P peut donc simplement se déduire semi-analytiquement en estimant la taille de la trame et les distorsions obtenues dans le meilleur et le pire cas de transmission.

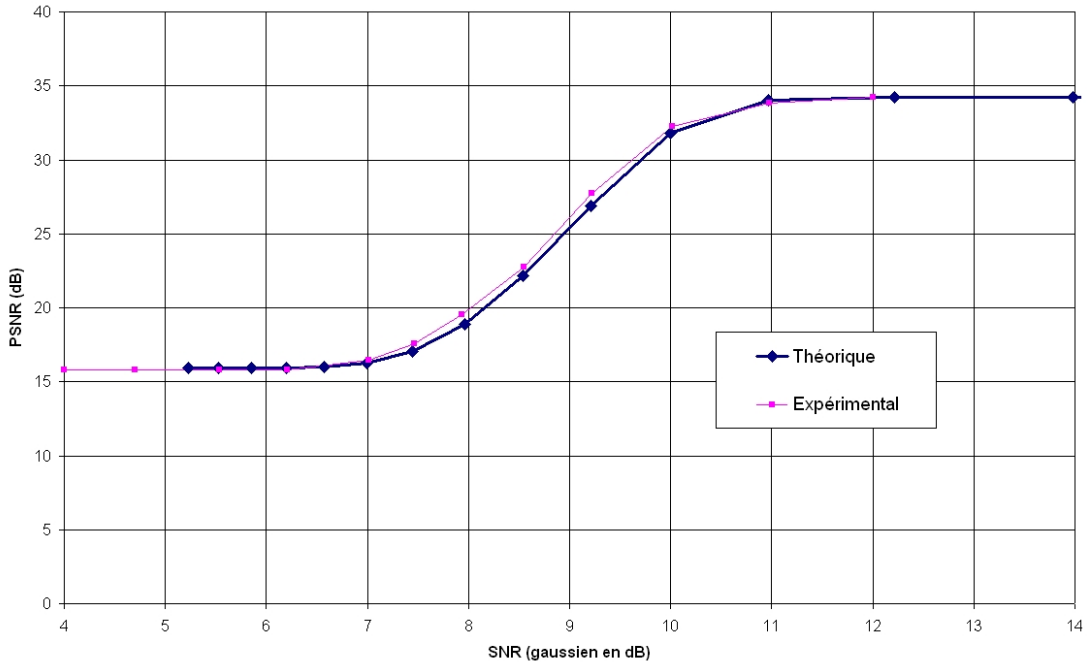

Figure 4.3 – Sensibilité globale en MSE des 14 trames P du premier GOP de la séquence 'Foreman'.

Sensibilité d'un groupe de trames

En généralisant l'équation (4.6) en rajoutant la sensibilité de la trame Intra, on obtient la formule de sensibilité pour un groupe de trames :

$$\hat{D}_{GOP} = \prod_{i=0}^{N}(1-P_e)^{\beta_i n_i} D_o + \sum_{i=0}^{N}\Big[\prod_{j=0}^{i-1}(1-P_e)^{\beta_j n_j}(1-(1-P_e)^{\beta_i n_i}) D_{loss_i}\Big] \quad (4.7)$$

avec D_{loss_0} la distorsion moyenne observée sur tout le GOP lors de la perte de la trame Intra.

Les résultats de simulations obtenus sur 6000 tirages dans un canal AWGN sont très proches des différentes formules analytiques présentées précédemment (comme le montrent les figures 4.4, 4.5 et 4.6). Il est à noter que la courbe de sensibilité de la trame Intra est quasiment confondue avec celle du GOP entier, ce qui s'explique par le fait qu'elle est la plus "sensible" aux erreurs de part sa taille et de part son impact dans le reste du GOP.

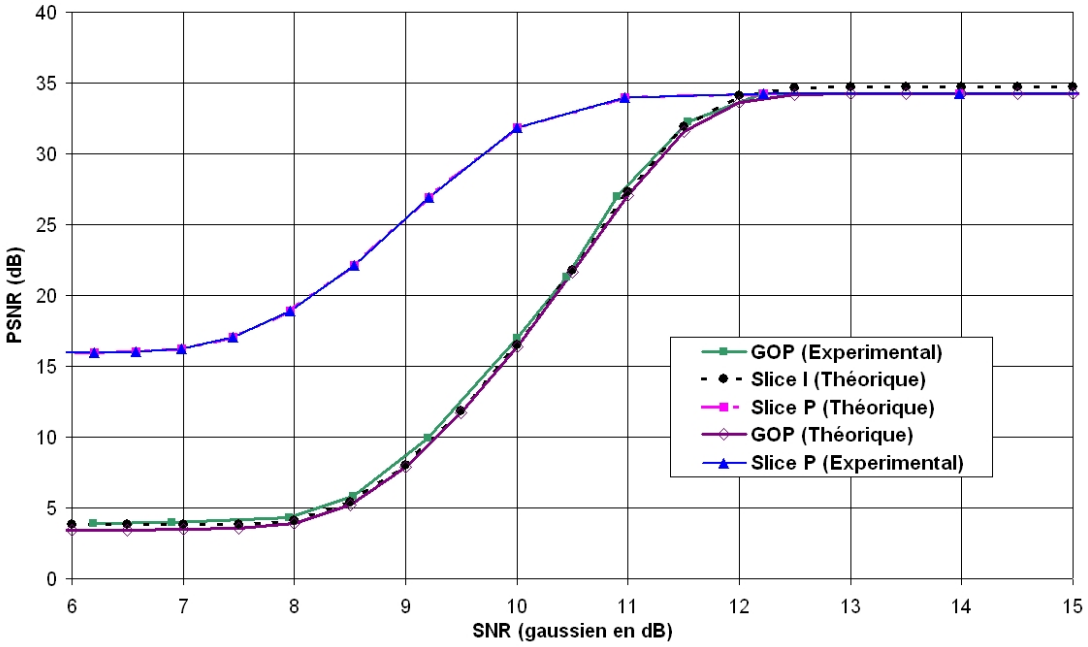

Figure 4.4 – Sensibilité en MSE du premier GOP (I_1P_{14}), des trames P, et de la trame I de la séquence 'Foreman' en QCIF avec $QP_I = 28$ et $QP_P = 30$

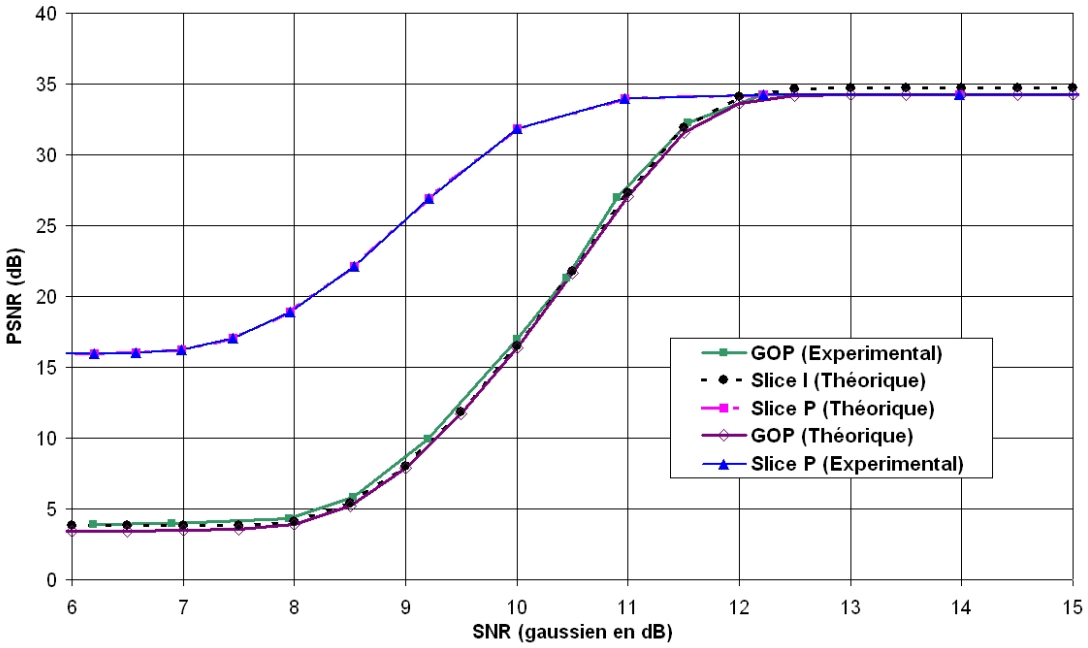

Figure 4.5 – Sensibilité en MSE du sixième GOP (I_1P_{14}), des trames P, et de la trame I de la séquence 'Foreman' en QCIF avec $QP_I = 31$ et $QP_P = 33$

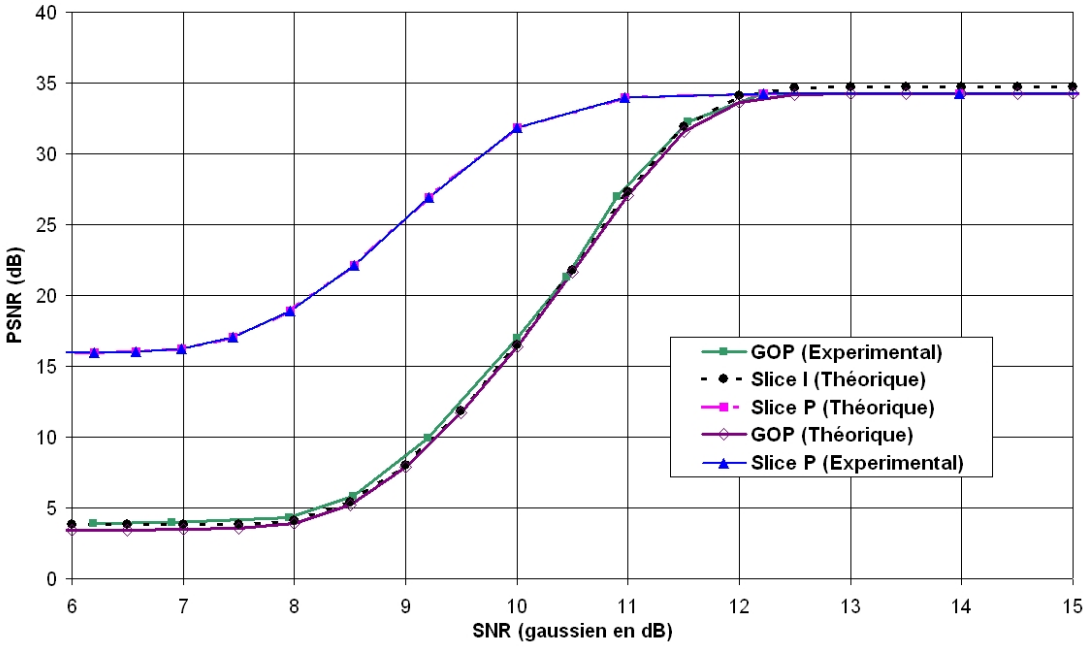

Figure 4.6 – Sensibilité en MSE du treizième GOP (I_1P_{14}), des trames P, et de la trame I de la séquence 'Foreman' en QCIF avec $QP_I = 35$ et $QP_P = 37$

Groupe de trames codées en "Data partition" (DP)

Le mode de partitionnement de données, prévu par la norme H.264 et présenté dans le paragraphe 1.2.3, réalise un découpage des trames Prédites en trois slices différentes et indépendantes :

- la NAL-A, qui est constituée l'ensemble des en-têtes de trames et des vecteurs de mouvements ;
- la NAL-B, qui contient les informations concernant les coefficients de la transformée entière des macroblocs codées en mode Intra ;
- la NAL-C, qui comprend les coefficients de la transformée entière résiduelle des macroblocs prédits.

