ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

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

PAR
Luc TRUDEAU

DÉTECTION ET DISSIMULATION DE LA DÉTÉRIORATION VISUELLE ISSUE DU DÉCODAGE DE SÉQUENCES H.264 CORROMPUES

MONTRÉAL, LE 19 MAI 2011

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

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

M. Mohamed Cheriet, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

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

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

LE 16 MAI 2011

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Ce mémoire de maîtrise fait le point sur certains concepts proéminents issus de plus de 2 ans de recherche sur la détection et la dissimulation d'erreurs. Vous conviendrez avec moi qu'un tel effort de recherche requiert des ressources financières et matérielles considérables. C'est pourquoi j'exprime ma reconnaissance et mes remerciements à Stéphane Coulombe, professeur à l'École de technologie supérieure, qui a accepté de me faire confiance dans ce projet et d'avoir consacré les ressources nécessaires à sa réalisation. J'en profite également pour souligner son engagement hors du commun en ce qui a trait à ma formation et à la qualité de ce mémoire.

Je tiens à remercier Evelyne, ma conjointe, qui m'a toujours soutenu dans cette aventure, à tous les niveaux. À nos deux filles, Joannie et Alexane, qui tout comme Evelyne, sont pour moi une source de motivation inépuisable.

Je tiens à souligner les efforts de Chantal Gamache, conseillère au Service d'appui en communication (SAC) à l'École de technologie supérieure, et Steven Pigeon, professionnel de recherche à la chaire de recherche industrielle Vantrix en optimisation vidéo. Chantal a consacré son temps, sa patience et son talent à l'amélioration de la qualité de la rédaction de cet ouvrage. Steven, de par son savoir-faire et ses connaissances, a enrichi, lors de nos nombreux échanges, mon approche et mes méthodes de recherches.

Finalement, cet ouvrage fut rendu possible, en partie, par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), bourse #356807-07 et, en partie, par Le Fonds québécois de la recherche sur la nature et les technologies (FQRNT), bourse #141698.

DÉTECTION ET DISSIMULATION DE LA DÉTÉRIORATION VISUELLE ISSUE DU DÉCODAGE DE SÉQUENCES H.264 CORROMPUES

Luc TRUDEAU

RÉSUMÉ

Compte tenu de leur nature, les réseaux mobiles sont plus fortement enclins à la corruption de données que leurs contreparties filaires. Même si les données se rendent à destination, les bits endommagés entrainent le rejet des paquets qui les encapsulent. Ces pertes ont un impact important sur la qualité de l'expérience de l'utilisateur lors de la consultation de flux vidéos ou de la vidéophonie, et ce, surtout lorsque la retransmission n'est pas envisageable. On restreint l'usage des approches conventionnelles de résilience aux erreurs, tels la retransmission de trames ou l'ajout de trames redondantes, car elles imposent un fardeau considérable aux réseaux mobiles, ceux-ci étant déjà fortement achalandés.

Dans cet ouvrage, nous proposons la réutilisation sélective des données corrompues afin d'augmenter la qualité visuelle de séquences endommagées. Cette sélection est guidée par une nouvelle approche de détection de la détérioration visuelle dans le domaine des pixels. Elle combine la mesure des effets de bloc (discontinuités spatiales en bordure de blocs) à l'estimation du mouvement.

Notre méthode a été testée sur un ensemble de 17 séquences QCIF codées en H.264 avec des QP de 16 à 28 et soumis à des taux d'erreurs de 0.0004 à 0.0032. Nos résultats de simulation démontrent qu'il est possible de décoder des trames corrompues. La probabilité d'un décodage réussi varie de 20 % à 70 % selon les paramètres d'encodage et le taux d'erreurs subies lors du transport. De plus, notre algorithme, développé en fonction de la norme H.264, réussit à effectuer le bon choix de 81 % à 86 % et 88 % à 91 % des cas (selon les conditions). Lorsque notre algorithme est combiné au décodeur de référence H.264, nous observons un gain moyen 0.65 dB à 0.86 dB de PSNR par rapport au calque de trame et calque de tranche respectivement pour nos conditions de test.

Mots-clés : Détection d'erreurs, dissimulation d'erreurs, H.264, vidéo mobile.

DETECTION AND CONCEALMENT OF VISUAL DEGRADATION RESULTING FROM ERRONEOUS H.264 SEQUENCES

Luc TRUDEAU

ABSTRACT

In mobile video applications, where unreliable networks are commonplace, corrupted video packets can have a profound impact on the quality of the user experience. Error resilient mechanisms like retransmission and redundant frames may impose an unacceptable burden on mobile networks. In these cases a decoder-based error resilience approach, like the one described in this work, can help improve the end-user experience without adding load to the network.

In this master's thesis, we show that, in a wide range of operating conditions, selectively reusing data resulting from decodable broken packets leads to better results than frame copy. This selection is guided by a novel concept that combines motion estimation and a measure of blocking artefacts at block edges to predict visual degradation caused by the decoding of erroneous packets.

The proposed solution was tested against 17, H.264 coded, QCIF sequence with quantization parameters from 16 to 28 and exposed to bit error rates of 0.0004 to 0.0032. Simulation results show that the probability of successfully decoding a broken sequence varies from 20 % to 70 % (depending on operating conditions). Combined with the proposed solution, the H.264/AVC JM reference software decoder, can select the best option between frame copy and the erroneous frame decoding in 81 % to 86 % and 88 % to 91 % of the test cases (depending on operating conditions). We also obtain an average gain of 0.65 dB to 0.86 dB when evaluated against frame copy and slice copy depending on our testing conditions.

Keywords: Error concealment, error detection, H.264, mobile video, pixel domain

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AIDB	*Average Inter-sample Difference across Boundaries*
AIDSB	*Average Internal Difference between Subsequent Blocks*
ASO	*Arbitrary Slice Order*
AVC	*Advanced video coding*
BER	*Bit error rate* Taux d'erreurs binaire
CABAC	*Context-adaptive binary arithmetic coding* Codage arithmétique binaire à contexte adaptatif
CAVLC	*Context-adaptive variable-length coding* Codage entropique à longueur variable
COD	Bit positionné en début de macrobloc identifiant s'il est encodé ou pas.
dB	décibel
DCT	*Discrete cosine transform* Transformée en cosinus discrète
FMO	*Flexible macroblock ordering* Ordonnancement flexible de macroblocs.
IAIDB	*Internal AIDB*
$IAIDB_{block}$	*Internal AIDB per block*
IP	*Internet protocol* Protocole Internet
JM	*Joint Model*
JPEG	*Joint photographic experts group*
ko/s	Kilooctet par seconde

LBP	*Local binary pattern* Patron binaire local
MCB	*Motion compensated blockiness* Effets de bloc compensés par le mouvement
MPEG	*Moving picture experts group*
NAL	*Network abstraction layer* Couche d'abstraction réseau
OSI	*Open Systems Interconnection*
PMVFAST	*Predictive Motion Vector Field Adaptive Search Technique*
PSNR	*Peak signal-to-noise ratio*
QCIF	*Quarter common international format* (176×144)
QP	*Quantization parameter* Paramètre de quantification
RBSP	*Raw Byte Sequence Payload*
RTP	*Real-time transport protocol* Protocole de streaming temps-réel
SAD	*Sum of absolute differences* Somme des différences absolues
SAE	*Sum of absolute error* Somme de l'erreur absolue
SDMCB	*Sum of the distributed motion compensated blockiness* Somme des effets de bloc compensés par le mouvement distribuée
SMCB	*Sum of motion compensated blockiness* Somme des effets de bloc compensés par le mouvement
SVM	*Support vector machine* Machine à vecteurs de support.

TC	*Texture consistancy* Consistance de la texture
UMHexagonS	*Unsymmetrical cross Multi-Hexagon-Grid-Search*
UDP	*User datagram protocol* Protocole de datagramme utilisateur
VCL	*Video coding layer* Couche de codage vidéo
YC_BC_R	Luminance, chrominance bleu, chrominance rouge

INTRODUCTION

Problématique

Tout comme l'univers, l'Internet est en expansion continue. Selon l'agence comScore, plus de 170 millions d'États-Uniens ont visionné de la vidéo par l'intermédiaire du réseau Internet durant le mois de février 2011 (comScore, 2011). De plus, comScore (2011) ajoute que durant ce mois, ce même public a effectué plus de 5 milliards de visionnements de séquences vidéos. Pour sa part, l'agence Juniper Research (2010) affirme que la vidéophonie atteindra 29 millions d'utilisateurs d'ici 2015.

Devant ces chiffres, on ne peut que constater la croissance fulgurante de la consommation de la vidéo sur des réseaux peu fiables, tels Internet ou, dans des conditions *temps réel*, comme c'est le cas avec la vidéophonie. Cette tendance de consommation se manifeste aussi dans les réseaux mobiles. Dans son rapport, Nielsen (2011) indique qu'en novembre 2010, aux États-Unis, 45% des téléphones achetés dans les six derniers mois étaient des téléphones intelligents et que, pour ce même mois, 31% des téléphones sur le marché états-unien étaient des téléphones intelligents, donc capables d'accéder à Internet et de consulter des flux vidéos sur des réseaux mobiles. L'augmentation du nombre d'utilisateurs et leur soif de contenu de haute qualité accaparent les ressources limitées des opérateurs de réseaux mobiles.

Dans ces conditions, il est crucial d'optimiser l'usage du canal de transmission. Cette optimisation ne vient pas seulement de l'amélioration des taux de compression, quoique la norme H.264 permette d'obtenir des taux de compression considérablement supérieurs aux normes antérieures, mais vient aussi de l'amélioration des mécanismes de résilience aux erreurs. L'amélioration de la résilience aux erreurs diminue la charge sur un réseau en réduisant le nombre de retransmissions, lorsque celles-ci sont envisageables. La vidéophonie sur des appareils mobiles est une des conditions où la retransmission n'est pas envisageable. De plus, la charge accrue sur les réseaux mobiles peut, elle aussi, limiter l'efficacité de la

retransmission. La retransmission peut aussi mener à des délais qui ne sont pas acceptables pour l'application vidéo envisagée. La vidéoconférence en est un parfait exemple.

La particularité du réseau mobile est qu'il est particulièrement vulnérable à l'erreur. Il est souvent considéré comme le maillon faible dans la chaine de transmission de flux vidéos ou de la vidéophonie. Le parcours des données vidéos du réseau Internet jusqu'au réseau de l'opérateur est fait, généralement, sur des réseaux câblés. En ce qui concerne ces pertes encourues lors de ce transport, il n'y a rien à faire, car les données ne se rendent pas à destination. Une fois rendues chez l'opérateur, les données se rendent à la station de base (*base transceiver station*) appropriée à la cellule de l'utilisateur et sont transmises sur le lien sans fil. Cette transmission est très sensible et, comme mentionnée, selon les conditions du réseau mobile, la retransmission sur ce lien n'est pas toujours possible ni envisageable. Lorsqu'il y a corruption sur le lien sans fil, les données se rendent à l'appareil mobile, mais un certain nombre de bits sont altérés par l'interférence. Dans ce cas, l'ensemble du paquet est rejeté, ceci représente un gaspillage important d'information utile, car les bits non altérés du paquet sont eux aussi supprimés. N'ayant pas la possibilité de retransmission, ne serait-il pas mieux d'exploiter l'information du paquet, même si une partie de celle-ci est corrompue ?

Pour résoudre ce problème, il faut changer de paradigme. La puissance de traitement n'est plus uniquement disponible dans les serveurs colossaux des opérateurs, mais aussi répartie à travers les appareils mobiles sur le réseau. Ceux-ci sont de plus en plus puissants, comme en témoignent les spécifications techniques du iPhone 4, avec son processeur d'un gigahertz et sa mémoire vive de 512 mégaoctets.

Il est possible de tirer profit de cette nouvelle puissance computationnelle pour reconstruire les données corrompues lors du transport sur un réseau peu fiable. Contrairement à l'encodeur qui cherche à extraire la redondance d'une séquence, ce nouveau type d'algorithme cherche à utiliser la redondance des images ou des portions d'images décodées pour reconstruire ce qui a été endommagé, et ce, sans l'ajout, apriori, de données redondantes de la part de l'encodeur.

Motivations

Un algorithme capable d'identifier et de dissimuler la détérioration visuelle issue des erreurs de transport pourrait transformer l'écosystème mobile. Non seulement serait-il capable de réduire la charge sur les réseaux mobiles, mais il pourrait aussi augmenter la qualité visuelle perçue par l'utilisateur dans des contextes où la retransmission est impraticable, comme c'est le cas en vidéophonie.

De plus, cet algorithme pourrait remplacer en partie certains mécanismes de résilience à l'erreur qui requièrent une quantité considérable de bande passante, tels la retransmission ou l'ajout de trames redondantes. Sans pour autant éliminer ces méthodes, les opérateurs, sachant que le décodeur est plus résilient à l'erreur, pourraient réduire considérablement la quantité de bits alloués à la résilience lors du transport.

Notons aussi qu'une solution qui ne requiert des altérations qu'au décodeur fait en sorte qu'elle est compatible avec la quantité incommensurable de contenu vidéo déjà encodé, et évite d'avoir à tout réencoder pour assurer la compatibilité avec un nouveau mécanisme.

Objectifs

Notre effort de recherche vise à déterminer, tout d'abord, s'il est possible de décoder avec succès (sans plantage [1]) une séquence corrompue et, si oui, quelle en est la probabilité. Par la suite, lorsque ce genre de décodage est réussi, nous cherchons à identifier les caractéristiques de la détérioration visuelle engendrée par le train de bits corrompu traité par le décodeur. Puis, nous évaluerons les possibilités de dissimuler cette détérioration visuelle.

Notre objectif ultime est de concevoir et de développer un algorithme capable de détecter et de dissimuler les dégradations visuelles. Cet algorithme doit satisfaire aux exigences logicielles suivantes :

1. Selon les auteurs, le terme plantage est considéré comme usuel, familier ou appartenant à l'argot des informaticiens. Des termes plus neutres, mais aussi plus généraux, comme incident et panne, ont été proposés antérieurement ou sont encore parfois employés dans une langue plus soutenue. Ils ne rendent toutefois pas l'idée première contenue dans crash (Office québécois de la langue française, 2011).

– L'algorithme doit identifier la détérioration visuelle importante résultant du décodage d'une trame corrompue.
– L'algorithme doit dissimuler la détérioration visuelle importante résultant du décodage d'une trame corrompue, dans le but d'améliorer la qualité visuelle de la trame.
– L'algorithme doit interagir uniquement avec le décodeur, et ce seulement dans le domaine des pixels, soit à la suite du décodage de la trame.

Organisation du mémoire

Dans cet ouvrage, les informations sont logiquement regroupées en chapitres. La chronologie des chapitres permet de présenter les notions élémentaires en premier, suivies de l'état de l'art, de la solution proposée et des résultats des simulations servant à mesurer l'efficacité de notre solution.

Les deux premiers chapitres présentent les notions de base requises pour une bonne compréhension de la norme H.264 et des mécanismes pour en effectuer le transport sur des réseaux peu fiables. Un lecteur avec ces connaissances peut les esquiver et poursuivre sa lecture au chapitre trois.

Le troisième chapitre fait état des conséquences du décodage de séquences H.264 corrompues. On y présente les répercussions de la corruption d'un encodage entropique, les différents types de détérioration visuelle ainsi que les sources de propagation de cette détérioration. Ce chapitre présente un contenu rare, de par le fait que, dans la majorité des cas, les paquets erronés sont rejetés et non pas décodés.

Au quatrième chapitre, nous passons en revue la littérature portant sur la détection de la détérioration visuelle dans les images et les séquences vidéos. Pour ce faire, nous présentons aussi l'évolution des efforts de recherche dans les domaines connexes, tels l'analyse syntaxique de l'encodage vidéo et le décodage conjoint source-canal (*Joint source channel decoding (JSCD)*).

Le cinquième chapitre présente la solution proposée. Tout d'abord, nous présentons une mesure de la détérioration visuelle, basée sur les effets de blocs compensés par le mouvement. Par la suite, nous présentons deux approches sélectives capables d'utiliser notre mesure pour guider les choix de dissimulation.

Au sixième chapitre, nous présentons les résultats de nos simulations dans le but de mesurer l'efficacité des approches proposées. Finalement, au dernier chapitre, nous concluons cet ouvrage.

CHAPITRE 1

NORME H.264

La norme de compression vidéo H.264, aussi connue sous le nom de MPEG-4 AVC, suscite un grand intérêt autant commercial que théorique, vu ses gains en compression et les technologies de pointe qui la composent. Cette norme, approuvée en mai 2003, raffine les technologies de ses prédécesseurs MPEG-2 et H.263. De plus, elle intègre plusieurs innovations avant-gardistes. En 2011, H.264 est une norme incontournable pour les applications vidéos tels le Blu-Ray, la télévision sur IP et la lecture vidéo en transit (*streaming*). Dans ce chapitre, nous présentons sommairement H.264 ainsi que les rudiments de l'encodage vidéo.

1.1 Notions élémentaires de la vidéo numérique

Avant de présenter les notions de blocs et de macroblocs, définissons une image numérique comme un ensemble fini d'éléments d'image, nommés pixels, contraction de la locution anglaise *picture element*. Un pixel est une unité de surface indivisible qui permet d'afficher une couleur (comme illustré à la figure 1.1). La variation de l'intensité des composantes d'un pixel permet d'afficher l'ensemble de couleurs contenues dans un espace colorimétrique. Le choix de ce dernier détermine les composantes, leur nombre et la portée de leur valeur. Un espace colorimétrique est un modèle mathématique permettant l'expression numérique d'un ensemble de couleurs.

L'espace colorimétrique RVB est très populaire, son nom est l'acronyme des trois couleurs qui le composent : rouge, vert et bleu. Cependant, les composantes de cet espace colorimétrique sont fortement corrélées avec la luminance (intensité lumineuse) ; une augmentation de celle-ci occasionnera une augmentation des valeurs des composantes RVB.

L'espace colorimétrique utilisé par la norme H.264 se nomme YC_BC_R, Y représente la luminance, et C_B et C_R sont, respectivement, les différentiels de Y par rapport au bleu et

Figure 1.1 Visualisation des pixels à l'intérieur d'une image numérique tirée de la suite d'images *Kodak Lossless True Color Image* rendue disponible par Eastman Kodak Company (1999).

au rouge, comme illustré à la figure 1.2. Les composantes C_B et C_R sont aussi connues sous le nom de composantes chromatiques, c'est-à-dire, relatives aux couleurs. La séparation de la luminance et des composantes chromatiques réduit la corrélation entre les composantes de cet espace colorimétrique.

(a) Lena (b) Composante Y (c) Composante C_B (d) Composante C_R

Figure 1.2 Composantes de l'espace colorimétrique YC_BC_R de l'image Lenna.

Notons deux faits intéressants, que nous pouvons observer grâce à la figure 1.2. Premièrement, dans une image naturelle, il y a souvent beaucoup moins de variation dans les composantes chromatiques que dans la composante Y. Deuxièmement, le système visuel humain est beaucoup plus sensible aux variations de luminance qu'à celles de C_B et C_R (Wang et al., 2001). Ces deux constats sont à la base du sous-échantillonnage chromatique. Cette

approche vise à conserver plus d'échantillons Y que d'échantillons chromatiques. La norme H.264 définit plusieurs ratios de sous-échantillonnage chromatique permettant le contrôle du compromis entre la qualité visuelle et le nombre d'échantillons chromatiques à encoder.

Les pixels d'une image naturelle témoignent d'une forte corrélation spatiale. Basées sur cette propriété, certaines opérations d'encodage sont effectuées sur des ensembles quadrilatéraux de pixels, nommés blocs. Dans un encodage par blocs, l'image est séparée en blocs non chevauchants. La taille des blocs varie selon la norme, mais sont généralement des multiples de quatre. Par exemple, H.264 définit les tailles de blocs suivantes : 16×16, 8×8, 16×8, 8×16, 4×4, 8×4 et 4×8. Un bloc de grande taille porte le nom de macrobloc. Dans la norme H.264, un macrobloc est défini comme un bloc de taille 16×16.

Avant d'être séparée en macroblocs, une image est séparée en une ou plusieurs tranches. Une tranche est un regroupement d'un ou plusieurs macroblocs. Les macroblocs qui composent une tranche ne sont pas nécessairement contigus. La particularité d'une tranche est qu'elle est encodée indépendamment des autres tranches de l'image. Quoique cette approche réduise l'efficacité de l'encodage, elle est particulièrement appréciée dans des contextes de parallélisme et de résilience aux erreurs.

Le (*peak signal-to-noise ratio (PSNR)*) est une mesure couramment utilisée pour évaluer la qualité visuelle d'une trame issue d'un codage. Il est exprimé à l'aide d'une échelle logarithmique et utilise le décibel (dB) comme unité. On mesure le PSNR entre une image de référence et une image à évaluer à l'aide de la formule suivante :

$$PSNR = 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right), \tag{1.1}$$

où MAX_I est la valeur maximale d'un pixel, dans le cas d'un pixel représenté par un entier non signé d'un octet, cette valeur est de 255. De plus, on définit l'erreur quadratique moyenne

(*Mean Square Error (MSE)*) :

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [C(i,j) - R(i,j)]^2 , \tag{1.2}$$

où C est la trame à évaluer, tandis que R est la trame de référence, celle utilisée pour évaluer la qualité visuelle. Toutes deux sont de taille $m \times n$.

1.2 Survol de la norme H.264

La norme H.264 offre, pour une fidélité visuelle comparable à MPEG-2, une économie de 50 % du débit (Sullivan et Wiegand, 2005). Cette réalisation n'est pas l'œuvre d'une seule innovation, mais bien de l'effet combiné de plusieurs innovations importantes dans divers aspects de l'encodage vidéo. Pour mieux comprendre cette technologie, décrivons d'abord sommairement les étapes d'encodage vidéo. Par la suite, chaque étape sera expliquée dans les sections subséquentes.

Comme illustré à la figure 1.3, le signal vidéo est divisé en macroblocs. L'efficacité de la prédiction de macroblocs repose sur la forte corrélation spatiotemporelle des pixels qui les composent. Dans un encodage prédictif, ce ne sont pas les pixels du bloc qui sont encodés, mais bien le différentiel entre le bloc et sa prédiction. Le différentiel, moins corrélé, augmente l'efficacité de la compression (Li et Drew, 2004). Néanmoins, le cout lié à l'utilisation du différentiel est l'encodage des données de prédiction. La norme H.264 permet deux types de prédictions : la première, basée sur les blocs d'une autre trame (*inter*) et la seconde, obtenue par l'interpolation du contenu des blocs voisins d'une même trame (prédiction *intra*). La norme prévoit aussi une transformée entière ainsi qu'une quantification, toutes deux appliquées au différentiel de la prédiction. Ces opérations augmentent l'efficacité de l'encodage entropique d'où résulte un meilleur taux de compression. La transformée entière ne fait que réorganiser les valeurs d'un bloc. La quantification est plus importante. Elle est cruciale et permet le contrôle du compromis entre le nombre de bits dédiés à l'encodage d'une

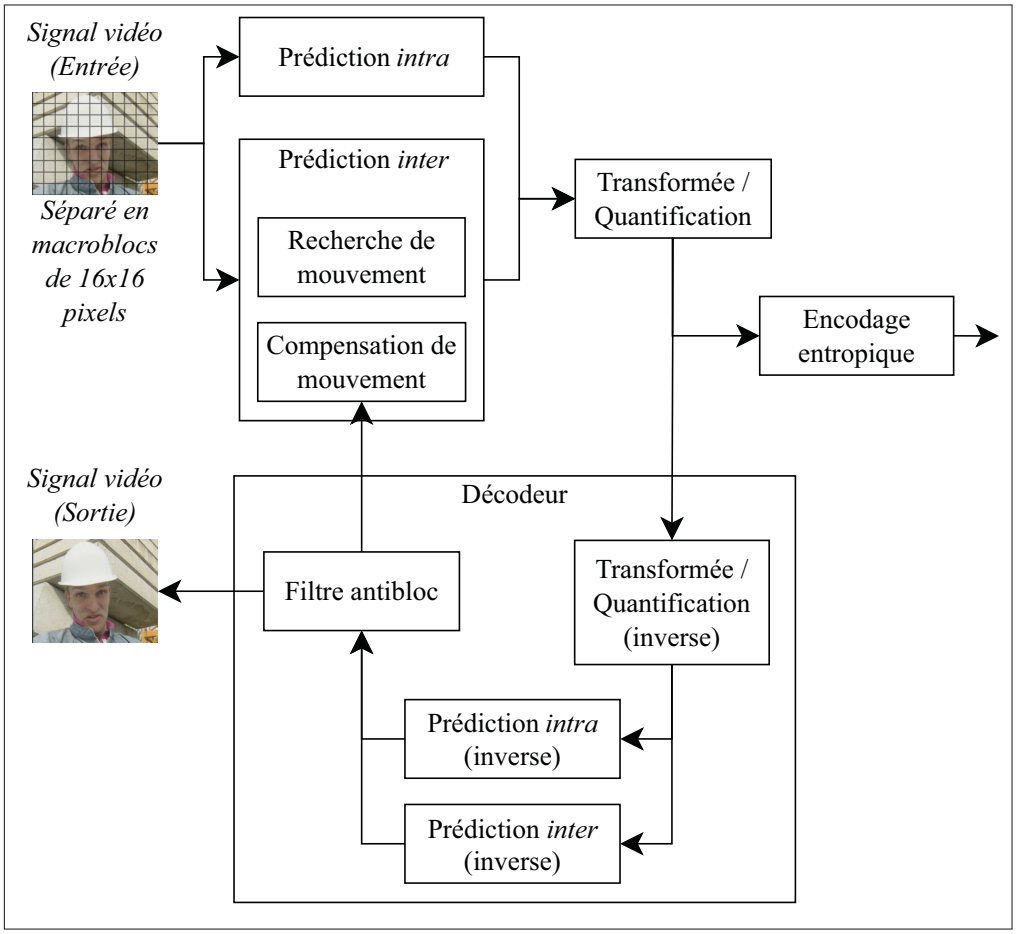


Figure 1.3 Survol des étapes de la norme H.264.
Adaptée de Schäfer et al. (2003, p. 2)

trame (débit) et la qualité visuelle. Finalement, la division du signal vidéo en macroblocs crée des effets de bloc dans l'image, qui sont mitigés à l'aide d'un filtre antibloc (*deblocking filter*).

1.3 Prédiction de macroblocs

1.3.1 Prédiction de macroblocs *inter*

Une source importante de redondance exploitée par la norme H.264 est la redondance inter-image, souvent appelée *inter*. La prédiction *inter* crée un modèle de prédiction basé sur une ou plusieurs trames vidéo préalablement décodées (Richardson, 2003). Ce modèle de prédiction est fondé sur la recherche et la compensation de mouvement dans le but d'accomplir l'appariement de blocs entre deux trames.

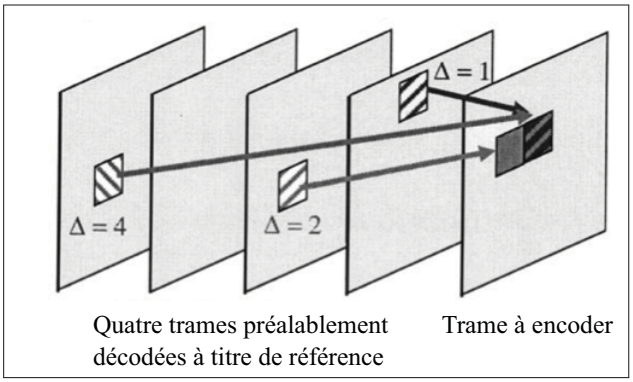


Figure 1.4 Vecteurs de mouvement provenant de quatre trames vidéo préalablement décodées, utilisées pour encoder les blocs dans la trame courante.
Adaptée de Wiegand et al. (2003, p. 570)

Un macrobloc contient 16×16 valeurs de luminances qui peuvent être décomposées de quatre manières distinctes : une partition 16×16, deux partitions 16×8, deux partitions 8×16 et quatre partitions 8×8. De plus, chacun des sous-macroblocs 8×8 peut également être décomposé : deux partitions 8×4, deux partitions 4×8, et quatre partitions 4×4. Le tout est résumé à la figure 1.5.

La décomposition de macroblocs permet d'optimiser l'encodage selon le niveau de détails d'une surface. Un macrobloc 16×16 est efficace pour les régions lisses où le différentiel de la prédiction possède peu d'énergie, ou pour de grandes zones où le mouvement est uniforme et translationnel. Des partitions plus petites sont utiles pour des régions plus complexes où

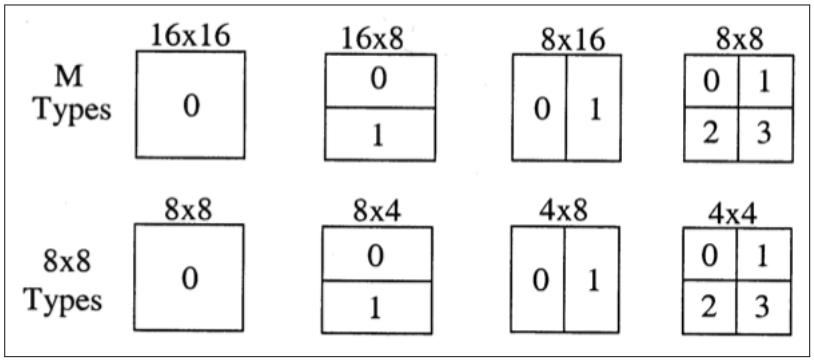


Figure 1.5 Partitions de macroblocs et de sous macroblocs.
Tirée de Wiegand et al. (2003, p. 569)

le différentiel entre la prédiction et le macrobloc à encoder est élevé. La décomposition en partitions permet d'utiliser plusieurs vecteurs de mouvement (un par partition) pour mieux modéliser la surface. Cependant, les vecteurs de mouvement supplémentaires doivent, eux aussi, être encodés, ce qui décourage l'usage inutile de petites partitions.

Lorsque le nombre de partitions est élevé, encoder les vecteurs de mouvement de chacune d'elles peut engendrer un cout binaire considérable et particulièrement à bas débit. C'est pourquoi, dans la norme H.264, les vecteurs de mouvement sont encodés différentiellement à l'aide de la médiane ou d'une prédiction directionnelle guidée par les blocs voisins (Wiegand et al., 2003). Cette prédiction est efficace, car la redondance spatiotemporelle cause une forte corrélation entre les vecteurs de mouvement de blocs avoisinants. Les blocs utilisés pour la prédiction sont des blocs préalablement décodés, qui appartiennent à la même tranche que le bloc à prédire.

On peut remarquer, à la figure 1.3 (p. 10), que la trame de référence n'est pas utilisée pour la recherche de mouvement, mais bien la trame décodée. Suite à l'encodage d'une trame, l'encodeur procède au décodage, afin d'obtenir une trame identique à celle que possèderait un décodeur. Cette approche améliore la précision des prédictions, car elle tient compte des artéfacts dus à l'encodage. Ceci explique pourquoi la recherche de vecteurs de mouvement est effectuée dans les trames préalablement décodées.

Soit (u,v) le vecteur de mouvement, issu de la recherche de mouvement à l'intérieur d'une surface $[-p,p]\times[-p,p]$, entre le bloc à encoder et le bloc le plus fidèle obtenu avec

$$(u,v) = \arg \min_{(u,v)\in[-p,p]\times[-p,p]} \mathrm{SAD}(u,v,\mathbf{C},\mathbf{R}), \tag{1.3}$$

où la somme de la différence absolue (SAD) d'un bloc de taille $M\times N$ à la position (x,y) dans la trame courante $\mathbf{C}$, par rapport à une trame de référence $\mathbf{R}$, se définit comme étant

$$\mathrm{SAD}(u,v,\mathbf{C},\mathbf{R}) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} \left|\mathbf{C}_{x+k,y+l} - \mathbf{R}_{x+k+u,y+l+v}\right| . \tag{1.4}$$

En supposant que le bloc à encoder et le bloc résultant de la recherche de vecteurs de mouvement soient identiques, il ne suffirait que d'encoder (u,v) et le numéro de la trame de référence (Δ, voir figure 1.4 (p. 11)) pour représenter le bloc à encoder. Cependant, il en est rarement ainsi ; il est souvent nécessaire d'encoder le différentiel entre le bloc prédit et le bloc à encoder.

De plus, (u,v) n'est pas nécessairement composé d'entiers (le déplacement des objets ne se fait généralement pas par des pas de pixels entiers). La norme H.264 permet l'utilisation de vecteurs de mouvement offrant un degré de précision au quart de pixel. Deux approches d'interpolation distinctes sont utilisées pour le demi et le quart de pixel. Le demi-pixel **b**, dans la figure 1.6, est obtenu à l'aide de l'équation suivante provenant de l'application d'un filtre à réponse impulsionnelle finie :

$$\mathbf{b} = \mathrm{round}\left(\frac{E-5F+20G+20H-5I+J}{32}\right) . \tag{1.5}$$

Dans l'équation précédente, round arrondit à l'entier le plus proche. De plus, E, F, G, H, I et J sont des positions des pixels illustrées à la figure 1.6.

À la figure 1.6, les lettres majuscules représentent des valeurs de pixel en position entière ; les minuscules sont des valeurs au demi-pixel, sauf j qui est au quart de pixel. Pour les valeurs au

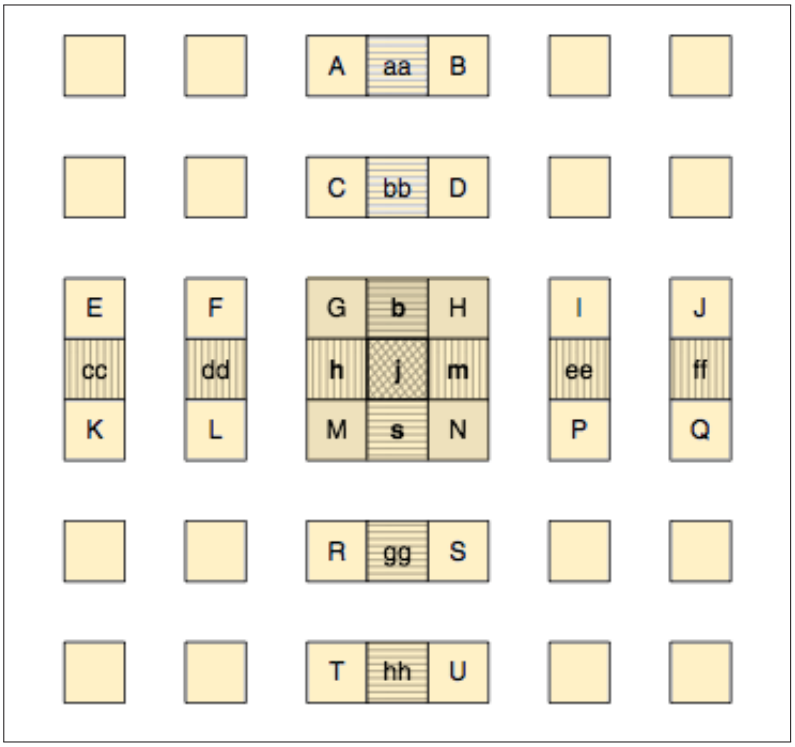


Figure 1.6 Interpolation au demi-pixel.
Adaptée de Richardson (2003, p. 173)

quart de pixel, l'interpolation linéaire de valeurs au demi-pixel est utilisée. Soit **a**, une valeur au quart de pixel située entre G et **b** dans la figure 1.6. On obtient **a** en effectuant

$$\mathbf{a} = \text{round}\left(\frac{G+\mathbf{b}}{2}\right). \tag{1.6}$$

Le filtre à réponse impulsionnelle finie de l'équation 1.5 ainsi que l'interpolation linéaire précédente sont utilisés seulement pour la luminance. Les composantes chromatiques sont obtenues à l'aide d'une interpolation bilinéaire, comme illustré à la figure 1.7.

Pour obtenir la composante chromatique de **a**, on a recourt à

$$\mathbf{a} = \text{round}\left(\frac{(8-d_x)\cdot(8-d_y)A + d_x\cdot(8-d_y)B + (8-d_x)\cdot d_y C + d_x\cdot d_y D}{64}\right). \tag{1.7}$$

Où A, B, C, D sont les valeurs chromatiques des pixels entiers à une distance (d_x, d_y) entourant la valeur à interpoler.

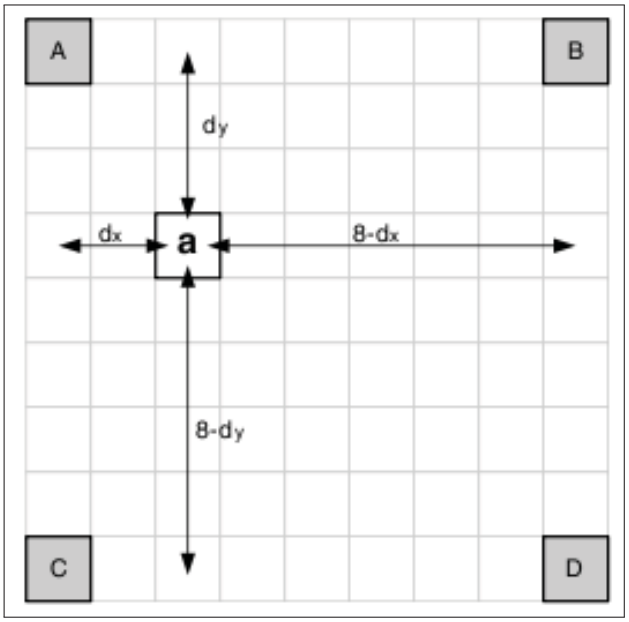


Figure 1.7 Interpolation de la composante chromatique au huitième de pixel.
Adaptée de Richardson (2003, p. 175)

1.3.2 Prédiction de macroblocs *intra*

Une seconde source considérable de redondance exploitée par la norme H.264 est la redondance spatiale à l'intérieur d'une image. La prédiction *intra* exploite cette redondance pour modéliser la texture d'un bloc à partir celle des blocs avoisinants.

Comme c'est le cas pour la prédiction *inter*, la prédiction *intra* repose sur le concept de blocs et de macroblocs et comprend la notion de blocs à tailles variables visant à améliorer la prédiction de régions complexes. La norme H.264 permet à l'encodeur de choisir entre des macroblocs 16×16 ou des sous-blocs 4×4 pour la luminance. Selon le choix, différents modes de prédiction sont offerts.

Deux faits intéressants sont à noter. Premièrement, la prédiction *intra* est accomplie dans le domaine spatial (contrairement à H.263 et MPEG-4 Visual qui utilisent le domaine fréquentiel (Wiegand et al., 2003)). Deuxièment, le recours aux pixels de blocs voisins pour établir une prédiction peut mener à une propagation d'erreurs, si celle-ci repose sur des pixels corrompus.

Une prédiction intra précise engendre un différentiel faible et décorélé, ce qui améliore le taux de compression issu des étapes subséquentes de l'encodage. Les gains obtenus par l'usage de la prédiction justifient le cout de l'encodage des données de la prédiction supplémentaire.

La norme H.264 définit neuf modes de prédiction liés à l'utilisation de blocs 4×4, illustrés à la figure 1.8. Huit de ces modes sont des extrapolations directionnelles à des angles de 26.6°, 45° et 90°, tandis que le mode 2, aussi connu sous le sigle DC, est la moyenne des pixels A à D et I à L (voir figure 1.8) préalablement encodés.

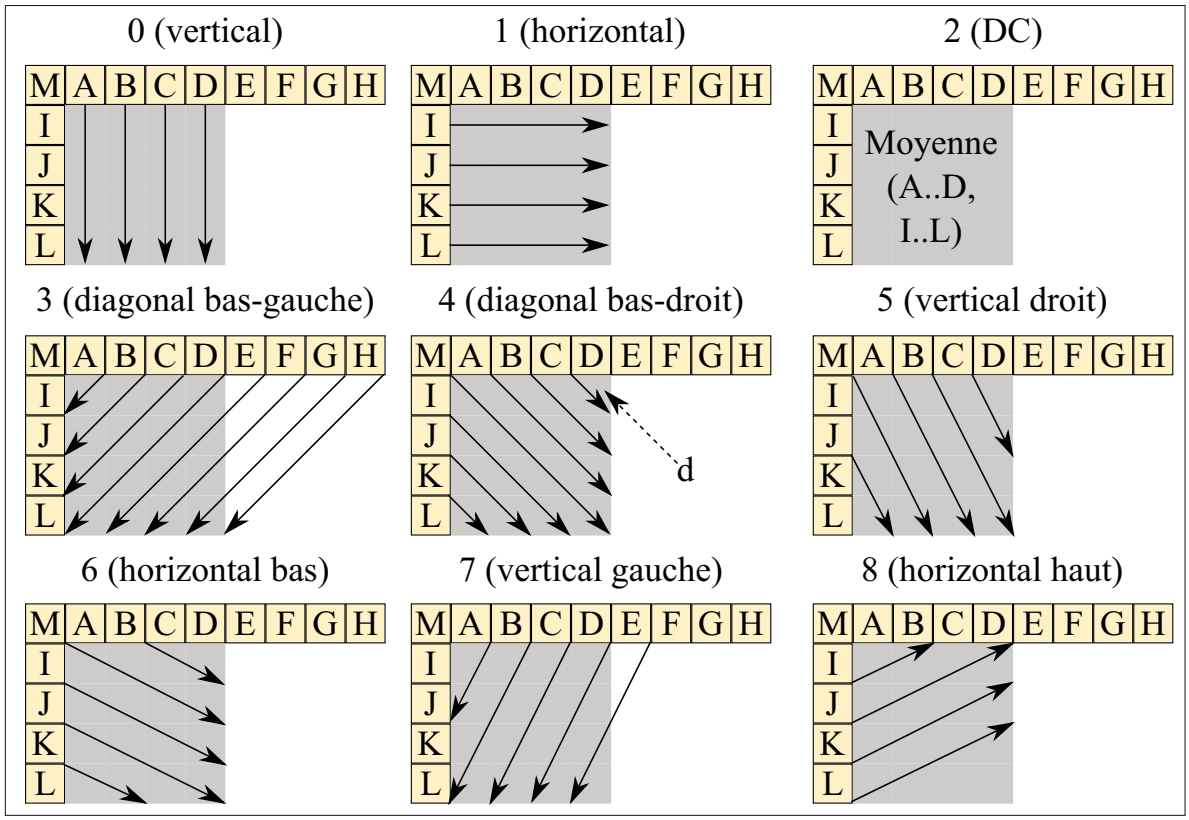


Figure 1.8 Les modes prédictifs pour la luminance de blocs 4×4.
Adaptée de Richardson (2003, p. 179)

Nous pouvons, par exemple, définir la valeur de **d**, le pixel situé dans le coin supérieur droit du bloc 4×4 prédit par le mode 4 (diagonale bas-droit). On obtient **d** à l'aide de la formule suivante :

$$\mathbf{d} = \text{round}\left(\frac{B+2C+D}{4}\right). \tag{1.8}$$

La figure 1.10 illustre la prédiction obtenue pour chaque mode offert pour un bloc 4×4 ainsi que la somme de l'erreur absolue (SAE) obtenue entre la prédiction et le bloc à encoder. Dans cet exemple, le meilleur mode de prédiction serait 8, car il offre le plus petit SAE, soit 203.

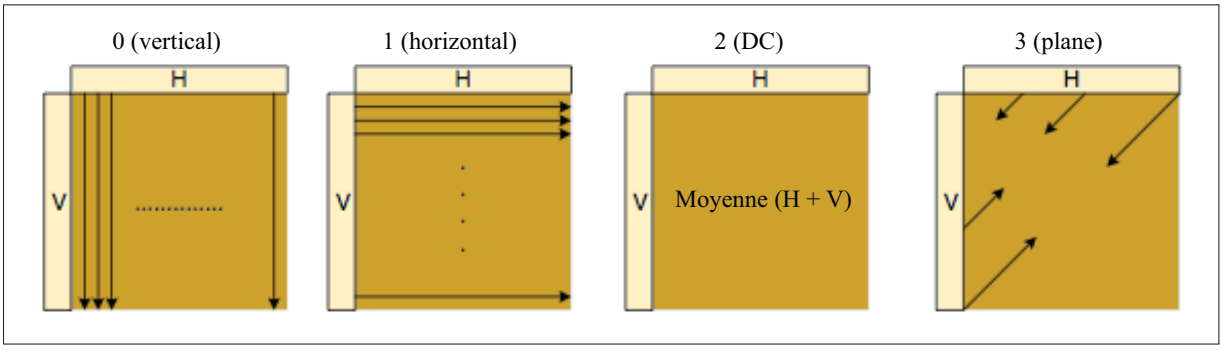
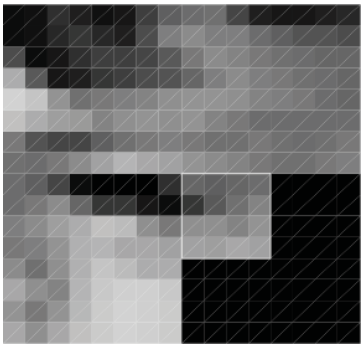


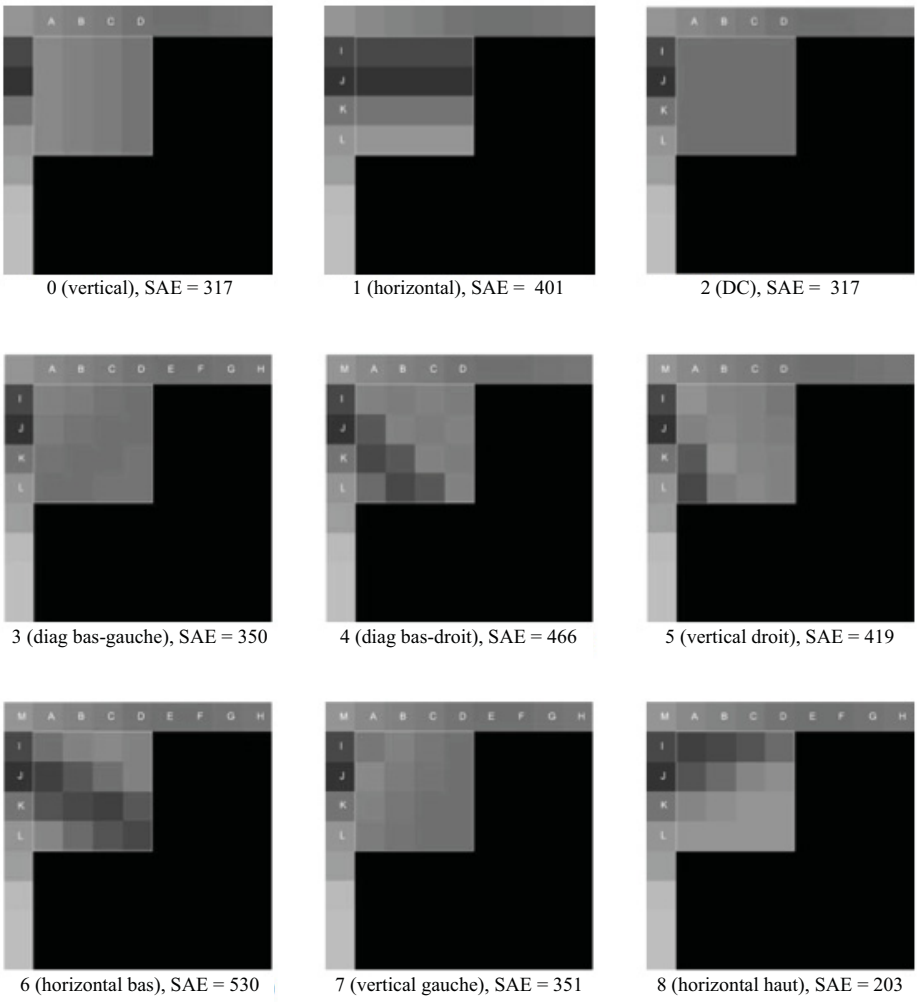
Figure 1.9 Modes de prédiction de la luminance de blocs 16×16.
Adaptée de Richardson (2003, p. 181)

En ce qui à trait à la prédiction des macroblocs 16×16, la norme H.264, en prévoit quatre modes : horizontal, vertical, DC et plane, tel qu'illustré à la figure 1.9. Le mode DC utilise la moyenne des pixels limitrophes horizontaux et verticaux, tandis que les trois autres approches sont des extrapolations. Contrairement aux 4×4, les modes de prédictions 16×16 sont destinés aux surfaces lisses avec peu de variation d'énergie. Ceci explique pourquoi le nombre de modes est restreint et que ceux-ci sont plus simples.

En ce qui concerne les composantes chromatiques, elles se regroupent dans un bloc 8×8, vu le sous-échantillonnage chromatique. Les mêmes types de prédictions que pour les blocs 16×16 sont employés soient : DC, horizontal, vertical et plane. Ceci est dû au fait que dans les images naturelles, les composantes chromatiques varient beaucoup moins que la luminance (Wang et al., 2001), ce qui en facilite la prédiction.



(a) Bloc 4×4 à prédire.



(b) Résultats de prédiction *intra* 4×4.

Figure 1.10 Comparaison des SAE issues des neuf modes de prédiction *intra* 4×4. Adaptée de Richardson (2003, p. 181)

1.4 Transformée entière

Les approches présentées jusqu'à présent cherchent à éliminer la redondance spatiotemporelle d'une séquence vidéo par l'utilisation de prédictions. Ces prédictions reposent sur les corrélations intrinsèques aux données d'une séquence vidéo. Ces prédictions sont imparfaites et requièrent que le différentiel entre la prédiction et le bloc à encoder soit, lui aussi, encodé. Toutefois, ce différentiel, appelé erreur résiduelle, possède une forte autocorrélation. La transformée entière définie dans la norme H.264 a pour objectif de décoreler spatialement l'erreur résiduelle afin d'en faciliter l'encodage. La matrice de transformation suivante :

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{1.9}$$

possède des propriétés similaires à une transformée en cosinus discrète (DCT) 4×4 (Malvar et al., 2003), qui est très prisée pour son aptitude à décorreler un ensemble de données.

La transformée entière possède plusieurs avantages par rapport à la DCT. D'une part, elle est plus simple et requiert seulement des additions et des décalages binaires (*bit shift*). D'autre part, son résultat étant entier implique qu'il n'y a pas de pertes fractionnaires (perte de précision) lors de la transformée inverse.

H.264 applique sa transformée sur des blocs 4×4. Ceci la distingue de ses prédécesseurs qui utilisent des blocs 8×8. L'utilisation de blocs plus petits est justifiée par les améliorations substantielles de la précision des prédictions *intra* et *inter*, qui réduisent considérablement le résiduel à transformer. De plus, l'utilisation de blocs plus petits réduit grandement le bruit autour des bordures (souvent appellé *ringing* ou *musquito noise*). L'ordre de balayage des différentiels de blocs 4×4 est présenté à la figure 1.11.

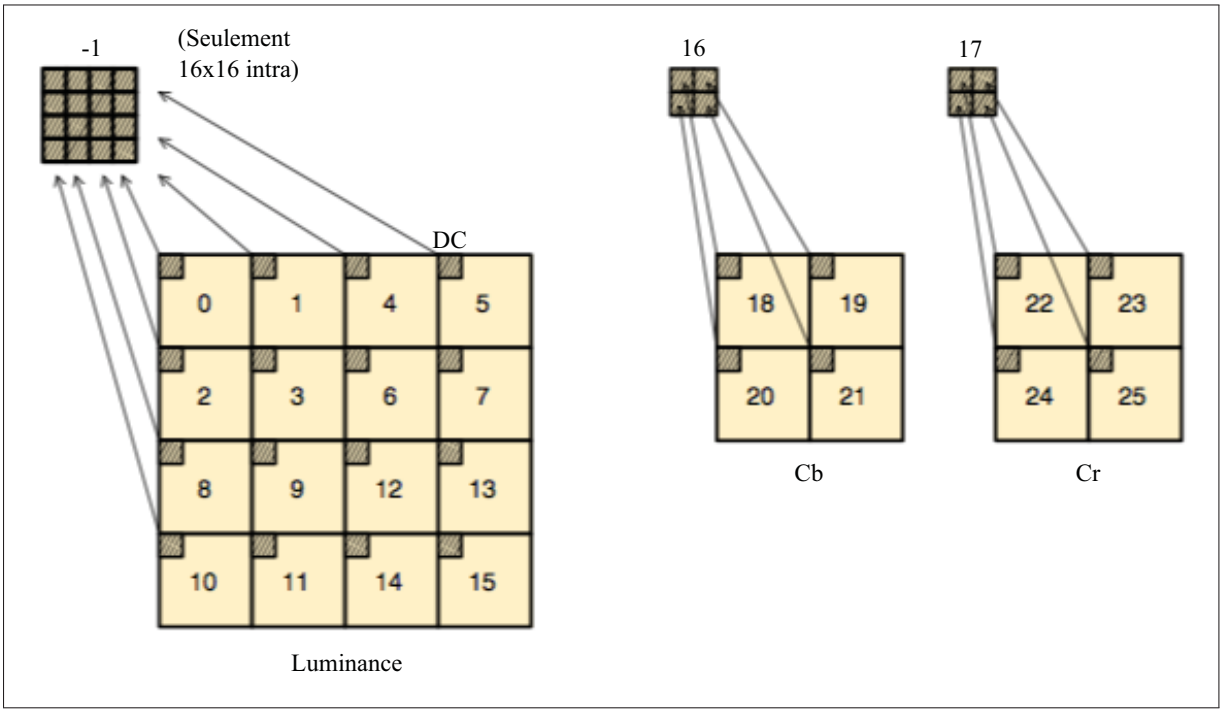


Figure 1.11 Ordre de balayage des coefficients de blocs à l'intérieur d'un macrobloc de l'erreur résiduelle lors de la transformée entière 4×4.
Adaptée de Richardson (2003, p. 189)

1.5 Quantification

Une nouveauté de la norme H.264 est qu'elle définit un paramètre de quantification qui exerce un contrôle logarithmique sur le pas de quantification alors que ses prédécesseurs utilisaient un contrôle linéaire. Le quantificateur est scalaire et peut varier pour la luminance et les composantes chromatiques. La valeur quantifiée Z de la valeur transformée W à la position i, j est défini comme étant :

$$Z_{ij} = \text{round}\left(\frac{W_{ij}}{Qstep}\right), \tag{1.10}$$

où $Qstep$ est le pas de quantification et round la fonction d'arrondissement afin d'obtenir des résultats entiers. La norme définit 52 valeurs de pas de quantification indexées par le paramètre de quantification (QP). Une augmentation de six de ce dernier double le pas de quantification. Ce grand nombre de valeurs permet à l'encodeur une meilleure précision, en ce qui a trait au compromis entre le débit binaire et à la qualité visuelle.

1.6 Encodage entropique

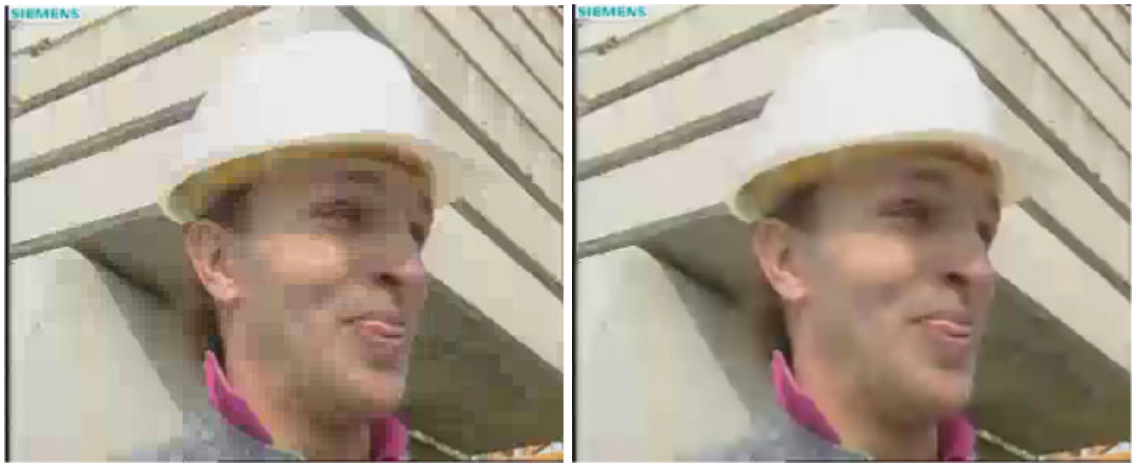
La norme H.264 innove dans le domaine du codage entropique en séparant le codage des paramètres de celui des données. Les paramètres sont représentés par des codes numériques qui sont, eux-mêmes, encodés avec des codes à longueur variable appelés codes exponentiels Golomb. Les valeurs des différentiels de prédiction sont encodées soit avec un codage entropique à longueur variable (CAVLC) ou un codage arithmétique binaire à contexte adaptatif (CABAC). Bien que CABAC offre un gain de compression d'environ 10 % par rapport à CAVLC, ce dernier est prédominant dans les applications mobiles à cause de sa simplicité. C'est pour cette raison que dans cet ouvrage, nous concentrons nos efforts exclusivement sur le CAVLC.

Un encodage à longueur variable, comme celui présenté par Huffman (1952), assigne des codes plus courts aux données plus fréquentes. Ceci implique que des codes plus longs sont attribués aux autres données. Cependant, la faible occurrence de longs codes assure que, comparé à un encodage à taille fixe, l'encodage à taille variable requiert moins de bits pour encoder la séquence.

Dans un encodage à longueur variable, un bit erroné engendre une désynchronisation beaucoup plus importante que pour un encodage à longueur fixe. Le décodeur ne connaissant pas la longueur des codes, l'erreur se propage jusqu'au prochain point de synchronisation prévue par la norme.

1.7 Filtre antibloc

Le filtre antibloc, mieux connu sous son nom anglophone *deblocking filter*, a pour objectif de réduire l'apparence d'effets de bloc (variation importante de la valeur des pixels en bordure de blocs). L'effet de bloc est l'effet secondaire le plus important produit par les algorithmes de compression vidéo présents dans la norme H.264. Il est le produit d'une discontinuité spatiale provenant de variations d'encodage de blocs adjacents.



(a) Sans filtre antibloc (b) Avec filtre antibloc

Figure 1.12 Exemple de l'application du filtre antibloc sur une trame fortement compressée. Adaptée de Schäfer et al. (2003, p. 7)

Le filtre antibloc agit en bordure de blocs dans le but de lisser les variations importantes d'intensités des valeurs de pixels. La figure 1.12 démontre l'efficacité du filtre antibloc appliqué sur une trame fortement compressée. Beaucoup plus qu'un simple filtre, le filtre antibloc tient compte du pas de quantification ainsi que du type de prédiction utilisé pour encoder les blocs, afin d'ajuster l'intensité du lissage.

La norme H.264 définit deux seuils qui régissent le comportement du filtre antibloc. Soit α pour l'écart entre les pixels en bordure de deux blocs :

$$|p_0 - q_0| < \alpha(\text{Indice}_A), \tag{1.11}$$

et β pour l'écart entre les pixels à la bordure d'un même bloc :

$$|p_1 - p_0| < \beta(\text{Indice}_B), \tag{1.12}$$

$$|q_1 - q_0| < \beta(\text{Indice}_B), \tag{1.13}$$

où p_i et q_i sont les pixels en bordure de blocs tel qu'illustré à la figure 1.13. L'encodeur peut

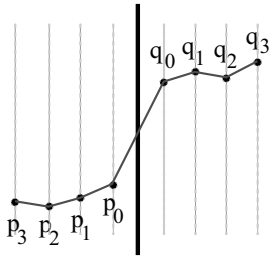


Figure 1.13 Visualisation unidimensionnelle d'une bordure de bloc qui requiert l'intervention du filtre antibloc. Adaptée de List et al. (2003, p. 616)

biaiser les indices à l'aide d'un décalage tel que

$$\text{Indices}_A = \min(\max(0, \text{QP} + \text{Decalage}_A), 51), \tag{1.14}$$

$$\text{Indices}_B = \min(\max(0, \text{QP} + \text{Decalage}_B), 51)\,. \tag{1.15}$$

La plage de valeur de 0 à 51 représente les valeurs possibles du paramètre de quantification (QP). Cela dit, aux formules 1.14 et 1.15, QP est le paramètre de quantification moyen entre les deux blocs évalués. Les valeurs des seuils α et β sont définies à l'aide des formules suivantes :

$$\alpha(x) = 0.8(2^{x/6} - 1), \tag{1.16}$$

$$\beta(x) = 0.5x - 7\,. \tag{1.17}$$

Afin de savoir si un lissage doit être appliqué et, si oui, à quelle intensité, le paramètre d'intensité de la bordure (bS) est défini selon des règles basées sur le mode d'encodage. Celles-ci sont résumées dans le tableau 1.1.

Comme le démontre le tableau 1.1, le type de prédiction et les conditions d'encodage produisent différents degrés d'effet de bloc. C'est pourquoi il est crucial que le filtre soit ajustable. De plus, les seuils α et β permettent de prendre en compte la texture propre à l'image afin de ne pas lisser des variations de valeurs de pixels propres à l'image situés en bordure de blocs.

Tableau 1.1 Règles guidant l'intensité du filtre antibloc en fonction des conditions d'encodages.

Condition d'encodage	**Intensité**
Prédiction *intra* et en bordure de macrobloc	4 (Lissage fort)
Prédiction *intra*	3
Prédiction *inter* et présence de résiduel	2
Différence des vecteurs de mouvement $\geqslant 1$	1
Prédiction *inter* avec des trames de références différentes	1
Sinon	0 (Aucun lissage)

L'usage du filtre antibloc permet, pour un même niveau de qualité visuelle (mesuré avec le PSNR), d'obtenir une réduction du débit de 5 à 10 % selon le contexte (List et al., 2003). L'ajout, à la norme H.264, du caractère obligatoire de ce mécanisme différencie cette dernière de ses prédécesseurs. Quoique plusieurs utilisent des filtres antibloc, peu les rendent obligatoires. Son ajout dans la chaine d'encodage décodage fait en sorte que ses améliorations sont prises en compte lors des prédictions inter-image.

Dans ce chapitre, nous présentons, sommairement, la norme de codage H.264 ainsi que les notions élémentaires de la vidéo numérique. Ces acquis permettront de mieux comprendre les concepts, plus avancés, des prochains chapitres ; où nous explorerons le transport de séquences H.264 de même que les impacts du décodage de paquets corrompus.

En conclusion, nous pouvons résumer l'encodage H.264 comme une opération visant à retirer la redondance présente dans une séquence vidéo. Cependant, retirer cette redondance amplifie considérablement les impacts de la corruption de cette dernière lors du transport. Des stratégies doivent être mises en place pour protéger les séquences lors du transport, un sujet réservé au chapitre 2 (p. 25). Cette corruption peut même se propager à travers la séquence. Ce phénomène est connu sous le nom de propagation d'erreur, et sera un des impacts des erreurs sur une séquence H.264 présentés au chapitre 3 (p. 36).

CHAPITRE 2

TRANSPORT DE SÉQUENCES H.264

Au chapitre précédent, nous avons décrit sommairement la norme de codage H.264. Dans ce chapitre, il sera question des considérations de transport d'une séquence H.264 sur des réseaux peu fiables. La norme H.264 inclut plusieurs mécanismes permettant d'améliorer la résilience aux erreurs et l'empaquetage d'une séquence vidéo encodée.

Tout d'abord, la notion de tranche, décrite au chapitre précédent, est revisitée à la section 2.1, mais cette fois, dans l'optique du transport sur des réseaux peu fiables. Par la suite, à la section 2.2, nous présentons des algorithmes de dissimulation de paquets perdus. Après quoi, d'autres approches de résilience aux erreurs sont décrites : l'ordonnancement flexible de macroblocs, à la section 2.3, ainsi que les ensembles de paramètres, à la section 2.4. Finalement, aux sections 2.5 et 2.6, des notions en lien avec la réseautique sont présentées, soit la couche d'abstraction réseau et la hiérarchie protocolaire RTP/UDP/IP.

2.1 Tranches

Nous avons déjà mentionné, lors de notre description de la norme H.264, la notion de tranche. On définit une tranche comme un regroupement de macroblocs encodés indépendamment du contenu des macroblocs appartenant aux autres tranches de la trame. En ce qui a trait à la résilience aux erreurs, l'usage de tranches offre deux avantages intéressants : empêcher la propagation d'erreurs et réduire les données manquantes lors de la perte de paquets. La séparation du contenu de la trame en plusieurs paquets fait en sorte que, si un de ceux-ci est perdu, seul le contenu de la tranche est manquant, et non la trame entière. Typiquement, l'assignation des macroblocs à une tranche se fait par un ordonnancement séquentiel de gauche à droite et du haut vers le bas (*raster*), comme illustré à la figure 2.1, où les 99 macroblocs d'une trame QCIF sont regroupés à l'intérieur de trois tranches.

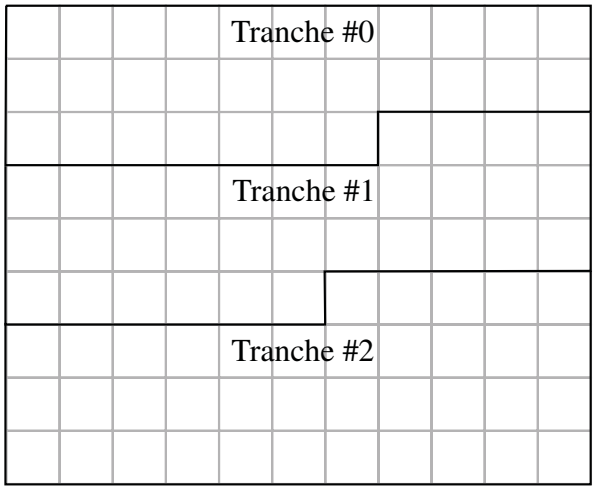


Figure 2.1 Séparation des macroblocs d'une trame QCIF en trois tranches.
Adaptée de Wiegand et al. (2003, p. 556)

Cependant, il y a des désavantages reliés à l'usage des tranches. Premièrement, les résultats des prédictions sont parfois moins précis, vu que ces dernières sont restreintes à utiliser uniquement le contenu de la tranche. Par exemple, le bloc offrant la meilleure prédiction pourrait être à l'extérieur de la tranche. Deuxièmement, les tranches augmentent le ratio de données d'entête par rapport aux données vidéo. Comme présenté à la section 2.6 (p. 33), chaque tranche doit avoir un entête NAL, ainsi que les entêtes des protocoles RTP, UDP et IP.

Le regroupement de macroblocs à l'intérieur de tranches permet aussi d'envoyer et de décoder ces dernières séparément. De plus, la norme H.264 prévoit la notion d'ordonnancement arbitraire des tranches (*Arbitrary Slice Order (ASO)*). L'ASO permet la transmission et le décodage des tranches dans le désordre. Ceci réduit considérablement les délais sur des réseaux où il n'y a pas de garantie de l'ordre de livraison des paquets, tels les réseaux IP (Sullivan et Wiegand, 2005).

2.2 Dissimulation d'erreurs dans le décodeur H.264

La perte de paquets est omniprésente sur les réseaux peu fiables, soit parce que ceux-ci ne se rendent pas à destination dans les délais requis ou parce qu'ils sont endommagés lors du

transport. Le décodeur inclus dans le logiciel de référence H.264/AVC JM offre deux options de dissimulation de tranches manquantes : le calque de la trame [1] et le calque des vecteurs de mouvement [2].

Comme son nom l'indique, le calque de la trame remplace, par la trame qui la précède, une trame corrompue. S'il y a plus d'une tranche par trame, alors le remplacement s'effectue au niveau de la tranche manquante. Tandis que les autres tranches de la trame sont intactes. Cette approche a deux avantages : le premier est sa facilité d'implémentation, et le deuxième est son efficacité lorsqu'il y a peu de variation entre les trames. La figure 2.2 illustre un exemple où la deuxième tranche d'une trame est corrompue ; le décodeur la remplace par la deuxième tranche de la trame qui la précède.

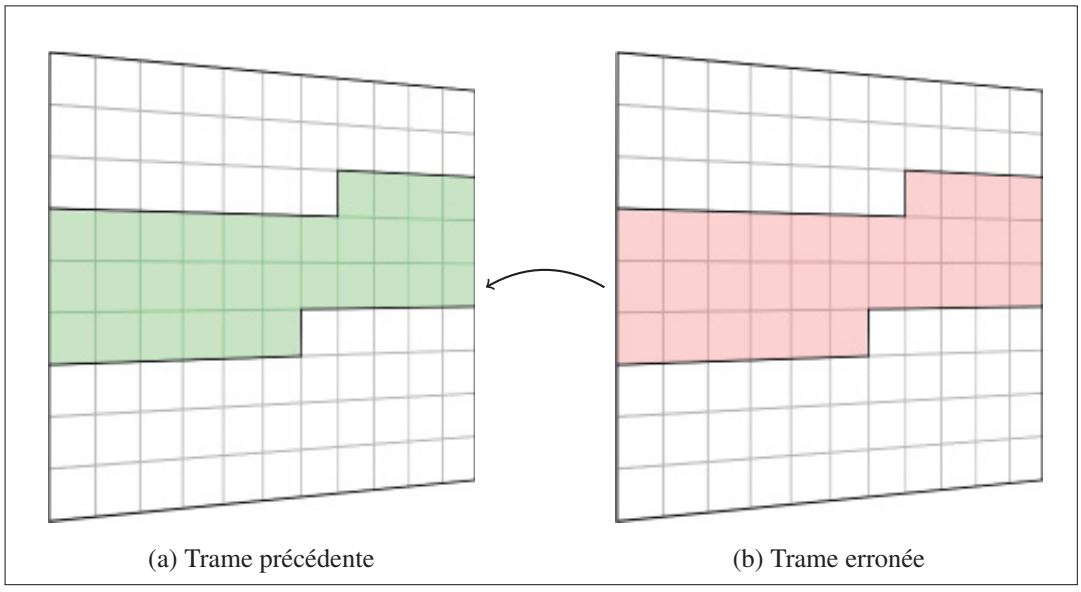
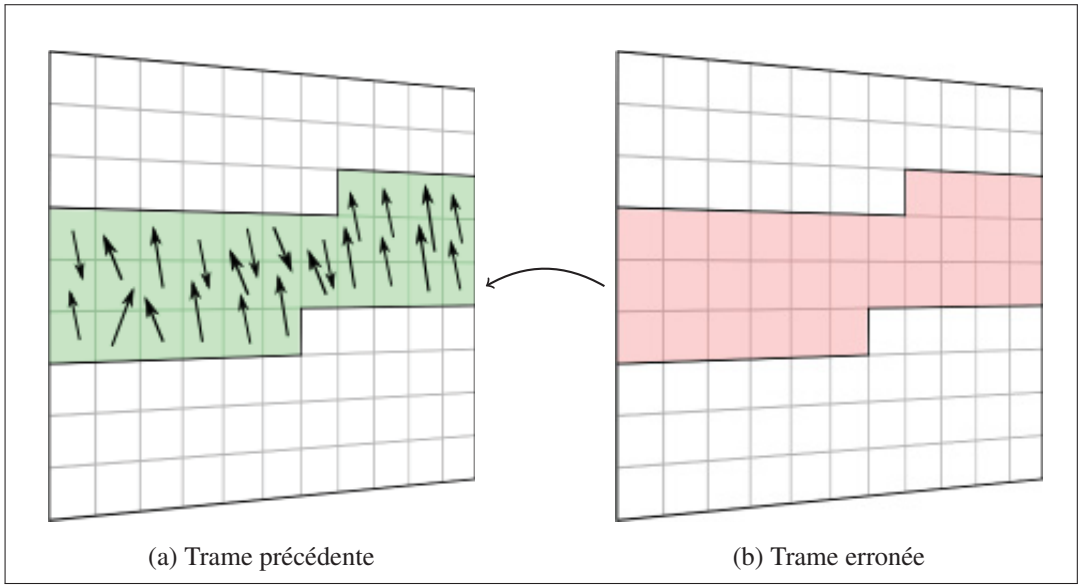


Figure 2.2 Visualisation du calque d'une tranche.

Une autre approche de dissimulation employée par le décodeur inclus dans le logiciel de référence H.264/AVC JM est le calque des vecteurs de mouvement, comme illustré à la figure 2.3. Cette approche repose sur la corrélation temporelle des vecteurs de mouvement, proposée par Wu et Boyce (2006). Lorsqu'une tranche est endommagée, les vecteurs de

1. Traduction employée dans cet ouvrage pour l'expression *slice copy*.
2. Traduction employée dans cet ouvrage pour l'expression *motion copy*.

mouvement de la tranche correspondante, dans la trame précédente, sont réutilisés. Cela a pour effet de perpétuer le mouvement. Selon Wu et Boyce, cette approche est plus efficace que le calque de la trame lorsqu'il y a du mouvement entre la trame précédente et la trame erronée.



(a) Trame précédente (b) Trame erronée

Figure 2.3 Visualisation du calque des vecteurs de mouvement.

2.3 Ordonnancement flexible de macroblocs

Comme mentionné à la section précédente, seules les tranches endommagées sont dissimulées. Pour améliorer l'efficacité de l'algorithme de dissimulation, la norme H.264 prévoit la possibilité de changer l'ordonnancement des macroblocs à l'intérieur d'une trame. Ce concept se nomme ordonnancement flexible de macroblocs (*Flexible Macroblock Ordering (FMO)*). Cette technique permet un meilleur contrôle sur la répartition des macroblocs dans les tranches de sorte à mieux disperser les pertes à travers l'image et, ainsi, à augmenter l'efficacité de la dissimulation.

La figure 2.1 (p. 26) montre l'ordonnancement typique d'une séquence H.264. Il existe plusieurs types d'ordonnancements flexibles. À titre d'exemple, nous présentons aux fig-

ures 2.4a et 2.4b les ordonnancements de macroblocs dispersé et entrelacé. Tous deux illustrent la répartition des macroblocs d'une trame séparée en deux tranches.

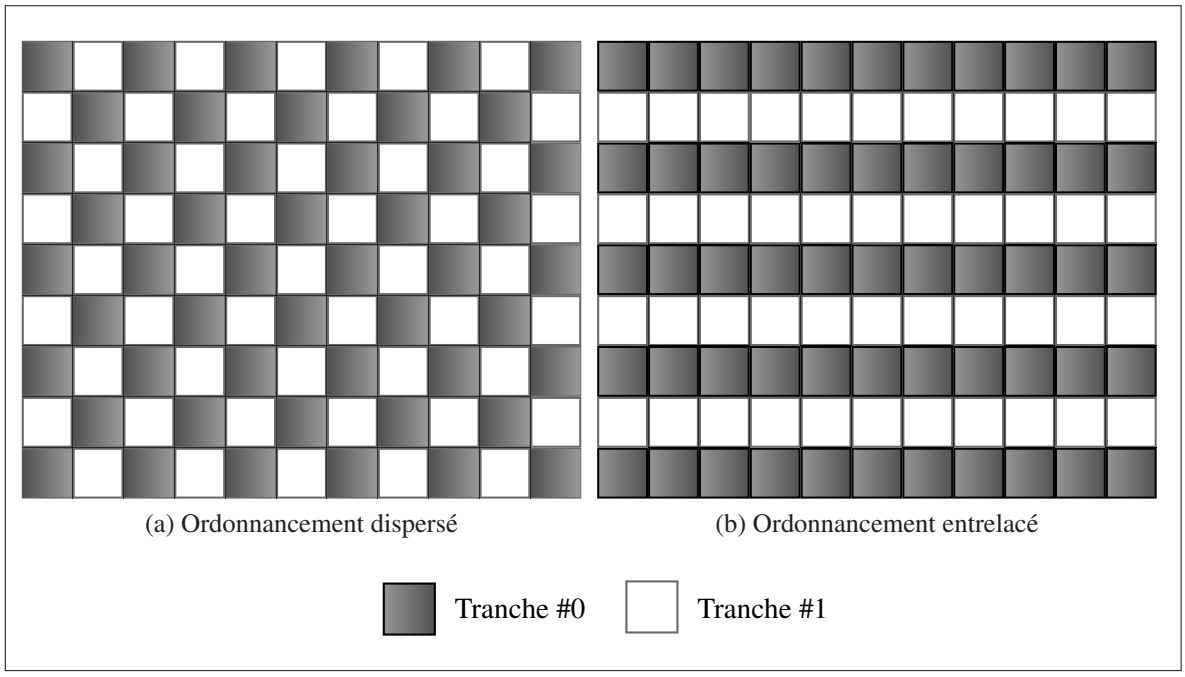


Figure 2.4 Exemples d'ordonnancements flexibles de macroblocs pour des trames QCIF séparées en deux tranches.

L'ordonnancement dispersé (2.4a) ressemble à un damier, où les cases alternent d'une tranche à une autre. Ceci fait en sorte que les quatre voisins d'un macrobloc sont des macroblocs provenant d'autres tranches. Un exemple de la détérioration visuelle issue du décodage d'une tranche corrompue contenue dans une trame avec un ordonnancement de type dispersé est présenté à la figure 2.5.

Pour sa part, l'ordonnancement entrelacé (2.4b) regroupe les macroblocs en rangées. Celles-ci sont séquentiellement assignées aux tranches de la trame. Lorsqu'utilisé avec le calquage de tranche, cet ordonnancement permet de conserver la corrélation spatiale horizontale des macroblocs, au détriment de la corrélation spatiale verticale. Dans le contexte de notre exemple, une telle approche de dissimulation fait en sorte que la moitié des rangées de macroblocs de la trame proviennent de la trame précédente. Un exemple de la détérioration

(a) Trame erronée (b) Détérioration visuelle (amplifiée)

Figure 2.5 Exemple de la détérioration visuelle issue du décodage d'une tranche corrompue contenue dans une trame avec un ordonnancement de type dispersé.

visuelle issue du décodage d'une tranche corrompue contenue dans une trame avec un ordonnancement de type entrelacé est présenté à la figure 2.6.

(a) Trame erronée (b) Détérioration visuelle (amplifiée)

Figure 2.6 Exemple de la détérioration visuelle issue du décodage d'une tranche corrompue contenue dans une trame avec un ordonnancement de type entrelacé.

2.4 Ensembles de paramètres

La notion d'ensembles de paramètres (*Parameter sets*) augmente considérablement la résilience aux erreurs, lors du transport d'une séquence H.264. Ces améliorations sont obtenues par la séparation des paramètres d'encodage du contenu encodé. En ce qui concerne les normes antérieures, les paramètres de codages sont souvent inclus dans les mêmes paquets que le contenu encodé, ce qui augmente la taille des paquets, les rendant ainsi plus susceptibles d'être erronés. De plus, l'information répétée dans chaque paquet en augmente considérablement les risques de corruption. Grâce à cette séparation, il est possible de transmettre ces paramètres de façon plus fiable, afin de s'assurer leur réception sans erreur. Ils peuvent être aussi transmis plus tôt, afin de compenser les délais de transport. La norme H.264 prévoit deux types d'ensembles de paramètres : les paramètres de séquences et les paramètres d'images. Ceux-ci sont présentés à la figure 2.7.

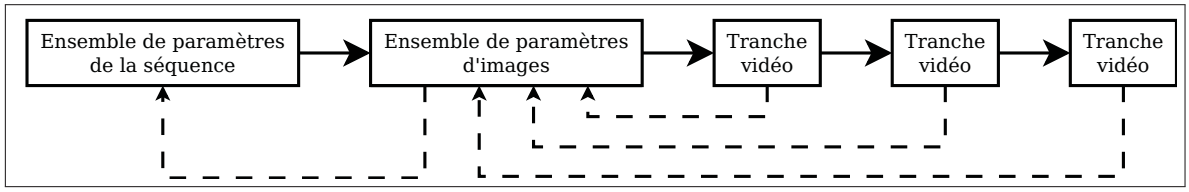


Figure 2.7 Liens entre les tranches, les ensembles de paramètres d'images et ceux de la séquence.
Adaptée de Superiori et Nemethova (2006, p. 53)

Premièrement, il y a les ensembles de paramètres de la séquence. Ceux-ci sont des paramètres généraux qui régissent la séquence dans son ensemble. Parmi les paramètres de la séquence, on retrouve, notamment, la largeur et la hauteur, en pixels, de la séquence ainsi que le nombre de trames de référence requises pour le décodage.