De la même manière que dans les exemples précédents, nous considérons que la sensibilité intrinsèque d'une slice suppose que les trames ou slices précédentes dans l'ordre de codage soient toutes correctes.

La sensibilité d'un GOP codé en mode DP s'écrit donc, par généralisation de l'équation (4.7) de la manière suivante :

$$\hat{D}_{GOP_{DP}} = \prod_{i=0}^{N}\prod_{k=1}^{3}(1-P_e)^{\beta_{i,k}n_{i,k}}D_o + \sum_{i=0}^{N}\sum_{k=1}^{3}\left[\prod_{j=0}^{N}\prod_{\ell=1}^{k-1}(1-P_e)^{\beta_{j,\ell}n_{j,\ell}}\prod_{j=0}^{i-1}(1-P_e)^{\beta_{j,k}n_{j,k}}\left(1-(1-P_e)^{\beta_{i,k}n_{i,k}}\right)D_{loss_{i,k}}\right] \tag{4.8}$$

avec $D_{loss_{i,k}}$ la distorsion moyennée sur tout le GOP lors de la perte de la $k^{ième}$ slice de la $i^{ème}$ trame, et $n_{i,k}$ taille en bits de la slice.

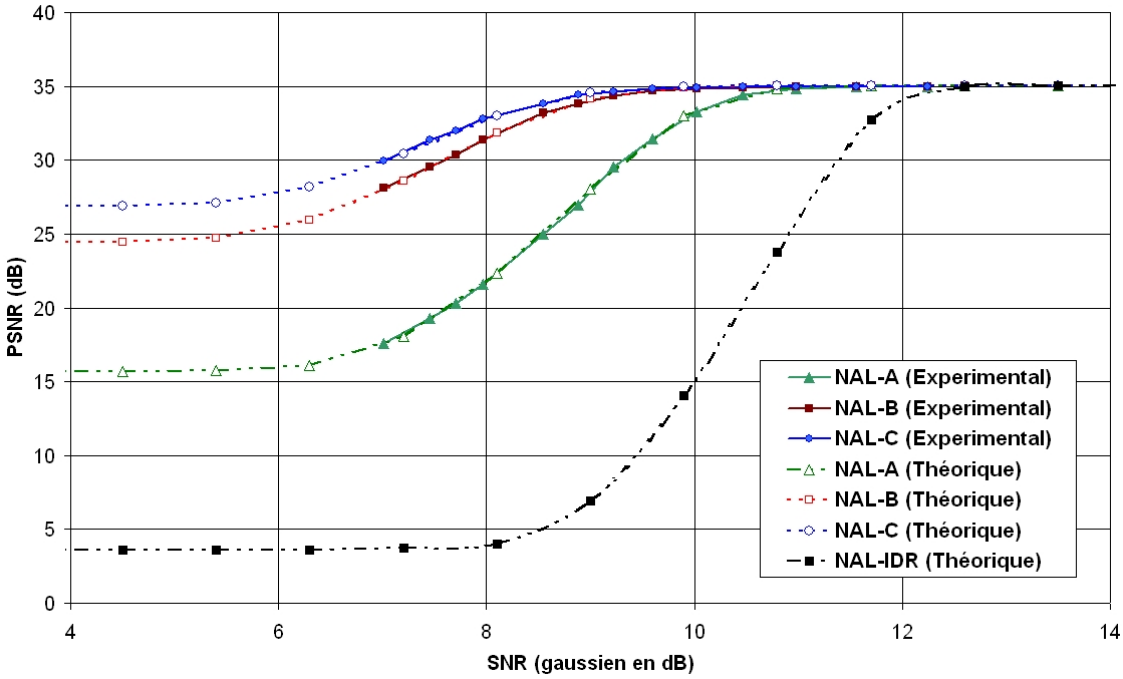

Figure 4.7 – Sensibilité en MSE du premier GOP (I_1P_{14}) codée en mode DP de la séquence 'Foreman' en QCIF avec $QP_I = 27$ et $QP_P = 30$

Les sensibilités, estimées sur 6000 tirages, des différentes slices du premier GOP de la séquence 'Foreman' codée en mode DP sont représentées dans la figure 4.7. Une fois de plus on constate la fiabilité du modèle par rapport aux simulations.

Étude de la sensibilité avec le "mélange de trames"

L'approche adoptée pour calculer la sensibilité des GOP dans une configuration classique ou "normale" peut aussi être utilisée pour calculer cette sensibilité en considérant d'autres

configurations du "mélange de trames". L'expression analytique donnée dans l'équation 4.7 suppose qu'une trame P a besoin pour pouvoir être correctement reconstruite, que toutes les trames précédentes soient correctement transmises . Or dans les diverses configurations, cette dépendance est différente pour chaque cas. Ainsi nous pouvons écire cette formule de la manière suivante :

$$\hat{D}_{GOP_{FS}} = \prod_{i=0}^{N}(1-P_e)^{\beta_i n_i} D_o + \sum_{i=0}^{N}\Big[\prod_{j\in FS_i}(1-P_e)^{\beta_j n_j}(1-(1-P_e)^{\beta_i n_i})D_{loss_i}\Big] \quad (4.9)$$

avec FS_i l'ensemble des trames sur lesquelles la $i^{ème}$ trame peut se référer.

Dans la partie 3.3.2, on a montré par le biais de la description en banc de filtres que la configuration "normale" avait autant de niveaux d'importance que de trames par GOP , alors que celle en "arbre" possède uniquement quatre niveaux d'importance pour des GOP de 15 trames. Ces niveaux d'importance vont apparaître dans les sensibilités des trames à travers le GOP entier, comme le montrent les figures 4.8 et 4.9.

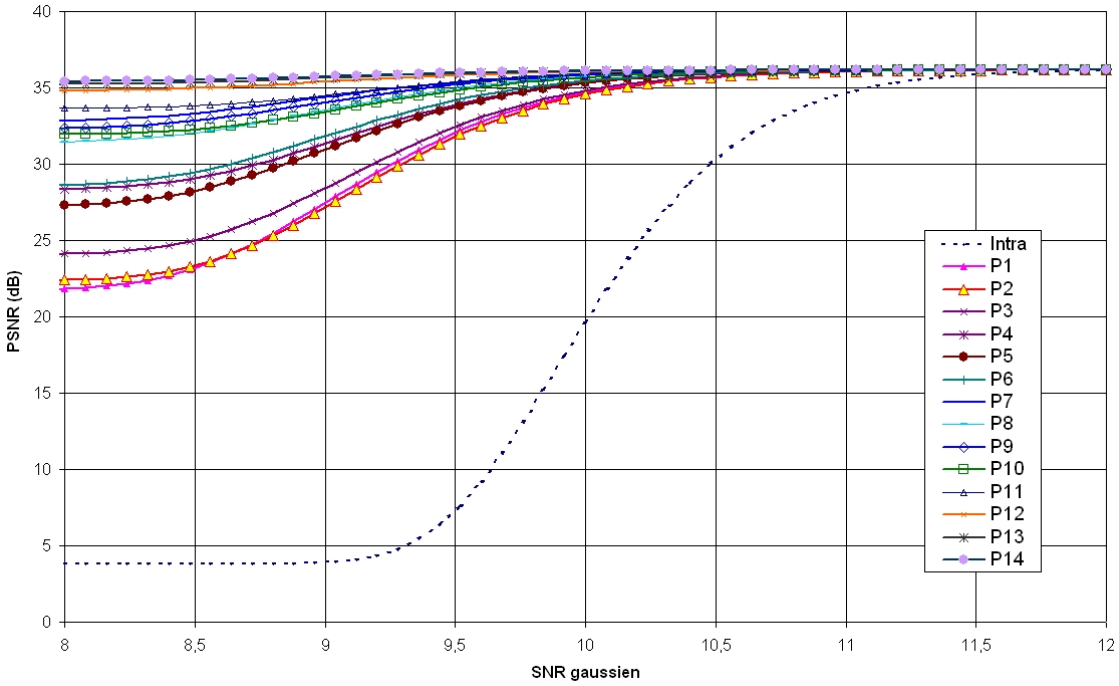

Figure 4.8 – Sensibilités globales théoriques en MSE de toute les trames du premier GOP (I_1P_{14}) codée en mode "normale" de la séquence 'Foreman' en QCIF avec $QP_I = 26$ et $QP_P = 28$.

Les différentes sensibilités montrent bien qu'outre le fait que la trame Intra soit la plus importante des trames, dans la configuration normale, chaque trame P a une sensibilité différente des autres : plus on "avance" temporellement dans le GOP et moins les trames P ont d'influence dans la sensibilité globale du GOP.

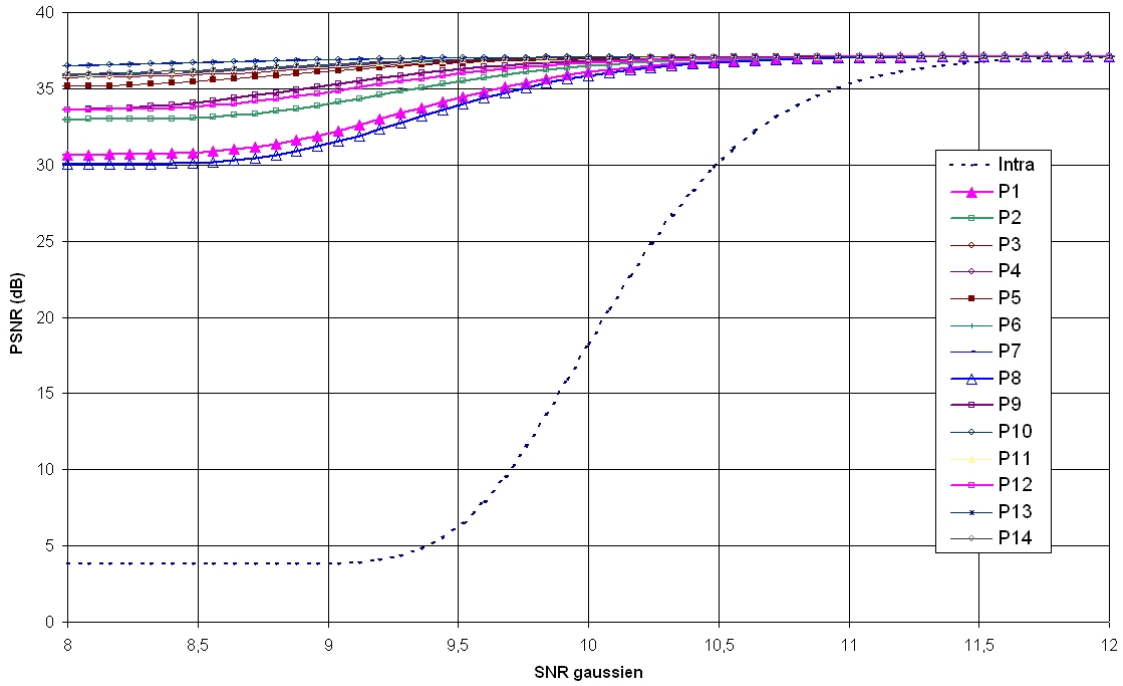

Figure 4.9 – Sensibilités globales théoriques en MSE de toute les trames du premier GOP (I_1P_{14}) codée en mode "arbre" de la séquence 'Foreman' en QCIF avec $QP_I = 25$ et $QP_P = 27$.

À l'opposé, la configuration scalable en "arbre" possède des sensibilités que l'on peut regrouper par niveau de raffinement temporel. Dans la figure 4.9, les trames P_1 et P_8 (correspondant dans le schéma 3.22 à 3_B et à 11_I) ont une sensibilité équivalente. De même, les trames P_2, P_5, P_9 et P_{12} (soit dans les figures 1_C, 5_F, 9_J et 13_M) ont la même influence dans le GOP, lors de leur perte. De cette manière, on vérifie bien que l'impact des trames erronées est moins important dans la configuration de type "arbre", et que cet impact dépend bien de la façon dont sont gérées les dépendances de prédictions des trames prédites.

Les résultats tout au long de ce chapitre ont montré que notre approche semi-analytique offre un modèle de prédiction très proche de la réalité, que l'on considère la mesure en MSE ou en PSNR.

4.2 Étude de sensibilité avec un codage canal

Ayant établi l'intérêt de transmettre les flux vidéos avec un taux d'erreur limité, mais aussi l'importance variable dans la reconstruction vidéo des différentes partitions, il est naturel de s'intéresser à la possibilité d'exprimer la sensibilité en présence de codage correcteur

d'erreur, apte à protéger, de manière égale ou non les différentes slices.

4.2.1 Protection d'erreurs à rendement variable : RCPC

Une manière d'appliquer facilement différents niveaux de protection à différentes parties de l'information est d'utiliser des codes à perforation variable, comme les RCPC (décrits au paragraphe 1.3). Ces codes, quasiment aussi performants que les meilleurs codes convolutifs de même rendement, permettent d'offrir plusieurs rendements en fonction des tables de perforation, tout en offrant une complexité d'implémentation restreinte, puisqu'ils ne nécessitent qu'un seul décodeur.

Ces codes offrent une probabilité d'erreur P_e sur un canal AWGN bornée par [24] :

$$P_e \leq \frac{1}{P} \sum_{d=d_{free}}^{\infty} a_d P_d \tag{4.10}$$

avec d_{free} la distance libre du code, a_d le nombre de trajets existants, et $P_d = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{d.E_S}{N_o}}\right)$ la probabilité que le mauvais trajet de distance d ait été sélectionné pour un rapport signal-à-bruit $SNR = E_S/N_0$ vu par le code convolutif.

On peut donc aisément calculer la distorsion d'une séquence codée lorsqu'elle est transmise sur un canal AWGN, et protégée par un code RCPC, en estimant sa probabilité P_e, et en la replaçant dans les expressions de sensibilité établies précédemment.

4.2.2 Protection égale aux erreurs (EEP)

Supposons tout d'abord que nous protégeons toutes les trames de la même façon, c'est-à-dire que nous appliquons une méthode de protection égale aux erreurs (EEP : *Equal Error Protection*). Si l'on considère différentes configurations de codage source et canal adaptées à une contrainte de débit global fixe, on peut comparer les différentes sensibilités obtenues en fonction du rapport signal-à-bruit (SNR). La figure 4.10 montre quatre configurations avec les courbes théoriques en pointillées et les résultats expérimentaux obtenus en moyennant sur 6000 réalisations en traits pleins. On constate à nouveau que l'approche théorique permet d'estimer assez précisément les résultats pratiques dans le cas d'une transmission avec un codage canal sur un canal AWGN.

Avec un tel outil, il est donc facile de déterminer, selon les caractéristiques du canal, le compromis compression/protection qui offrira la plus forte moyenne de PSNR. Ainsi, dans l'exemple de la figure 4.10, pour un rapport signal à bruit de 3 dB, la meilleure configuration est celle où le rendement du code RCPC est égal à 1/3, et où la séquence est codée à $21,3$ $kbit/s$.

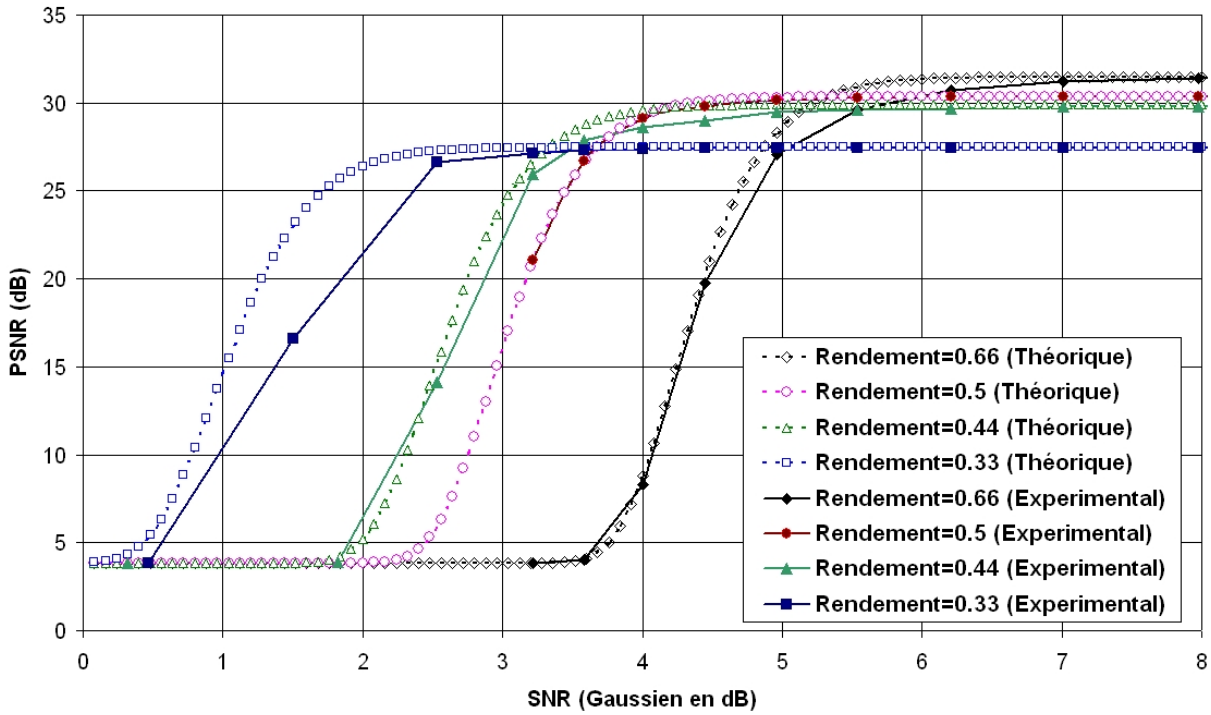

Figure 4.10 – Sensibilités en PSNR de différentes configurations de la séquence 'Foreman' (I_1P_{14}) adaptée à un débit total de 64 kbit/s.

4.2.3 Protection inégale aux erreurs (UEP)

Supposons à présent que l'on souhaite protéger différemment certaines parties du flux, et donc que nous nous plaçons dans un contexte de protection inégale aux erreurs ou "UEP" (*Unequal Error Protection*) [18]. Dans ce cas, on appliquera un niveau de protection différent à chaque partie définie dans le flux, en fonction de sa sensibilité. C'est dans ce type d'approche que l'intérêt d'un codage par codes perforés compatibles prend tout son sens, car le codage et le décodage restent uniques, avec une simple variation des tables de perforations selon les partitions. Chaque type de trame possède alors son propre niveau de protection (i.e. sa table de perforation pour le code RCPC), adapté à sa propre sensibilité.

Protection inégale aux erreurs en mode "IPPP"

Le premier cas de définition de partitions que l'on peut envisager est de séparer, dans une séquence de type "IPPP...", les Intras des Prédites. Dans ce cas, où seules deux classes apparaissent, on ne peut donc appliquer que deux types de rendements différents pour s'adapter aux contraintes de débit canal.

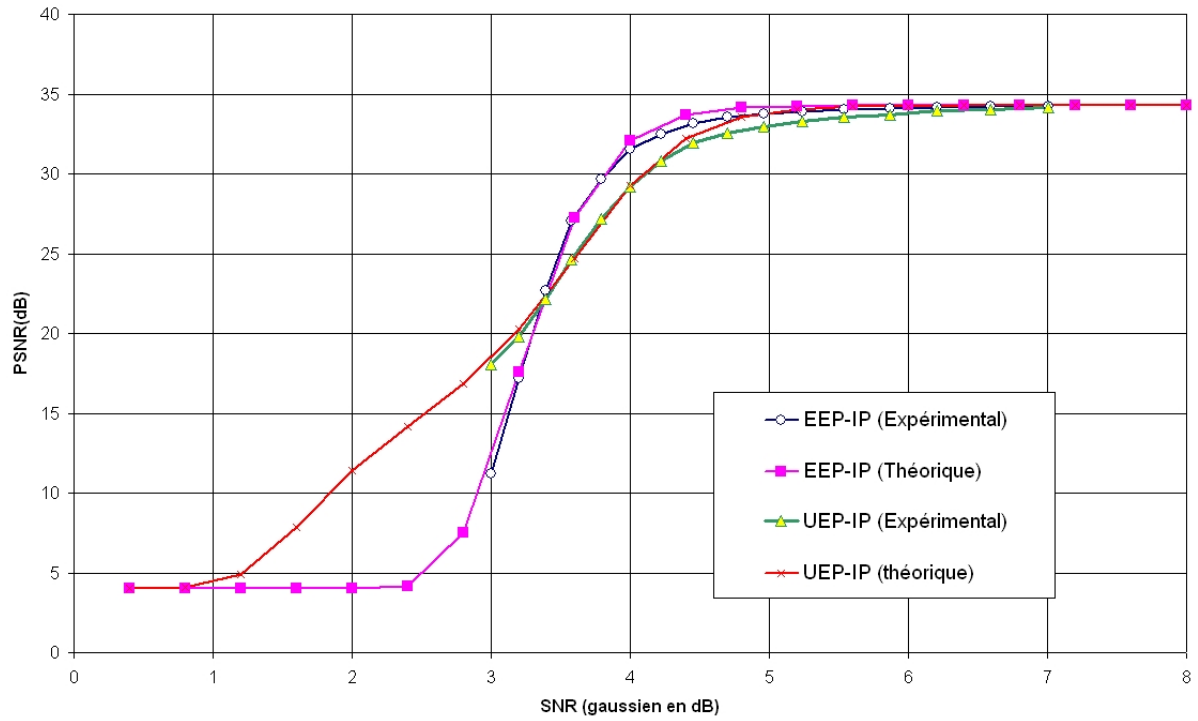

Figure 4.11 – Sensibilités en PSNR du premier GOP codé en mode normal (I_1P_{14}) de la séquence 'Foreman' EEP/UEP.