Deuxièmement, il y a les ensembles de paramètres d'images qui régissent une ou plusieurs trames de la séquence. Parmi les paramètres d'images, on retrouve, notamment, un identificateur des paramètres de séquences, auxquels sont associés les paramètres d'images, un autre permettant d'identifier l'encodage entropique utilisé et un autre, le paramètre de quantification.

2.5 Couche d'abstraction réseau

La norme H.264 sépare les notions de codage vidéo de celles du transport en deux couches distinctes, comme illustré à la figure 2.8. La couche de codage vidéo (*Video Coding Layer (VCL)*) est conçue pour être indépendante du canal de transmission. Cette couche est présentée en détail dans notre chapitre sur H.264 (p. 6). La couche d'abstraction réseau (*Network Abstraction Layer (NAL)*) a pour responsabilité d'adapter le contenu de la VCL selon les caractéristiques du canal de transmission afin d'être empaquetée dans des unités NAL. Grâce à cette couche, la norme H.264 peut être adaptée au transport sur les réseaux de type : RTP/UDP/IP, H.324M, MPEG-2 TS et H.320 (Wenger, 2003).

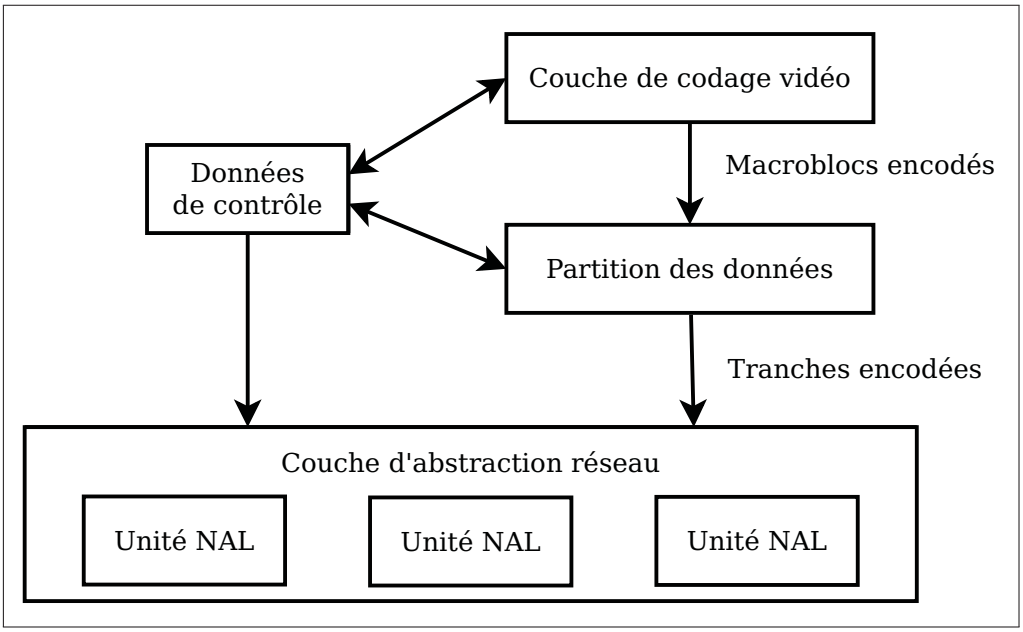


Figure 2.8 Couches conceptuelles de la norme H.264.
Adaptée de Superiori et Nemethova (2006, p. 52)

Une unité NAL (*NAL unit*) est le nom donné à un paquet logique dont le but est de contenir les données provenant de la VCL. Une unité NAL combine un entête NAL et une séquence d'octets utiles (*Raw Byte Sequence Payload (RBSP)*). Ces octets sont ceux issus de l'encodage d'une séquence H.264.

2.6 Hiérarchie protocolaire RTP/UDP/IP

La hiérarchie protocolaire majoritairement utilisée pour la consultation de flux vidéo (*streaming*) et la vidéophonie est RTP/UDP/IP (Wenger, 2003). De par son nom, cette hiérarchie identifie les protocoles utilisés aux couches trois, quatre et cinq des sept couches du modèle OSI (Zimmermann, 1980), illustré à la figure 2.9. De plus, à la figure 2.10, nous présentons l'encapsulation d'une unité NAL pour la hiérarchie protocolaire RTP/UDP/IP. Le nombre d'octets requis par chaque entête y est aussi indiqué. Ceci révèle le fardeau binaire lié à l'encapsulation, soit 40 octets.

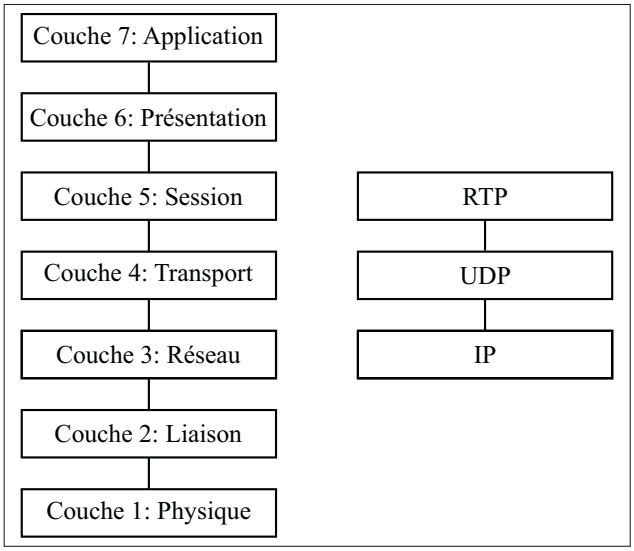


Figure 2.9 Hiérarchie protocolaire RTP/UDP/IP et les couches du modèle OSI. Adaptée de Smith et Collins (2007, p. 339)

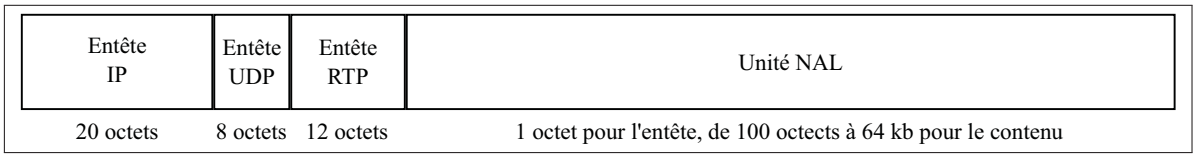


Figure 2.10 Données d'entêtes des protocoles IP, UDP, RTP pour l'encapsulation d'une unité NAL.

Le plus grand avantage à utiliser le protocole Internet (*Internet Protocol (IP)*) est son omniprésence dans le domaine des télécommunications (Smith et Collins, 2007). Ce protocole

est supporté, non seulement par l'ensemble des ordinateurs modernes, mais aussi par une panoplie d'appareils portables, tels les tablettes, les téléphones intelligents ainsi que les décodeurs numériques. Proprement dit, le protocole IP permet d'acheminer des paquets d'un routeur à un autre, à travers le réseau, en direction de la destination appropriée, définie par l'adresse de destination IP contenue dans l'entête IP. Cependant, le protocole IP ne fournit aucune protection contre la perte ou la corruption de paquets. Notons aussi que le protocole IP ne fournit pas d'information sur l'ordre de reconstruction des paquets. Pour ce faire, il faudra avoir recours à des protocoles de couches supérieures.

Le protocole de *datagramme* utilisateur (*User Datagram Protocol (UDP)*) se situe à la couche quatre du modèle OSI. Il régit un service, simple, de transport de *datagrammes* sans pour autant en garantir la bonne livraison. Contrairement à d'autres protocoles de la couche transport, on y note l'absence d'un mécanisme permettant la validation des *datagrammes* reçus par le destinataire et leur retransmission au besoin. Ce mécanisme n'étant pas souhaitable dans nos conditions de transport, car il ajouterait un fardeau additionnel au réseau et dans certains cas, ne permettrait pas de respecter la contrainte *temps réel* (Wenger, 2003). Toutefois, une somme de contrôle (*checksum*) est incluse dans l'entête UDP. Elle permet de valider la fidélité du *datagramme* reçu. Cependant, notons toujours l'absence d'information sur l'ordre de reconstruction des paquets.

Au dessus d'UDP, on retrouve le protocole de transmission en temps réel (*real-time transport protocol (RTP)*). Il établit et maintient une session entre un émetteur et un ou plusieurs destinataires. L'entête de ce protocole est composé d'informations tels le numéro de séquence, l'estampille temporelle et le type de données utilisées. Ces informations viennent boucler la boucle en ce qui concerne le transport de vidéo sur des réseaux peu fiables. Le numéro de séquence, incrémenté de un pour chaque paquet de la session, permet d'identifier les paquets perdus. L'estampille temporelle permet la synchronisation entre l'émetteur et le destinataire. Les informations sur le type de données utilisées permettent l'identification de la norme de codage des données, tel H.264.

Ce chapitre sur le transport de séquences H.264, regroupe un grand nombre de concepts appartenant à deux domaines distincts. D'une part, les notions liées à la norme de codage H.264, tels les tranches, la dissimulation d'erreurs, l'ordonnancement flexible de macroblocs, les ensembles de paramètres et la couche d'abstraction réseau. D'autre part, les notions de réseautique, tels les protocoles IP, UDP et RTP. Même lorsque tous ces concepts sont employés, il est fort probable que du contenu vidéo transmis sur un canal de communication peu fiable soit endommagé. Au prochain chapitre, nous étudions cette erreur et les conséquences de son décodage, soit la détérioration visuelle voire même le plantage du décodeur.

CHAPITRE 3

DÉTÉRIORATION VISUELLE SUITE AU DÉCODAGE DE SÉQUENCES VIDÉO H.264 CORROMPUES

Jusqu'à présent, nous avons présenté la norme H.264 ainsi que les mécanismes lui permettant de mieux résister aux pertes encourues lors de son transport sur des réseaux peu fiables. L'approche typique adoptée pour réduire ces pertes est d'augmenter la redondance. Cette dernière peut se manifester sous plusieurs formes, tels l'encodage de trames redondantes ou la retransmission de paquets manquants. Toutefois, cette redondance accroit le nombre de bits requis pour le transport d'une séquence. Mais en pratique, pour plusieurs réseaux mobiles, la bande passante est restreinte et, pour accommoder le nombre grandissant d'utilisateurs ou pour respecter les délais *temps réel*, ces mécanismes ne peuvent pas être employés.

Dans ce chapitre, il sera question des notions reliées au décodage de séquences H.264 corrompues et à la détérioration visuelle qui peut en résulter. À la section 3.1, nous expliquons l'impact de la corruption de bits sur l'encodage entropique. Par la suite, la détérioration visuelle, issue du décodage de trames *inter* corrompues, est expliquée à la section 3.2. Finalement, nous présentons, à la section 3.3, les différentes formes de propagations d'erreurs.

3.1 Décodage de paquets corrompus

Vu la situation précédemment décrite, une solution intéressante pour la résilience aux erreurs, qui n'augmente pas le fardeau des réseaux mobiles, est d'améliorer l'efficacité de la dissimulation d'erreurs effectuée par le décodeur. Le chapitre précédent fait mention des techniques de dissimulation d'erreurs incorporées au décodeur inclus avec le logiciel de référence H.264/AVC JM (p. 26). Ces techniques présupposent que l'ensemble du paquet est corrompu. Ce nivèlement par le bas est partiellement dû à l'architecture en couche utilisée pour l'interconnexion des systèmes, décrite par Zimmermann (1980) et présentée dans cet ouvrage, au chapitre 2 (p. 33). Cette architecture sépare, dans deux couches distinctes, la réception des

paquets et le décodage de leur contenu. Ceci a plusieurs avantages, mais l'inconvénient majeur est de restreindre l'interaction entre les couches. Par exemple, lorsque l'on détermine au niveau d'UDP qu'un paquet est corrompu, ce dernier est rejeté. Ceci empêche une application située aux couches supérieures, tel le décodeur H.264, d'avoir accès aux paquets corrompus.

Le *Joint source-channel decoding* (Duhamel et Kieffer, 2010) est une approche qui vise à établir une meilleure interaction entre les couches en combinant la logique de la source (c.-à-d. la norme H.264) à l'analyse du contenu erroné d'un paquet, afin de déterminer l'emplacement de l'erreur. Cette localisation de l'erreur fait en sorte qu'au lieu de dissimuler l'ensemble du contenu du paquet, la dissimulation serait effectuée seulement sur des codes contenant les bits corrompus. Dans le cadre de la norme H.264, un algorithme de détection d'erreurs d'une telle précision n'a pas encore été développé.

Une des raisons de la complexité de la détection d'erreurs dans des données codées provient du fait que le codage cherche à éliminer la redondance. Cette absence de redondance complexifie grandement la détection des erreurs. Commençons par regarder le train de bits produit par un codage entropique à taille variable, tels CAVLC ou Exp-Golomb. Dans ce genre de codage, une erreur peut complètement changer le message, comme illustré à la figure 3.1 (voir tableau 3.1 pour les codes). Ceci est dû au fait qu'altérer des bits change les codes et donc, leurs tailles et, par conséquent, altère les codes subséquents jusqu'au prochain marqueur de synchronisation. Un marqueur de synchronisation est une suite unique de bits servant à réinitialiser la lecture de codes à taille variable.

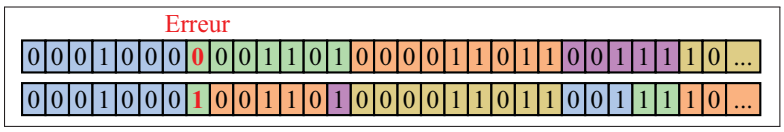


Figure 3.1 Désynchronisation du train de bits.
Adaptée de Ikuno (2007, p. 11)

Tableau 3.1 Codes exponentiels Golomb.
Adaptée de Ikuno (2007, p. 11)

Code	Code Exp-Golomb
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001

Cette désynchronisation peut mener à la désynchronisation du décodeur, voire même, à faire planter [1] ce dernier. D'une part, le décodeur plante suite à une défaillance causée par la lecture de codes invalides. D'autre part, si le décodeur ne plante pas, le décodage sera désynchronisé, c'est-à-dire que de mauvais codes seront utilisés jusqu'au prochain marqueur de synchronisation. Lorsque le décodeur reconstruit la trame avec ces mauvais codes, ceci peut occasionner de la détérioration visuelle, comme celle présentée à la figure 3.2. C'est le cas à la figure 3.2d, où la détérioration visuelle n'est pas toujours évidente pour l'utilisateur.

En ce qui concerne cette détérioration visuelle, elle est le sujet de la prochaine section. Cependant, pour plus d'information sur la résilience du décodeur, inclus dans le logiciel de référence H.264/AVC JM, face à la désynchronisation des codes entropiques, veuilliez vous référer à la section 6.3 «Analyse de la résilience aux erreurs du décodeur de référence H.264» (p. 81).

1. Selon les auteurs, le terme plantage est considéré comme usuel, familier ou appartenant à l'argot des informaticiens. Des termes plus neutres, mais aussi plus généraux, comme incident et panne, ont été proposés antérieurement ou sont encore parfois employés dans une langue plus soutenue. Ils ne rendent toutefois pas l'idée première contenue dans crash (Office québécois de la langue française, 2011).

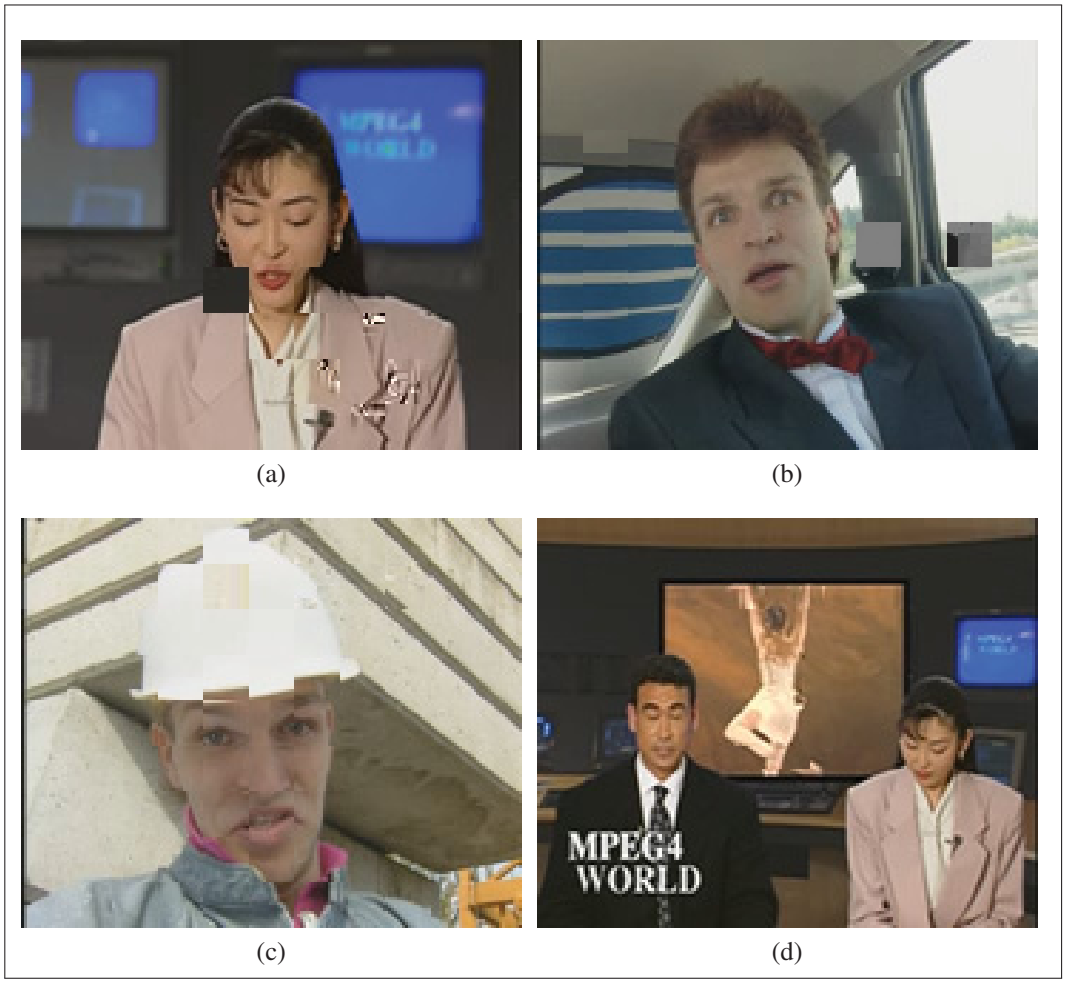


Figure 3.2 Exemples de la détérioration visuelle issue du décodage désynchronisé de paquets corrompus.

3.2 Détérioration visuelle

Ici, notre analyse de la détérioration visuelle porte uniquement sur celle issue du décodage de trames *inter*, car c'est le type de détérioration visuelle détectée par la solution proposée dans cet ouvrage. La désynchronisation d'un bloc *inter* peut survenir dans une ou plusieurs de ces composantes : la trame de référence, le vecteur de mouvement ou le résiduel.

D'une part, la détérioration visuelle engendrée par la corruption de la trame de référence ou du vecteur de mouvement est très similaire, c'est-à-dire que le contenu du bloc ne provient pas du bon endroit. Il provient soit de la mauvaise trame de référence ou du mauvais

emplacement dans la bonne trame. Cette détérioration cause souvent des effets de bloc issus de la discontinuité spatiale entre le contenu des bons blocs et celui du bloc erroné, comme illustré à la figure 3.3.

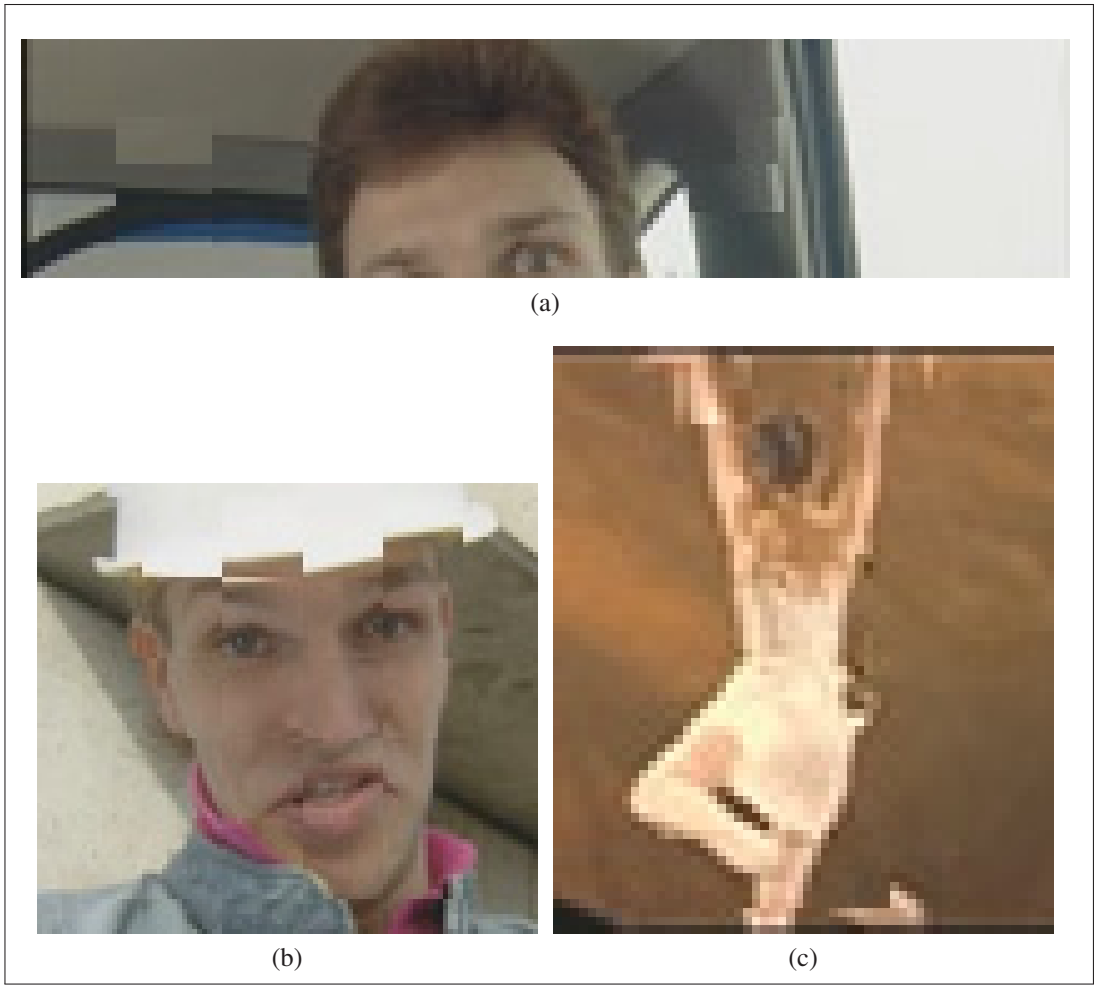


Figure 3.3 Exemples de la détérioration visuelle issue de la corruption de la trame de référence ou du vecteur de mouvement.

D'autre part, le résiduel corrompu produit un effet secondaire très différent de celui issu la corruption de la trame de référence ou du vecteur de mouvement. Il s'agit, ici, d'un bloc qui n'a pas l'apparence d'une image naturelle. Ceci est dû à la corruption des indices transformés qui se produit dans le domaine de la transformée. Le résultat de la transformée inverse des indices corrompus est un bloc d'une seule couleur ou avec une texture non naturelle, comme c'est le

cas à la figure 3.4. Toutefois, cette détérioration produit souvent, elle aussi, une discontinuité spatiale dans l'image.

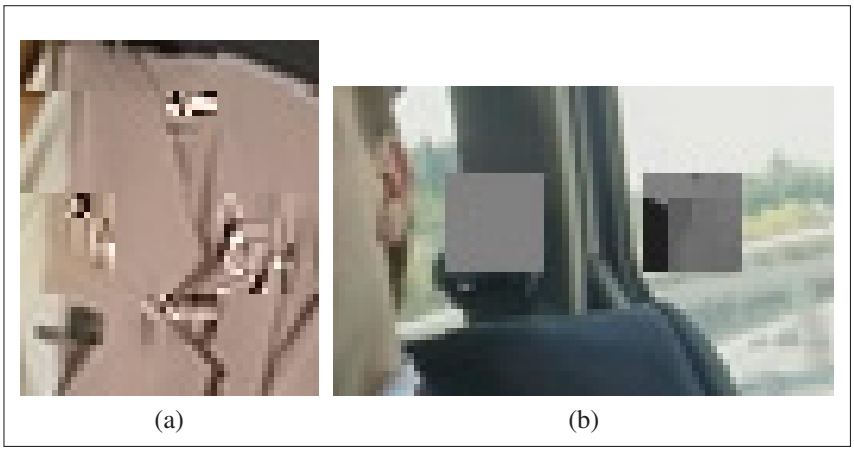


Figure 3.4 Exemples de la détérioration visuelle issue de la corruption du résiduel.

3.3 Propagation de la détérioration visuelle

Comme démontré à la figure 3.1 (p. 37), une erreur binaire peut avoir un impact considérable sur un train de bits, vu la désynchronisation qu'elle engendre en ce qui a trait aux codes entropiques. Néanmoins, cette erreur peut aussi avoir d'autres répercussions, si elle sert à la prédiction d'autres blocs. Ce phénomène se nomme propagation d'erreurs. Il peut survenir autant dans le domaine spatial que temporel.

Lorsque des pixels endommagés servent à la prédiction de trames subséquentes, ceci permet à l'erreur de voyager d'une trame *inter* à une autre. Il s'agit, ici, d'une propagation temporelle de l'erreur. Cette propagation peut se poursuivre jusqu'à la prochaine trame *intra*. La figure 3.5 permet de visualiser la propagation d'erreurs dans le temps.

La propagation spatiale d'erreurs peut survenir de deux façons, soit par les vecteurs de mouvement ou par l'intermédiaire du filtre antiblocs. Les vecteurs de mouvement sont sujets à la propagation d'erreurs, car ils sont prédits. Ce qui fait en sorte qu'un vecteur de mouvement endommagé nuira aux autres vecteurs de mouvement dont la prédiction repose sur celui-ci.

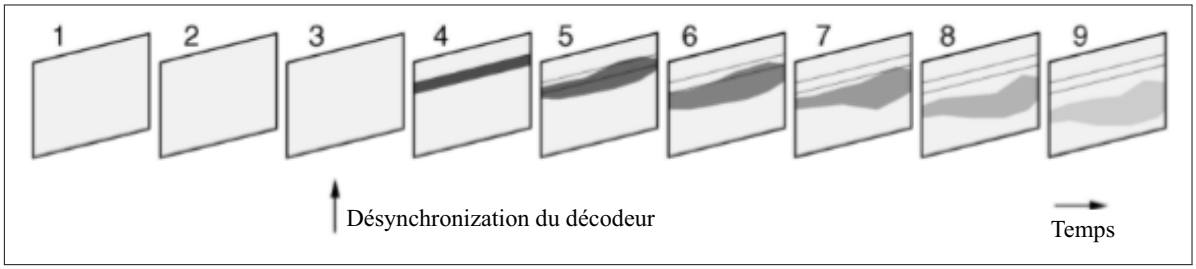


Figure 3.5 Propagation temporelle de l'erreur.
Adaptée de Girod et Färber (1999, p. 1711)

Le filtre antiblocs effectue, malgré lui, une propagation de l'erreur en bordure des blocs lorsqu'il lisse ces dernières, afin d'en réduire les effets de bloc. Le lissage d'une bordure d'un bloc corrompu propage la détérioration visuelle au bloc voisin. Cette propagation est locale et ne s'étend pas plus loin que le lissage, soit jusqu'à deux pixels à l'extérieur d'un bloc corrompu.

Dans ce chapitre, nous avons décrit la détérioration visuelle issue du décodage de paquets corrompus. Typiquement, les paquets corrompus sont rejetés et ce genre de détérioration ne se produit pas. Cependant, pour tirer profit du contenu valide à l'intérieur de paquets corrompus, il faut décoder ces paquets et donc faire face à cette détérioration. Pour ce faire, un nouveau type d'algorithme a été développé, celui capable de détecter et de dissimuler la détérioration visuelle. Avant de présenter l'algorithme proposé (chapitre 5), nous présentons d'abord ceux qui existent déjà (chapitre 4).

CHAPITRE 4

ÉTAT DE L'ART DE LA DÉTECTION DE LA DÉTÉRIORATION VISUELLE DANS LE DOMAINE DES PIXELS

Dans ce chapitre, nous faisons état de la littérature sur les approches de détection de la détérioration visuelle. Il ne s'agit pas, ici, d'une simple énumération, mais bien d'une suite logique d'ouvrages permettant de comprendre l'introduction et l'évolution de plusieurs concepts fondamentaux, sur lesquels reposent nos contributions.

La détection de la détérioration visuelle sert à identifier la dégradation visuelle engendrée par le décodage de paquets corrompus. Pour décoder ces paquets, le décodeur du canal de transmission doit interagir avec le décodeur vidéo (source) afin d'identifier et de rendre disponible les paquets corrompus. Ce type d'interaction porte le nom de *Joint Source Channel Decoding*. Notre prologue, section 4.1, expose les origines du *Joint Source Channel Decoding*. La section 4.2 offre un compte rendu des notions du *Joint Source Channel Decoding* qui s'appliquent spécifiquement au transport de séquences vidéos. Les améliorations apportées à une de ces notions, l'analyse syntaxique, sont présentées à la section 4.3. Finalement, à la section 4.4, les approches combinant l'analyse syntaxique à la détection de la détérioration visuelle dans le domaine des pixels sont décrites.

4.1 Prologue

Dans son ouvrage *A Mathematical Theory of Communication*, Shannon (1948) explique sous l'allégorie ingénieuse du télégraphe, la notion même du *Joint Source Channel Decoding*. L'usage d'un langage redondant, tel l'anglais, comme encodage pour le télégraphe a l'avantage d'être résistant aux erreurs du canal de transmission. Le contexte du message permet la reconstruction des mots, même si plusieurs lettres de ceux-ci sont bruitées (Shannon, 1948, p. 24).

En 1979, Modestino et Daut sont les premières personnes à proposer l'utilisation du *Joint Source Channel Decoding* pour améliorer le transport d'images numériques. Déjà en 1979, Modestino et Daut déplorent le gaspillage de la bande passante, introduit par les stratégies de résilience aux erreurs. Un gaspillage qui, 30 ans plus tard, est toujours présent sur les réseaux sans fil (Duhamel et Kieffer, 2010, p.1).

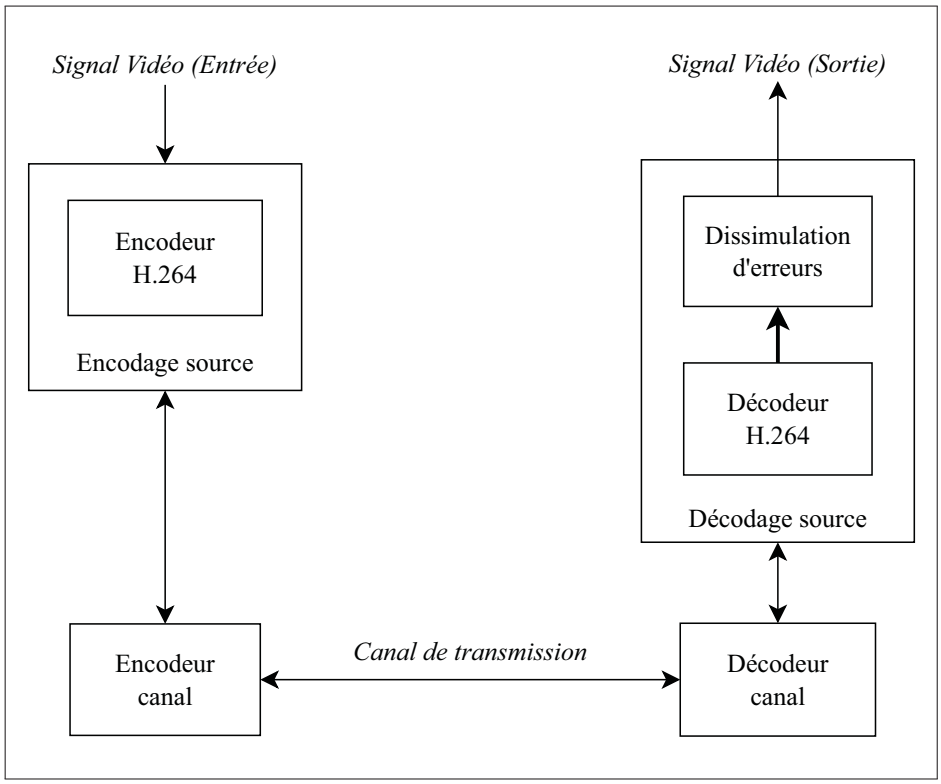


Figure 4.1 Diagramme à blocs des composantes liées au transport de séquences vidéos. Adaptée de Wang et Zhu (1998, p. 976)

Avant de poursuivre, décrivons les composantes liées au *Joint Source Channel Decoding*. La figure 4.1 résume les étapes et les interactions du *Joint Source Channel Decoding*. Tout d'abord, l'encodeur H.264 retire la redondance d'un signal vidéo. Par la suite, l'encodeur du canal va empaqueter le contenu et insérer les mécanismes de résilience à l'erreur nécessaires selon le canal de transmission. Notons l'ajout de données redondantes comme un des mécanismes de résilience à l'erreur. Dès la réception de paquets chez le destinataire, le décodeur du canal dépaquète ceux-ci et valide que leur contenu est intact. À l'aide des données

dépaquetées, le décodeur H.264 reconstruit le signal vidéo. Cependant, si des données sont manquantes ou corrompues, le décodeur H.264 fait appel à un algorithme de dissimulation d'erreurs pour estimer ces dernières.

4.2 Lien entre le décodeur vidéo et celui du canal de transmission

Les techniques de résilience aux erreurs sur des réseaux non fiables ont suscité beaucoup d'intérêt dans les années 1980 et 1990. En 1998, dans leur revue sur les techniques de contrôle et de dissimulation d'erreurs lors du transport de séquences vidéos [1] (Wang et Zhu, 1998), Wang et Zhu récapitulent, entre autres, l'état de l'art, de l'époque, des stratégies de résilience aux erreurs. Wang et Zhu présentent une multitude d'approches. Parmi celles-ci, ils identifient un type d'approches qu'ils qualifient d'approches de dissimulation par post-traitement effectué par le décodeur (Wang et Zhu, 1998, Chap. 5). Ce type résume, de façon exacte, les approches que nous présentons dans cet ouvrage. Leurs composantes sont illustrées à la figure 4.1. Parmi les approches de dissimulation présentées en 1998, on remarque : *Motion-Compensated Temporal Prediction* (Ghanbari, 1993), *Maximally Smooth Recovery* (Wang et al., 1993), *Projection onto Convex Sets (POCS)* (Sun et Kwok, 1995), *Spatial- and Frequency-Domain, Interpolation* (Hemami et Meng, 1995) et (Sun et al., 1992).

Toutes ces approches de post-traitement, effectuées par le décodeur, présentées par Wang et Zhu (1998), supposent que les blocs endommagés sont connus. Pratiquement, cette identification est accomplie en ignorant les paquets corrompus, ce qui force le décodeur à dissimuler tous les blocs que contiennent ces derniers même si l'ensemble des blocs n'est pas endommagé. Pour reprendre l'exemple du télégraphe, ceci revient à rejeter l'ensemble des mots d'une phrase qui possède une lettre en erreur. Cela complique grandement la compréhension du message. Ce nivèlement par le bas, quoique plus simple, est un exemple du gaspillage identifié par Modestino et Daut, où le décodeur de la source ne communique pas avec le décodeur du canal de transport.

1. Traduction de l'auteur du titre *Error Control and Concealment for Video Communication : A Review*.

Le *Joint Source Channel Decoding* (Duhamel et Kieffer, 2010) cherche, à l'aide de la logique de la source, à identifier précisément où se trouve l'erreur dans un ensemble de bits corrompus. Par exemple, pour l'encodage MPEG-4 Visual, Talluri propose des règles permettant de vérifier la validité de la portée des vecteurs de mouvement, des codes entropiques décodés, de la portée des valeurs de coefficient DCT et du nombre de coefficients DCT afin que ces derniers ne dépassent pas le nombre permis par la norme (Talluri, 1998). Une fois l'erreur identifiée, il existe deux types de solutions possibles : la reconstruction et la dissimulation.

D'une part, la reconstruction vise à retrouver les valeurs originales des bits corrompus à l'aide de modèles mathématiques complexes, comme présentée par Duhamel et Kieffer (2010).

D'autre part, la dissimulation tente de décoder les paquets corrompus et, par la suite, identifier et dissimuler la détérioration visuelle engendrée par cette opération. L'article de Ye et al., paru en mai 2003, présente un algorithme destiné au décodage d'images JPEG corrompues. Ce dernier est capable d'identifier et de classifier des blocs erronés, selon leurs caractéristiques dans le domaine des pixels (Ye et al., 2003).

Ye et al. définissent quatre mesures permettant d'identifier les blocs erronés. La première est la validité de la valeur assignée à un pixel. Il s'agit ici de déterminer si un pixel est dans un intervalle de données plausibles. Par exemple, pour les valeurs d'un pixel non signées de niveaux de gris assignées sur huit bits, les valeurs possibles sont de 0 à 255. La deuxième mesure est l'hétérogénéité des bordures de blocs horizontales mesurée à l'aide d'un filtre Sobel. La troisième est l'hétérogénéité des bordures de blocs verticales, encore une fois mesurée à l'aide d'un filtre Sobel. Dans d'autres oeuvres, l'hétérogénéité en bordure de blocs est aussi connue sous le nom d'effets de bloc. Le filtre Sobel est défini par les opérateurs suivants (horizontal et vertical respectivement) :

$$S_h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_v = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{4.1}$$

La dernière mesure, présentée par Ye et al., évalue la continuité des bordures propres au contenu de l'image qui traversent un bloc. Par la suite, ces mesures sont comparées à des seuils, afin de déterminer si le bloc évalué est erroné. Le résultat de la détection d'erreurs à l'aide de ces mesures est présenté à la figure 4.2, où les blocs noirs de l'image 4.2b représentent les erreurs détectées. On y constate de faux positifs, surtout dans le chapeau et les cheveux de Lena Söderberg.

(a) Image endommagée

(b) Détection de la détérioration (blocs noirs)

Figure 4.2 Exemple du résultat de la détection d'erreurs avec les mesures de Ye et al.. Tirée de Ye et al. (2003, p. 371)

La détérioration visuelle observée dans l'image de la figure 4.2a résulte d'un taux d'erreurs binaires de 3×10^{-4}. Avec ce taux, Ye et al. (2003) observent des gains de PSNR allant de 5 à 10 dB selon l'image, l'emplacement de la détérioration visuelle et le type d'approche de dissimulation d'erreurs utilisée (l'interpolation linéaire, l'interpolation directionnelle ou par prédiction des coefficients DCT).

4.3 Amélioration des approches d'analyse syntaxique d'encodage vidéo

Dans le but d'améliorer l'efficacité des règles de validation de syntaxe de la norme MPEG-4 Visual présentées par Talluri (1998), Yan et Wing proposent une nouvelle règle de validation qui augmente considérablement l'efficacité de ce genre d'approche. Cette règle permet de valider, entre deux marqueurs de synchronisation, que le nombre de COD soit égal au nombre de blocs (Yan et Wing, 2003). Voici le raisonnement derrière cette approche. Tout d'abord, les marqueurs de synchronisation sont des séquences binaires uniques qui servent de point de repère pour le décodeur. Le COD est un bit positionné en début de macrobloc, qui identifie si ce dernier est encodé ou pas. Le nombre de macroblocs contenus dans une tranche est disponible dans les paramètres d'encodages. S'il y a eu désynchronisation, ces deux nombres risquent d'être différents. Pour des séquences QCIF, encodées à 96 kb/s, Yan et Wing (2003) affirment que leur approche est capable de détecter entre 90% et 100% des erreurs binaires, selon la séquence et le taux d'erreurs binaires utilisés (allant de 0.2% à 1.0%).

Toujours dans le même ordre d'idée, en 2006, Superiori et Nemethova proposent un valideur de syntaxe H.264. Ce valideur identifie des séquences de codes binaires invalides engendrées par la désynchronisation des codes entropiques, suite à une erreur (Superiori et Nemethova, 2006). Ce valideur est exhaustif et définit des restrictions sur les valeurs d'un grand nombre de paramètres de la norme.

Superiori et Nemethova proposent trois catégories d'erreurs de décodage selon les caractéristiques de l'erreur (Superiori et Nemethova, 2006) :

- **Code numérique invalide :**

Il s'agit ici d'un code qui n'est pas dans la table de codes numériques associés à ce paramètre.

- **Code hors de portée :**

Cette erreur survient lorsqu'une valeur décodée est à l'extérieur de l'ensemble des valeurs valides pour le paramètre en question.

- **Erreur contextuelle :**

Cette erreur survient si le décodeur doit effectuer une opération invalide.

Les tests de Superiori et Nemethova ont été effectués sur la séquence QCIF *Foreman* encodée avec un QP fixé à 28. Dans ces conditions, leur approche réussit à détecter 60 % des erreurs binaires provenant de trames corrompues intra et 47 % de celles provenant de trames inter, selon (Superiori et Nemethova, 2006).

4.4 Combinaison de l'analyse syntaxique et de la détection de la détérioration visuelle dans le domaine des pixels

En 2007, Superiori et al. combinent leur approche de validation syntaxique à un algorithme d'identification de la détérioration visuelle inspiré de Ye et al. (2003), et auquel un système de vote a été ajouté, afin d'améliorer le résultat de la détection (Superiori et al., 2007). Dans son mémoire, Ikunoq explique en détail l'algorithme proposé (Ikuno, 2007).

La détection de la détérioration visuelle est réalisée dans le domaine des pixels. Pour isoler l'erreur, Superiori et al. utilisent le différentiel de la trame à évaluer par rapport à celle qui la précède (c.-à-d. la différence entre ces deux trames). La figure 4.3 montre une trame endommagée ainsi que la trame qui la précède. Le différentiel entre ces deux images est calculé en effectuant une différence absolue entre les pixels correspondants (même position spatiale). Le résultat du différentiel des images de la figure 4.3 est présenté à la figure 4.4a.

Par la suite, les valeurs du différentiel sont regroupées en blocs 8×8 et la moyenne est assignée comme étant représentative de l'énergie du bloc (Ikuno, 2007, p. 27-28), comme illustré à la figure 4.4b. Les erreurs détectées à la figure 4.4c sont obtenues à l'aide d'un seuil. Si la valeur de l'énergie d'un bloc est supérieure à ce seuil, le bloc est considéré comme erroné.

De plus, la détection d'effets de bloc en bordure de blocs est effectuée. Pour trouver les bordures à l'intérieur des trames, les composantes horizontales 4.5a et verticales 4.5b, d'un filtre de Haar (Haar, 1911), sont combinées et seulement les valeurs en bordure de blocs sont conservées 4.5c. L'analyse des effets de bloc est accomplie pour les blocs 8×8 et pour

(a) Trame endommagée (b) Trame précédente

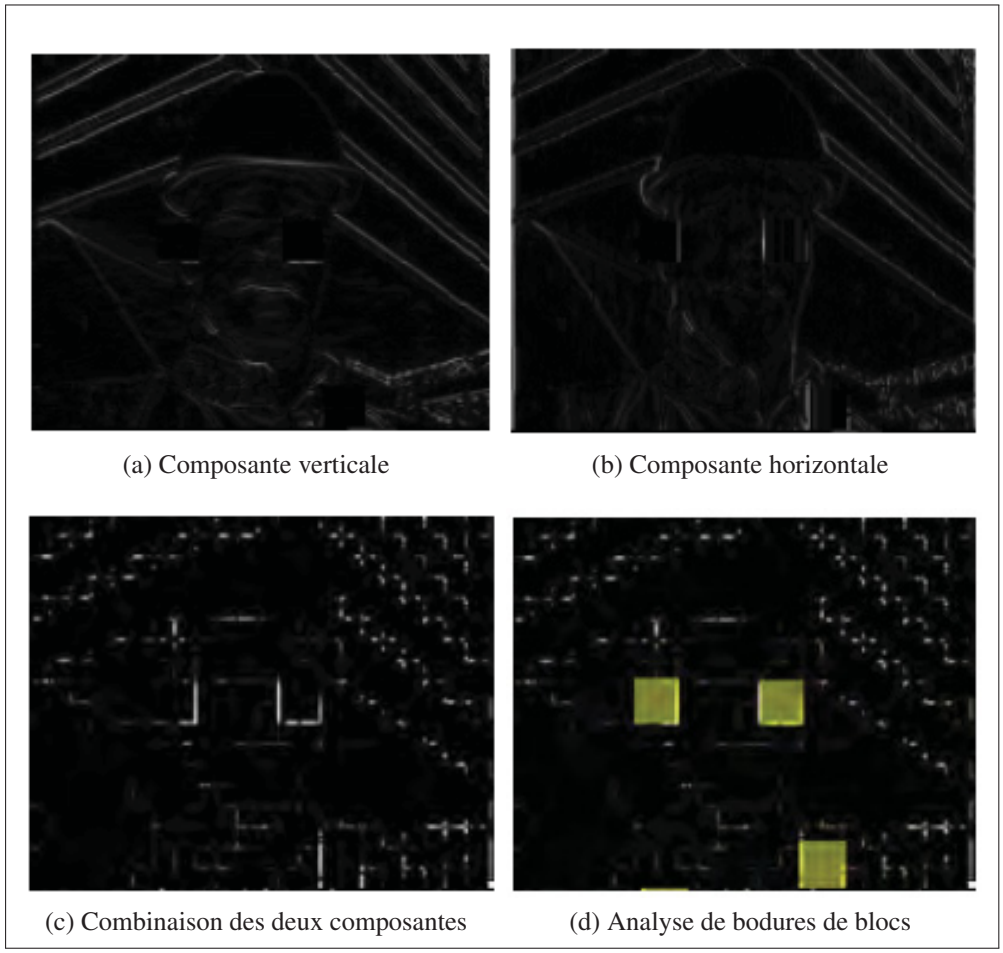
Figure 4.3 Trames utilisées par Ikuno pour démontrer l'algorithme de détection de la détérioration visuelle. Tirée de Ikuno (2007, p. 25)

(a) Différentiel (b) Analyse du différentiel (c) Erreurs détectées

Figure 4.4 Exemple de l'analyse et de la détection d'erreurs.
Tirée de Ikuno (2007, p. 25)

les macroblocs 16×16. Ceux-ci sont comparés à des seuils, afin de savoir s'il y a présence d'erreurs.

Les mesures d'énergie et d'effets de bloc sont interprétées par un système de vote qui détermine où se trouvent les blocs erronés à l'intérieur de la trame évaluée. Selon Superiori et al. (2007), l'approche proposée améliore de 1.36 dB, en moyenne, de PSNR des trames de la séquence QCIF *foreman*. Celles-ci sont encodées avec un QP fixé à 28 et sont exposées à



(a) Composante verticale (b) Composante horizontale

(c) Combinaison des deux composantes (d) Analyse de bodures de blocs

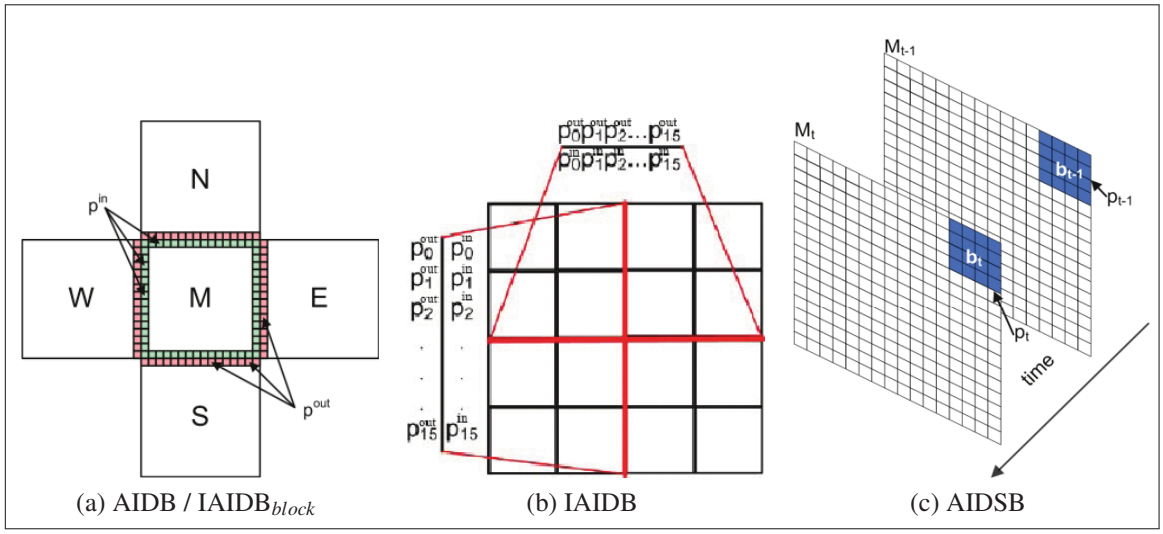
Figure 4.5 Composantes et analyse du filtre de Haar.
Tirée de Ikuno (2007, p. 26)

un taux d'erreurs de 10^{-5}. Ce résultat est obtenu par la moyenne des résultats de l'exécution de 20 simulations.

L'année suivante, Farrugia et Debono reprennent les travaux de Superiori et al. (2007) et retirent les seuils et le système de vote. Ceux-ci sont remplacés par une machine à vecteurs de support (Cortes et Vapnik, 1995). Cette dernière est une méthode d'apprentissage supervisée qui, une fois entrainée sur un ensemble de données, est capable de classifier d'autres données. Pour ce genre d'algorithme, le temps d'entrainement est long, mais la classification est très rapide, ce qui en fait un bon choix pour des considérations *temps réel* (Farrugia et Debono, 2008).

Le vecteur d'entrée (*feature vector*) définit par Farrugia et Debono possède huit composantes :

1. AIDB ;
2. La moyenne du IAIDB$_{block}$;
3. L'écart type du IAIDB$_{block}$;
4. IAIDB vertical ;
5. IAIDB horizontal ;
6. La moyenne du AIDSB ;
7. L'écart type du AIDSB ;
8. TC.



(a) AIDB / IAIDB$_{block}$ (b) IAIDB (c) AIDSB

Figure 4.6 Visualisation des composantes du vecteur d'entrée.
Tirée de Farrugia et Debono (2010, p. 79-80)

Ce vecteur représente un macrobloc contenu à l'intérieur d'une trame d'une séquence vidéo. Le AIDB est la différence moyenne des pixels en bordure de macroblocs (M, à la figure 4.6a). Le IAIDB$_{block}$ est la même mesure que le AIDB, mais appliquée aux blocs 4×4 à l'intérieur d'un macrobloc. Le IAIDB est la moyenne de la différence des pixels pour les bordures internes (verticales et horizontales) d'un macrobloc (lignes rouges figure 4.6b). Le AIDSB

est l'énergie d'un macrobloc 16×16, tel que défini par Ikuno (2007), soit la moyenne du différentiel par rapport à la trame précédente (voir la figure 4.6c). La consistance de la texture (TC) est basée sur une approche d'analyse de texture présentée par Wang et He (1990), connue sous le nom de patron binaire local (*Local Binary Pattern (LBP)*).

Farrugia et Debono ont effectué des tests avec des observateurs humains pour déterminer l'importance de la détérioration visuelle. Par la suite, ils utilisent cette information pour mesurer leur solution. Celle-ci identifie en moyenne 95.25% des détériorations qu'ils ont qualifiées de *Very Annoying*, *Annoying* et *Slightly Annoying*. Pour la détérioration visuelle qu'ils qualifient de *Perceptible but not Annoying* le taux de détection est de 62.91%. Les séquences utilisées sont de résolution QCIF, encodées à 128 kb/s.

En somme, ce chapitre offre un bref survol des travaux de recherche en lien avec le *Joint Source Channel Decoding*, la résilience aux erreurs lors du transport de vidéo, la validation de syntaxe pour les normes d'encodage vidéo et la détection de la détérioration visuelle dans le domaine des pixels. Ces concepts sont présentés dans un ordre quasi chronologique, où la cohésion conceptuelle a raison de la chronologie. Conséquemment, ils servent de notions fondatrices aux approches proposées dans cet ouvrage et présentées dans le prochain chapitre.

CHAPITRE 5

NOUVELLE APPROCHE D'IDENTIFICATION ET DE DISSIMULATION DE LA DÉTÉRIORATION VISUELLE

Les chapitres antérieurs constituent les éléments d'un panorama technologique permettant l'assimilation des notions présentées dans ce chapitre. Ces éléments recouvrent un large territoire technologique s'étendant de l'encodage vidéo à son transport ainsi que l'impact, la détection et la dissimulation de la détérioration de séquences vidéos codées en respectant la norme H.264.

Nous rappelons d'abord, à la section 5.1, les limitations des approches actuelles de détection d'erreurs. Nous présentons ensuite, à la section 5.2, une amélioration fondamentale aux approches actuelles de détection de la dégradation visuelle à l'intérieur de trames corrompues. Cette approche de détection améliorée sert de base pour un nouveau type d'approche de dissimulation : les approches sélectives. À la section 5.3, deux variantes de ces dernières sont décrites.

5.1 Limitations des approches actuelles de détection de la détérioration visuelle

Les approches de détection de la détérioration visuelle étudiées ici sont celles présentées dans notre revue de la littérature. Avant de définir les limitations de ces approches, commençons par uniformiser la nomenclature utilisée pour les décrire. Pour ce faire, nous présentons la figure 5.1, soit un survol des concepts liés à la détection de la détérioration visuelle.

Tout d'abord, nous représentons les trames comme des matrices bidimensionnelles de pixels. Cela dit, $\mathbf{F}(t)$ fait référence à la $t^{\text{ième}}$ trame d'une séquence vidéo. Cependant, pour alléger la notation, quand le contexte est clair, nous n'utilisons pas l'indice de la trame. Par exemple, $\mathbf{F}(t)$ devient $\mathbf{F}$. L'image de référence, celle non compressée, est définie par $\mathbf{F}_O(t)$. De plus, on note $\mathbf{P}$, la trame qui précède la trame $\mathbf{F}$, ce qui équivaut à $\mathbf{F}(t-1)$.

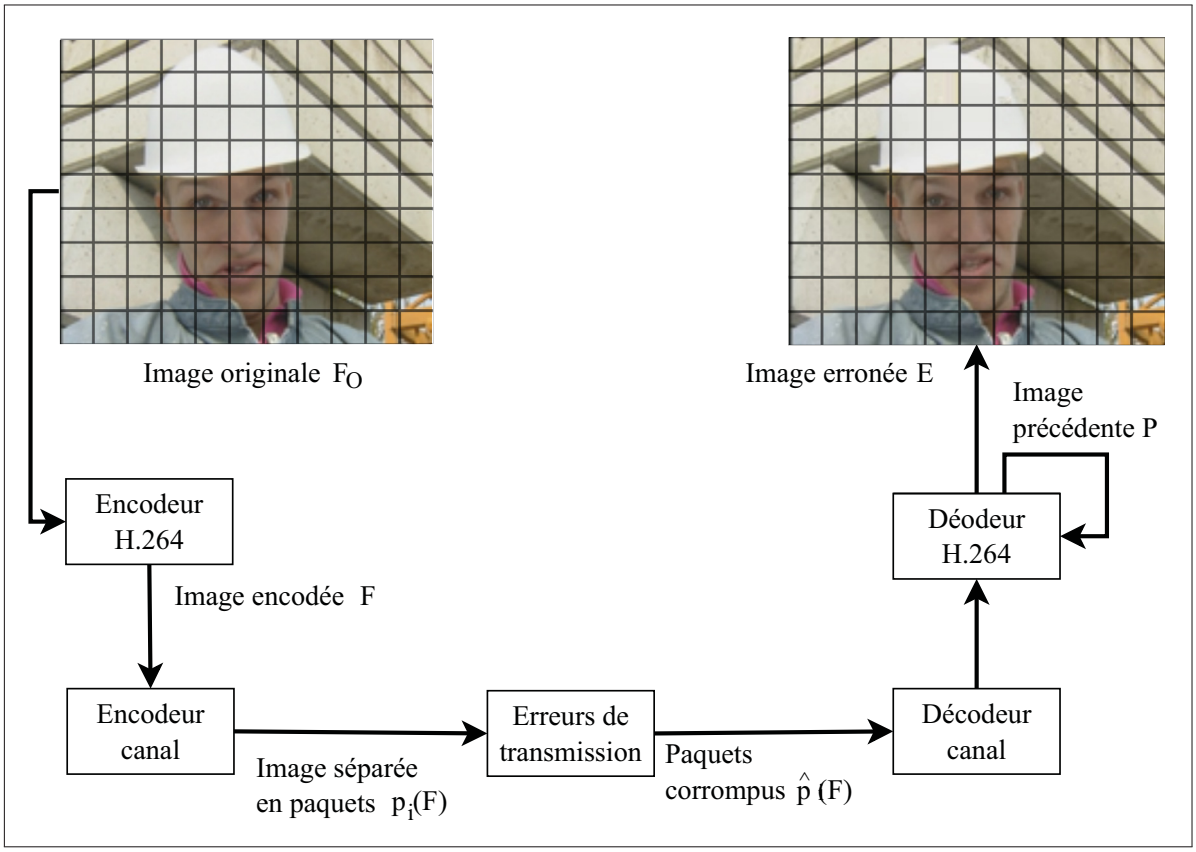


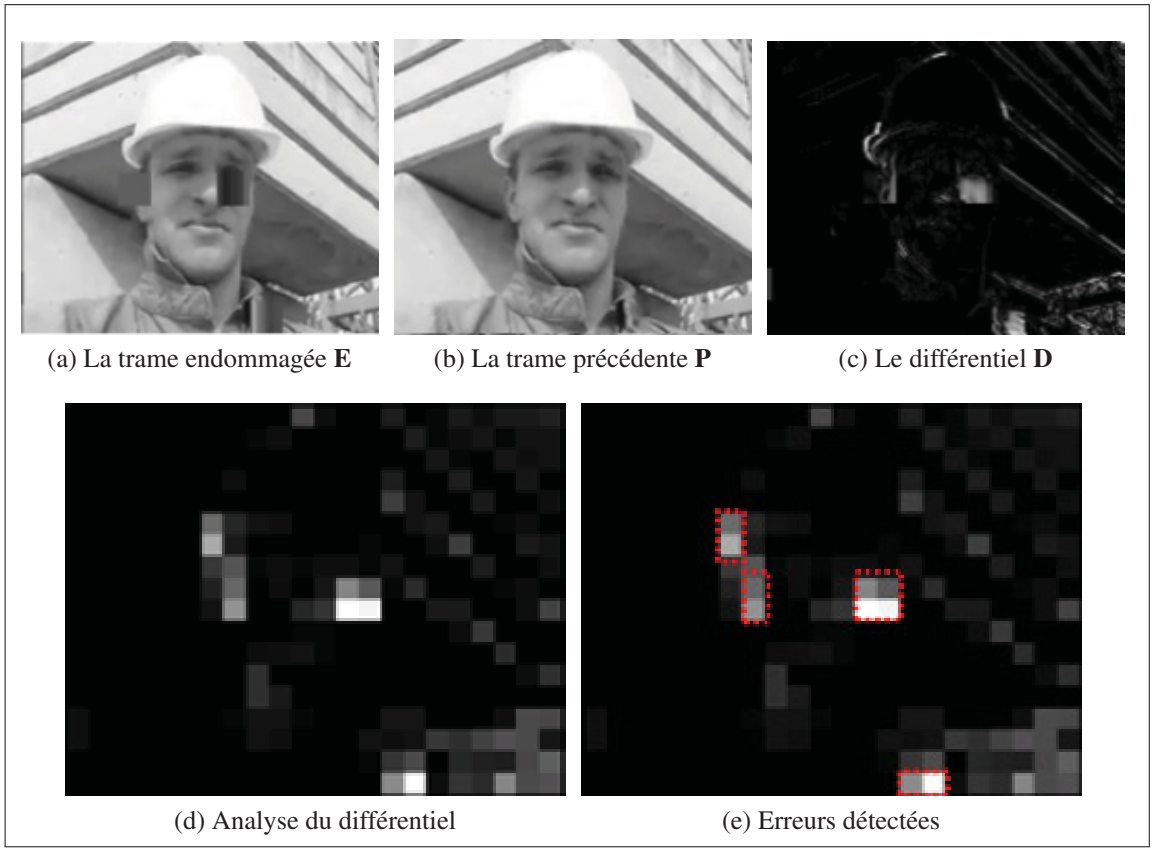
Figure 5.1 Survol des concepts liés à la détection de la détérioration visuelle.

Pour la transmission sur le canal, une trame est divisée en paquets. Cet empaquetage est représenté par $p_i(\mathbf{F})$, où i est l'indice du paquet associé à la trame $\mathbf{F}$. Nous désignons la corruption du paquet $p_i(\mathbf{F})$ par $\hat{p}_i(\mathbf{F})$. La trame issue du décodage des paquets corrompus $\hat{p}_i(\mathbf{F})$, lorsqu'il est possible, est représentée par $\mathbf{E}(t)$, ce qui équivaut à $\hat{\mathbf{F}}(t)$.

Les auteurs, Superiori et al. (2007) ainsi que Ikuno (2007), présentent une approche de détection où l'on mesure l'énergie des blocs ainsi que leurs effets de blocs à l'intérieur d'un différentiel $\mathbf{D}$. Ce différentiel, issu de la différence absolue entre la trame erronée $\mathbf{E}$ et la trame qui la précède $\mathbf{P}$, est défini par la formule suivante :

$$\mathbf{D} = |\mathbf{E} - \mathbf{P}|. \tag{5.1}$$

Les données obtenues par ces mesures sont comparées à des seuils et un système de votes détermine s'il y a erreur. La méthode se base sur l'hypothèse que les pixels ne varient pas beaucoup d'une image à l'autre et que les grandes variations sont probablement dues à des erreurs de transmission. Nous présentons, à titre d'exemple de cette approche, la figure 5.2 tirée de Ikuno (2007, p. 25). Pour éviter le recours aux seuils, les auteurs Farrugia et Debono (2008) collectent une série de mesures provenant de **D** et utilisent une machine à vecteurs de support (Cortes et Vapnik, 1995) pour identifier les blocs erronés.



(a) La trame endommagée **E** (b) La trame précédente **P** (c) Le différentiel **D**

(d) Analyse du différentiel (e) Erreurs détectées

Figure 5.2 Exemple de l'analyse et de la détection d'erreurs.
Tirée de Ikuno (2007, p. 25)

Le problème commun que nous identifions, comme limitation majeure, avec ces approches est l'usage du différentiel **D**. Quoiqu'il puisse sembler alléchant, ce dernier, illustré à la figure 5.2c, ne contient pas uniquement l'erreur due aux erreurs de transmission, mais aussi la variation des valeurs de pixels propre à la séquence. Cela signifie que le différentiel résultant

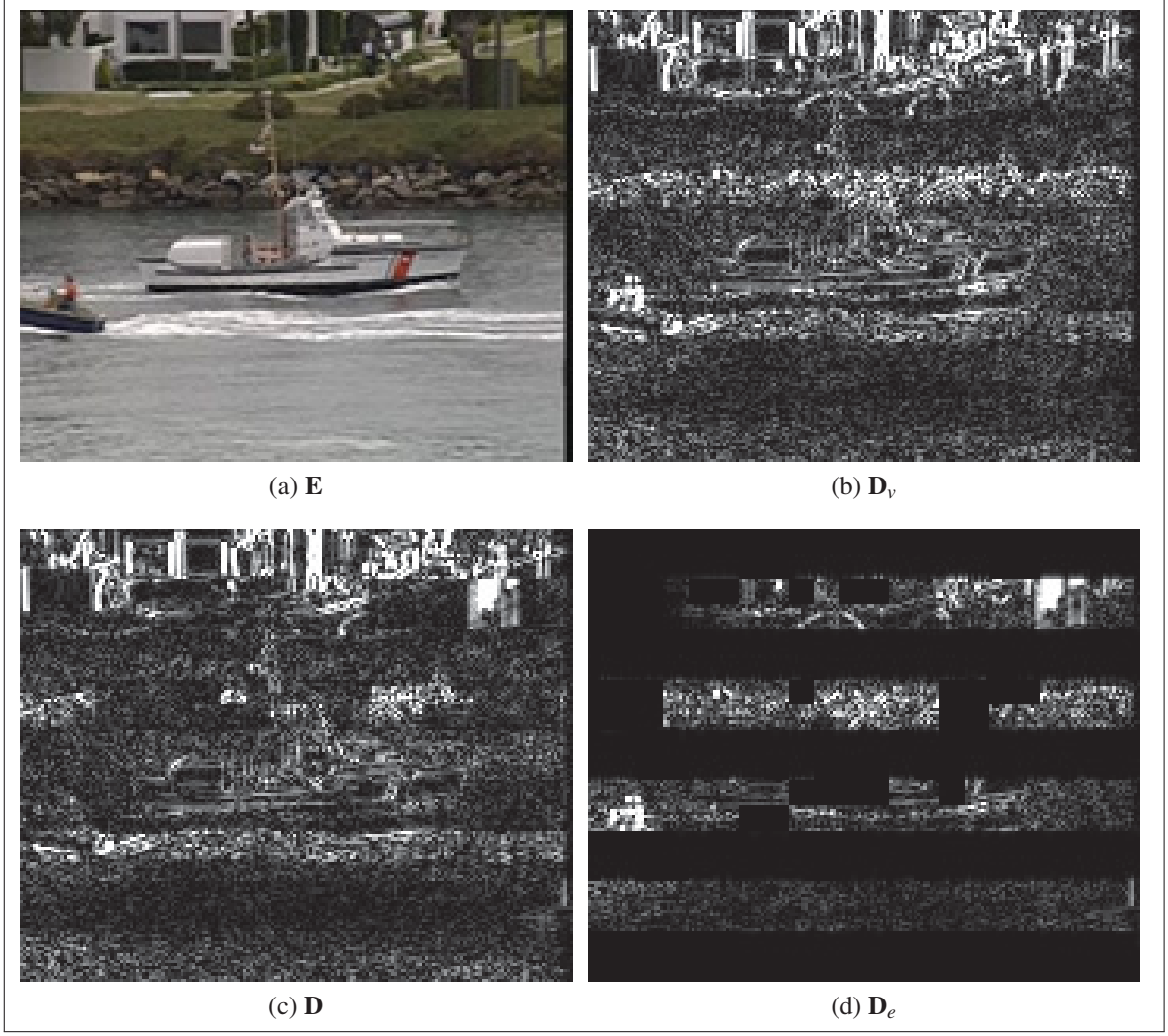
des transformations entre les images non erronées, entre **F** et **P**, tel le mouvement des objets, est aussi contenu dans **D**. En effet, utilisant l'inégalité du triangle ($|a-b| \leq |a-c|+|c-b|$), on peut réécrire l'équation précédente comme :

$$\mathbf{D} = |\mathbf{E}-\mathbf{P}| \leq \underbrace{|\mathbf{E}-\mathbf{F}|}_{\mathbf{D}_e} + \underbrace{|\mathbf{F}-\mathbf{P}|}_{\mathbf{D}_v}, \tag{5.2}$$

avec $\mathbf{D}_e$ l'erreur produite par les détériorations visuelles dues aux erreurs de transmission et $\mathbf{D}_v$ l'erreur produite par les variations entre les trames. On peut remarquer que sans **F**, il est impossible de distinguer, à l'intérieur de **D**, les contributions individuelles de $\mathbf{D}_e$ et de $\mathbf{D}_v$. Les auteurs mentionnés précédemment (Superiori et al. (2007), Ikuno (2007) ainsi que Farrugia et Debono (2008)) misent sur les caractéristiques spatiales de la dégradation visuelle pour tenter d'effectuer cette distinction. Cependant, il est probable que la variation entre les trames $\mathbf{D}_v$ manifeste ce genre de caractéristiques. Comme nous le démontrons aux figures 5.3b et 5.3c.

Par notre contrexemple, la figure 5.3, nous démontrons l'inefficacité de la détection d'erreurs par **D**. La séquence *coastguard* est caractérisée par un déplacement horizontal de la caméra. Ce déplacement crée une très grande variation entre deux trames consécutives, comme illustré à la figure 5.3b. L'erreur à détecter est identifiée à la figure 5.3d. Il n'est pas simple d'extraire cette information de **D** (figure 5.3c) sans connaitre **F**.

Bref, ce manque de précision, engendré par la variation *intertrame* présente dans le différentiel, force les auteurs à employer des systèmes complexes de votes ou de machines à vecteur de support. À l'opposé, le résultat de notre effort de recherche est une approche de détection simple qui oeuvre dans un contexte différent de **D**, où la variation entre les trames est minimisée par la compensation de mouvement.



(a) **E** (b) $\mathbf{D}_v$

(c) **D** (d) $\mathbf{D}_e$

Figure 5.3 Contrexemple de l'usage de **D** pour effectuer la détection d'erreurs. Le contenu du différentiel 5.3c provient de la variation entre la trame 5.3a et la trame qui la précède.

5.2 Effets de blocs compensés par le mouvement

Nous venons de présenter la variation entre les trames comme un obstacle majeur à la détection d'erreurs. En s'intéressant aux origines de cette variation, on constate qu'elle peut provenir d'une translation, d'une déformation, d'un changement d'intensité lumineuse ainsi que de l'insertion ou du retrait de contenu. Pour plus d'information sur la détérioration visuelle référez-vous au chapitre sur la détérioration visuelle (p. 36).

En pratique, l'erreur de transmission agit comme un changement entre les trames équivalant soit à une déformation ou à une insertion de contenu. Ce qui fait en sorte que ces types de changements sont les plus difficiles à distinguer de l'erreur.

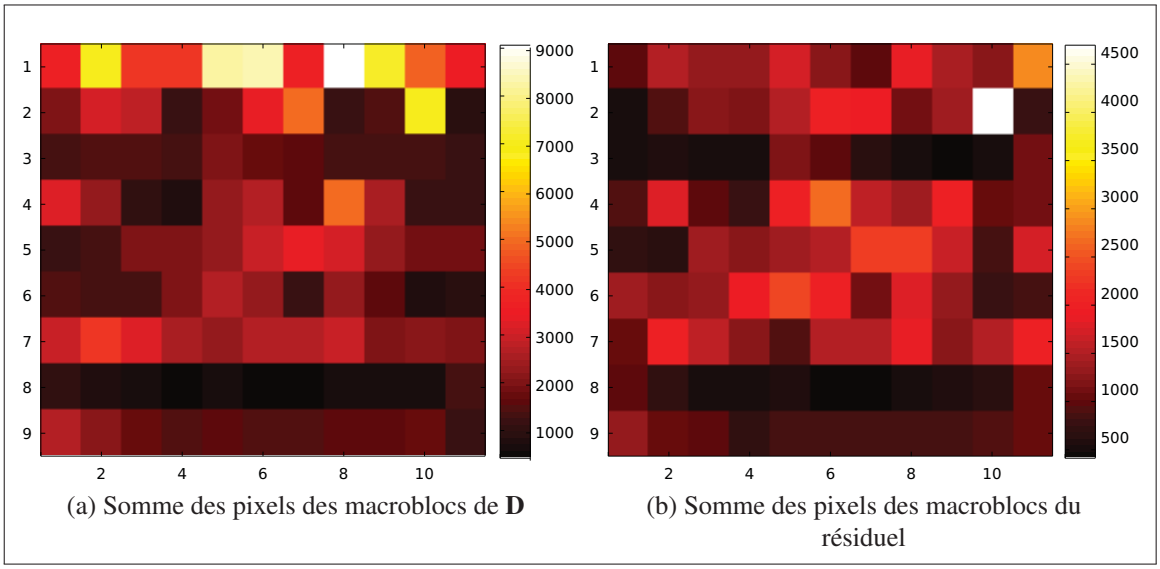
Dans le domaine de l'encodage vidéo numérique, la variation entre les trames est un concept qui a suscité une grande quantité d'efforts de recherche. Comme présentées dans notre section sur H.264 (p. 6), les approches de recherche et de compensation de mouvement sont raffinées, depuis maintenant plusieurs années, pour modéliser cette variation afin d'augmenter le taux de compression issue de leur encodage. La formule de la recherche des vecteurs de mouvement a déjà été décrite à la formule 1.3 de la page 13, pour plus d'information veuillez consulter (Li et Drew, 2004).

En ce qui a trait à la détection de la détérioration visuelle, la recherche de vecteurs de mouvement peut servir à prendre en compte la variation entre les trames ($\mathbf{D}_v$). En contrepartie, les approches qui reposent sur $\mathbf{D}$ insinuent que $\mathbf{D}_v = 0$.

Pour illustrer cette affirmation, nous recourons à la figure 5.4. Celle-ci montre la somme des pixels pour chaque macrobloc de la trame à la figure 5.3a, pour $\mathbf{D}$ (figure 5.4a) et le résiduel (figure 5.4b). Le résiduel est le différentiel absolu entre le bloc à évaluer et son meilleur candidat dans la trame précédente.

La première rangée de blocs de la figure 5.4a nous permet de constater que pour plusieurs blocs valides, comme le démontre la figure 5.3d, le pointage obtenu est supérieur à ceux des blocs corrompus. Ceci implique qu'une approche qui exploite l'énergie des blocs pour effectuer sa détection effectuerait de fausses détections.

De plus, le bloc avec le pointage le plus élevé, dans la figure 5.4b, est non seulement une erreur, mais l'erreur la plus dérangeante de la figure 5.3a. L'écart de pointage, par rapport au prochain bloc, est considérable, soit plus de 2000. Notons que le second bloc au pointage le plus élevé est lui aussi erroné. Cependant, il est beaucoup moins dérangeant, ce qui justifie l'écart de 2000.



(a) Somme des pixels des macroblocs de **D**

(b) Somme des pixels des macroblocs du résiduel

Figure 5.4 Valeurs des macroblocs issues de **D** 5.4a et du résiduel 5.4b mesurés sur la figure 5.3a.

Quoiqu'elle améliore la détection d'erreurs, la compensation de mouvement n'est pas en mesure d'identifier les vecteurs de mouvement corrompus. Comme décrit au chapitre 3 (p. 36), ce type d'erreur insère, dans l'image erronée, du contenu fautif provenant du mauvais endroit dans la trame de référence. Pour ce genre d'erreur, la recherche de mouvement retrouve ce contenu fautif. Ce qui trompe l'algorithme de détection, et ce dernier n'est pas en mesure d'identifier l'erreur.

Cette limitation de la recherche de mouvement nous oblige à incorporer un élément important pour la détection d'erreurs : La notion de nuisance. Ce que nous cherchons à détecter n'est pas l'erreur absolue, mais bien l'erreur qui engendre une dégradation visuelle perceptible par le système visuel humain. Dans le chapitre sur l'effet des erreurs sur les séquences vidéos, nous avons conclu que, pour être dérangeantes, les erreurs doivent exhiber un ou plusieurs effets de blocs.

Les effets de blocs compensés par le mouvement (MCB), que nous proposons dans cette recherche, sont la combinaison de ces deux faits. Comme décrit à la figure 5.5, le MCB

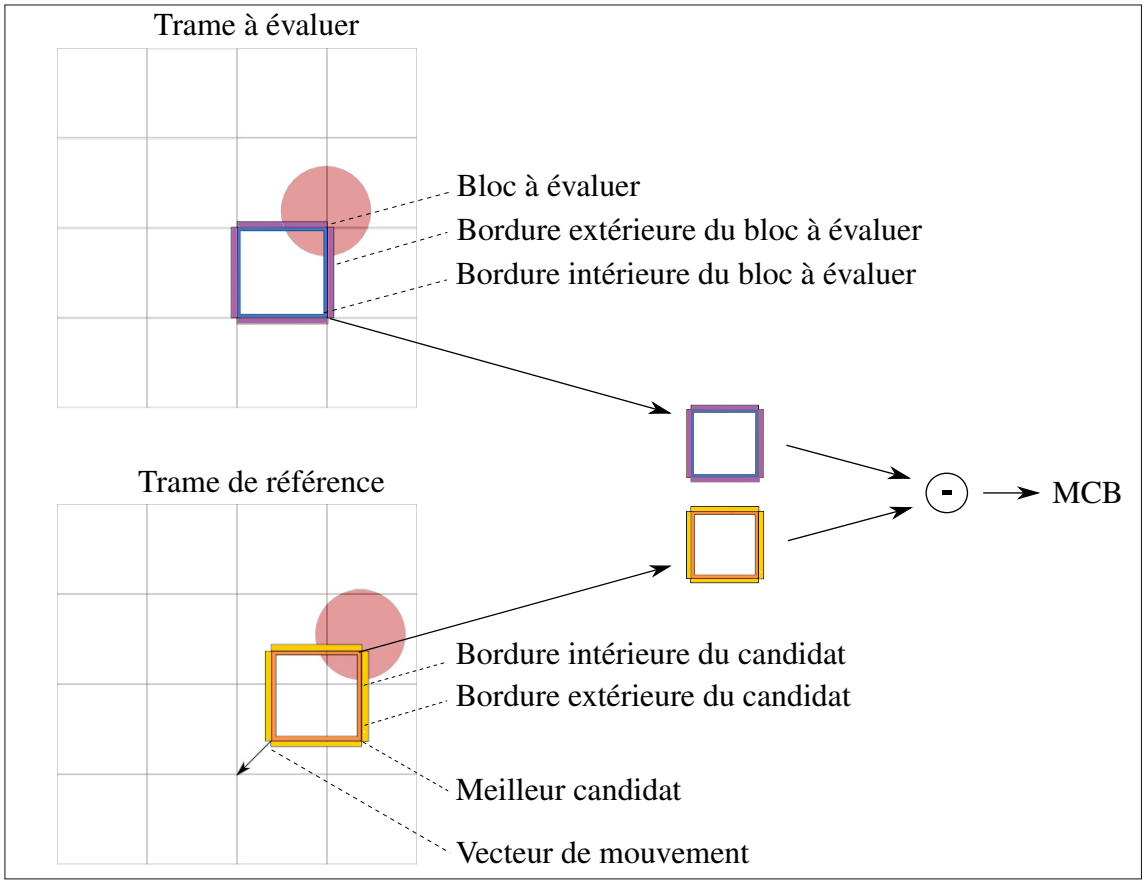


Figure 5.5 Visualisation des composantes liées au calcul du MCB.

recherche les discontinuités spatiales en bordure de blocs entre la trame à encoder et la trame de référence. Ne connaissant pas la trame de référence, nous utilisons la trame précédente.

Avant de procéder à l'explication pratique du fonctionnement du MCB (à la sous-section 5.2.2), nous présentons en détail, à la section 5.2.1, sa définition mathématique afin de bien comprendre comment le MCB est mesuré.

5.2.1 Définition théorique du MCB

Pour cette démonstration, et à des fins de simplicité, nous avons recours à des blocs carrés [1] de taille B. Nous définissons un bloc comme un ensemble de pixels de taille $B \times B$. La hauteur et

1. Il est aussi possible d'utiliser des blocs rectangulaires.

la largeur d'une trame sont définies, respectivement, par H et W. Les intervalles $I_m = [0, \frac{W}{B} - 1]$ et $I_n = [0, \frac{H}{B} - 1]$ représentent les indices de blocs à l'intérieur d'une trame. Nous désignons p, comme la demi-hauteur de la fenêtre de recherche de mouvement.

Nous employons (u, v), comme le vecteur optimal issu de la recherche de mouvement. Cette opération s'exprime par la formule suivante :

$$(\mathbf{U}_{m,n}, \mathbf{V}_{m,n}) = \arg \min_{(u,v) \in K} \text{SAD}(mB, nB, u, v, \mathbf{E}, \mathbf{P}), \tag{5.3}$$

où $K = [-p, p] \times [-p, p]$, $m \in I_m$, $n \in I_n$ et

$$\text{SAD}(x, y, u, v, \mathbf{E}, \mathbf{P}) = \sum_{q=0}^{B-1} \sum_{r=0}^{B-1} \left| \mathbf{E}_{x+q,y+r} - \mathbf{P}_{x+q+u,y+r+v} \right| . \tag{5.4}$$

Les composantes des vecteurs de mouvement optimaux (u, v) sont insérées à l'intérieur des matrices $\mathbf{U}$ et $\mathbf{V}$, à la position correspondante à l'indice du bloc. Cette opération est effectuée sur l'ensemble des blocs qui composent la trame $\mathbf{E}$. Comme il a déjà été mentionné, nous effectuons la recherche de vecteurs de mouvement afin de tenir compte des variations issues de $\mathbf{D}_v$ afin de les séparer de la mesure de détérioration visuelle. Ainsi, au lieu de faire une différence entre les blocs situés à la même position spatiale, notre métrique effectuera la différence sur les blocs correspondants d'une trame à l'autre, en considérant leur mouvement. De cette manière, nous réduisons l'effet du mouvement des objets entre les trames sur la mesure de différence.