La figure 4.11 montre que la méthode UEP peut se révéler ne pas être efficace dans ce type de configuration. En effet, avec seulement deux types de trames, on ne peut avoir assez de "liberté" pour choisir en fonction des sensibilités le rendement le mieux adapté, et donc il devient souvent difficile, lorsque ce nombre de degrés de liberté est trop faible, de présenter un gain notable. Cependant, on note que ceci est moins vrai si dans un contexte à deux classes la première est plus sensible et fortement plus petite que la seconde, ce qui n'est pas le cas pour les Intras dans notre cas de figure.

Protection inégale aux erreurs en mode "Data Partitionning"

Pour augmenter le nombre de classes, mais également tirer parti des fonctionnalités du standard H.264/AVC, nous proposons de considérer le mode de partitionnement de données, dans lequel, comme nous l'avons expliqué au paragraphe 1.2.3, il existe quatre types de NAL : Intra, et NAL-A, -B, et -C. On peut donc définir 4 rendements différents associés aux quatre sensibilités, comme illustré sur la figure 4.7. En utilisant l'expression de sensibilité globale de l'équation (4.9), il est possible de choisir les meilleurs paramètres de rapport de poinçonnement des RCPC pour chaque type de slice, tout en satisfaisant les contraintes de débit total.

Par exemple, la figure 4.12 donne les résultats obtenus à la fois pour les séquences 'Foreman' (64 kbits/s, avec $R_{moyen} = 1/2$) et 'Stefan' (185 kbit/s, avec $R_{moyen} = 2/3$), avec le même rapport R_{moyen} pour les modes EEP et UEP. Pour les deux cas, le gain de la méthode UEP est supérieur à 5 dB en terme de PSNR par rapport au mode EEP.

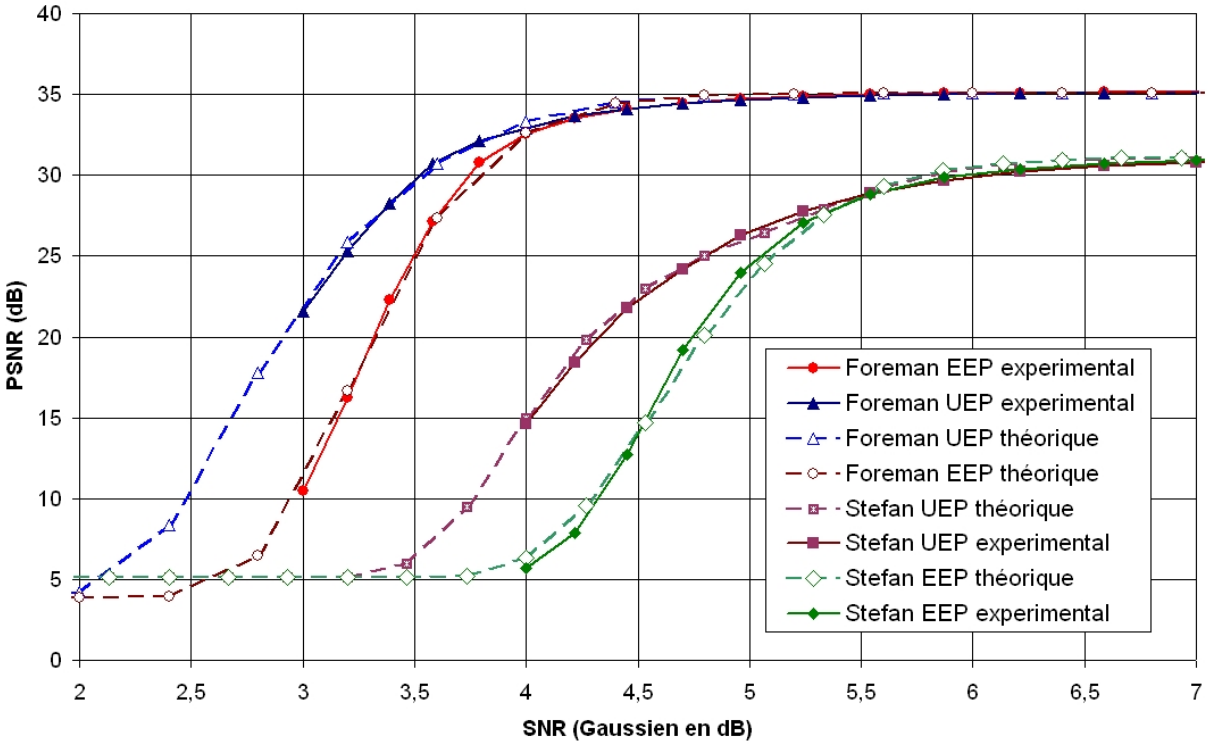

Figure 4.12 – Sensibilités en PSNR du premier GOP codé en DP (I_1P_{14}) des séquences 'Foreman' et 'Stefan' premier GOP en EEP/UEP.

Protection inégale aux erreurs avec le "mélange de trames"

Enfin, un autre mode à considérer pour la protection inégale aux erreurs est le mode de "mélange de trames", pour lequel différents niveaux de sensibilité ont été définis. Considérerons ici la configuration nommée "arbre" introduite au paragraphe 3.3.2, puisque cette configuration offre à la fois des propriétés de scalabilité et une bonne efficacité de compression. Cette solution scalable permet donc de définir des raffinements temporels possédant leur propre type de sensibilité comme illustré par la figure 3.22. Dans le cas où le GOP regroupe quinze trames, on définit alors quatre niveaux de raffinement et donc quatre types de sensibilité différentes, sur lesquelles nous pouvons appliquer des rendements de protection canal différents.

Si nous prenons le cas de la séquence de 'Foreman' codée en QCIF à 189 $kbit/s$ avec

la configuration en "arbre" ($QP_I = 25$, $QP_{P^{(1)}} = 26$, $QP_{P^{(2)}} = 27$ et $QP_{P^{(3)}} = 29$) par comparaison au mode "normal" ($QP_I = 26$ et $QP_P = 28$), alors pour un rendement moyen $R_{moyen} = 1/2$, on peut estimer la sensibilité du premier GOP dans un canal AWGN en mode UEP et EEP comme illustré par la figure 4.13.

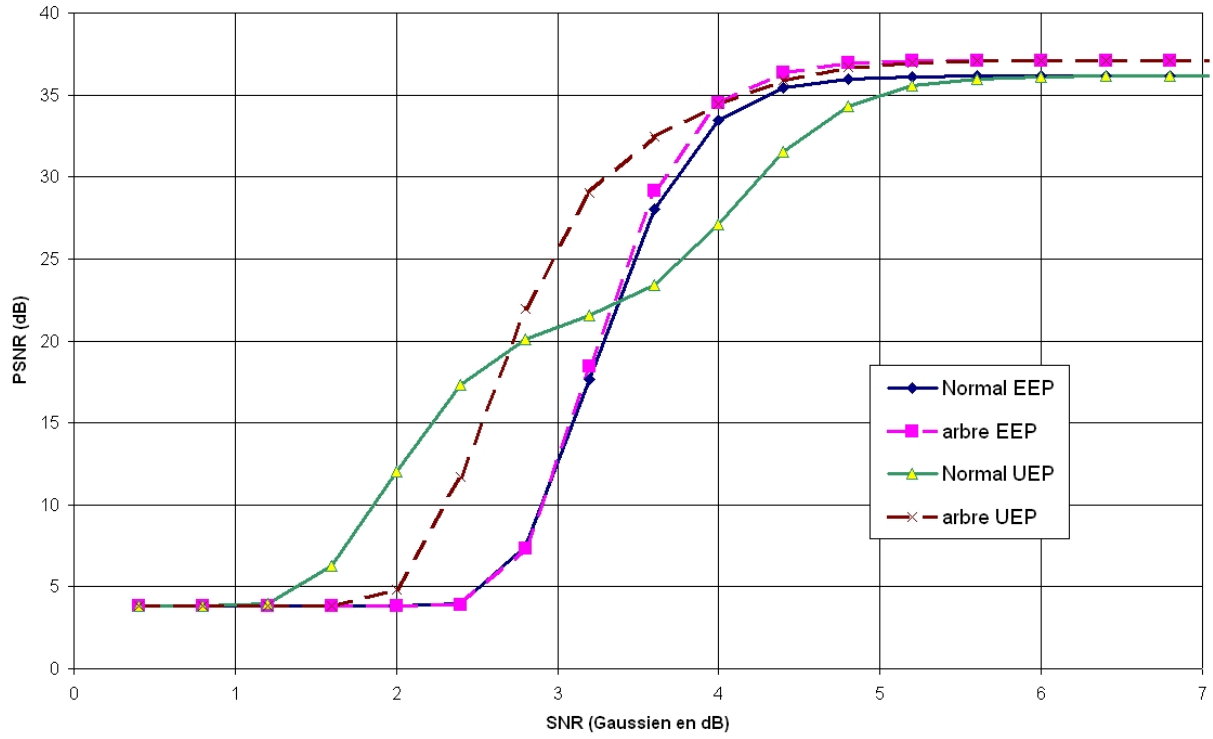

FIGURE 4.13 – Sensibilités en PSNR du premier GOP codé mode "normal" et en "arbre" (I_1P_{14}) de la séquences 'Foreman' en EEP/UEP.

De même que dans la figure 4.11, la configuration dite "normale" montre bien que le mode UEP s'adapte assez mal à ce type de configuration. En effet, elle offre trop peu de degrés de liberté pour adapter les rendements de manière optimales sur chaque partie de l'information. Néanmoins, la configuration "normale" semble mieux se comporter pour des rapports signal à bruit inférieurs à $2,5\ dB$, puisque le rendement de protection associé à la trame Intra étant trop important, à faible SNR, la distorsion estimée est alors celle d'un GOP ayant perdu toutes ses trames sauf la trame Intra (ce qui donne en PSNR moyen autour de 20 dB).

Quant au mode "frame shuffle" en "arbre", il tire pleinement profit du mode UEP, puisqu'il permet de gagner jusqu'à environ 0.50 dB en rapport signal-à-bruit (SNR) à qualité équivalente.

4.3 Contrôleur applicatif

Nous avons vu dans la partie précédente que l'on peut aisément estimer la sensibilité de n'importe quel flux codé en H.264 sur un canal bruité si l'on connaît les caractéristiques de celui-ci. Nous présenterons dans cette dernière partie l'emploi de cette méthode d'optimisation dans une chaîne de transmission vidéo de bout à bout.