Lors de nos expérimentations, il a été constaté que les gains réalisés par l'usage d'un niveau de précision au quart et au demi-pixel étaient minimes. Alors, pour abréger cette démonstration, ces notions ne seront pas présentées ici. Dans ce chapitre, nous utilisons une précision au pixel entier. Cependant, nous tenons à informer que des niveaux de précision au quart et au demi-pixel ont été considérés dans le cadre de nos recherches.

Pour établir l'appariement des blocs entre la trame erronée et la trame de référence, nous avons recours à la somme de la différence absolue (SAD) entre deux blocs (eq.5.4). Quoiqu'il n'en

résulte pas une identification exacte du mouvement, cette approche est peu complexe et permet l'identification du bloc candidat avec la plus faible variation par rapport au bloc désiré. Ainsi, lorsque, pour un bloc donné, le SAD minimum est élevé, on peut souvent conclure qu'une erreur s'est produite.

Toutefois, l'usage du SAD minimum, tel que calculé en (eq. 5.3), n'est pas efficace comme approche de détection d'erreurs, lorsque le contenu erroné est présent dans la trame précédente **P** (c.-à-d. lorsqu'un bloc possède un bloc similaire dans la trame précédente). Cette situation se présente souvent lorsque les vecteurs de mouvement sont corrompus. Dans cette situation, ces vecteurs pointent sur le mauvais contenu même si le SAD est faible, ce qui peut souvent entrainer des effets de blocs importants.

Comme illustré à la figure 5.5 (p. 61), c'est pour cette raison que nous mesurons plutôt les effets de blocs en bordure du bloc à évaluer, par rapport aux effets de blocs du candidat résultant, indiqué par le vecteur (u,v), de la recherche de mouvement (eq.5.3). Nous nommons ainsi cette opération : mesure d'effets de bloc compensés par le mouvement. On y réfère aussi par son acronyme anglais MCB, et elle est définie par la formule :

$$\mathrm{MCB}_d(\mathbf{E},\mathbf{P},m,n) = \sum_{l=0}^{B-1} \left|\mathbf{b}_{d,l}(\mathbf{E},mB,nB) - \mathbf{b}_{d,l}(\mathbf{P},mB+\mathbf{U}_{m,n},nB+\mathbf{V}_{m,n})\right|, \tag{5.5}$$

où $m \in I_m$, $n \in I_n$. De plus, **U** et **V** proviennent de l'équation (5.3). Le MCB est appliqué à toutes les bordures d'un bloc ($d \in \mathcal{B}$). Nous référons aux bordures d'un bloc par l'ensemble suivant : $\mathcal{B} = \{\mathcal{N},\mathcal{E},\mathcal{S},\mathcal{W}\}$ et nous mesurons les vecteurs d'effets de bloc $\mathbf{b}_{d,l}$ pour chacune d'elles, à l'aide de :

$$\mathbf{b}_{\mathcal{N},l}(\mathbf{F},x,y) = \mathbf{F}_{x+l,y} - \mathbf{F}_{x+l,y-1}, \tag{5.6}$$

$$\mathbf{b}_{\mathcal{E},l}(\mathbf{F},x,y) = \mathbf{F}_{x+B,y+l} - \mathbf{F}_{x+B-1,y+l}, \tag{5.7}$$

$$\mathbf{b}_{\mathcal{S},l}(\mathbf{F},x,y) = \mathbf{F}_{x+l,y+B} - \mathbf{F}_{x+l,y+B-1}, \tag{5.8}$$

$$\mathbf{b}_{\mathcal{W},l}(\mathbf{F},x,y) = \mathbf{F}_{x,y+l} - \mathbf{F}_{x-1,y+l}, \tag{5.9}$$

où $l \in [0, B-1]$, et (x, y) sont les coordonnées en pixels des blocs.

Pour évaluer l'effet global des effets de blocs en bordure d'un bloc, nous utilisons la somme des effets de blocs compensés par le mouvement de ce dernier (SMCB). Cette mesure est obtenue, pour un bloc aux coordonnées de bloc (m, n), à l'aide de la formule suivante :

$$\mathrm{SMCB}(\mathbf{F}, \mathbf{P}, m, n) = \sum_{d \in \mathcal{B}} \mathrm{MCB}_d(\mathbf{F}, \mathbf{P}, m, n)\,. \tag{5.10}$$

Dans cette formule, on effectue la sommation des valeurs MCB pour chacune des bordures $d \in \mathcal{B}$ du bloc évalué. Ce qui est intéressant avec cette nouvelle approche, c'est que nous mesurons le niveau de discontinuité présent à la bordure du bloc, obtenue en soustrayant le bloc considéré du bloc qui lui correspond le mieux dans l'image précédente. S'il n'y a aucun bloc candidat correspondant bien au bloc considéré ou si le bloc correspond bien, mais qu'il provient de l'application d'un mauvais vecteur de mouvement et, par conséquent il est ainsi hors de son contexte initial, la mesure est élevée. Sinon, un objet s'est seulement déplacé et, par conséquent le contexte visuel est le même autour du bloc considéré et du bloc candidat, même s'il y a beaucoup de détails ou de bordures marquées autour de ces blocs, alors la mesure obtenue sera faible par l'effet soustractif du MCB.

Les figures 5.6, 5.7 et 5.8 montrent la mesure des effets de bloc compensés par le mouvement dans trois situations différentes choisies parmi celles rencontrées lors du décodage de séquence H.264. Pour chaque situation, les quatre bordures (nord, est, sud, ouest) extérieures du bloc à évaluer, encadré en rouge dans 5.6a, sont comparées à celles du meilleur bloc candidat, de la trame précédente, encadré en rouge dans 5.6b.

Dans la première situation, illustrée à la figure 5.6, la trame à évaluer ne contient pas d'erreur. Cependant, on y constate un changement important de l'emplacement du cercle rouge. Dans une telle situation, le différentiel ne serait pas efficace. Néanmoins, la translation est prise en compte, grâce à la recherche de mouvement, ce qui fait que les quatre bordures externes sont identiques.

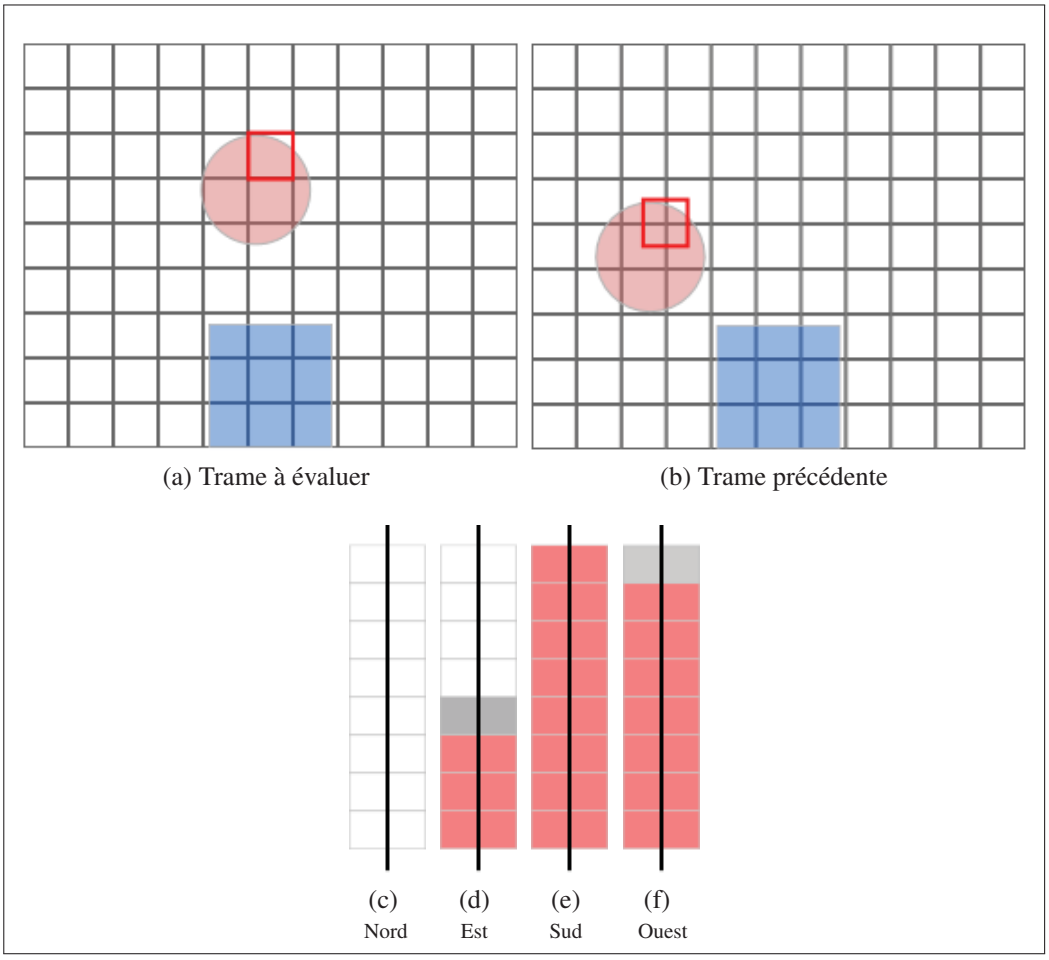


Figure 5.6 Exemple de la mesure des effets de bloc compensés par le mouvement, pour une trame normale.

La seconde situation est celle du macrobloc corrompu, comme présenté à la figure 5.7. Dans un tel contexte, le macrobloc corrompu n'existe pas dans la trame précédente. Ceci fait en sorte que la recherche de mouvement ne réussira pas à trouver un bon candidat. Certaines implémentations d'algorithmes de recherche de vecteurs de mouvement, pour des SAD équivalents, vont favoriser les vecteurs de mouvement plus petits. Dans la situation présentée à la figure 5.7a, le candidat présenté par la figure 5.7b est donc plausible. La comparaison des bordures compensées par le mouvement indique que trois bordures sont différentes (5.7d, 5.7e, 5.7f), ce qui augmente le pointage du macrobloc de sorte à en permettre l'identification comme macrobloc erroné.

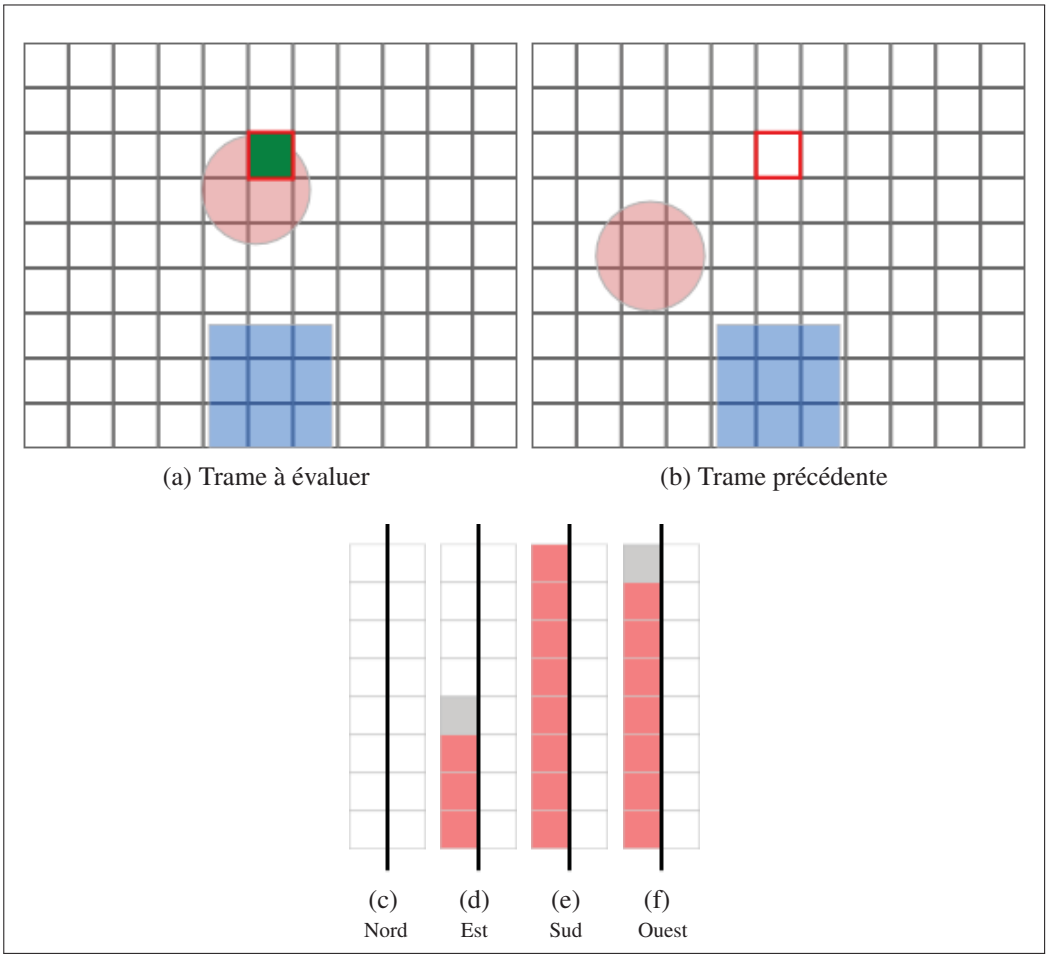


Figure 5.7 Exemple de la mesure des effets de bloc compensés par le mouvement, pour un macrobloc corrompu.

Le vecteur de mouvement corrompu est notre troisième et dernière situation. Celle-ci est illustrée à la figure 5.8. La particularité de cette situation est que le macrobloc erroné est présent dans la trame précédente et sera retrouvé par l'algorithme de recherche de vecteur de mouvement. Cependant, la comparaison des bordures extérieures démontre que les pixels externes ne s'agencent pas (5.8d, 5.8e, 5.8f). Ceci fait en sorte que nous sommes en mesure d'identifier ce macrobloc erroné.

La détection d'effets de bloc par la mesure de pixels en bordure de blocs peut toutefois introduire de fausses détections pour les bordures de blocs avoisinant un bloc erroné. Comme

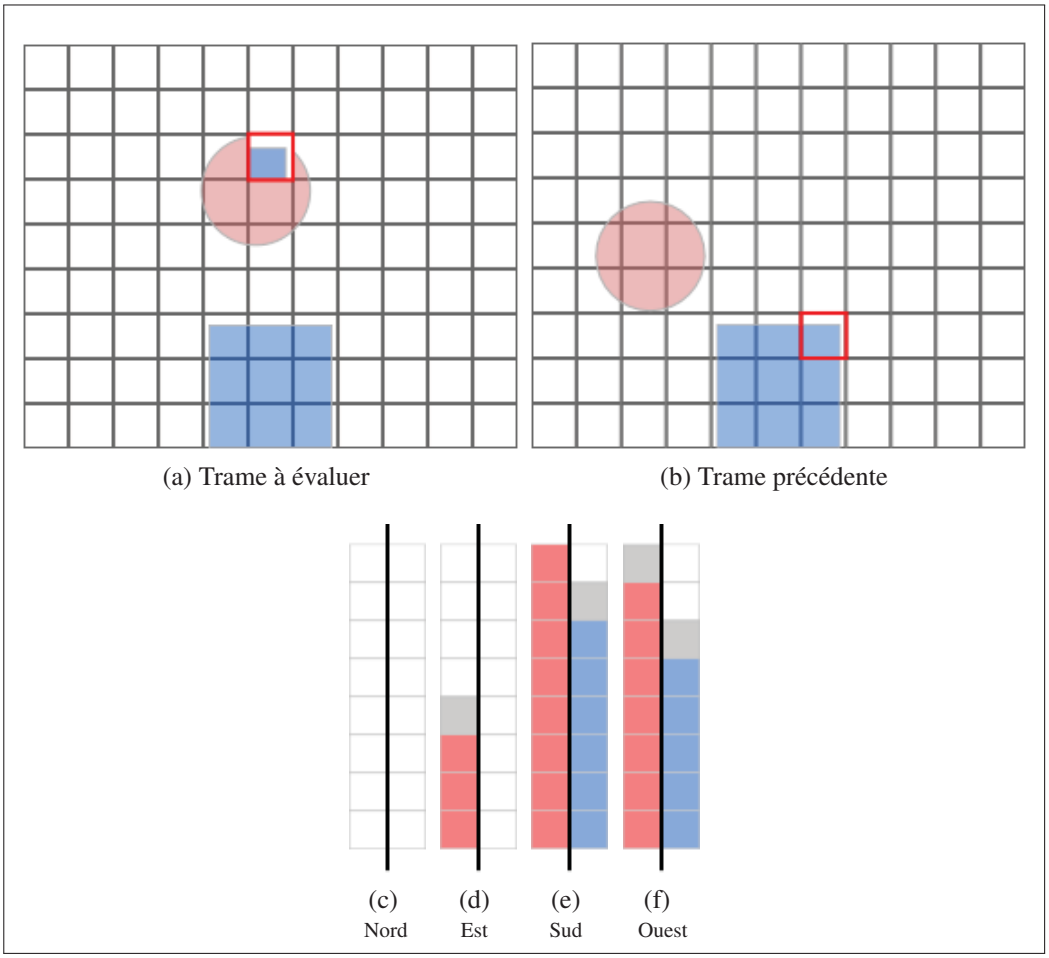
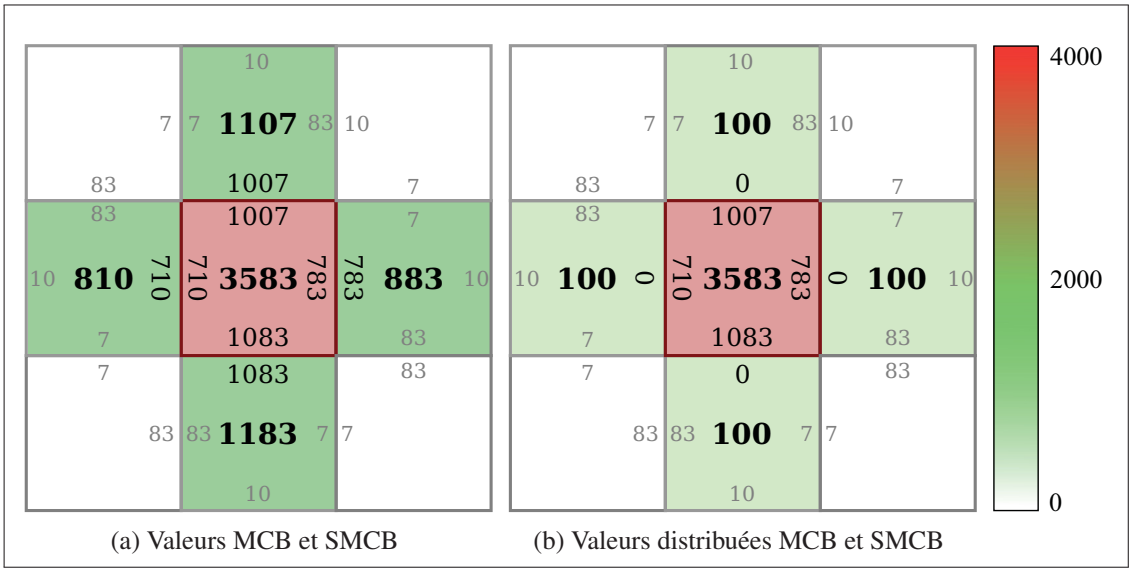


Figure 5.8 Exemple de la mesure des effets de bloc compensés par le mouvement, pour un vecteur de mouvement corrompu.

illustré à la figure 5.9a, dans une telle situation, la valeur du SMCB des blocs voisins de la détérioration visuelle est considérablement augmentée par la bordure limitrophe de l'erreur.

Pour résoudre ce problème, nous proposons le SDMCB, soit une version du SMCB où les bordures à valeurs élevées sont distribuées. La distribution de bordures est guidée par la valeur du SMCB. La valeur d'une bordure élevée est assignée au bloc avec le plus grand SMCB. La figure 5.9 montre le résultat de la distribution et l'algorithme 5.1 (p.69) démontre comment celle-ci est effectuée.

Cet algorithme de distribution de bordures, montré à l'algorithme 5.1 (p.69), est appliqué, en ordre décroissant de valeur de SMCB, sur tous les blocs dont la valeur de SMCB est



(a) Valeurs MCB et SMCB (b) Valeurs distribuées MCB et SMCB

Figure 5.9 Valeurs MCB des bordures et valeurs SMCB (au centre) des blocs avant 5.9a et après 5.9b la distribution de bordures.

supérieure à un seuil T_b. Le recours au seuil est justifié, car si la valeur du SMCB est trop basse, la distribution n'est pas précise et ne sert à rien. Le but de la distribution est d'éviter la propagation des effets de bloc importants aux blocs voisins. Le seuil T_b établit la valeur à partir de laquelle un bloc est considéré comme ayant des effets de bloc importants.

Algorithme 5.1 Distribution de bordures

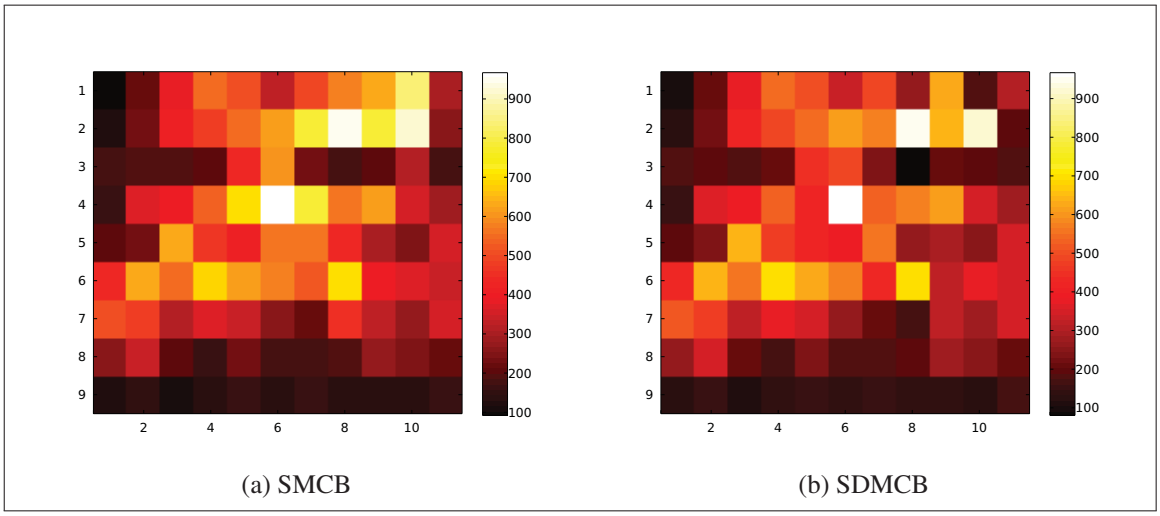
```
1   %Bordure supérieure
2   si SMCB(F,P,m,n) < SMCB(F,P,m,n−1) alors
3     MCB_N(F,P,m,n) = 0
4   sinon
5     MCB_S(F,P,m,n−1) = 0
6   fin si
7
8   %Bordure droite
9   si SMCB(F,P,m,n) < SMCB(F,P,m+1,n) alors
10    MCB_E(F,P,m,n) = 0
11  sinon
12    MCB_W(F,P,m+1,n) = 0
13  fin si
14
15  %Bordure inférieure
16  si SMCB(F,P,m,n) < SMCB(F,P,m,n+1) alors
17    MCB_S(F,P,m,n) = 0
18  sinon
19    MCB_N(F,P,m,n+1) = 0
20  fin si
21
22  %Bordure gauche
23  si SMCB(F,P,m,n) < SMCB(F,P,m−1,n) alors
24    MCB_W(F,P,m,n) = 0
25  sinon
26    MCB_E(F,P,m−1,n) = 0
27  fin si
```

5.2.2 Explication de l'approche MCB

L'approche MCB comporte deux composantes clés pour l'identification d'erreurs. La première est spatiale, l'analyse de bordures identifie les discontinuités entre les pixels en bordure de blocs. Toutefois, ces discontinuités pourraient appartenir à l'image. La seconde composante est temporelle et, dans ce cas, recherche ces discontinuités dans la trame précédente.

L'insertion ou la déformation de contenu causant un effet de bloc important est le seul cas problématique. Il est rare et l'effet de bloc doit être considérablement important pour

être considéré comme erroné. Nous présentons à la section 5.3 notre approche de détection d'erreurs à l'aide des valeurs de MCB.



(a) SMCB (b) SDMCB

Figure 5.10 Approches de détection d'erreurs basées sur le MCB, mesurées par rapport à la figure 5.3a. Pour l'approche SDMCB, un seuil $T_b = 900$ a été utilisé.

Nous présentons, à la figure 5.10, les résultats obtenus de l'utilisation des approches SMCB et SDMCB sur la trame erronée de la figure 5.3a (p. 58). On y constate en 5.10b une réduction importante des valeurs des blocs, non erronés, voisins aux macroblocs erronés (c.-à-d. SMCB $>$ 800). Rappelons que la propagation d'erreur due à l'usage des bordures extérieures cause cette augmentation des valeurs des macroblocs non erronés.

En ce qui a trait à la détection d'erreur, nous observons que le SDMCB détecte l'erreur la plus dérangeante de la figure 5.3a (p. 58). De plus, en regardant l'erreur identifiée à la figure 5.3d (p. 58), nous constatons que les valeurs les plus élevées de SDMCB (c.-à-d. SDMCB $>$ 800) sont assignées à des macroblocs qui ont subi une détérioration visuelle. Les valeurs basses de SDMCB (c.-à-d. SDMCB $<$ 200) sont assigné à des blocs qui ne contiennent pas d'erreurs. Certes, une zone d'incertitude existe entre 200 et 800, cependant, la détérioration visuelle contenue par les macroblocs de cette plage de valeurs n'est pas importante, et les gains de leur dissimulation sont minimes.

À première vue, la comparaison du SDMCB par rapport aux autres approches comme le résiduel et le différentiel **D** n'est pas évidente. Cependant, en regardant la zone d'incertitude de **D**, soit d'environ 1000 à 9000 (voir figure 5.4a (p. 60)), on constate qu'il n'est pas possible de distinguer la détérioration visuelle la plus dérangeante $\mathbf{D}_e$ à la variation entre les trames $\mathbf{D}_v$.

5.3 Approches sélectives de dissimulation de la détérioration visuelle

Dans cette section, nous sortons du contexte théorique de la détection d'erreurs et nous offrons une solution pratique de l'application du MCB. Cette solution a pour but d'améliorer la qualité visuelle des dissimulations effectuées par le décodeur de référence H.264. Dans le cadre de nos recherches, nous avons intégré notre solution dans le décodeur du logiciel de référence H.264/AVC JM, mais il est aussi concevable de le voir comme un module d'extension (*plugiciel*) destiné à d'autres décodeurs.

L'approche MCB assigne une valeur numérique à un bloc évalué en fonction de l'écart de son effet de bloc par rapport au bloc qui lui ressemble le plus dans la trame précédente. Nous affirmons, résultats à l'appui [2], que cette approche permet d'identifier les détériorations visuelles importantes dans une trame corrompue. La quantification de la notion d'une dégradation visuelle importante a été un problème colossal que nous avons eu à surmonter, lors de notre implémentation pratique.

Une solution simple est l'usage d'un seuil. Cette valeur numérique définit un point après lequel un bloc avec un certain SDMCB est considéré comme erroné. Ce type d'approche, employée par Superiori et al. (2007) ainsi que Ikuno (2007), ne tient pas compte du caractère variant du contenu d'une séquence vidéo.

L'effet de bloc est relatif à l'image dans laquelle il est perçu. Ceci implique que l'usage d'un seuil fixe ne sera pas en mesure d'identifier les erreurs dans des séquences où les caractéristiques spatiotemporelles divergent.

2. Voir le chapitre Résultats et analyse (p. 75).

De cette observation est née l'approche sélective. Comme son nom l'identique, cette approche repose sur la notion d'un choix. Ce changement de paradigme provient du constat suivant : dans un contexte pratique, la dissimulation est inutile si l'erreur ne peut être dissimulée. Il est préférable de se concentrer sur le contenu que nous sommes en mesure de dissimuler.

En nous basant sur le fait que nous cherchons à effectuer une dissimulation, nous prenons d'un côté une dissimulation traditionnelle, tels la trame calquée ou le calque des vecteurs de mouvement, et nous la comparons à la trame à évaluer. Par la suite, nous choisissons la plus petite valeur SDMCB, que ce soit pour une trame, comme à la section 5.3.1, ou pour un macrobloc, section 5.3.2.

Ces approches se comportent comme des seuils dynamiques capables de s'adapter non seulement au contenu, mais aussi à l'aptitude de dissimulation du macrobloc évalué. Les approches sélectives effectuent une dissimulation d'un macrobloc, seulement lorsque du contenu de qualité visuelle supérieure (jugée par le SDMCB) est disponible.

Une considération pratique importante est le temps de calcul requis par notre algorithme. Ce temps est restreint, vu les contraintes du *temps réel* ainsi que les limitations matérielles des appareils mobiles. Ce qui est dispendieux en terme de temps de calcul pour le MCB est la recherche des vecteurs de mouvement. Cependant, les récentes avancées dans ce domaine, telles PMVFAST (Tourapis et al., 2001) et UMHexagonS (Cai et al., 2009) réduisent de façon substantielle le temps de calcul requis pour trouver les vecteurs de mouvement. De plus, notons que dans ce contexte, le décodeur est informé par les entêtes RTP des paquets corrompus et mesure le SDMCB uniquement pour les trames qui en résultent.

5.3.1 Approche sélective au niveau de la trame

Notre première approche sélective repose sur le choix d'une trame. Lorsque des trames sont corrompues, le décodeur peut recourir à plusieurs trames pour la dissimulation : la tranche calquée, la tranche issue du calque des vecteurs de mouvement et la trame endommagée.

Le pointage SDMCB d'une trame est obtenu par la sommation des valeurs du SDMCB pour chacun de ses macroblocs. Cette sommation est décrite par la formule :

$$c_c = \sum_{m=0}^{\frac{W}{B}-1} \sum_{n=0}^{\frac{H}{B}-1} \text{SDMCB}(\mathbf{F}', \mathbf{P}, m, n)\,, \tag{5.11}$$

où $\mathbf{F}'$ est une trame dissimulée par le décodeur, soit une tranche calquée ou une tranche issue du calque des vecteurs de mouvement, etc. On mesure une seconde sommation des valeurs SDMCB pour chaque bloc d'une trame. Cette fois, nous utilisons la trame endommagée $\mathbf{E}$:

$$c_e = \sum_{m=0}^{\frac{W}{B}-1} \sum_{n=0}^{\frac{H}{B}-1} \text{SDMCB}(\mathbf{E}, \mathbf{P}, m, n)\,, \tag{5.12}$$

Par la suite, la trame avec le plus petit pointage est choisie comme dissimulation $\mathbf{F}^*$:

$$\mathbf{F}^* = \begin{cases} \mathbf{E}, & \text{si } c_e < c_c \\ \mathbf{F}', & \text{sinon} \end{cases}. \tag{5.13}$$

5.3.2 Approche sélective au niveau du macrobloc

Notre deuxième approche offre un niveau de granularité supérieur à celle de la première approche, en effectuant la sélection pour chaque macrobloc. Le but de cette approche est de tirer profit du gain mutuel de différentes approches en fonction des caractéristiques spatiotemporelles des régions d'images.

Dans certaines conditions, la tranche calquée est optimale ; dans d'autres, c'est la trame erronée ou bien le calque des vecteurs de mouvement. De cette approche résulte une trame de dissimulation formée d'un conglomérat de macroblocs optimaux (jugés selon le SDMCB) issus de différents macroblocs candidats pour la dissimulation.

La dissimulation résultant de la sélection d'un bloc parmi ceux aux indices (m,n) provenant de la trame corrompue $\mathbf{E}$ ou de la trame issue de la dissimulation par calquage de tranches $\mathbf{F}'$ est obtenue à l'aide de

$$\mathbf{F}^*_{x,y} = \begin{cases} \mathbf{E}_{x,y}, & \text{si SDMCB}(\mathbf{E},\mathbf{P},m,n) < \text{SDMCB}(\mathbf{F}',\mathbf{P},m,n) \\ \mathbf{F}'_{x,y}, & \text{sinon} \end{cases}, \tag{5.14}$$

pour tout $x \in mB + [0,B-1]$, tout $y \in nB + [0,B-1]$, et pour tout bloc de l'image (c.-à-d. $m \in I_m = [0, \frac{W}{B} - 1]$, et $n \in I_n = [0, \frac{H}{B} - 1]$).

Dans ce chapitre, nous avons présenté deux variantes d'un nouveau type d'algorithme sélectif apte à la dissimulation d'erreurs. Ces variantes agissent à deux différents niveaux de la hiérarchie de la vidéo numérique, soit ceux des macroblocs et ceux des trames. Pour guider la sélection, nous proposons un algorithme de détection d'erreurs fondé sur une nouvelle notion : les effets de bloc compensés par le mouvement. Cette dernière prend en compte une partie des changements *intertrame*. Ceux-ci, comme nous avons démontré, entravent la détection des algorithmes existants.

Le chapitre subséquent porte sur l'analyse des résultats expérimentaux obtenus à l'aide de notre banc d'essai, conçu pour valider les notions présentées dans ce chapitre.

CHAPITRE 6

RÉSULTATS DE SIMULATIONS ET ANALYSE

Dans le chapitre précédent, nous avons présenté les concepts théoriques des nouvelles approches sélectives proposées au niveau des tranches et au niveau des blocs. À l'aide de ces approches et dans des conditions de vidéo mobile, nous faisons état, dans ce chapitre, des essais pratiques réalisés pour valider les gains réalisables. Nous démontrons, non seulement, l'efficacité des approches sélectives proposées, mais aussi leur supériorité par rapport à l'approche de dissimulation utilisée par le décodeur inclus dans le logiciel de référence H.264/AVC JM.

Nous présentons d'abord nos hypothèses de validation à la section 6.1. Nous expliquons ensuite, à la section 6.2, les composantes de notre banc d'essai utilisées pour créer notre jeu de tests. Dans le but de valider nos hypothèses sur le décodeur, les taux de succès observés lors du décodage d'une séquence erronée sont présentés à la section 6.3. Un succès sera considéré lorsque le décodeur retournera une image sans planter. Finalement, à la section 6.4, nous présentons les résultats résultants de l'application de nos algorithmes de détection et de dissimulation d'erreurs sur les images décodées.

6.1 Hypothèses de validation

Commençons par la présentation des hypothèses posées dans le cadre de nos expérimentations. La figure 6.1, illustre les trames issues de la 726^{e} sous-séquence de nos données d'essai (expliqué plus en détail à la section suivante) où l'image erronée 6.1b correspond au résultat obtenu par le décodage d'une tranche corrompue.

La première hypothèse est la suivante : la trame qui précède l'erreur 6.1a ne contient jamais d'erreurs, une hypothèse posée aussi par (Superiori et al., 2007). Nous aurons recours à cette trame comme source de contenu pour la dissimulation d'erreurs de la trame 6.1b. La trame

(a) Trame précédente (b) Trame erronée

(c) Dissimulation par tranche calquée (d) Trame de référence

Figure 6.1 Les trames résultantes de la sous-séquence #726 du jeu de tests. (Séquence : Foreman, QP=16 BER=0.0008, FMO=Dispersé)

6.1c est reconstruite à l'aide du calquage de la tranche de la trame précédente 6.1a. Cette trame représente la solution d'un décodeur moderne, ce à quoi nous allons comparer nos dissimulations. La figure 6.1d est la trame de référence, qui n'a pas subi l'encodage. C'est la trame que nous utilisons pour mesurer le PSNR.

La seconde hypothèse se lit comme suit : seul le contenu des paquets est corrompu et non les entêtes. Cette hypothèse, aussi mentionnée lorsque nous présentons notre banc d'essai, simplifie considérablement le décodage des paquets. La petite taille des entêtes, par rapport

au contenu d'un paquet, implique qu'il est beaucoup moins probable que les entêtes soient corrompus. Pratiquement, les paquets aux entêtes corrompus seraient éliminés par les couches inférieures du modèle de communications.

La dernière hypothèse est celle-ci : plus d'une tranche est employée pour encoder un paquet. Cette hypothèse est réaliste, car l'usage de plusieurs tranches est pratique courante sur des réseaux non fiables. Les types d'ordonnancement de macroblocs à l'intérieur des tranches sont présentés dans la prochaine section.

6.2 Description des données d'essai

Afin de valider notre approche de détection et de dissimulation vidéo, nous avons construit un banc d'essai constitué d'un grand nombre de séquences vidéos encodées à divers débits et soumises à des taux d'erreurs similaires à ceux proposés dans l'ouvrage (Stockhammer et al., 2003). Ces erreurs découlent des conditions difficiles de transport de séquences vidéos H.264 vers des appareils évoluant sur des réseaux mobiles.

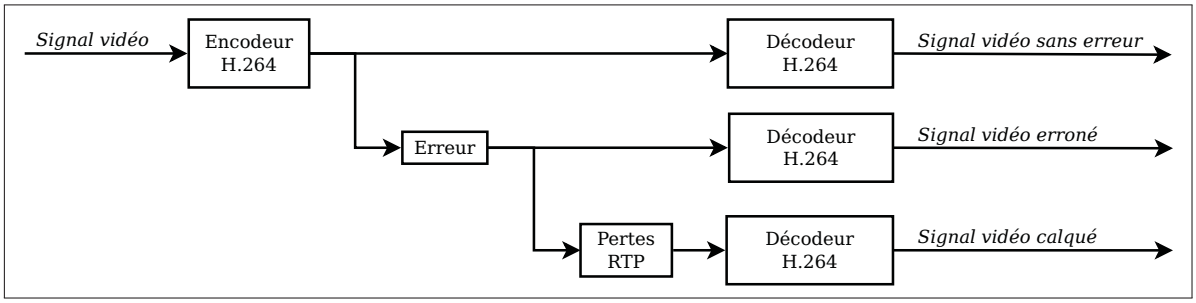


Figure 6.2 Diagramme des étapes du banc d'essai.

Nous présentons, à la figure 6.2 [1], les étapes de notre banc d'essai. Nous y observons qu'un signal vidéo est encodé, transmis, puis décodé trois fois. La première opération de décodage décode les données sans que celles-ci soient exposées à l'erreur. Cette étape est cruciale afin de déterminer la détérioration visuelle engendrée par l'encodage du signal. Dans la seconde opération, la séquence encodée est sujette à un motif d'erreur binaire (*Bit Error Pattern*)

1. Le diagramme de la figure 6.2 n'illustre qu'une partie du banc d'essai. Le diagramme complet se trouve à l'annexe I (p. 101).

gaussien. Les données corrompues sont envoyées au décodeur. S'il réussit à les décoder, celles-ci sont conservées, sinon la sous-séquence est rejetée. Dans la dernière opération de décodage, la sous-séquence corrompue est analysée par un module de détection de pertes RTP (*RTP loss*). Celui-ci identifie et retire les paquets RTP corrompus. Les données résultant de l'analyse de pertes RTP sont décodées. Lorsque le décodeur s'aperçoit d'un paquet manquant, il utilise une approche de dissimulation d'erreurs. Par exemple, il remplace la tranche manquante par son équivalent dans la trame précédente [2] (*slice copy*, traduit dans cet ouvrage par la locution tranche calquée).