4.3.1 Présentation

D'une manière générale, la plupart des transmissions multimédia s'établissent en plusieurs étapes. Selon les caractéristiques du canal considéré (SNR, taux d'effacement, type de canal, ...), on sélectionne la protection canal adaptée pour un débit alloué et fournissant un taux d'erreur résiduelle très faible (de l'ordre de 10^{-6}). De cette manière, on supposera alors que le taux d'erreur résiduelle espéré en sortie du décodeur canal est assez faible pour ne pas corrompre l'intégrité des données en entrée du décodeur source. Ensuite, en fonction du débit total alloué et du rendement utilisé par le codage canal sélectionné, on obtient le débit source alloué, qui servira de consigne pour l'algorithme d'allocation du débit du codeur source.

Cette méthode a pour défaut de ne pas s'adapter aux fluctuations des caractéristiques du canal (dans le cas par exemple d'une transmission sans fil), et surtout de ne pas tenir compte des sensibilités des différentes parties de l'information générées par le codeur source.

Notre vision est différente, puisque nous considérons le système dans sa globalité dans le but d'optimiser non pas le taux d'erreurs résiduelles (comme précédemment), mais la qualité de rendu visuel en sortie du décodeur source. De plus, nous supposons posséder des estimations des caractéristiques du canal. Nous proposons donc ici d'utiliser un contrôleur applicatif prenant en compte les conditions de canal estimées à intervalles réguliers pour optimiser la qualité espérée en réception, en contrôlant dynamiquement les débits sources et le taux de protection canal.

Grâce à notre étude de sensibilité (décrite au paragrpahe 4.2.2), lorsque l'on connaît les caractéristiques du canal, et on peut choisir, pour un débit total alloué, le meilleur compromis débit source et rendement utilisé pour maximiser la qualité de restitution d'une séquence vidéo. Cette méthode fonctionnera donc uniquement si l'on a accès à toutes les données intrinsèques (tailles, PSNR, ...) de chaque séquence codée pour différents débits cible.

4.3.2 Expériences et résultats

Chaîne de transmission considérée

Nos expériences ont été effectuées dans le cadre du projet européen PHOENIX suite aux travaux présentés dans [34]. La chaîne de transmission considérée, illustrée au début de ce document par la figure 2, comporte différentes parties, développées en collaboration avec les autres membres du consortium.

Le simulateur utilisé par la suite inclut donc non seulement le codeur source et le codeur RCPC utilisé au niveau du module "Application Processing", mais également des blocs qui simulent véritablement tous les échanges à travers les différentes étapes du système.

Notre chaîne de simulation comprend donc, en addition des éléments utilisés précédemment :

- une partie simulant les différentes couches de paquétisation et mise en réseau (RTP, UDP, IPv6), et prenant en compte les pertes possibles de paquets et les délais dus aux congestions possibles d'un réseau filaire,
- une autre partie simulant la transmission radio sur un canal sans fil comprenant un (dé-)codeur canal, un (dé-)entrelaceur, un (dé-)modulateur et un canal radio bruité.

Nous avons donc implémenté dans ce simulateur le contrôleur d'optimisation conjointe [35], qui optimise à chaque instant la répartition de la bande passante entre le flux compressé et la redondance pondérée sur chaque partie du flux, en fonction des informations de contrôle de qualité du canal (CSI) avec une mise à jour des paramètres d'optimisation par intervalle fixe d'une seconde. Par conséquence, la taille de GOP a été choisie aussi à une seconde pour permettre une ré-actualisation des paramètres de codages pour chaque GOP.

La partie réseau nous oblige à exploiter le mode multi-slice du standard H.264/AVC, permettant d'encapsuler dans plusieurs NALs de 180 octets les différentes slices d'une même trame. Quant à la distinction des ensembles sur lesquels sont appliqués les différents rendements de protections inégales, elle s'effectue selon le type de NAL ; typiquement, les NALs 5 (de type IDR) seront davantages protégées que les NALs 3 ou 4, pour le mode partitionnement de données.

Au niveau du lien radio, nous avons considéré soit une modulation BPSK, soit une modulation OFDM, avec dans les deux cas un codage correcteur d'erreur standard de type convolutif à protection égale, sauf pour les entêtes comrpessées qui ont été mieux protégées. Le codage UEP du flux de données, réalisé en marge du codeur correcteur RCPC précédemment introduit, a été réalisé au plus près du codage de soure, c'est-à-dire dans le module "Application Processing".

Pour obtenir le meilleur couple compression-protection, nous estimons toutes les sensibilités résultant des différentes configurations de codage correcteur et de codage source pour un débit total fixe de manière exhaustive, d'après les expressions formulées dans le paragraphe précédent (4.2), et choisissons la solution maximisant la qualité de rendu en bout de transmission d'après les caractéristiques du canal pour le prochain GOP.

Canal Gaussien et BPSK

Nous avons donc représenté sur la figure 4.14 l'évolution, avec et sans notre contrôleur applicatif, du PSNR de la séquence reconstruite au niveau décodeur en fonction du temps et à différents niveaux de rapport signal-à-bruit (SNR) d'un canal AWGN (variant aléatoirement de 1 à 6 dB).

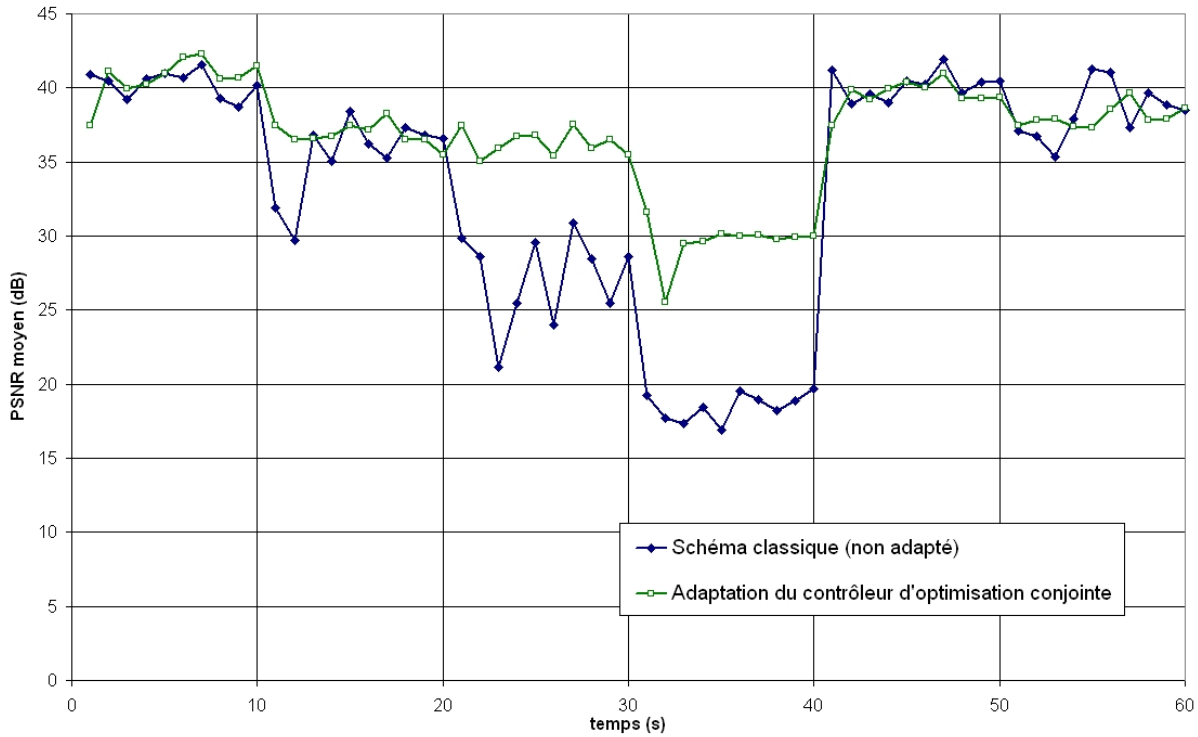

Figure 4.14 – Évolution du PSNR de la séquence 'Akiyo' en QCIF, à 15Fps pour un débit total de 128 $kbit/s$, avec et sans adaptation du contrôleur d'optimisation conjointe.

La première courbe "non-adaptée" représente l'évolution du PSNR moyen par GOP dans le cas où l'on suppose que l'on fixe à 50% le rapport de la bande passante allouée à la protection canal de manière uniforme à tout le GOP. La deuxième suppose que l'on a alloué dynamiquement en fonction des informations des caractéristiques estimées du

canal du GOP précédent, le rendement RCPC "optimal" à appliquer sur chaque partie du flux. En moyenne, le gain en PSNR est d'environ 3.4 dB dans les conditions de cette expérience.

Dans la figure 4.15, nous proposons un exemple qui montre le gain visuel du schéma adapté par rapport à celui d'une approche classique.

Figure 4.15 – Exemples de résultats visuels obtenus avec la simulation : à gauche, avec adaptation, et à droite sans adaptation du canal.

Canal à évanouissement et OFDM

Nous avons aussi considéré par ailleurs le cas d'un canal suivant les recommandations standard ETSI (channel A) [36], soit un canal de type WLAN, auquel nous avons rajouté un évanouissement suivant une loi log-normale avec un temps de cohérence de 5 secondes, canal sur lequel nous avons utilisé une modulation OFDM, pour un rapport signal-à-bruit moyen E_b/N_o de 13.2 dB.

L'évolution de qualité vidéo résultante GOP par GOP, en mode adapté et en mode non adapté, de la séquence reconstruite au niveau décodeur en fonction du temps sont représentés sur la figure 4.16. En moyenne, le gain en PSNR est d'environ 4.7 dB dans les conditions de cette expérience.

La figure 4.17 illustre visuellement ces résultats par deux images montrant le gain du schéma adapté par rapport à celui d'une approche classique dans le cas d'une transmission de type WLAN.

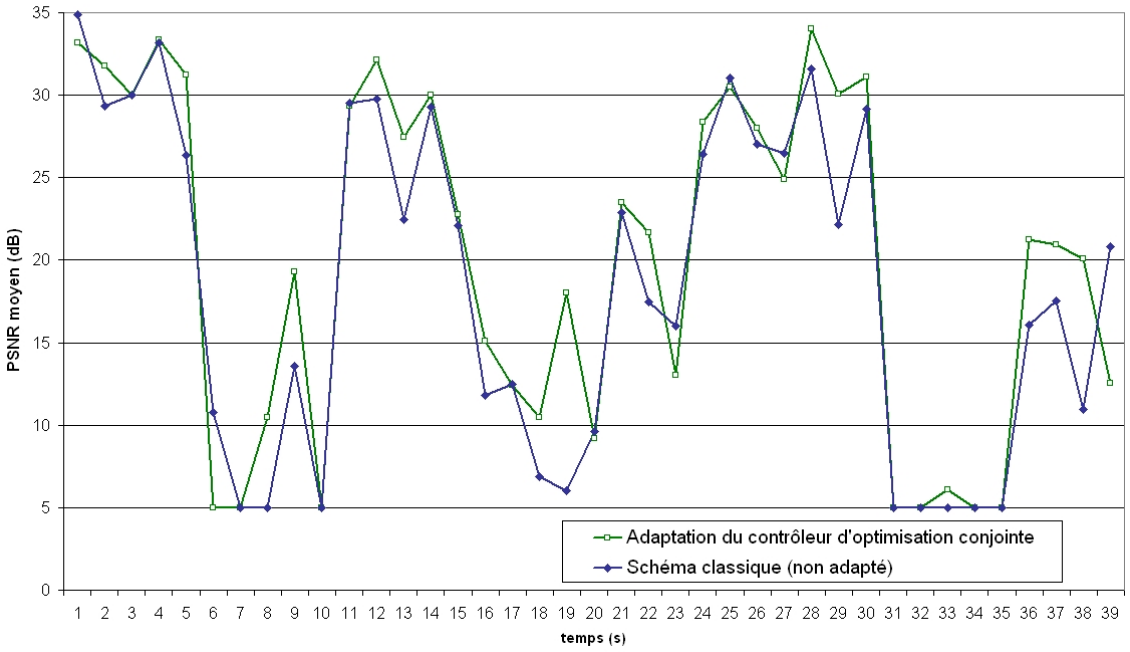

Figure 4.16 – Évolution du PSNR de la séquence 'Foreman' en QCIF à 15Fps pour un débit total de 256 $kbit/s$, avec et sans adaptation du contrôleur d'optimisation conjointe.

Figure 4.17 – Exemples de résultats visuels obtenus avec la simulation : à gauche, avec adaptation, et à droite sans adaptation du canal.

4.4 Conclusions

Nous avons donc montré que cette architecture de transmission multimédia adapté sur un canal sans fil permet d'obtenir un gain en PSNR sur l'ensemble de la plage de rapport signal à bruit d'intérêt. En effet, le contrôleur applicatif réalise une adaptation en fonction des caractéristiques de canal, transmises dans notre dispositif par un canal sans erreur, et lentement variable, les choix d'adaptation permettent de protéger plus ou moins certaines parties du flux d'information pour optimiser la qualité de rendu en sortie du décodeur source.

Conclusions et perspectives

Conclusions

Cette thèse est consacrée à l'optimisation conjointe d'une chaîne de communication pour des transmissions multimédia, et plus particulièrement pour la transmission de vidéo codées selon le standard H.264/AVC, choisi pour ses bonnes performances en terme de compression. Nous avons considéré trois points d'étude qui s'intéressent à trois aspects différents et indépendants d'optimisation sur la chaîne globale de transmission :

- exploitation de la redondance résiduelle inhérente à ce type de codeur,
- adaptation de la mise en forme de l'information source,
- analyse des sensibilités pour une allocation optimale des rendements respectifs de compression et de protection.

Dans le cadre de l'exploitation de la redondance résiduelle offerte par le codeur CAVLC du standard H.264/AVC, nous avons proposé, d'une part, un décodeur souple adapté à la norme H.264, permettant un réel gain en terme de fidélité de reconstruction (environ 2 dB de SNR dans le cas d'un bruit blanc Gaussien) et, d'autre part, une méthode de chiffrement difficilement attaquable par cryptanalyse, s'appuyant sur les propriétés statistiques particulières du codeur VLC de la norme H.264/AVC.

Dans le cadre de l'adaptation de la mise en forme de l'information source par mélange de trames, nous avons démontré que notre solution était compatible avec la norme sans aucune modification du décodeur, et qu'elle permettait d'offrir non seulement un gain en compression mais aussi qu'elle apportait les propriétés de scalabilité temporelle attendue. De plus, on notera que cette approche est très générique, car elle permet dès à présent de proposer une solution simple et efficace pour tout codeur vidéo par blocs, à la condition d'accepter alors une modification éventuelle du décodeur qui ne dsiposerait pas comme H.264 d'une double indexation des trames. Cette approche offre un nouveau degré de liberté dans la manière de coder une séquence vidéo.

Dans le cadre de l'analyse des sensibilités, nous avons proposé une méthode fiable permettant d'estimer, en fonction de la qualité du canal considéré, la distorsion de reconstruction en bout de chaîne de transmission. Cette méthode nous permet d'allouer de manière optimale, et dynamiquement en fonction du canal et de la source, les rendements à utiliser

pour maximiser la qualité de reconstruction en bout de chaîne de transmission. Nous avons montré que cette approche d'adaptation au moyen de formules semi-analytique apporte un véritable gain en terme de qualité de reconstruction que ce soit sur un canal de type AWGN, ou sur un canal de type WLAN.

Perspectives

Différents travaux pourraient permettre de continuer dans ces différents axes de travail afin d'étendre ou géneraliser les résultats obtenus. Ainsi, concernant les travaux sur le décodeur souple, il serait intéressant de prendre en compte le fait qu'en fonction de la taille et du type d'information contenue dans le flux les statistiques réelles des symboles VLC évoluent. On peut imaginer ajuster, à chaque instant, les statistiques des symboles dans l'algorithme de décodage pondéré, en fonction du nombre de bits et de la quantité d'informations restants à décoder. De plus, dans le cas de l'utilisation d'un décodeur canal à entrée et sortie souple, on peut envisager adapter notre décodeur source souple pour pouvoir sortir des informations souples, et permettre d'appliquer un décodage itératif.

Nous projetons enfin de continuer les travaux menés sur l'analyse des sensibilités, d'une part, en généralisant ces formules pour d'autres codeurs comme H.264/SVC, et d'autre part, en proposant de nouvelles méthodes semi-analytiques estimant plus rapidement et simplement les sensibilités intrinsèques des trames. Par exemple, on pourra chercher à estimer l'évolution du PSNR en fonction du débit source en supposant que les coefficients de la transformée entière suivent une loi de probabilité donnée (par exemple loi gaussienne ou de Cauchy). En outre, nous pensons prendre en compte dans de prochaines études les pertes dues aux congestions réseaux dans la formulation de la sensibilité.

Annexe A

Tables VLC du standard H.264 et bits transparents pour le chiffrement partiel

Cette annexe regroupe l'ensemble des tables VLC utilisées dans le procédé de chiffrement. Les bits susceptibles d'être chiffrés sont notés en *italique*. Nous n'avons pas ici répertorié les tables "trailing_ones" et "suffix_level", puisque tous les mots de codes, issus de ces tables, sont suceptibles d'être entièrement chiffrés. Nous ne mentionnerons pas ici les contextes exacts dans lesquels sont applicables les règles de chiffrement pour chaque table VLC. Comme indiqué dans le paragraphe 2.2.2, certaines mots commr ceux des macroblocs en bordure de slice ne doivent pas être chiffrés si l'on veut éviter d'introduire de potentiels configurations non compatible du standard.

Prediction Intra 4x4

	Mode considéré	Index	Mots de code
0	Vertical	0	*000*
1	Horizontal	1	*001*
2	DC	X	X
3	Diagonal Bas gauche	2	*010*
4	Diagonal Bas droite	3	*011*
5	Diagonal Vertical droit	4	*100*
6	Diagonal Horizontal bas	5	*101*
7	Diagonal Vertical gauche	6	*110*
8	Diagonal Horizontal haut	7	*111*

TABLE A.1 – Table VLC : Intra_4x4_pred

Type de macroblocs

Index	Mot de code	Mb_type	Pred_Mode	Coded_Block Chroma	Coded_Block Luma
0	1	I_4x4	Na	Na	Na
1	01*0*	I_16x16_0_0_0	0	0	0
2	01*1*	I_16x16_1_0_0	1	0	0
3	0010*0*	I_16x16_2_0_0	2	0	0
4	0010*1*	I_16x16_3_0_0	3	0	0
5	0011*0*	I_16x16_0_1_0	0	1	0
6	0011*1*	I_16x16_1_1_0	1	1	0
7	000100*0*	I_16x16_2_1_0	2	1	0
8	000100*1*	I_16x16_3_1_0	3	1	0
9	0001*010*	I_16x16_0_2_0	0	2	0
10	0001*011*	I_16x16_1_2_0	1	2	0
11	0001*100*	I_16x16_2_2_0	2	2	0
12	0001*101*	I_16x16_3_2_0	3	2	0
13	000111*0*	I_16x16_0_0_1	0	0	15
14	000111*1*	I_16x16_1_0_1	1	0	15
15	000010000	I_16x16_2_0_1	2	0	15
16	000010001	I_16x16_3_0_1	3	0	15
17	000010*010*	I_16x16_0_1_1	0	1	15
18	000010*011*	I_16x16_1_1_1	1	1	15
19	000010*100*	I_16x16_2_1_1	2	1	15
20	000010*101*	I_16x16_3_1_1	3	1	15
21	00001*0110*	I_16x16_0_2_1	0	2	15
22	00001*0111*	I_16x16_1_2_1	1	2	15
23	00001*1000*	I_16x16_2_2_1	2	2	15
24	00001*1001*	I_16x16_3_2_1	3	2	15
25	000011010	I_PCM	Na	Na	Na

Table A.2 – Table VLC : Mb_type

Prediction chroma

Index	Mot de code	Mode de prediction chroma
0	1	DC
1	01*0*	Horizontal
2	01*1*	Vertical
3	00100	Plan

Table A.3 – Table VLC : Chroma_Pred

Paramètre de quantification

Index	Mot de code	Mb_QP_Delta
0	1	0
1	01*0*	1
2	01*1*	-1
3	001*00*	2
4	001*01*	-2
5	001*10*	3
6	001*11*	-3
7	0001*000*	4
8	0001*001*	-4
...	...	...

Table A.4 – Table VLC : Mb_QP_Delta

Nombre de coefficients nuls

Index	Mot de code	Total_zeros
0	1	0
1	01*0*	1
2	01*1*	2
3	001*0*	3
4	001*1*	4
5	0001*0*	5
6	0001*1*	6
7	00001*0*	7
8	00001*1*	8
9	000001*0*	9
10	000001*1*	10
11	0000001*0*	11
12	0000001*1*	12
13	00000001*0*	13
14	00000001*1*	14
15	00000000	15

Table A.5 – Table VLC : Total_zeros pour des blocs 4x4, lorsque total_coeff=1

Index	Mot de code	Total_zeros
0	1	0
1	01	1
2	00*1*	2
3	00*0*	3

Table A.6 – Table VLC : Total_zeros pour des blocs 2x2, lorsque total_coeff=1

Plage de zeros

Run_before /Zeros_left	1	2	3	4	5	6	plus de 6
0	*1*	1	*11*	*11*	1*1*	11	*111*
1	*0*	0*1*	*10*	*10*	1*0*	*000*	*110*
2		0*0*	*01*	*01*	0*11*	*001*	*101*
3			*00*	00*1*	0*10*	*011*	*100*
4				00*0*	0*01*	*010*	*011*
5					0*00*	*101*	*010*
6						*100*	*001*
7							0001
8							00001
9							000001
10							0000001
11							00000001
12							000000001
13							0000000001
14							00000000001

Table A.7 – Table VLC : Run_before

Annexe B

Analyse de la redondance temporelle dans les schémas hiérarchiques

Dans le cas des trames prédites (*P-frames*), on utilise la redondance temporelle en ne codant principalement que l'erreur de prédiction (différence entre l'image prédite par la compensation de mouvement et l'original). Considérons ici que le codage de trames prédites temporellement puisse se modéliser comme le codage de variables aléatoires, pour lequel on ne code que la différence entre celle-ci et celle de l'instant précédent, et observons l'influence de l'emploi de différents pas de quantification temporellement.