Les séquences sont encodées et décodées à l'aide de la version 16.2 du logiciel de référence H.264/AVC JM (Joint Video Team (JVT) d'ISO/IEC MPEG et ITU-T VCEG, 2010). Ce dernier, n'étant pas un produit commercial, a pour objectif principal la démonstration de nouvelles fonctionnalités visant la norme H.264. Notre banc d'essai est constitué de sous-ensembles de quatre trames consécutives commençant à un emplacement aléatoire dans diverses séquences vidéos. Ces sous-ensembles débutent par une trame *intra* suivi de trois trames *inter* (IPPP). Nous retenons cinq sous-ensembles pour chaque séquence de résolution QCIF contenue dans l'ensemble de séquences de référence de l'Université l'État de l'Arizona (Arizona State University, 2010).

Dans le contexte d'applications mobiles, le débit est souvent fixe (typiquement entre 64 kb/s et 128 kb/s pour les séquences QCIF). Cependant, pour simplifier nos expérimentations, nous imposons plutôt un paramètre de quantification (QP) fixe. Ceci élimine les variations de la qualité visuelle dues aux changements de QP requis pour garder le débit fixe. Parmi les valeurs de 0 à 51 du paramètre de quantification, nous avons retenu : 16, 20, 24 et 28. Ces valeurs sont choisies, car elles représentent un intervalle de données plausibles pour une application mobile selon la bande passante disponible. De plus, nous n'avons pas retenu des QP trop élevés (c.-

2. Dans cet ouvrage, *slice copy* a été utilisé au détriment de *motion copy*, car aucune implémentation fonctionnelle de *motion copy* n'était disponible. Lors de l'utilisation de *motion copy*, nous avons éprouvé des problèmes avec les versions 15.1 à 16.2 du décodeur inclus dans le logiciel de référence H.264/AVC JM. Ces problèmes sont connus et confirmés par plusieurs forums Internet et catalogués dans le logiciel de suivi de problèmes Mantis relié au logiciel de référence H.264/AVC JM (Joint Video Team (JVT) d'ISO/IEC MPEG et ITU-T VCEG, 2009). C'est pour cette raison, quoique notre ouvrage soit aussi compatible avec *motion copy*, que nous avons testé nos solutions avec *slice copy*.

à-d. QP $>$ 30), car la quantité de bits requis pour encoder une trame QCIF est si minime qu'il arrive souvent que les taux d'erreurs utilisés ne parviennent pas à briser un seul bit de la trame.

Les taux d'erreur binaire (BER) utilisés sont : 0.0004, 0.0008, 0.0016, et 0.0032. Cet intervalle de taux est considérablement large et agressif par rapport aux intervalles d'erreurs retrouvés dans la littérature de la norme H.264 (Stockhammer et al., 2003). Nous avons choisi un tel intervalle pour obtenir des résultats témoignant des gains possibles dans un grand nombre de conditions d'erreur de transport. Le taux d'erreur utilisé est appliqué au niveau des bits et non pas au niveau des paquets (Wenger, 2003) ou des blocs (comme dans les ouvrages de dissimulation d'erreurs).

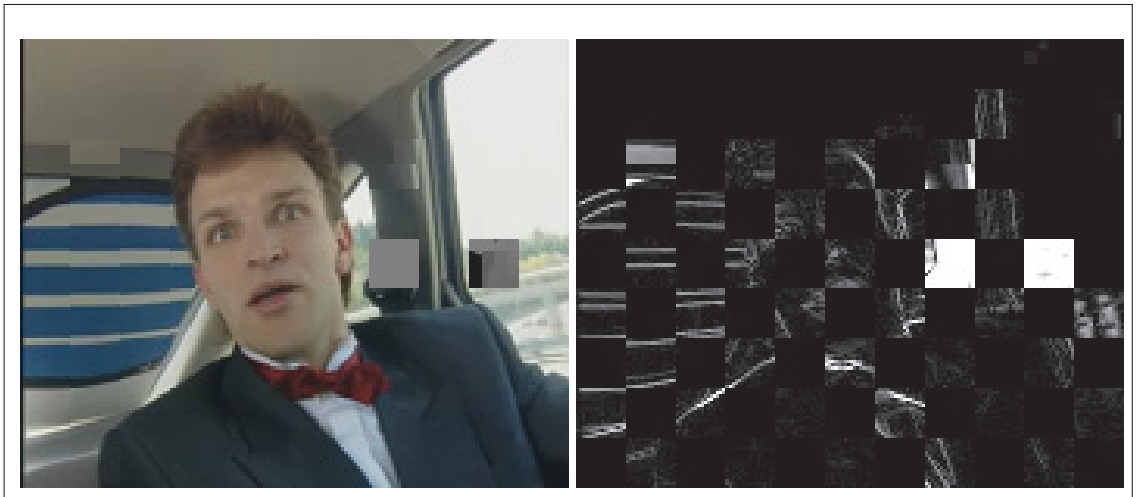
Seule la troisième trame (une trame P) est exposée à l'erreur. En ce qui concerne l'ordonnancement de macroblocs flexible (FMO), deux types sont employés : dispersé (*dispersed*) et entrelacé (*interlaced*). Tous deux sont composés de deux tranches. À la figure 6.3 et 6.4, nous présentons deux exemples d'erreurs pour chacun des types d'ordonnancement. Les sous-séquences sont encodées à 30 images par seconde, tandis que les autres paramètres sont propres au profil de base (*baseline*) défini dans la norme H.264. Le format de sortie est RTP et la résolution est QCIF, c'est-à-dire 176×144 pixels. Rappelons l'hypothèse, précédemment posée, que les entêtes NAL et RTP n'ont pas été exposés à l'erreur.

Notre banc d'essai est constitué d'un jeu de tests de 2720 sous-ensembles de trames. Plus précisément, nous avons 1360 trames dispersées et 1360 trames entrelacées. Ces 1360 trames proviennent de 17 séquences vidéos distinctes. Pour chacune d'elles, cinq sous-séquences distinctes ont été utilisées. Celles-ci ont été encodées avec nos quatre paramètres de quantification et exposées à nos quatre taux d'erreur ($17 \times 5 \times 4 \times 4 = 1360$).

En ce qui concerne les données d'entrainement utilisées pour établir les seuils, celles-ci proviennent du même banc d'essai, mais avec des trames de départs aléatoires, distinctes de celles utilisées pour mesurer les performances de nos solutions.

(a) Trame erronée (b) Erreur (amplifiée)

Figure 6.3 Exemple de l'erreur présente dans la sous-séquence #392 du jeu de tests. (Séquence : Carphone, QP=16 BER=0.0032, FMO=Entrelacé)



(a) Trame erronée (b) Erreur (amplifiée)

Figure 6.4 Exemple de l'erreur présente dans la sous-séquence #1752 du jeu de tests. (Séquence : Carphone, QP=16 BER=0.0032, FMO=Dispersé)

À la prochaine section, le banc d'essai sera employé pour déterminer la résilience du décodeur inclus dans le logiciel de référence H.264/AVC JM. La section 6.4 résume notre utilisation du banc d'essai pour valider les approches sélectives de détection et de dissimulation d'erreur proposées.

6.3 Analyse de la résilience aux erreurs du décodeur de référence H.264

Dans certaines conditions, le décodeur inclus dans le logiciel de référence H.264/AVC JM réussit à décoder des séquences binaires corrompues. Cette découverte, initiatrice de notre effort de recherche, a été réalisée très tôt dans nos expérimentations. Notons que, un décodage réussi est pour nous l'opération de décoder une tranche corrompue qui ne *plante* pas le décodeur, sans pour autant garantir l'intégrité du contenu en résulte. En d'autres mots, l'image issue du décodage d'une tranche corrompue va souvent contenir de la détérioration visuelle (voir figures 6.1b (p. 76), 6.3a (p. 80) et 6.4a (p. 80)). Le taux de décodages réussis varie, non seulement selon le taux d'erreur binaire appliqué à la séquence, mais aussi selon le paramètre de quantification (QP) utilisé lors de l'encodage. Ceci s'explique par le fait qu'un QP plus faible force l'encodeur à utiliser plus de bits pour encoder une trame ; et plus il y a de bits sujets à erreur, plus la probabilité de corruption de la trame augmente.

Les histogrammes de la figure 6.5 présentent les taux de décodages réussis observés en fonction des paramètres de quantification et du taux d'erreur sur les bits présents dans notre jeu de tests. Ces taux sont intéressants, car ils confirment que le décodage des données corrompues peut avoir un taux de réussite allant de 20 % à 70 %, selon les conditions. On peut concevoir une stratégie de résilience à l'erreur, où une prochaine génération de décodeurs seraient en mesure d'effectuer une tentative de décodage d'une tranche corrompue, dans un environnement contrôlé (possiblement à l'aide d'un second décodeur), afin d'éviter que le décodeur principal *plante*.

Lors du décodage d'une tranche corrompue, le décodeur inclus dans le logiciel de référence H.264/AVC JM fait la transition entre trois états distincts. Le résultat de ceux-ci est illustré à la figure 6.6. Le premier état est celui du décodage des bits de la tranche corrompue situés avant l'erreur. Cet état est équivalent au décodage normal d'une trame. Il est illustré par la partie supérieure, tout en noir (signifiant l'absence de différence), du différentiel 6.6b de la figure 6.6.

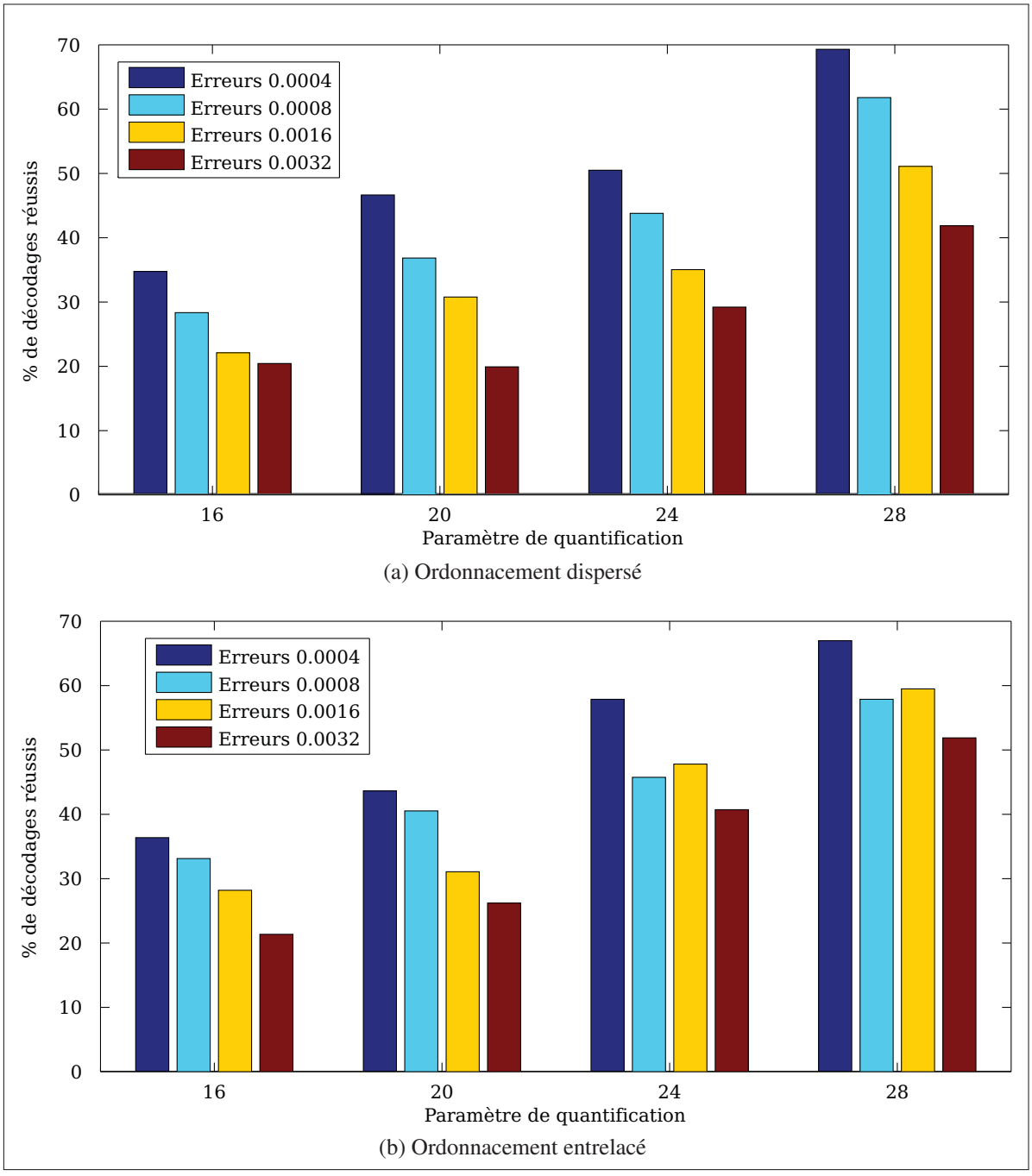
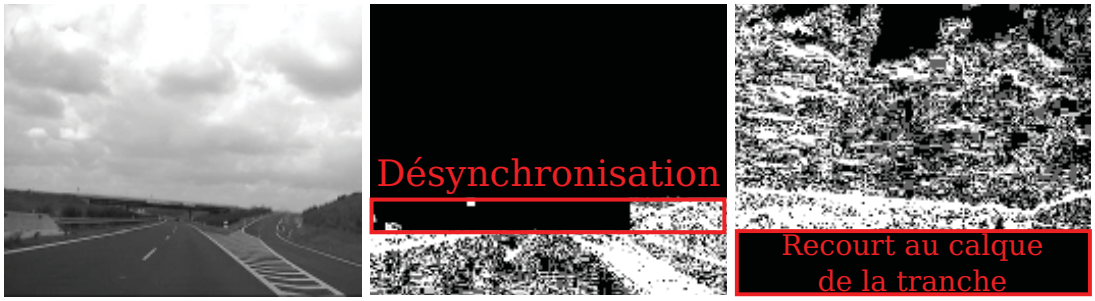


Figure 6.5 Histogrammes des pourcentages de décodages réussis en fonction du paramètre de quantification utilisé lors de l'encodage ainsi que du taux d'erreurs sur les bits (BER).

Par la suite, le décodeur décode le premier bit corrompu. Ceci cause une désynchronisation des codes binaires de l'encodage entropique. Même si les autres bits de la séquence sont



(a) Trame Corrompue

(b) Différentiel entre (a) et la trame non-erronée

(c) Différentiel entre (a) et la trame précédente

Figure 6.6 Exemple du comportement du décodeur. En 6.6b, le différentiel de 6.6a, par rapport à la trame codée/décodée sans erreur, est utilisé pour démontrer la désynchronisation. Le différentiel entre 6.6a et la trame précédente permet de démontrer, en 6.6c, l'utilisation du calque de tranche.

valides, la désynchronisation fait en sorte qu'ils sont associés à la mauvaise valeur dans la table de référence CAVLC. Le décodeur interprète ces mauvaises valeurs, ce qui entraine une détérioration visuelle dans l'image. Cette détérioration peut se manifester fortement, comme c'est le cas dans l'image de la figure 6.1b, ou être presque imperceptible, comme c'est le cas dans la figure 6.6a. Afin de mieux voir la détérioration visuelle de la figure 6.6a, elle est encadrée à la figure 6.6b. C'est dans ce second état que le décodeur est le plus vulnérable aux défaillances.

Le second état se termine lorsque le décodeur cesse d'insérer de la détérioration visuelle dans l'image et utilise le contenu provenant de la tranche précédente. Ce changement est illustré en 6.6c. Ici, on suppose que le décodeur est complètement désynchronisé ; il ne sait que faire de ce qu'il décode ; il applique donc le contenu des blocs précédents, ce qui, grâce à la corrélation temporelle, est le contenu de remplacement correct s'il y a absence de mouvement. De plus, notons que dans l'image 6.6a, il y a très peu d'effets de blocs causés par la désynchronisation. Ceci est dû au filtre antibloc. Lorsqu'il est appliqué sur la trame, il identifie et lisse les effets de blocs. Ceci explique aussi pourquoi, aux figures 6.3a et 6.3b, l'erreur se répand à la tranche non erronée. Dans ce cas, le filtre antibloc lisse le bloc erroné, ce qui répand l'erreur dans les blocs avoisinants.

6.4 Analyse de l'approche sélective

Comme démontré à la section précédente, l'image résultant d'une tranche corrompue peut souvent être décodée. Cependant, celle-ci possède, à cause de la désynchronisation du décodeur lors du décodage, une détérioration visuelle plus ou moins importante. Elle peut toutefois être utilisée pour guider un algorithme de dissimulation d'erreurs grâce à un nouveau type d'algorithme, proposé dans ce mémoire, permettant la détection de la détérioration visuelle.

L'algorithme de détection présenté dans cet ouvrage est basé sur la mesure des effets de blocs compensés par le mouvement (MCB), et détecte la détérioration visuelle causant des effets de blocs absents de la trame précédente. La prémisse de notre approche sélective est de déterminer entre la tranche calquée (figure 6.1c (p. 76)) et la tranche corrompue (figure 6.1b (p. 76)), celle qui produirait la meilleure dissimulation (c.-à-d. qui produirait le meilleur PSNR par rapport à la trame de référence). Cette opération peut être accomplie à deux niveaux : celui de la trame ou celui des blocs. Nous présenterons tout d'abord la configuration de notre banc d'essai conçu pour mesurer l'approche sélective. Par la suite, la sous-section 6.4.1 présente les résultats obtenus avec une approche par trames et la sous-section 6.4.2, les résultats d'une approche par blocs.

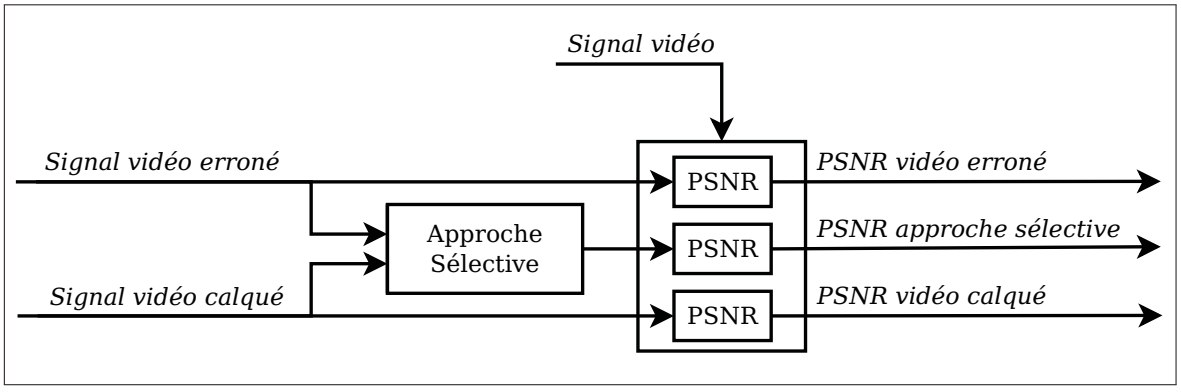


Figure 6.7 Configuration du banc d'essai pour mesurer l'approche sélective.

Tout d'abord, à l'aide du PSNR comme discriminant, nous mesurons, par rapport au signal vidéo original, la qualité visuelle des dissimulations pour la tranche calquée et la tranche

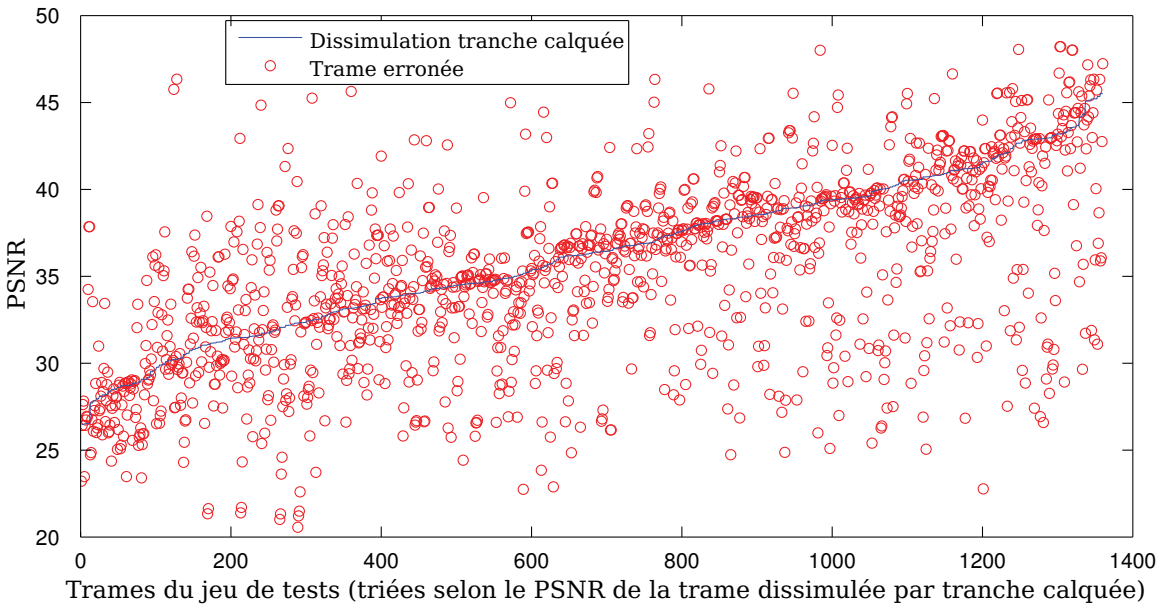
corrompue. La figure 6.7 [3] démontre la configuration élaborée pour obtenir ces mesures. Cette configuration s'ajoute à la suite de celle présentée à la figure 6.2 (p. 77). On y constate que le PSNR est mesuré par rapport au signal vidéo de référence et qu'il est mesuré pour l'image résultant du décodage de la tranche corrompue, pour le candidat de dissimulation basée sur la tranche calquée et pour le résultat des approches sélectives basées sur la trame et les blocs.

Par la suite, les résultats mesurés sur nos données d'essai sont présentés à la figure 6.8. On y remarque une importante quantité de trames et de blocs erronés qui possèdent un PSNR supérieur aux trames calquée. Cela s'explique par le fait que, dans ces cas, la détérioration visuelle engendrée par la désynchronisation du décodeur est moins importante que la variation temporelle par rapport à la trame précédente, ce qui pénalise la tranche calquée. En lien avec les trois états d'un décodeur traitant une tranche corrompue, nous pouvons conclure que, dans l'état 1, la tranche endommagée va produire un meilleur résultat que le calquage de la tranche. Dans l'état 2, la tranche endommagée va produire un résultat inférieur à la tranche calquée. Finalement, à l'état 3, les deux sont équivalents. La proportion de blocs compris dans les états 1 et 2, l'intensité de la détérioration produite par le décodeur ainsi que la variation temporelle par rapport à la trame précédente sont les facteurs qui déterminent laquelle des deux trames constituera la meilleure alternative.

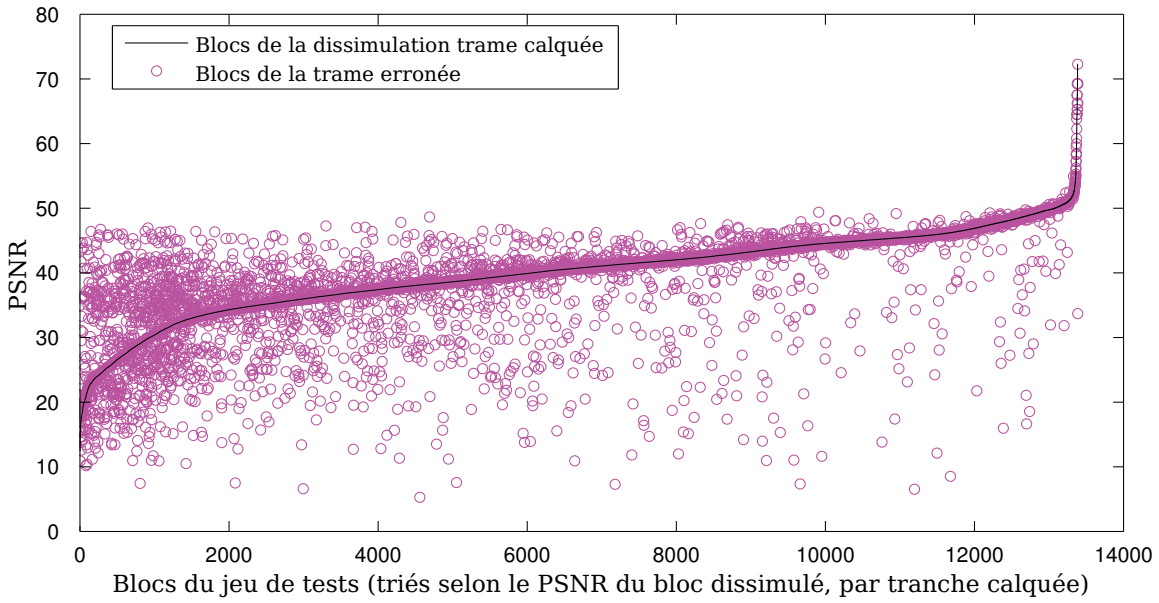
Plus précisément, les mesures obtenues de notre banc d'essai nous permettent de conclure que l'image résultant du décodage de la tranche corrompue offre un PSNR plus élevé dans 44 % des cas pour l'ordonnancement dispersé et 49 % pour l'entrelacé. Pour ces trames, on observe respectivement des gains moyens de 2.12 dB et 1.68 dB par rapport à leurs trames équivalentes calquées. De manière similaire, les blocs issus de trames corrompues sont supérieurs dans 42 % des cas pour l'ordonnancement dispersé et 49 % pour l'entrelacé. Ces blocs produisent, pour les approches d'ordonnancement précédentes, des gains moyens de 1.44 dB et 1.02 dB.

Bref, nous avons déjà démontré qu'il est possible de réussir à décoder des tranches erronées. Ce que nous venons de démontrer, dans cette section, est qu'une quantité importante des

3. Le diagramme de la figure 6.7 n'illustre qu'une partie du banc d'essai. Le diagramme complet se trouve à l'annexe I (p. 101).



(a) Distribution des trames corrompues (cercles rouges) par rapport aux trames calquées (ligne bleue)



(b) Distribution des blocs issus des trames corrompues (cercles magenta) par rapport aux blocs issus des trames calquées (ligne noire). À des fins de visualisation, seulement un bloc sur dix est affiché.

Figure 6.8 Visualisation de la distribution des valeurs du PSNR de la trame erronée et de la dissimulation par tranche calquée.

images résultant d'un tel décodage exhibent une fidélité visuelle supérieure au calque de cette tranche (approche couramment employée par le décodeur fourni avec le logiciel de référence H.264/AVC JM).

6.4.1 Approche sélective par trame

Maintenant, toujours avec la même configuration du banc d'essai, mesurons l'aptitude de la somme du SDMCB, pour l'ensemble d'une trame, à identifier le meilleur candidat entre la trame corrompue et la dissimulation par tranche calquée. Pour ce faire, nous utilisons une taille de bloc $B = 16$ (vu notre indépendance au train de bits, nous ne connaissons pas la taille exacte des blocs) et un seuil d'effet de bloc $T_b = 5000$. La taille des macroblocs étant 16×16, ceci nous permet de conclure que s'il y a erreur, les effets de blocs vont souvent se manifester en bordure de macroblocs. La valeur du seuil T_b est obtenue avec des données d'essai provenant du même jeu de test, mais avec des trames de départ différentes.

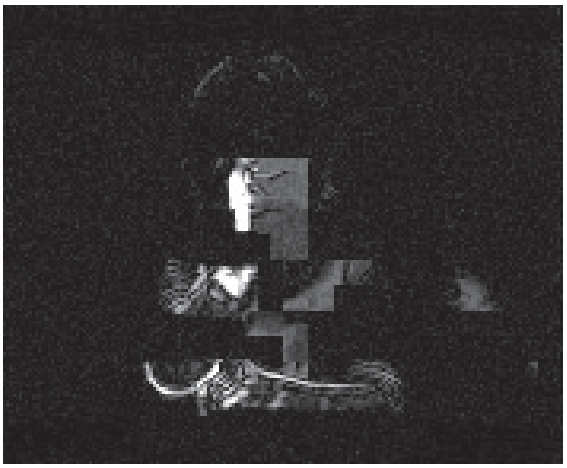
Deux exemples de trames et de leur pointage MCB sont présentés aux figures 6.9 et 6.10. Dans les deux cas, on constate que l'erreur de la trame endommagée (6.9c et 6.10c) est moins nuisible que celle de la trame calquée (6.9d et 6.10d). Ceci se reflète dans le pointage SDMCB. Ce dernier permet d'identifier la trame avec le moins d'erreurs, dans un contexte où la trame de référence n'est pas disponible.

Les résultats obtenus sont présentés à la figure 6.11. La figure 6.11a montre que l'approche sélective, dans le cadre d'un ordonnancement dispersé, fait le bon choix dans 81 % des cas. Pour ceux-ci, un gain moyen de 1.98 dB est mesuré, tandis que pour l'ensemble de la séquence le gain moyen est de 0.72 dB. Les choix de l'approche sélective offrent un PSNR supérieur aux trames dissimulées par calquage de tranche dans 31 % des cas. Pour la figure 6.11b, la bonne trame est choisie dans 86 % des cas. Il en résulte un gain moyen de 1.24 dB pour celles-ci. Sur l'ensemble des trames, le gain moyen est de 0.65 dB et un PSNR supérieur au calquage de trame est obtenu dans 43 % des cas.

(a) Trame endommagée ($SDMCB = 9983$)

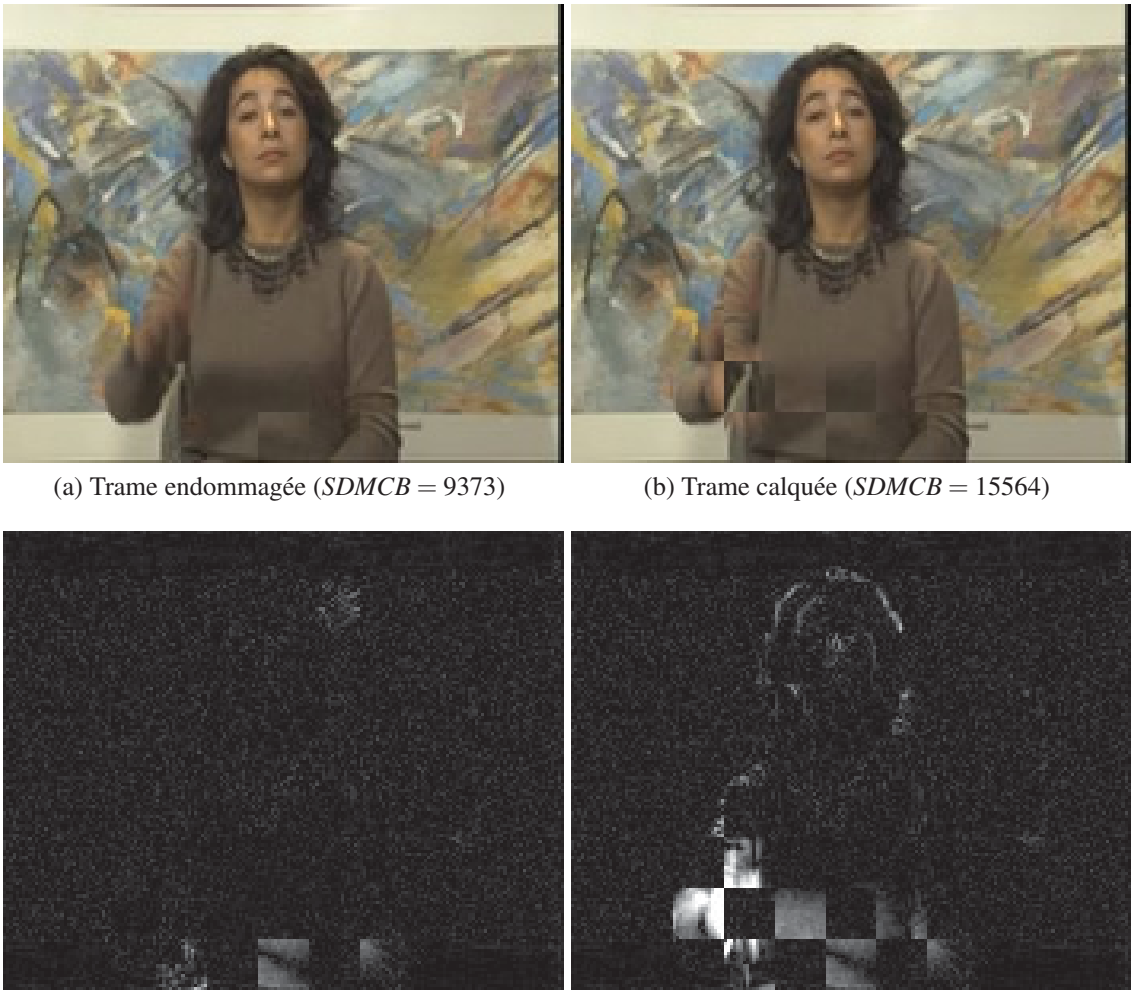
(b) Trame calquée ($SDMCB = 13142$)

(c) Différentiel trame erronée et trame de référence



(d) Différentiel trame calquée et trame de référence

Figure 6.9 Visualisation des pointages SDMCB obtenus par rapport à l'erreur présente dans les trames. (Séquence : Silent, QP=20 BER=0.0008, FMO=Dispersé)



(a) Trame endommagée ($SDMCB = 9373$)

(b) Trame calquée ($SDMCB = 15564$)

(c) Différentiel trame erronée et trame de référence

(d) Différentiel trame calquée et trame de référence

Figure 6.10 Visualisation des pointages SDMCB obtenus par rapport à l'erreur présente dans les trames. (Séquence : Silent, QP=16 BER=0.0004, FMO=Entrelacé)

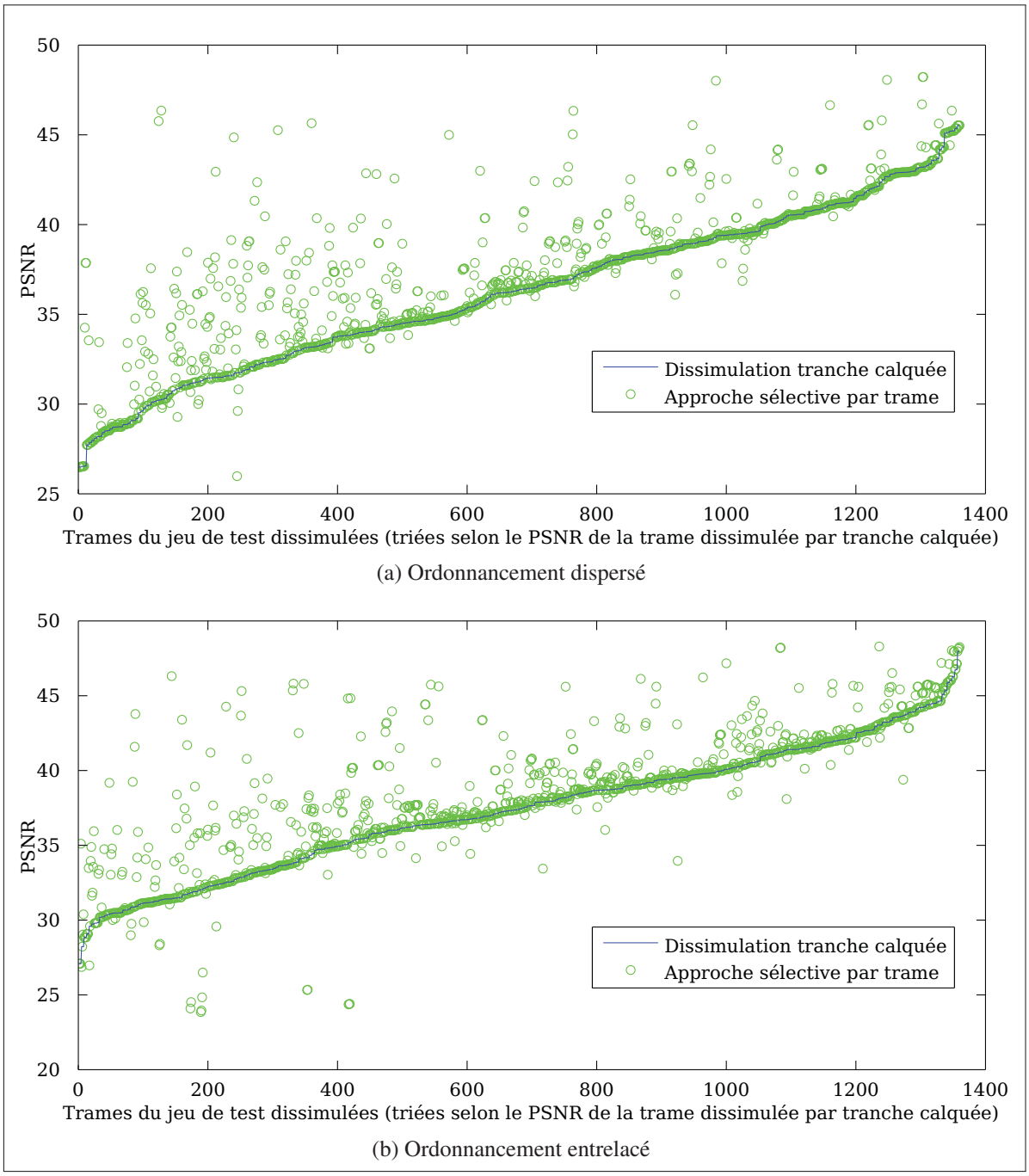


Figure 6.11 Visualisation de la distribution du PSNR de l'approche sélective par trame (cercles verts) à l'égard de la dissimulation par tranche calquée (ligne bleue).

6.4.2 Approche sélective par blocs

Le niveau de granularité plus fin de l'approche sélective par bloc est son attrait principal. En remplaçant uniquement les blocs endommagés, on maximise l'usage des données correctement décodées.

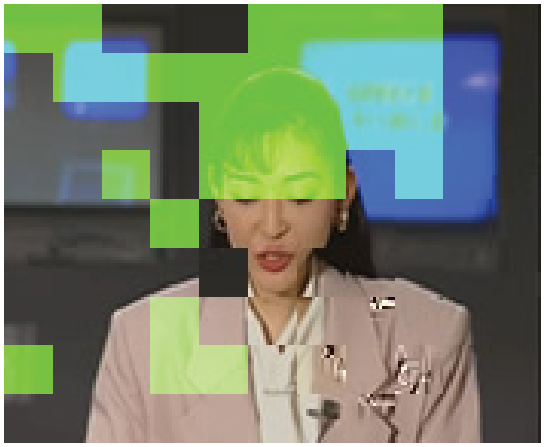
Sans altérer la configuration de notre banc de test, nous mesurons l'aptitude du MCB à identifier le meilleur bloc candidat entre ceux provenant de la trame corrompue et ceux, de la dissimulation par tranche calquée. Pour ce faire, nous réutilisons la taille de bloc $B = 16$ et le seuil d'effet de bloc $T_b = 5000$.

Aux figures 6.12 et 6.13, on constate les gains en PSNR issus de l'approche sélective. Ces gains proviennent des choix de blocs provenant de la trame calquée (6.12d, 6.13d) et de la trame endommagée (6.12c, 6.13c). Dans les deux cas, l'approche sélective écarte les blocs possédant une détérioration visuelle en faveur du bloc analogue dans l'autre trame.

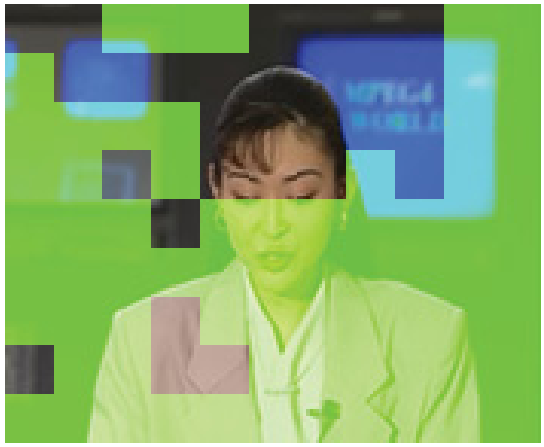
Les résultats obtenus sont présentés à la figure 6.14. La figure 6.14a démontre que l'approche sélective par bloc choisit le bon bloc dans 88 % de cas, ce qui produit un gain moyen de 0.86 dB sur l'ensemble des trames avec un ordonnancement de type dispersé. Pour l'ordonnancement entrelacé (figure 6.14b), l'approche sélective par bloc effectue le bon choix dans 91 % des cas, permettant un gain moyen de 0.69 dB.

(a) Trame endommagée (PSNR = 25.35 dB)

(b) Trame calquée (PSNR = 39.84 dB)



(c) Blocs choisis dans la trame endommagée



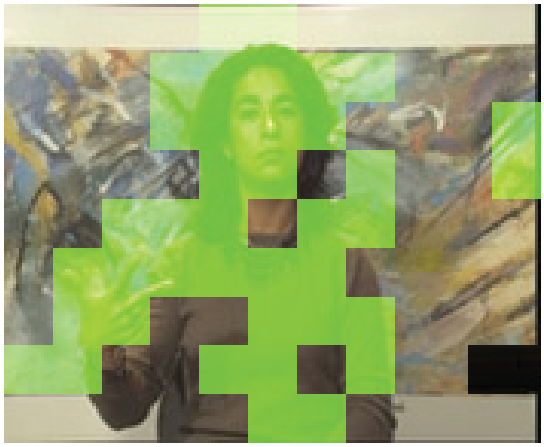
(d) Blocs choisis dans la trame calquée

(e) Trame sélective par blocs (PSNR = 44.07 dB)

Figure 6.12 Visualisation des blocs choisis dans la trame corrompue et la trame calquée par l'approche sélective par bloc. (Séquence : Akiyo, QP=16 BER=0.0008, FMO=Dispersé)

(a) Trame endommagée (PSNR = 24.76 dB)

(b) Trame calquée (PSNR = 36.38 dB)



(c) Blocs choisis dans la trame endommagée

(d) Blocs choisis dans la trame calquée

(e) Trame sélective par blocs (PSNR = 38.94 dB)

Figure 6.13 Visualisation des blocs choisis dans la trame corrompue et la trame calquée par l'approche sélective par bloc. (Séquence : Silent, QP=16 BER=0.0008, FMO=Entrelacé)

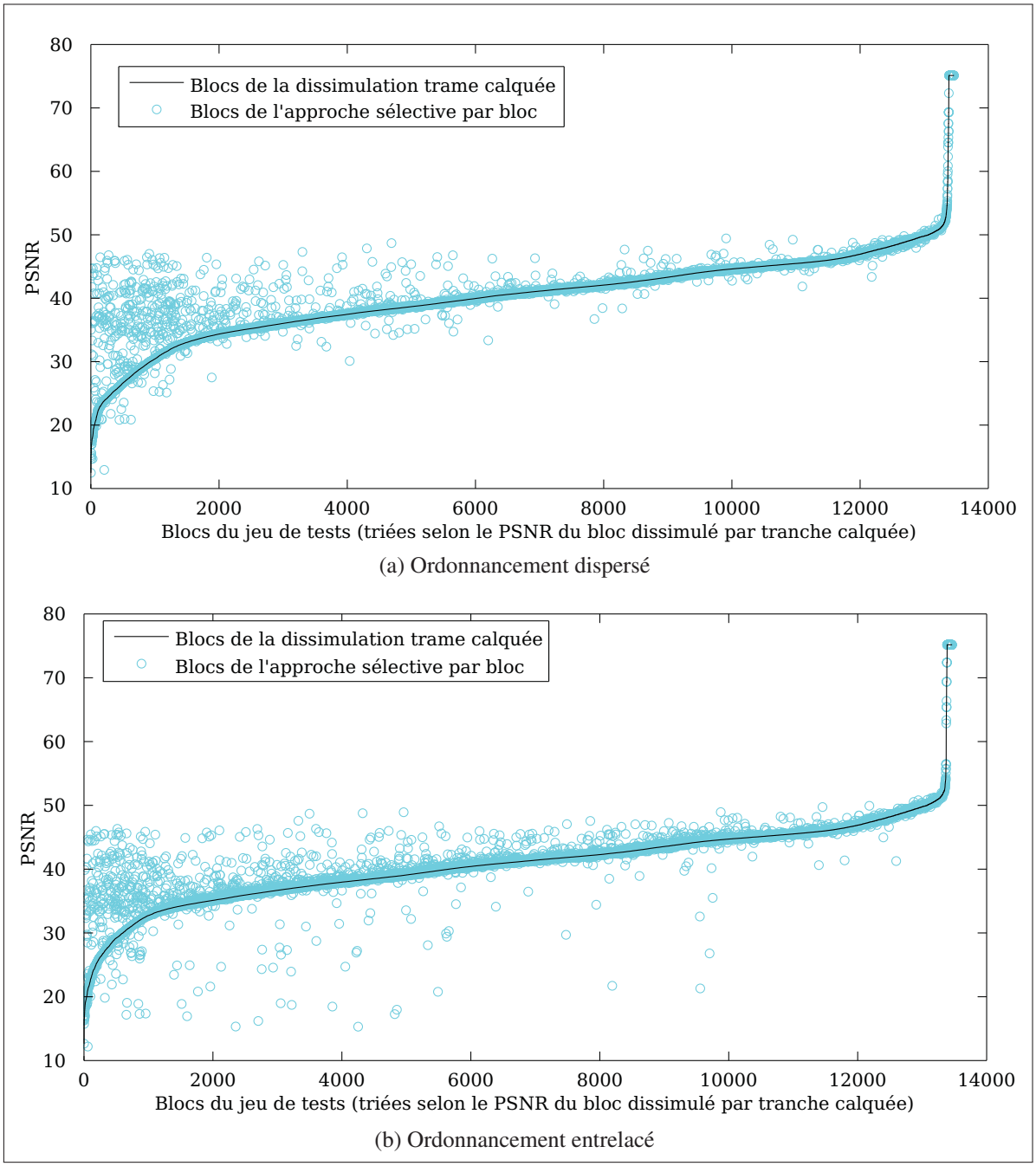


Figure 6.14 Visualisation de la distribution du PSNR issue des blocs résultants de l'approche sélective par bloc (cercles cyan) à l'égard de ceux produits par la dissimulation par tranche calquée (ligne bleue).

À l'aide de la figure 6.15, nous sommes en mesure d'analyser la distribution des PSNR résultant de la trame erronée, de la dissimulation par tranche calquée et des approches

sélectives. Cette figure expose la variation du nombre de trames selon des intervalles de 5 dB de PSNR. Ces intervalles s'étalent de 20 dB à 50 dB, permettant ainsi de visualiser la variation de la qualité visuelle issue de chaque approche.

Commençons l'analyse en soulignant la présence d'une importante quantité de trames erronées dans les intervalles de faible qualité, où le PSNR est inférieur à 30 (les intervalles centrés autour de 20 et 25). Pour ces intervalles, nous constatons, autant dans l'histogramme de la figure 6.15a que 6.15b, que la majorité de ces trames sont dissimulées par les tranches calquées et les approches sélectives.

Poursuivons l'analyse des approches sélectives avec le constat suivant : pour l'intervalle centré autour de 30 dB, le nombre de trames liées aux approches sélectives est inférieur à celui des trames liées à la dissimulation par tranche calquée. Cette observation n'est pas due à une perte de qualité, mais bien à une augmentation de la qualité, car on retrouve ces trames dans les intervalles supérieurs. Pour ces intervalles, les approches sélectives produisent un plus grand nombre de trames que la dissimulation par tranche calquée.

Notons aussi que plus le taux d'erreurs augmente, plus l'écart entre l'approche sélective et la trame calquée diminue. Ce qui est logique, car plus l'erreur augmente, moins il y a d'information utile dans la trame corrompue et plus la détérioration visuelle sera importante. Il est donc crucial, pour qu'une trame erronée soit supérieure à une trame calquée, que la détérioration visuelle soit moindre que la variation *intertrame*.

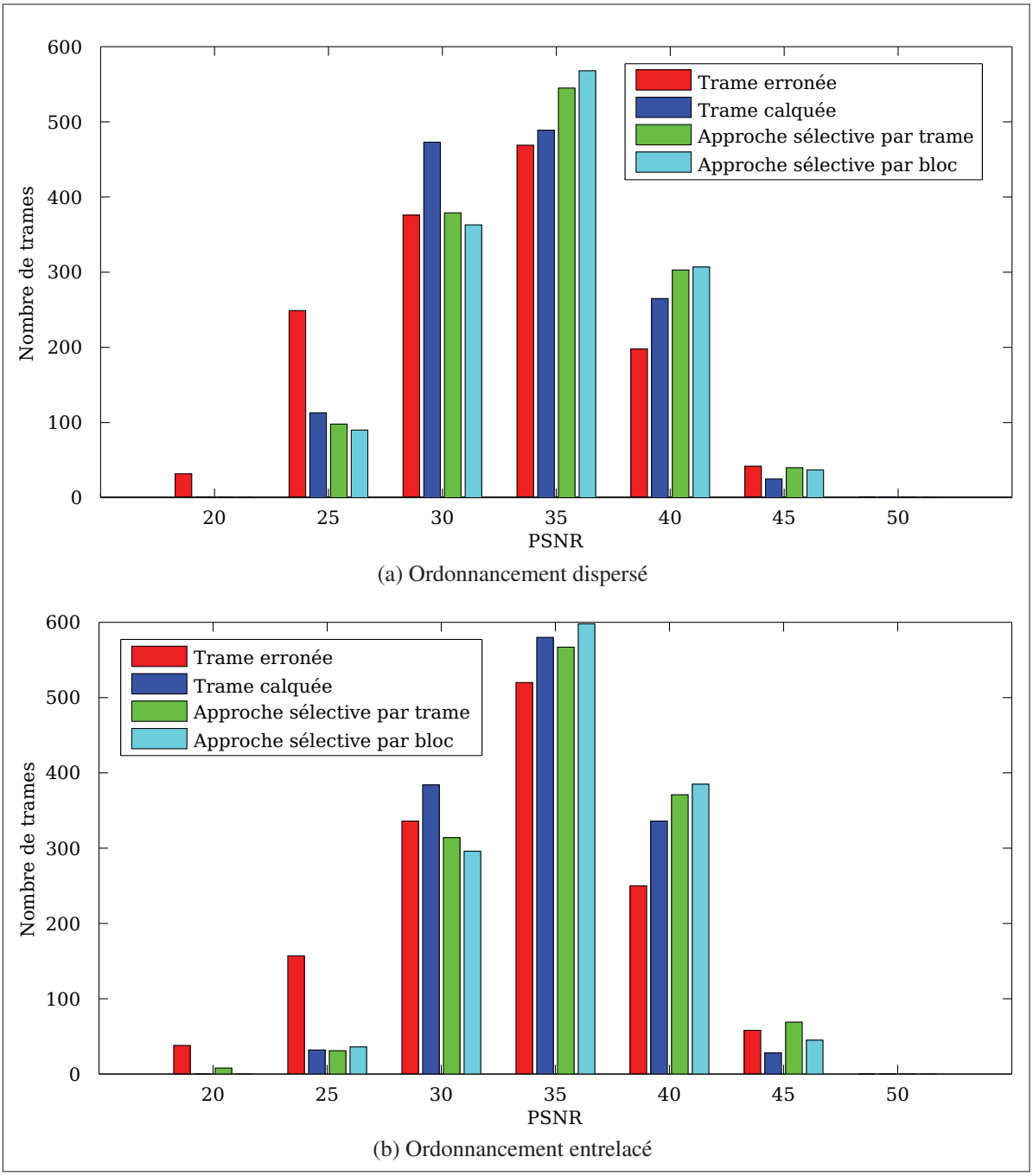
L'analyse des figures 6.11 et 6.14 nous démontre que notre approche est plus efficace avec l'ordonnancement flexible de macroblocs de type dispersé que celui de type entrelacé. Ce fait est mis en évidence à l'annexe II « Résultats détaillés de simulations ». Ce phénomène s'explique par le nombre de blocs voisins non erronés au bloc erroné, qui varie selon le type d'ordonnancement utilisé. Pour l'ordonnancement dispersé, les quatre voisins d'un bloc erroné ne sont pas erronés. Tandis que pour l'ordonnancement entrelacé, seulement deux voisins ne sont pas erronés. Ceci réduit les effets de blocs mesurés par le MCB.

Grâce à ces observations, nous sommes en mesure de conclure que les approches sélectives produisent de meilleurs résultats en exploitant les trames erronées de haute qualité visuelle, tout en évitant les trames erronées de mauvaise qualité. En ce qui a trait à la comparaison des approches sélectives, l'approche sélective par bloc offre un plus grand nombre de trames de qualité visuelle pour les intervalles de 35 dB et 40 dB vis-à-vis l'approche par trame. Cependant, l'approche sélective par trame se reprend dans l'intervalle de 45 dB. Cette reprise est attribuée au fait que le remplacement de blocs sporadiques dans une trame peut engendrer une perte de qualité causée par le mauvais arrimage des valeurs de pixels en frontière de bloc. Cette légère détérioration réduit le nombre de trames de très haute qualité. Toutefois, à des niveaux de 40dB et plus, la différence de qualité est souvent imperceptible. Pour éviter d'avoir à manipuler des valeurs de PSNR infinies, nous avons saturé le PSNR à 75 dB. Cette valeur a été choisie, car elle surpassait l'ensemble des valeurs mesurées, comme on peut le constater à la figure 6.14.

Tableau 6.1 Résumé du PSNR moyen obtenu par les diverses approches présentées dans cette section sur le jeu de tests.

Approche	**PSNR Moyen dispersé (dB)**	**PSNR Moyen entrelacé (dB)**
Encodée (sans erreur)	41.22	41.20
Sélective par bloc (avec référence)	37.69	38.53
Sélective par tranche (avec référence)	37.12	38.08
Sélective par bloc	**37.03**	**37.95**
Sélective par tranche	**36.90**	**37.91**
Dissimulation tranche calquée	36.18	37.26
Trame corrompue	35.07	36.08

Le tableau 6.1 résume les résultats présentés dans cette section. De par ces résultats, nous sommes en mesure de conclure deux choses. La première est que les trames issues du décodage de séquences corrompues possèdent une qualité visuelle importante et exploitable. La seconde est que les approches sélectives issues de nos travaux et appliquées à un tel décodage peuvent guider la détection d'erreurs et servir à l'amélioration de la dissimulation. De plus, le tableau valide que nos algorithmes de sélection sont en mesure d'alterner efficacement entre la trame



(a) Ordonnancement dispersé

(b) Ordonnancement entrelacé

Figure 6.15 Histogrammes des PSNR des trames, avec des intervalles de 5 dB, centrés à chaque incrément de 5 dB, à partir de 20 dB. Les trames considérées sont celles résultantes : du décodage de la trame erronée (rouge), du calquage de tranche (bleu), de l'approche sélective par tranche (vert) ainsi que l'approche sélective par bloc (cyan).

calquée et la trame corrompue afin d'offrir une dissimulation supérieure à celle offerte par le calquage de trame. En derniers lieux, les valeurs obtenues avec une sélection guidée par la trame de référence sont présentées dans le tableau 6.1. Ceci a pour but de relativiser les résultats de nos approches sélectives sans référence.

Les résultats présentés dans ce chapitre sont des moyennes pour l'ensemble des séquences, des taux d'erreurs binaires et des QP utilisés. Ils démontrent d'une façon concise et efficace les gains issus de l'usage des approches sélectives. Toutefois, un lecteur assidu désirant une analyse plus pointue dans le but de connaitre les gains engendrés selon divers paramètres est référé à l'annexe II « Résultats détaillés de simulations » (p. 102), pour une décomposition des résultats présentés ici. On y retrouve les résultats obtenus pour les séquences *Akiyo*, *Carphone*, *Coastguard* et *Foreman*. De plus, les résultats obtenus pour chaque taux d'erreurs binaires et chaque QP y sont présentés.

CONCLUSION

Dans ce mémoire, nous avons présenté, tout d'abord, les notions de base sur la norme H.264 et son transport. Par la suite, nous avons étudié la détérioration visuelle issue du décodage de trames corrompues. Une revue de la littérature a aussi été effectuée, portant non seulement sur les approches de détection de la détérioration visuelle, mais aussi sur l'analyse syntaxique de vidéos encodés et le *Joint source channel decoding*.

Par ailleurs, nous avons présenté une nouvelle mesure permettant d'identifier les effets de bloc propres à la détérioration visuelle. Nos analyses des simulations permettent les conclusions suivantes :

– Il est possible de décoder des trames corrompues. La probabilité d'un décodage réussi varie de 20 % à 70 % selon les paramètres d'encodage et le taux d'erreurs subi lors du transport.
– Les trames endommagées peuvent offrir une dissimulation de meilleure qualité que le calque de la trame. Pour notre banc d'essai, 44 % des séquences démontraient un PSNR supérieur au calque de la trame. Ce pourcentage peut fluctuer selon la variation *intertrame* et l'importance de la détérioration visuelle présente dans la trame.
– La mesure des effets de bloc compensés par le mouvement permet de départager entre deux candidats de dissimulation, lequel possède le moins de détérioration visuelle importante. Pour notre banc d'essai, lorsque évaluée en fonction des trames de références, l'approche sélective a effectué le bon choix, par rapport à une méthode avec référence, dans 81 % et 86 % des cas, selon le FMO, pour l'approche sélective par trame et, dans 88 % et 91 % des cas, selon le FMO, pour celle par bloc.

L'algorithme réalisé répond aux objectifs du projet. Comme démontré, il est capable d'identifier et de dissimuler la détérioration visuelle. Il offre des gains moyens de 0.72 dB et de 0.65 dB, selon le FMO, pour l'approche sélective par trame et de 0.86 dB et de 0.69dB, selon le FMO, pour celle par bloc, par rapport au calquage de la trame. De plus, il n'utilise que l'information disponible dans le domaine des pixels.

Quoique la solution proposée offre des résultats intéressants, il reste encore beaucoup de possibilités d'amélioration, comme, par exemple, l'implémentation de notre solution dans un décodeur H.264 commercial. Il serait aussi intéressant d'effectuer l'intégration avec d'autres algorithmes de dissimulation, tel le calque des vecteurs de mouvements.

De plus, il serait aussi possible de combiner notre approche avec de l'information provenant des paramètres d'encodage, tels l'ordonnancement des macroblocs, le nombre de tranches, etc. Ce type d'information pourrait améliorer l'efficacité du MCB. Ce type d'approche n'a pas été étudié dans cet ouvrage, car notre objectif était de travailler uniquement dans le domaine des pixels.

Sous leur forme actuelle, les concepts présentés dans cet ouvrage peuvent non seulement améliorer la dissimulation chez des décodeurs H.264, mais aussi servir d'outils d'analyse de la détérioration visuelle sans référence. Ce genre d'outils peuvent être insérés dans des décodeurs numériques ou des appareils mobiles, comme ceux offerts par les opérateurs comme Rogers, Telus, Bell et Vidéotron, et faire état de la présence de détérioration visuelle s'il y a lieu.

Tout cela réuni fait en sorte que, quoique notre effort de recherche se termine, les contributions issues de cet ouvrage ont un potentiel important de développement et d'évolution dans d'autres projets de recherche ou des initiatives commerciales.

ANNEXE I

CONFIGURATION DU BANC D'ESSAI

La figure I.1 décrit les différentes étapes de traitement de notre banc d'essai. Ce diagramme permet d'établir le lien entre les figures 6.2 et 6.7 de la section « Résultats de simulations et analyse ».

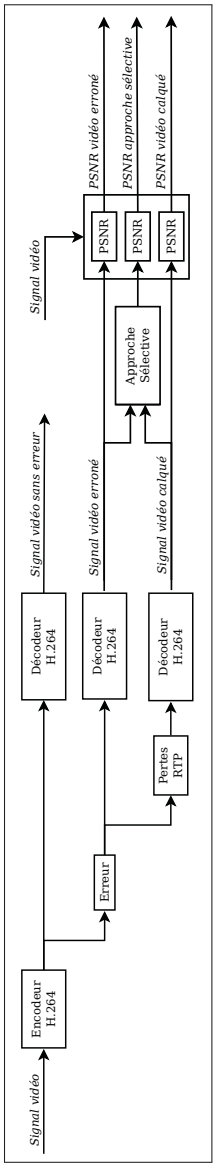


Figure I.1 Configuration du banc d'essai utilisé pour réaliser l'expérimentation.

ANNEXE II

RÉSULTATS DÉTAILLÉS DE SIMULATIONS

1 Résultats par séquences

Pour complémenter les résultats présentés dans ce mémoire, cette annexe présente les résultats par séquence. Pour se faire, quatre séquences ont été choisies pour leurs caractéristiques distinctes. La séquence *Akiyo* présente une actrice qui bouge peu, la caméra est fixe et l'arrière-plan est constant. Il s'agit ici de conditions similaires à la vidéophonie. La séquence *Carphone* est filmée à l'intérieur d'une voiture, ce qui fait en sorte que l'arrière-plan varie continuellement. De plus, on note que l'acteur bouge considérablement et est plus expressif qu'*Akiyo*. La séquence *Coastguard* se distingue des autres par le fait qu'il n'y a pas d'acteur humain et que le plan est en mouvement. Finalement, la séquence *Foreman* présente un acteur humain expressif avec un changement de plan.

De par ces résultats, nous démontrons que notre solution offre des gains pour chacune des séquences. Les QP et les taux d'erreurs sont les mêmes que ceux décris dans le mémoire. Pour chaque séquence, un tableau est présenté avec un résumé du PSNR moyen obtenu selon l'approche de dissimulation utilisée. Par la suite, des diagrammes permettent de visualiser le PSNR des blocs et des trames choisis par les algorithmes sélectifs. Finalement, un histogramme présente la distribution des PSNR des trames résultant des deux approches de dissimulation sélectives présentées dans ce mémoire, du calque de la trame et de la trame erronée.

1.1 Séquence *Akiyo*

Tableau II.1 Résumé des résultats obtenus pour la séquence *Akiyo*.

Approche	**PSNR Moyen dispersé (dB)**	**PSNR Moyen entrelacé (dB)**
Encodée (sans erreur)	42.90	42.87
Sélective par bloc (avec référence)	42.12	41.63
Sélective par tranche (avec référence)	41.39	41.18
Sélective par bloc	**41.72**	**41.01**
Sélective par tranche	**41.26**	**40.99**
Dissimulation tranche calquée	40.77	40.20
Trame corrompue	38.92	39.61

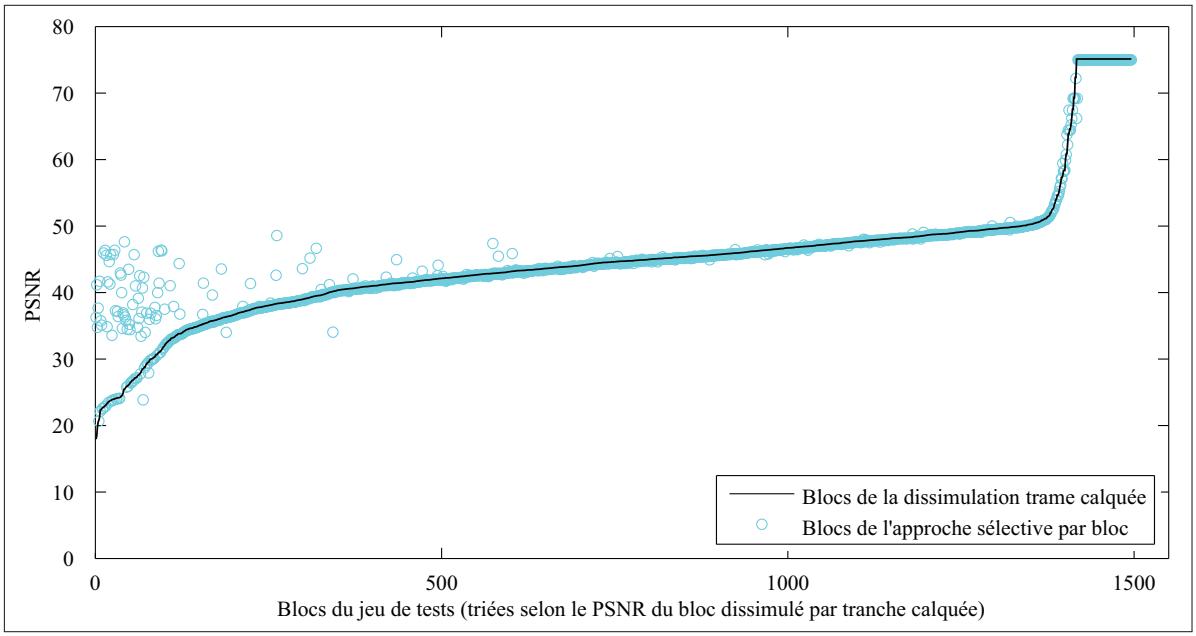


Figure II.1 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Akiyo*, FMO = Dispersé)

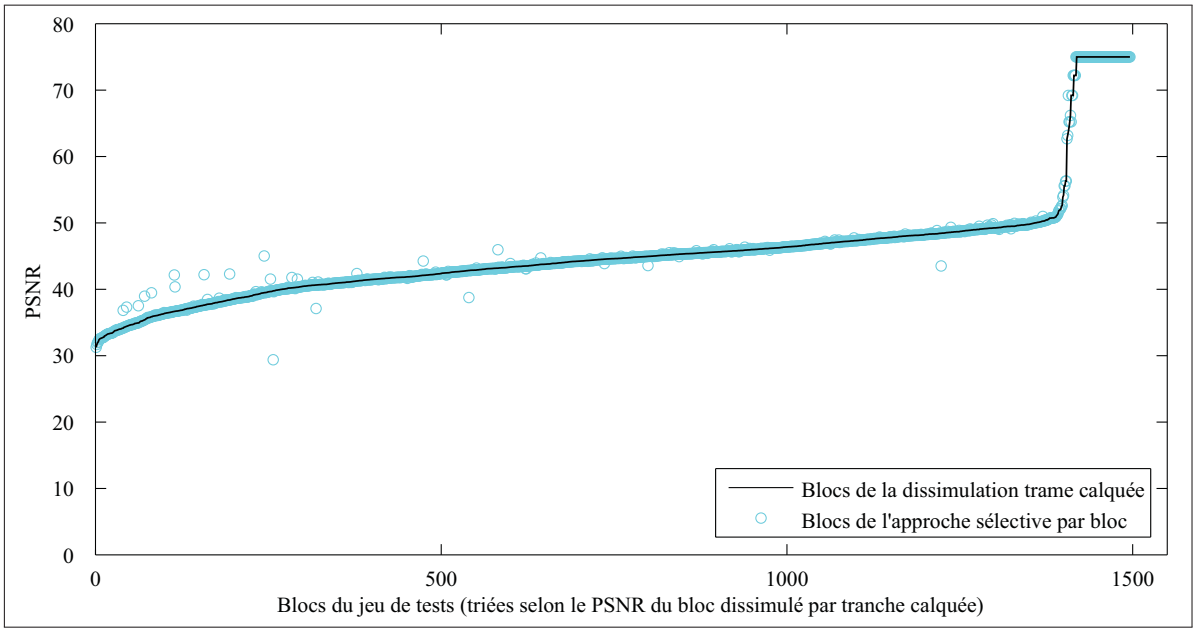


Figure II.2 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Akiyo*, FMO = Entrelacé)

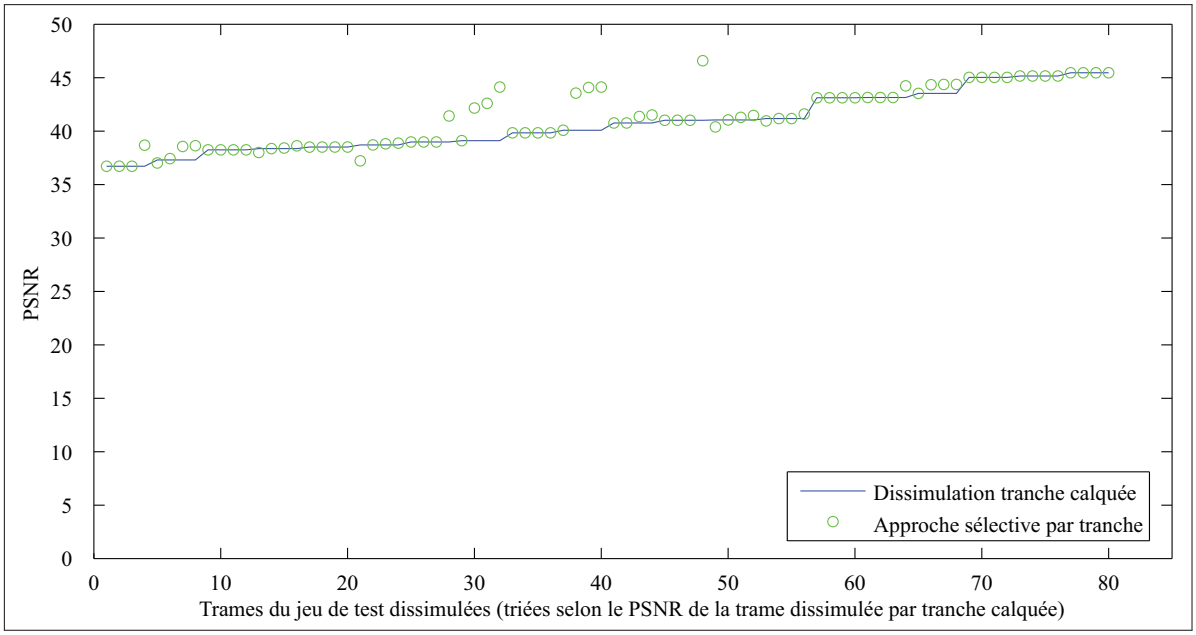


Figure II.3 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Akiyo*, FMO = Dispersé)

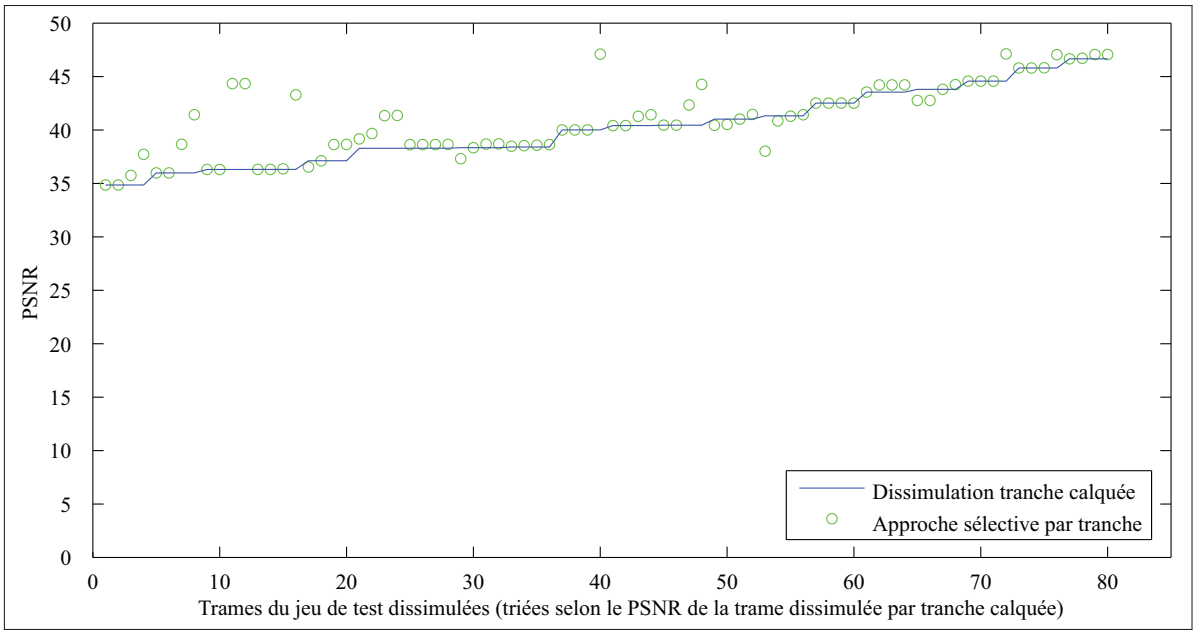


Figure II.4 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Akiyo*, FMO = Entrelacé)

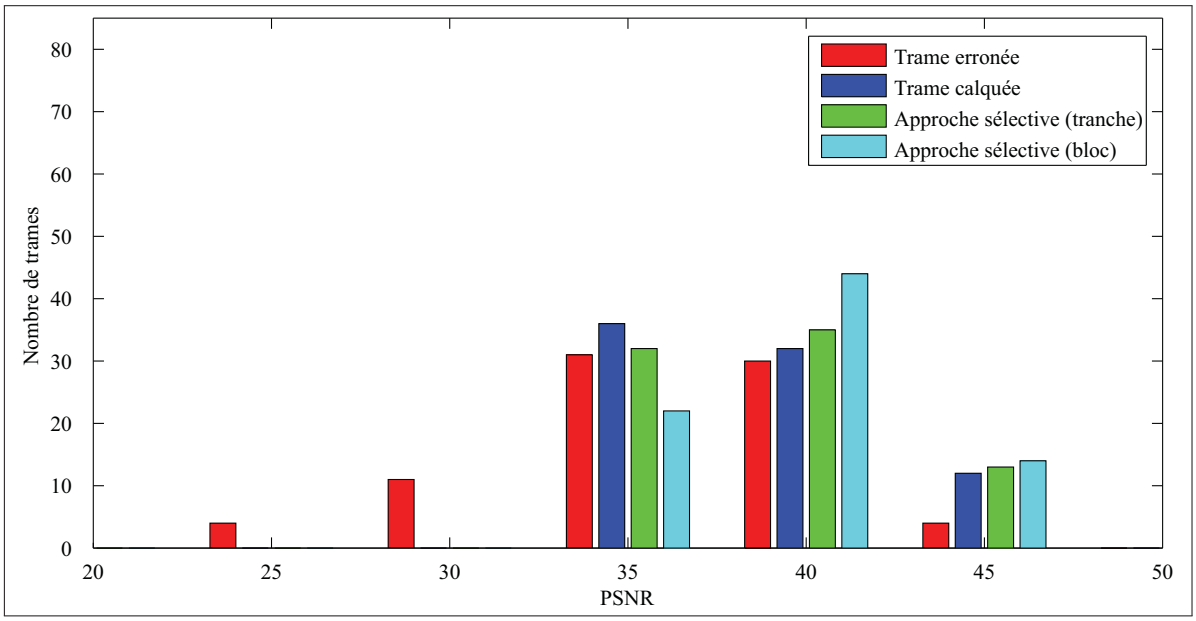


Figure II.5 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Akiyo*, FMO = Dispersé)

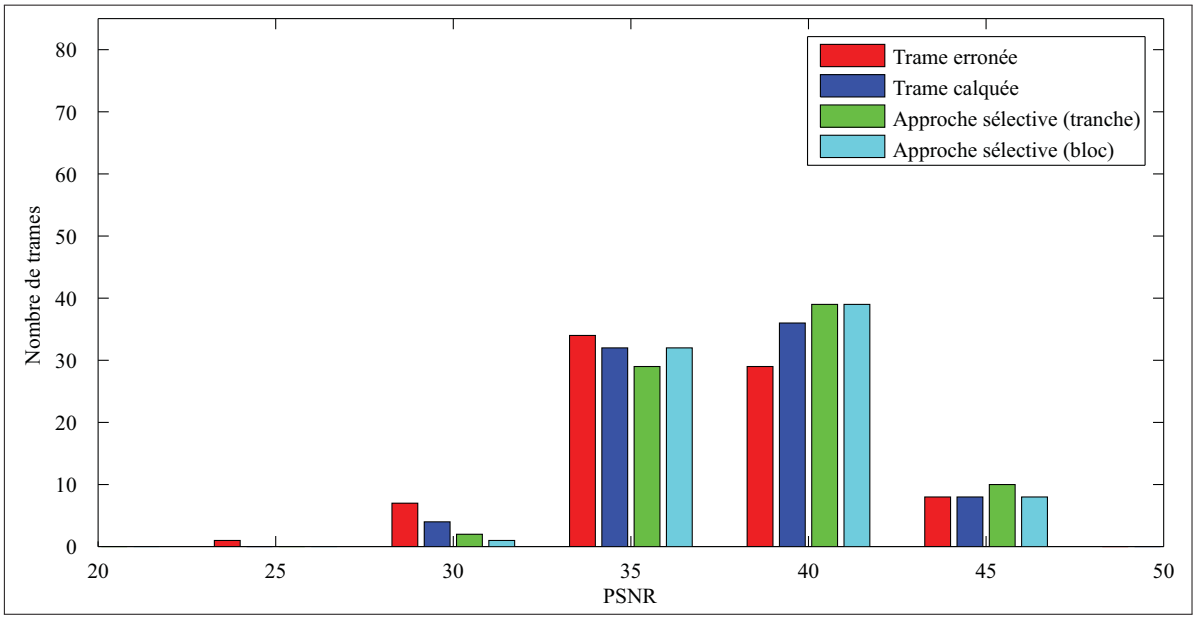


Figure II.6 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Akiyo*, FMO = Entrelacé)

1.2 Séquence *Carphone*

Tableau II.2 Résumé des résultats obtenus pour la séquence *Carphone*.