Soit $\{X_n\}_n$ une suite de variables aléatoires corrélées, telles que X_n est dépendante seulement de X_{n-1} :

$$X_n = X_{n-1} + \varepsilon_n \tag{B.1}$$

où $\{X_n\}_n$ peut suivre n'importe quelle densité de probabilité et $\{\varepsilon_n\}_n$ est une suite de variables aléatoires indépendantes, suivant une densité de probabilité symétrique et centrée, de variance σ^2.

Quantification

Supposons que l'on applique une quantification scalaire uniforme Q_n à chaque instant n sur les valeurs $\{\varepsilon_n\}_n$ à transmettre. Le bruit de quantification va conduire à une distorsion D_n pour chaque n, dépendante du pas de quantification δ_n appliqué, qui s'écrit sous l'hypothèse de haute résolution :

$$D_n = \frac{\delta_n^2}{12}.$$

De plus, on supposera que l'on travaille en boucle fermée, de manière à modeliser fidèle-

ment le codage de trames prédites. Ainsi on peut écrire :

$$X_n = \widehat{X}_{n-1} + \widetilde{\varepsilon}_n \tag{B.2}$$

et

$$\widehat{X}_n = \widehat{X}_{n-1} + \widehat{\varepsilon}_n \tag{B.3}$$

avec $\widetilde{\varepsilon}_n$ variable aléatoire différente de ε_n à cause des étapes de quantifications et d'estimations précédentes et $\widehat{\varepsilon}_n = Q_n(\widetilde{\varepsilon}_n)$.

Grâce à (B.1) et (B.2), ont peut écrire :

$$\widehat{X}_{n-1} - X_{n-1} = \widetilde{\varepsilon}_n - \varepsilon_n \tag{B.4}$$

De plus grâce à (B.3), on peut écrire que :

$$\widehat{X}_{n-2} + \widehat{\varepsilon}_{n-1} - X_{n-1} = \widetilde{\varepsilon}_n - \varepsilon_n \tag{B.5}$$

en réutilisant (B.2),

$$\widehat{\varepsilon}_{n-1} - \widetilde{\varepsilon}_{n-1} = \widetilde{\varepsilon}_n - \varepsilon_n \tag{B.6}$$

Donc

$$Q_{n-1}(\widetilde{\varepsilon}_{n-1}) - \widetilde{\varepsilon}_{n-1} = \widetilde{\varepsilon}_n - \varepsilon_n \tag{B.7}$$

$$\widetilde{\varepsilon}_n = \big(Q_{n-1}(\widetilde{\varepsilon}_{n-1}) - \widetilde{\varepsilon}_{n-1}\big) + \varepsilon_n. \tag{B.8}$$

Or, si $\widetilde{\varepsilon_n}$ suit une loi de probabilité symétrique et centrée, alors $d_n = Q_n(\widetilde{\varepsilon}_n) - \widetilde{\varepsilon}_n$ suit une loi de probabilité symétrique et centrée qui tend à être uniforme sur $[-\frac{\delta_n}{2}, \frac{\delta_n}{2}]$ avec une variance égale à $\frac{\delta_n^2}{12}$ si on utilise une quantification uniforme avec $\delta_n \ll \sigma_n$.

Pour montrer que $\widetilde{\varepsilon}_n$ suit une loi de probabilité symétrique et centrée, on peut le prouver par récurrence : $\widetilde{\varepsilon}_1 = \big(Q_0(\widetilde{\varepsilon}_0) - \widetilde{\varepsilon}_0\big) + \varepsilon_1$ or, $\widetilde{\varepsilon}_0 = \varepsilon_0$, donc $d(0)$ est symétrique et centrée, ainsi que ε_1, donc aussi $\widetilde{\varepsilon}_1$.

Si au rang n, $\widetilde{\varepsilon}_n$ est symétrique et centrée, alors $\widetilde{\varepsilon}_{n+1} = \big(Q_n(\widetilde{\varepsilon}_n) - \widetilde{\varepsilon}_n\big) + \varepsilon_{n+1}$. Comme $\widetilde{\varepsilon}_n$ est symétrique et centrée, d_n l'est aussi. ε_{n+1} étant par définition symétrique et centré, alors $\widetilde{\varepsilon}_{n+1}$ l'est aussi. On conclue par récurrence.

En considérant tous ces résultats, on a :

$$\widetilde{\varepsilon}_n = d_{n-1} + \varepsilon_n, \tag{B.9}$$

avec d_n uniforme sur $[-\frac{\delta_n}{2}, \frac{\delta_n}{2}]$.

Donc $\widetilde{\varepsilon}_n$ dépend uniquement de l'étape de quantification à l'instant $n-1$, et sa densité de probabilité est égale à la convolution de celle de ε_n avec d_{n-1}, et sa variance $\widetilde{\sigma}_n^2 \simeq \sigma_n^2 + \frac{\delta_{n-1}^2}{12}$ $\forall\ n$.

Adaptation dans le cas de séquence vidéo

Dans une séquence vidéo contenant une unique "scène", i.e. qui ne contient pas de changement brusque de contenu, on peut considérer que la redondance temporelle suit une même loi d'une image à l'autre, i.e. la densité de probabilité de ε_n est identique $\forall\ n$.
Si $\delta_n > \delta_{n-1}$ alors $\widetilde{\sigma}_{n+1} > \widetilde{\sigma}_n$.

Donc $\widehat{\varepsilon}_n = Q_n(\widetilde{\varepsilon}_n)$ a une densité de probabilité qui dépend des quantifications aux instants n et $n-1$, et donc $H(\widehat{\varepsilon}_{n+1}) > H(\widehat{\varepsilon}_n)$. Par contre, si on considère pour le même $\widetilde{\varepsilon}_n$ deux quantifications Q_n et Q'_n avec δ_n et δ'_n leurs pas de quantification respectifs, si $\delta_n < \delta'_n$, alors $H(\widehat{\varepsilon}_n) > H(\widehat{\varepsilon}'_n)$.

Dans le cas d'une quantification uniforme de même pas de quantification $\forall\ n$, l'ordre des trames successivement codées (i.e. quantifiées) n'influe pas sur la distorsion et sur le débit (i.e. l'entropie). En revanche, dans le cas où la quantification est uniforme mais de pas de quantification variable à chaque instant n, on obtient alors pour différentes configurations de codage différents résultats de débit et de distorsion.

Ainsi, si l'on compare deux configurations types du "mélange de trame" appelées "normal" et "miroir", de taille de GOP de 7, avec pour pas de quantification $\delta_0 < \delta_1 = \delta_2 < \delta_3 = \delta_4 < \delta_5 = \delta_6$, alors $\widetilde{\varepsilon}_n = d_{n-1} + \varepsilon_n$ avec d_n uniforme sur $[-\frac{\delta_n}{2}, \frac{\delta_n}{2}]$ dans le cas du "normal" et $\widetilde{\varepsilon'}_n = d_{E(n/2)-1} + \varepsilon_n$ avec d_n uniforme sur $[-\frac{\delta_n}{2}, \frac{\delta_n}{2}]$ dans le cas du "miroir", $\forall\ n$ et $\widetilde{\sigma}_n^2 \simeq \sigma_n^2 + \frac{\delta_{n-1}^2}{12}$, et $\widetilde{\sigma'^2_n} \simeq \sigma_n^2 + \frac{\delta_{E(n/2)-1}^2}{12}$

Donc $\forall\ n \in \{1, \ldots, 6\}$, $H(\widehat{\varepsilon}_n) < H(\widehat{\varepsilon'}_n)$. Ce qui signifie que le débit sera théoriquement plus faible pour la configuration "mirror", ce qui est conforté par les résultats de nos simulations.

En revanche, $D(1)+D(2)+D(3)+...+D(6) = D'(1)+D'(2)+...+D'(6) = \frac{\delta_1^2}{12}+\frac{\delta_2^2}{12}+...+\frac{\delta_6^2}{12}$.
La distorsion totale est donc égale pour les deux configurations.

Bibliographie

[1] ISO/IEC 13818-2 : *2000 Information Technology.* Coding of Moving Pictures and Associated Audio Information. Part 2 : Video.

[2] T. Wiegand and G. Sullivan : *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification.* (ITU-T Rec H.264|ISO/IEC 14496-10 AVC) Mars 2003.

[3] "Advanced Encryption Standard", National Institute of Standards and Technology, FIPS-197, November 2001.

[4] *Joint verification model for H.264*, JM 10.3 (http ://iphome.hhi.de/suehring/tml/), July 2005.

[5] Y. Altunbasak and N. Kamaci : "An analysis of the DCT coefficient distribution with the H.264 video coder", in *Proceedings IEEE Acoustics, Speech, and Signal Processing. (ICASSP'04)*, vol. 3, pp. 177-180, Montreal, Canada, May 2004 .

[6] S.B. Zahir Azami, P. Duhamel and O. Rioul : "Combined Source-Channel Coding : Panorama of Methods", in *CNES Workshop on Data Compression*, Toulouse, France, 13-14 Nov. 1996.

[7] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv : "Optimal decoding of linear codes for minimal symbol error rate", *IEEE Transactions On Information Theory*, vol. IT-20, pp. 284-287, March 1974.

[8] V.B. Balakirsky : "Joint source-channel coding with variable length codes", in *Proceedings IEEE International Symposium Information Theory*, pp. 415-420, Ulm, Germany, 1997.

[9] R. Bauer and J. Hagenauer : "Turbo-FEC/VLC-decoding and its application to text compression", in *Proceedings IEEE CISS'00*, pp. 6-11, Princeton University, New

Jersey, USA, March 2000.

[10] L. Blaszak, M. Domanski, A. Luczak and S. Mackowiak : "AVC video coders with spatial and temporal scalability", in *Proceedings of the Picture Coding Symposium (PCS'03)*, pp. 41-47, April 2003, St-Malo, France.

[11] V. Buttigieg : *Variable-length error-correcting codes.*, Ph.D. thesis, University of Manchester, Manchester, United Kingdom, 1995.

[12] M. Bystrom, S. Kaiser and A. Kopansky : "Soft source decoding with applications", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, n. 10, pp. 1108-1120, Octber 2001.

[13] N. Demir and K. Sayood : "Joint source/channel coding for variable length codes", in *Proceedings IEEE DCC'98*, pp. 139-148, Snowbird, Utah, USA, March-April 1998.

[14] G. Boisson, E. François and C. Guillemot : "Efficient scalable motion coding for wide-range scalable video coding", in *Proceedings of XII. European Signal Processing Conference (EUSIPCO-2004)*, Vienna, Austria September 2004.

[15] J. B. Cain, G. C. Clark and J. M. Geist : "Punctured convolutional codes of rate (n-1)/N and simplified maximum likelihood decoding", *IEEE Transactions on Information Theory*, vol. IT-25, pp. 97-100, January 1979.

[16] Q. Chen and K.P. Subbalakshmi : "Joint source-channel decoding for MPEG-4 video transmission over wireless channels," in *IEEE Journal on Selected Areas in Communications*, vol. 21, n. 10, pp. 1780-1789, 2003.

[17] R.M. Fano : "A heuristic discussion of probabilistic decoding", *IEEE Transactions on Information Theory*, vol. 64, n. 9, pp. 64-73, April 1963.

[18] K. Fazel and J.J. Lhuillier : "Application of Unequal Error Protection Codes on Combined Source-Channel Coding of Images," in *Proceedings of ICC'90*, vol. 320, n. 5, pp. 1-6, 1990.

[19] G.D. Forney : "The Viterbi Algorithm", in *Proceedings of the IEEE*, vol. 61, n. 3, pp. 268-278, March 1973.

[20] C. Guillemot and P. Siohan : "Joint source-channel decoding with soft information : A survey", *Elsevier Journal on Applied Signal Processing, special issue on the turbo principle*, vol. 6, pp. 906-927, 2005.

[21] T. Guionnet and C. Guillemot : "Soft decoding and synchronization of arithmetic codes : Application to image transmission over noisy channels.", *IEEE Transactions on Image Processing*, vol. 12, pp. 1599-1609, Dec. 2003.

[22] L. Guivarch, J.C. Carlach and P. Siohan : "Joint source-channel soft decoding of Huffman sources with turbo-codes". in *Proceedings Data Compression Conf.*, pp. 83-92, March. 2000.

[23] J. Hagenauer : "A Viterbi Algorithm with Soft-Decision Output and its Applications", in *Proceedings GLOBECOM'89*, vol. 47, n. 1, pp. 1-7, Dallas, TX, Nov. 1989.

[24] J. Hagenauer : "Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications", *IEEE Transactions on Communications*, vol. 36, pp. 389-400, April 1988.

[25] Z. He, J. Cai and C.W. Chen : "Joint source channel rate-distorsion analysis for adaptive mode selection and rate control in wireless video coding", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 511-523, June 2002.

[26] M. Jeanne, P. Siohan, J-C. Carlach and L. Guivarch : "Source and Joint Source-Channel Decoding of Variable Length Codes" in *Proceedings ICC*, New York, USA, April/May 2002.

[27] H. Jenkac, T. Stockhammer and W. Xu : "Cross-layer assisted reliability design for wireless multimedia broadcast", in *EURASIP Signal Processing Journal, Special Issue on Advances in Signal Processing-assisted Cross-layer Designs*, pp. 1933-1949, 2006.

[28] C. Lamy and L. Perros-Meilhac : "Low complexity iterative decoding of variable-length codes", in *Proceedings of the Picture Coding Symposium (PCS'03)*, pp. 275-280, April 2003, St-Malo, France.

[29] C. Lamy and O. Pothier : "Reduced complexity maximum a posteriori decoding of variable-length codes", in *Proceedings IEEE Globecom'01*, pp. 1410-1413, San Antonio, Texas, USA, Nov. 2001.

[30] C.M Lee, M. Kieffer and P. Duhamel : "Robust motion vectors and texture transmission for the H263 video encoder family", in *Proceedings of the Picture Coding Symposium (PCS'03)*, pp. 247-251, April 2003, St-Malo, France.

[31] S. Lin and D.J. Costello Jr : "Error Control Coding : Fundamentals and applications", *Prentice Hall* 1983

[32] J.L. Massey : "Variable-length codes and the Fano metric", *IEEE Transactions on Information Theory* vol. 18, n. 1, pp. 196-198, Jan. 1972.

[33] M.G. Martini and M. Chiani : "Rate-Distortion models for Unequal Error Protection for wireless video trnasmission", in *Proceedings IEEE Trans. on Wireless Comm.*, vol. 3, n. 1, pp. 258-268, Jan. 2004.

[34] M. Martini et al : "A demonstration platform for network aware joint optimization of wireless video transmission", in *IST Mobile Summit 2005*, Dresden, Germany, June 2005.

[35] M. Martini, M. Mazotti, C. Lamy-Bergot, J. Huusko and P. Amon : "Content Adaptive Network Aware Joint Optimization of Wireless Video Transmission", in *IEEE Journal on Communication Magazine*, Vol. 45, n. 1, pp. 84-90, January 2007.

[36] J. Medbo, H. Andersson, P. Schramm, H.Asplund and J.-E. Berg : "Channel models for Hiperlan/2 in different indoor scenarios", in *COST 259*, TD(98)70, Bradford, UK, Apr. 1998.

[37] H. Nguyen and P. Duhamel : "Optimal Soft-Output Viterbi Algorithm", in *Proceedings Third Conference on Networking (ICN'04)*, Mars 2004.

[38] H. Nguyen and P. Duhamel : "Iterative joint source-channel decoding of variable length encoded video sequences exploiting source semantics," in *Proceedings of International Conference on Image Processing (ICIP 2005)*, vol. 5, pp. 3221-3224, Singapore, 2004.

[39] J.-R. Ohm : "Multimedia Communication Technology", *Springer*, 2004.

[40] L. Perros-Meilhac and C. Lamy : "Huffman tree based metric derivation for a low-complexity sequential soft VLC decoding" *Proceedings of ICC'04* vol. 2, pp. 783-787, New York, USA, April-May 2002.

[41] M. Park and D.J. Miller : "Joint source-channel decoding for variable length encoded data by exact and approximate MAP sequence estimation", *IEEE Transactions on Communications*, vol. 48, n. 1, pp. 1-6, January 2000.

[42] A.H. Murad and T.E. Fuja : "Joint source-channel decoding of variable-length encoded sources", in *Proceedings IEEE ITW'98*, pp. 95-95, Killarney, Ireland, June 1998.

[43] G-R. Mohammad-Khani, C-M. Lee, M. Kieffer and P. Duhamel : "Simplification of VLC tables with application to ML and MAP decoding algorithms", *IEEE*

Transactions on Communications, vol. 54, n. 10, pp. 1835–1844, October 2006.

[44] G. Pau, C. Tillier, B. Pesquet-Popescu and H. Heijmans : "Motion compensation and scalability in lifting-based video coding" *Signal Processing : Image Communication*, vol. 9, n. 7, pp. 577-600, Aug 2004.

[45] J. Proakis : "Digital Communications", *McGraw-Hill*, 1983.

[46] J. Daemen and V. Rijmen : "Rijndael, the advanced encryption standard.", in *Dr. Dobb's Journal*, vol. 26, n. 3, pp. 137-139, March 2001.

[47] C.E. Shannon : "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pt. II, pp. 623-656, 1948.

[48] C. Shi, S. Wang and B. Bhargava : "MPEG video encryption in real-time using secret key cryptography", in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pp. 2822-2829, Las Vegas, Nevada, 1999.

[49] H. Schwarz, D. Marpe and T. Wiegand : "Subband extension for H.264/AVC", Doc JVT-K023, in *VCEG meeting - ITU-T SG16/Q.6*, Munich, Germany, March 2004.

[50] H. Schwarz, D. Marpe and T. Wiegand : "Analysis of hierarchical B pictures and MCTF", in *Proc. IEEE ICME'06*. Toronto, July 2006.

[51] H. Schwarz, D. Marpe and T. Wiegand : "Hierarchical B pictures", Doc JVT-P014 , *Joint Video Team* , Poznan, Poland, July 2005.

[52] T. Stockhammer and M. Bystrom : "Dependent source and channel rate allocation for video transmission", in *IEEE Transactions on Wireless Communications*, vol. 3, n 1, pp. 258-268, Jan. 2004.

[53] K. Stuhlmuller, N. Farber, M. Link and B. Girod : "Analysis of video transmission over lossy channels,", in *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1012-1032, June 2000.

[54] Y. Takishima, M. Wada and H. Murakami : "Reversible variable length codes", *IEEE Transactions on Information Theory*, vol. 43, n. 3, pp. 158-162, 1995.

[55] L. Tang : "Methods for encrypting and decrypting MPEG video data efficiently", in *Proceedings of the ACM International Multimedia Conference 1996*, pp. 219-229, Boston, Nov. 1996.

[56] Vinai A. Vaishampayan : "Design of multiple description scalar quantizers", *IEEE Transactions on Information Theory*, vol. 39, n. 3, pp. 821-834, 1993.

[57] P.P. Vaidyanathan : "Multirate systems and filter banks". *Prentice Hall, Englewood Cliffs*, NJ, USA, 1993.

[58] M. Van der Schaar, S. Krishnamachari, S. Choi and X. Xu : "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs", in *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1752-1763, 2003.

[59] C. Weidmann, P. Kadlec, O. Nemethova and A. Al Moghrabi : "Combined sequential decoding and error concealment of H.264 video", in *Proceedings IEEE MMSP'04*, pp. 299-302, Siena, September 2004.

[60] J. Wen and J.D. Villasenor : "Utilizing soft information in decoding of variable length codes", in *Proceedings IEEE DCC'99*, pp. 131-139, Snowbird, Utah, USA, March 1999.

[61] T. Wiegand, G. Sullivan, G. Bjøntegaard and A. Luthra : "Overview of the H.264/AVC video coding standard", *IEEE Transactions Circuits Syst. Video Technol.* vol. 13, pp. 560-576, July 2003.

[62] J. Xu, Z. Xiong, S. Li and Y. Zhang : "Three-dimensional embedded subband coding with optimized truncation (3D EBCOT)" *Applied and Computational Harmonic Analysis*, vol. 10, pp. 290-315, 2001.

[63] J. Zheng, X. Ji, G. Ni, W. Gao and F. Wu : "Extended direct mode for hierarchical B picture coding", in *Proceedings of International Conference on Image Processing (ICIP 2005)*, vol. 2, pp. 265-268, Genova, September 2005.

[64] C. Bergeron and C. Lamy-Bergot : "Soft-input decoding of variable-length codes applied to the H.264 standard", in *Proceedings IEEE MMSP'04*, pp. 87-90, Siena, September 2004.

[65] C. Bergeron and C. Lamy-Bergot : "Compliant Selective Encryption for H.264/AVC video streams", in *Proceedings IEEE MMSP'05*, pp. 447-480, Shanghai, November 2005.

[66] C. Bergeron, C. Lamy-Bergot, and B. Pesquet-Popescu : "Adaptive M-Band Hierarchical filterbank for compliant temporal scalability in H.264 standard", in *Proceedings IEEE ICASSP'05*, Philadelphia, March 2004.

[67] C. Bergeron, C. Lamy-Bergot, G. Pau and B. Pesquet-Popescu : "Temporal Scalability through adaptive M-band filterbanks for robust H.264/AVC video coding", in *EURASIP Journal on Applied Signal Processing*, Article ID 21930, March 2006.

[68] C. Lamy-Bergot, N. Chautru and C. Bergeron : "Unequal Error Protection for H.263+ bitstreams over a wireless IP network" in *Proceedings IEEE ICASSP'06* , Toulouse, May 2006.

[69] C. Bergeron and C. Lamy-Bergot : "Modelling H.264/AVC sensitivity for error protection in wireless transmissions" in *Proceedings IEEE MMSP'06* , Victoria, October 2006.