Approche	**PSNR Moyen dispersé (dB)**	**PSNR Moyen entrelacé (dB)**
Encodée (sans erreur)	41.40	41.20
Sélective par bloc (avec référence)	36.57	34.81
Sélective par tranche (avec référence)	35.69	34.13
Sélective par bloc	**35.84**	**33.80**
Sélective par tranche	**35.41**	**33.99**
Dissimulation tranche calquée	34.63	33.10
Trame corrompue	33.71	32.41

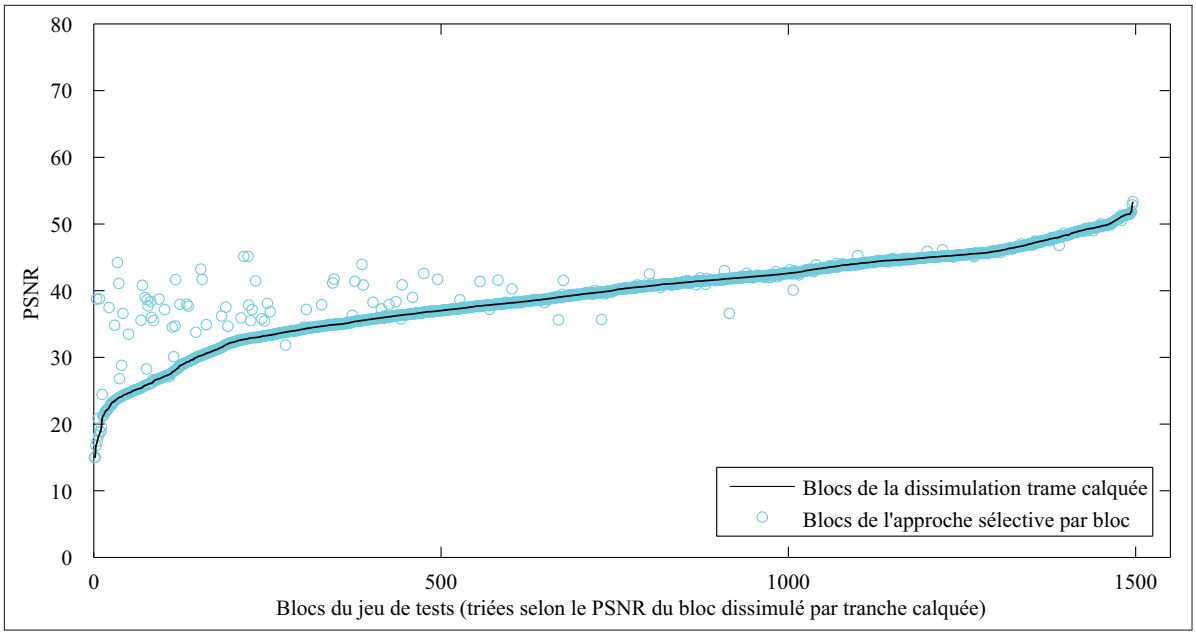


Figure II.7 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Carphone*, FMO = Dispersé)

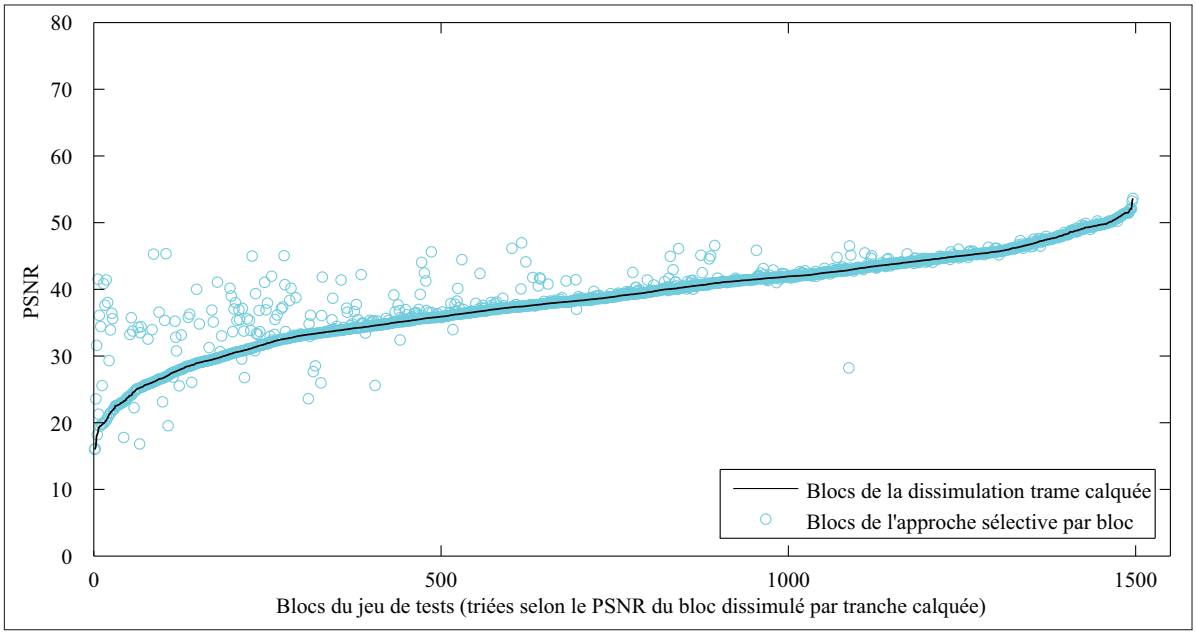


Figure II.8 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Carphone*, FMO = Entrelacé)

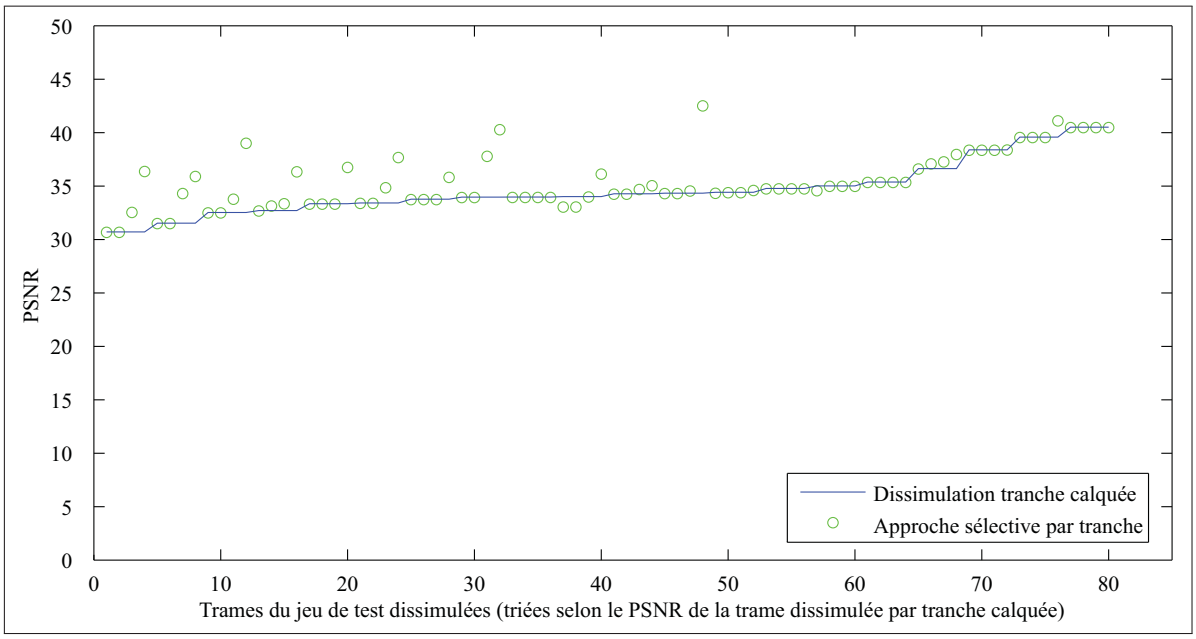


Figure II.9 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Carphone*, FMO = Dispersé)

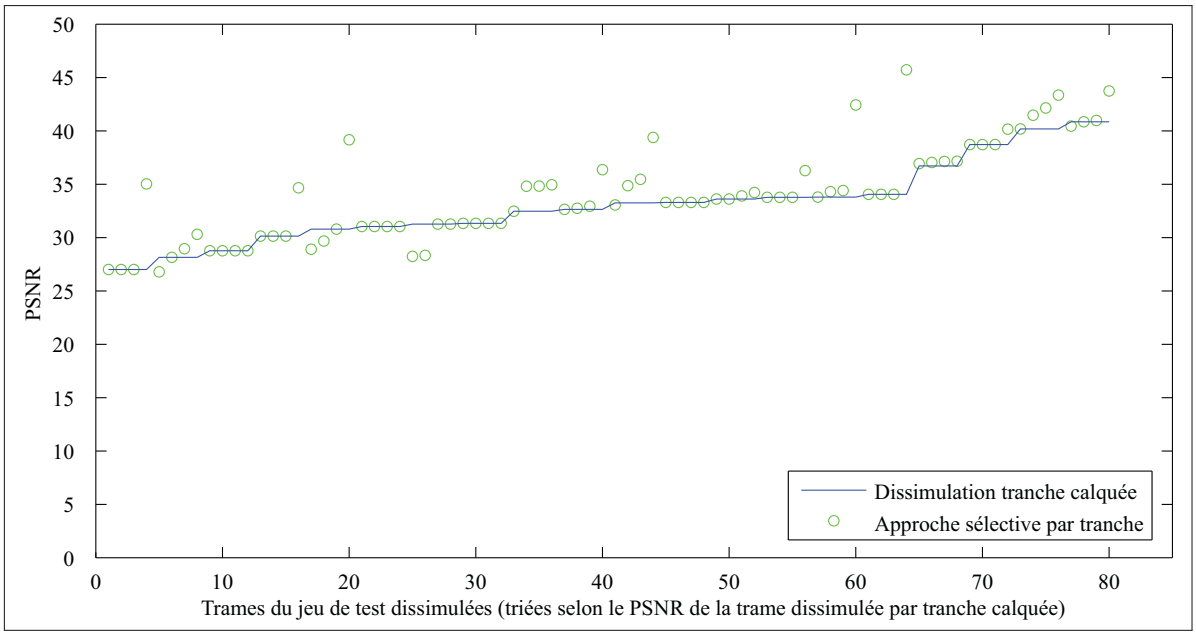


Figure II.10 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Carphone*, FMO = Entrelacé)

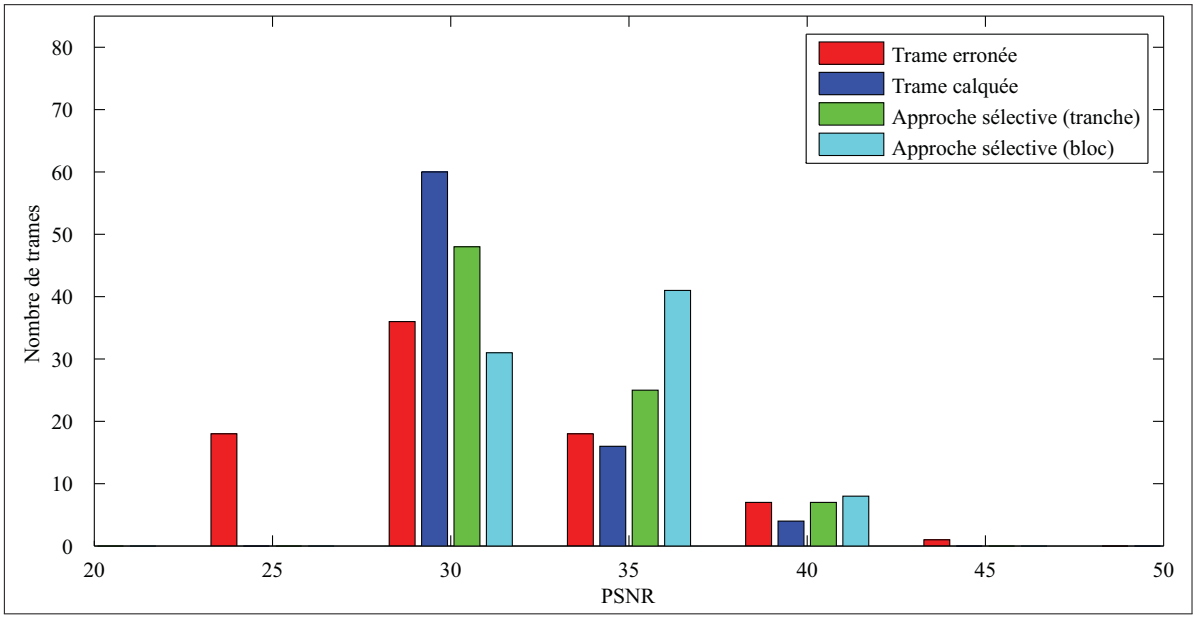


Figure II.11 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Carphone*, FMO = Dispersé)

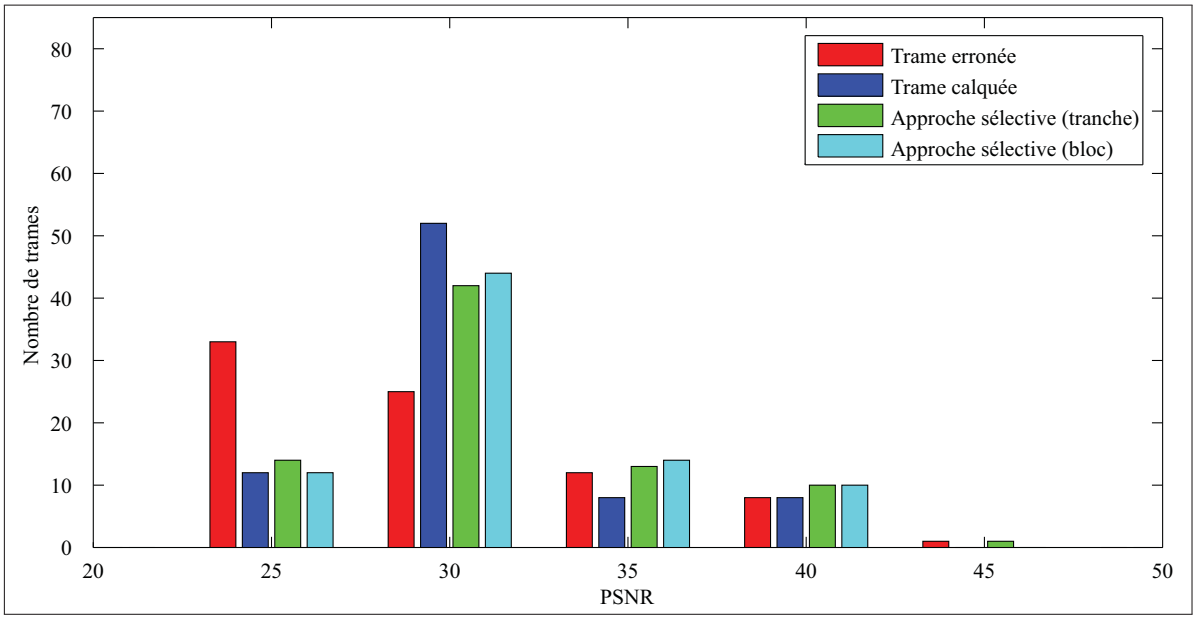


Figure II.12 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Carphone*, FMO = Entrelacé)

1.3 Séquence *Coastguard*

Tableau II.3 Résumé des résultats obtenus pour la séquence *Coastguard*.

Approche	**PSNR Moyen dispersé (dB)**	**PSNR Moyen entrelacé (dB)**
Encodée (sans erreur)	39.78	39.90
Sélective par bloc (avec référence)	34.00	35.01
Sélective par tranche (avec référence)	32.94	34.10
Sélective par bloc	**33.37**	**34.04**
Sélective par tranche	**32.79**	**33.98**
Dissimulation tranche calquée	31.62	33.01
Trame corrompue	30.51	31.21

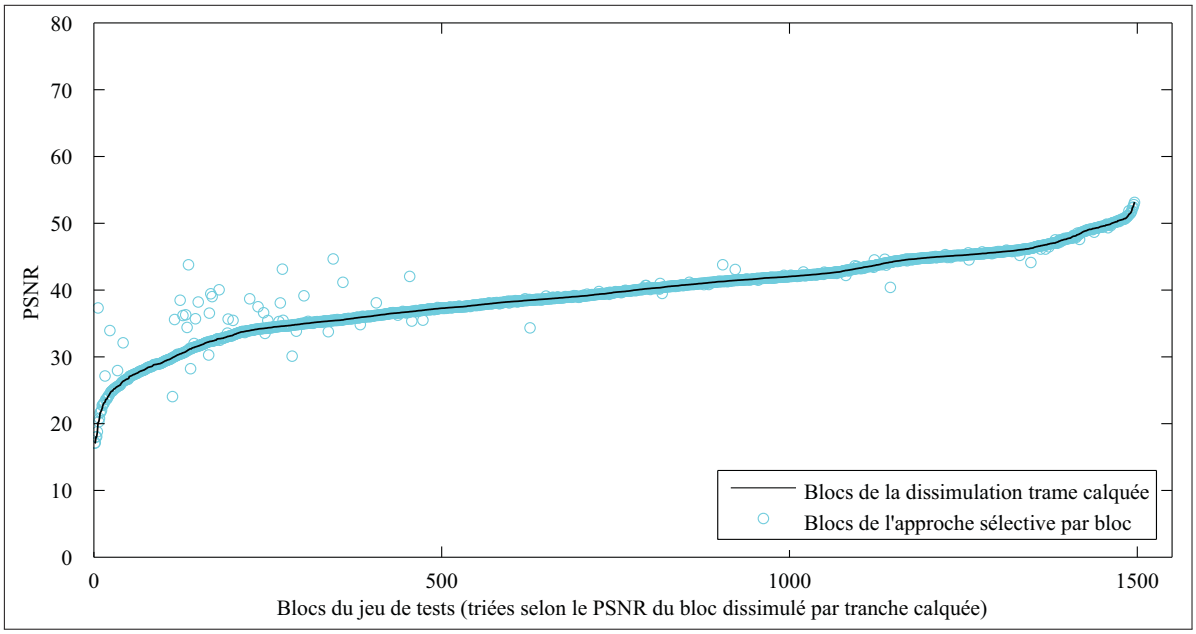


Figure II.13 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Coastguard*, FMO = Dispersé)

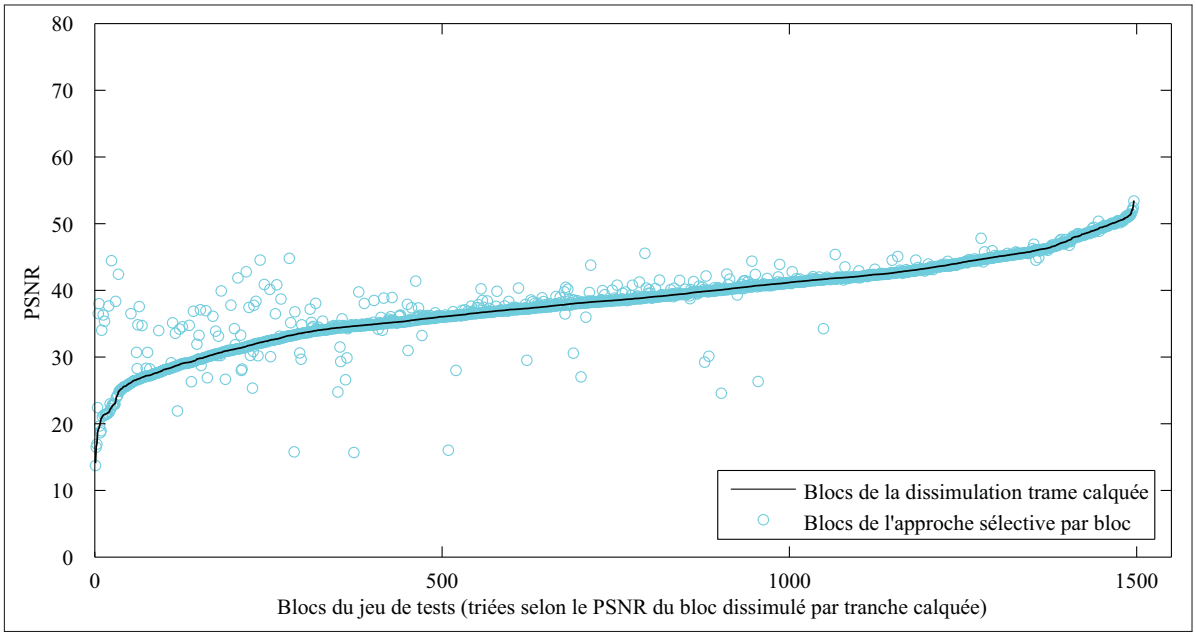


Figure II.14 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Coastguard*, FMO = Entrelacé)

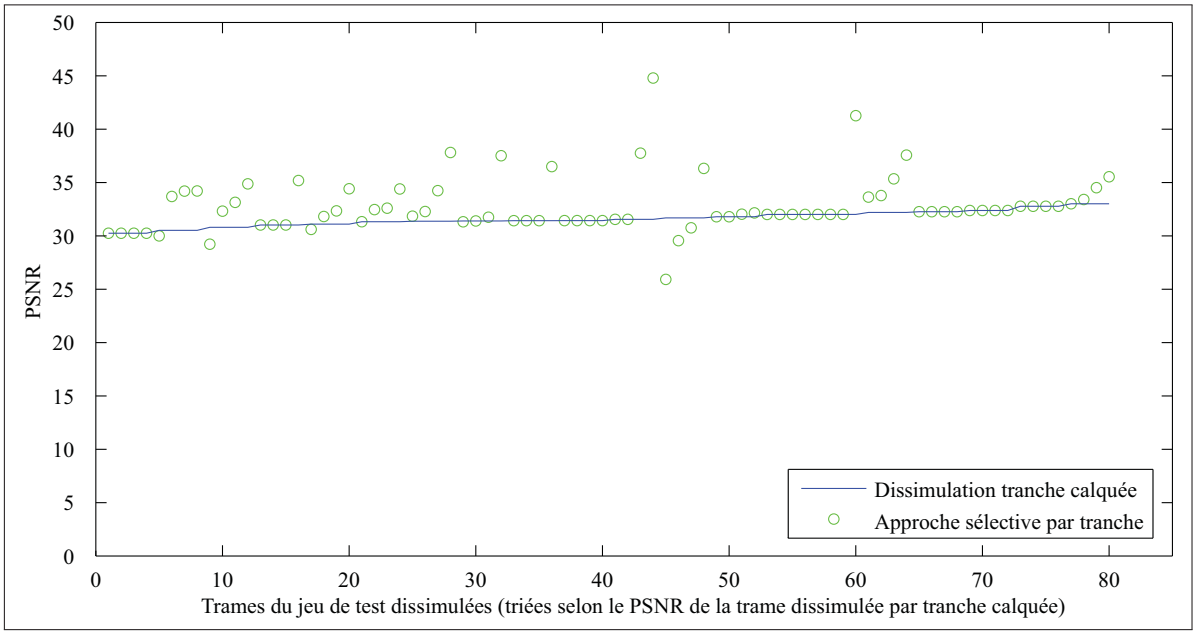


Figure II.15 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Coastguard*, FMO = Dispersé)

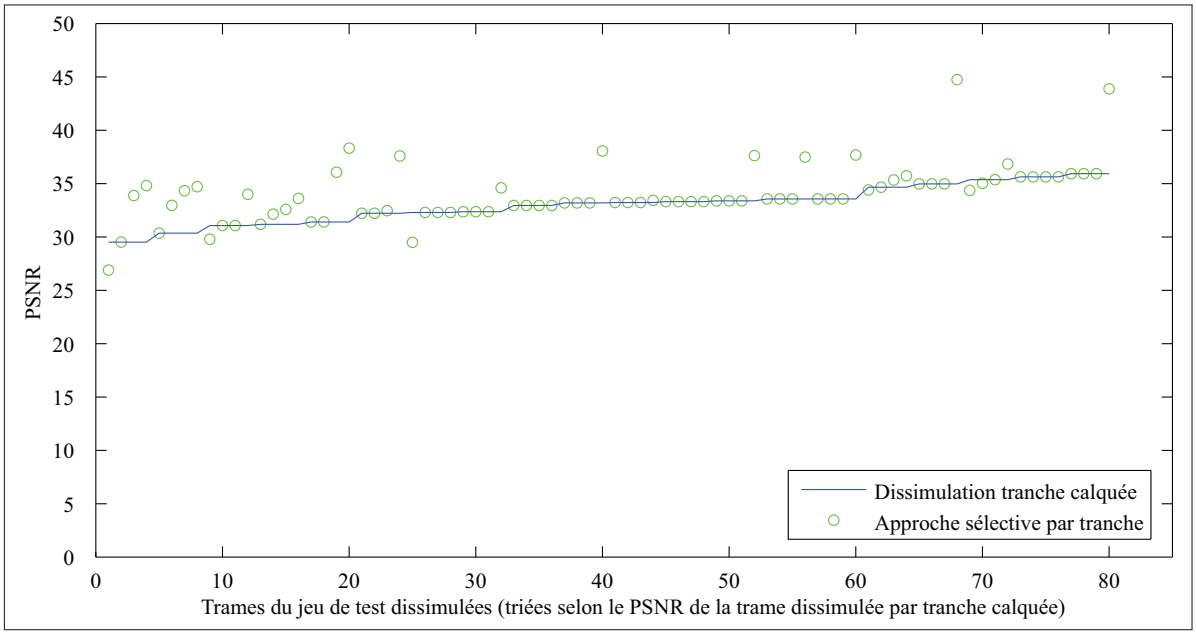


Figure II.16 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Coastguard*, FMO = Entrelacé)

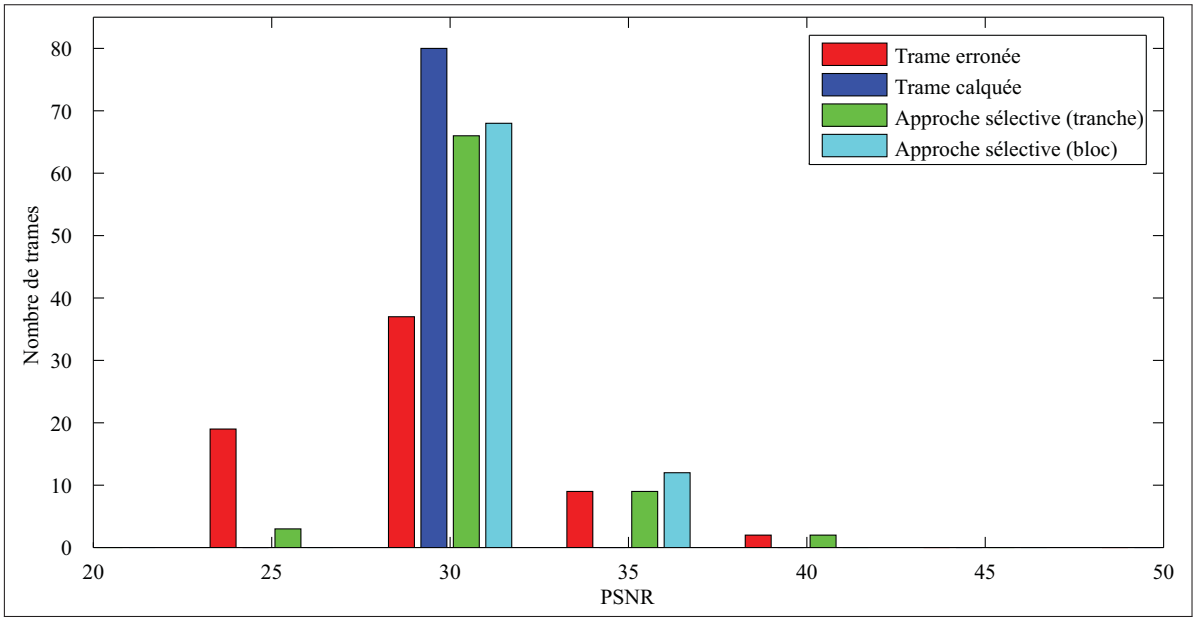


Figure II.17 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Coastguard*, FMO = Dispersé)

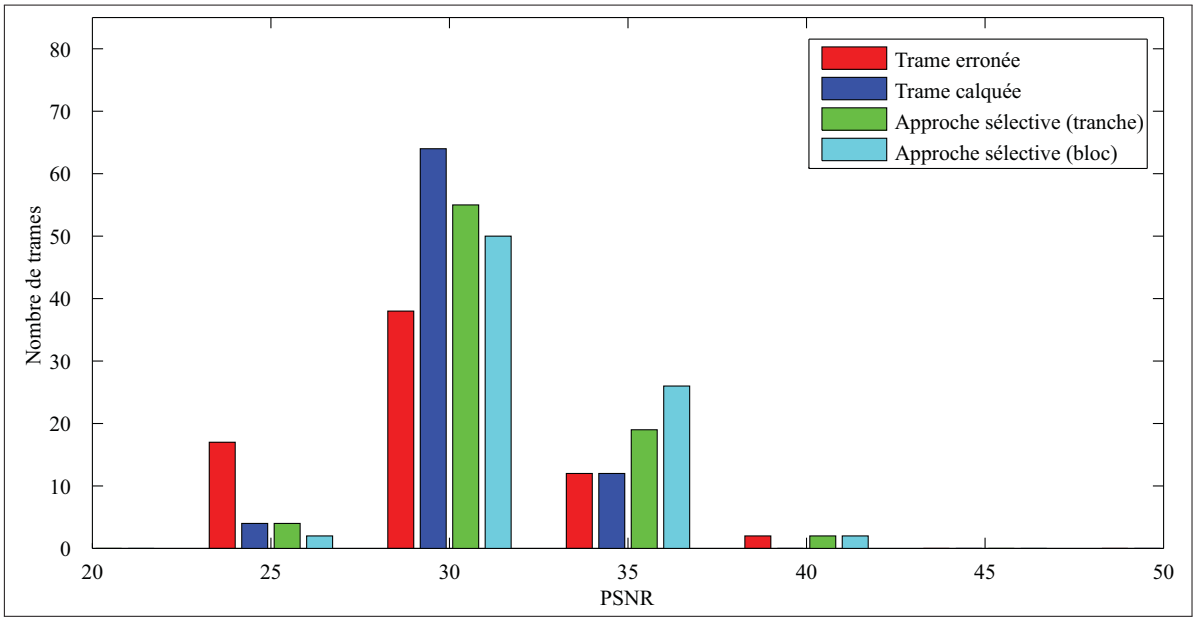


Figure II.18 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Coastguard*, FMO = Entrelacé)

1.4 Séquence *Foreman*

Tableau II.4 Résumé des résultats obtenus pour la séquence *Foreman*.

Approche	**PSNR Moyen dispersé (dB)**	**PSNR Moyen entrelacé (dB)**
Encodée (sans erreur)	40.51	40.14
Sélective par bloc (avec référence)	34.79	35.05
Sélective par tranche (avec référence)	33.55	33.99
Sélective par bloc	**33.89**	**33.02**
Sélective par tranche	**33.45**	**32.70**
Dissimulation tranche calquée	32.15	32.69
Trame corrompue	31.87	31.72

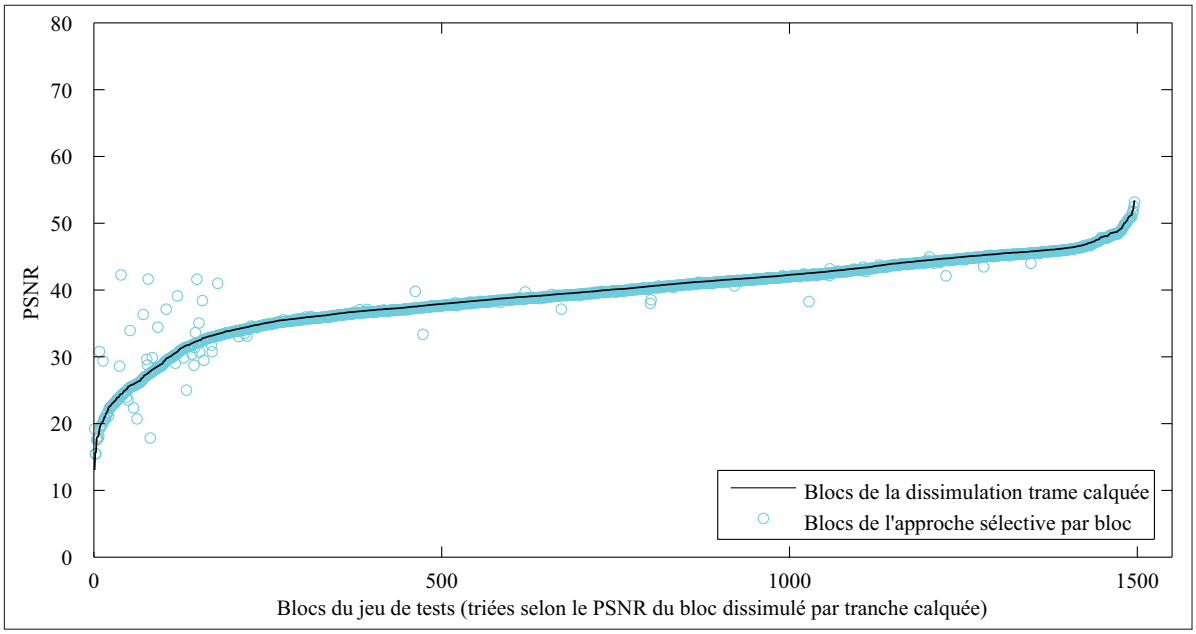


Figure II.19 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Foreman*, FMO = Dispersé)

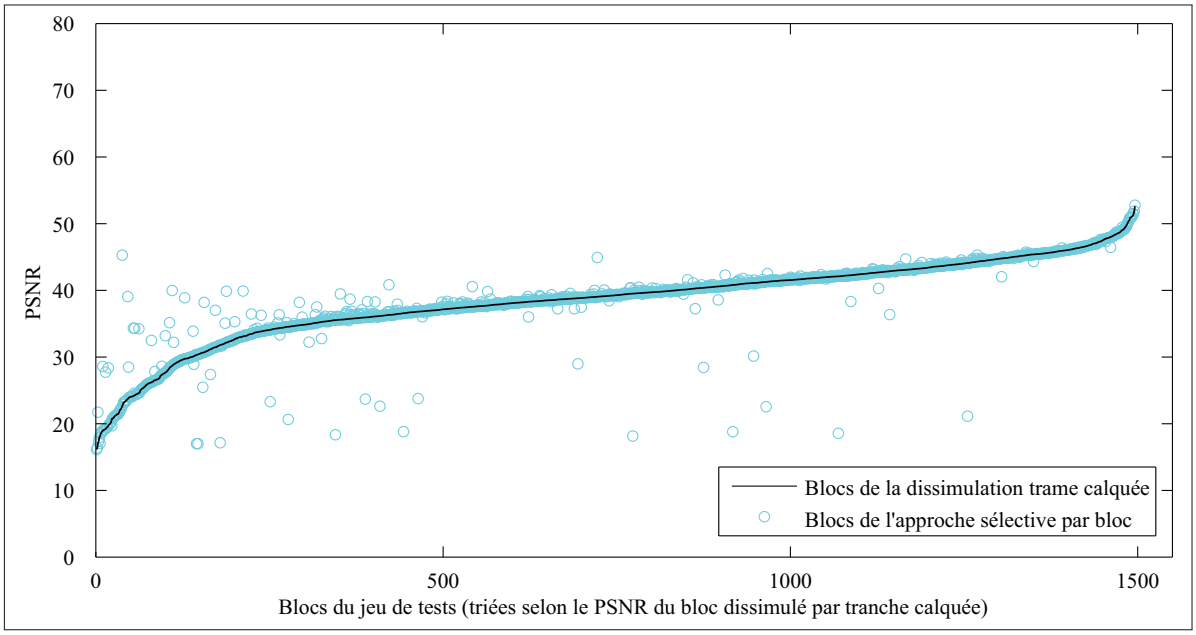


Figure II.20 PSNR des blocs de l'approche sélective par bloc par rapport au calquage de trame. (Séquence=*Foreman*, FMO = Entrelacé)

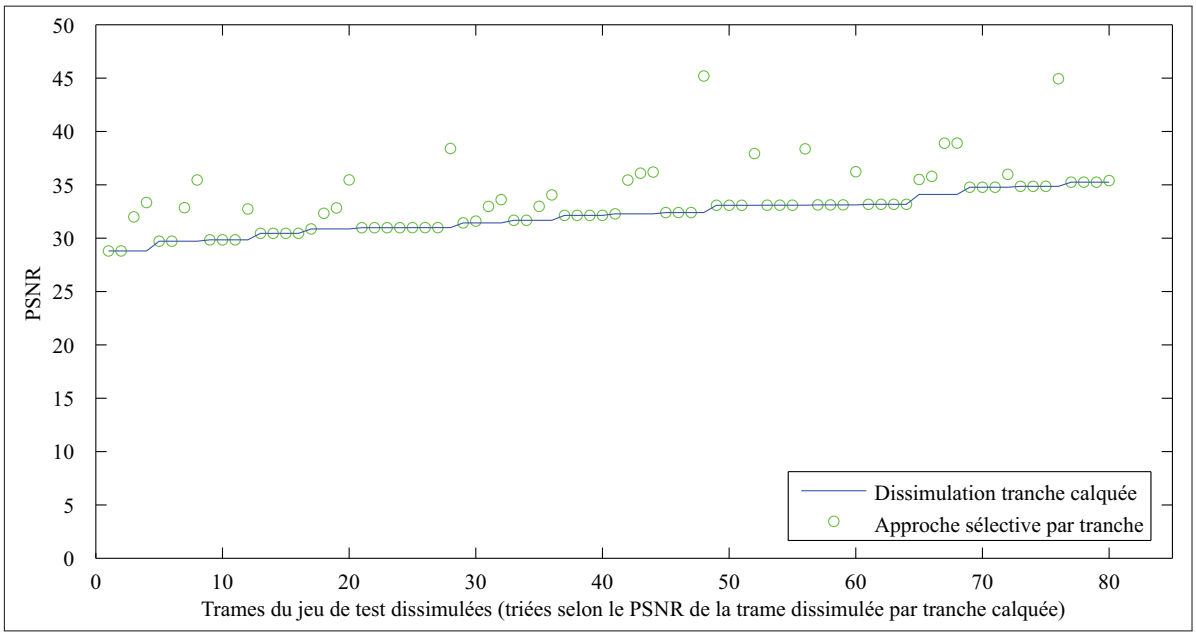


Figure II.21 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Foreman*, FMO = Dispersé)

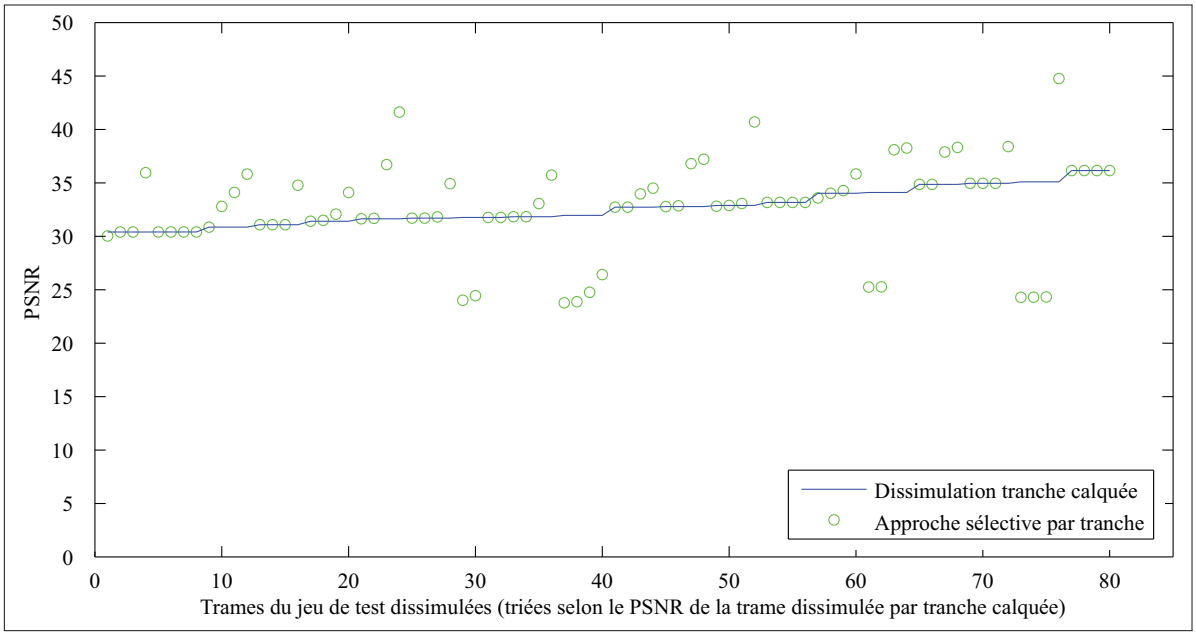


Figure II.22 PSNR des trames de l'approche sélective par trame par rapport au calquage de trame. (Séquence=*Foreman*, FMO = Entrelacé)

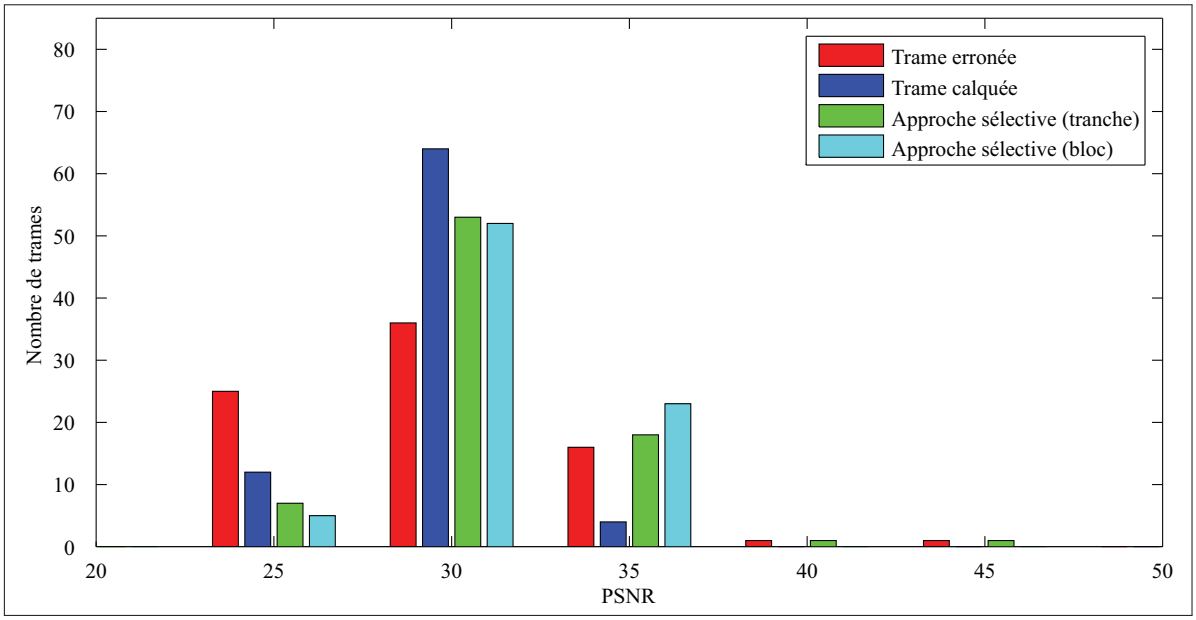


Figure II.23 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Foreman*, FMO = Dispersé)

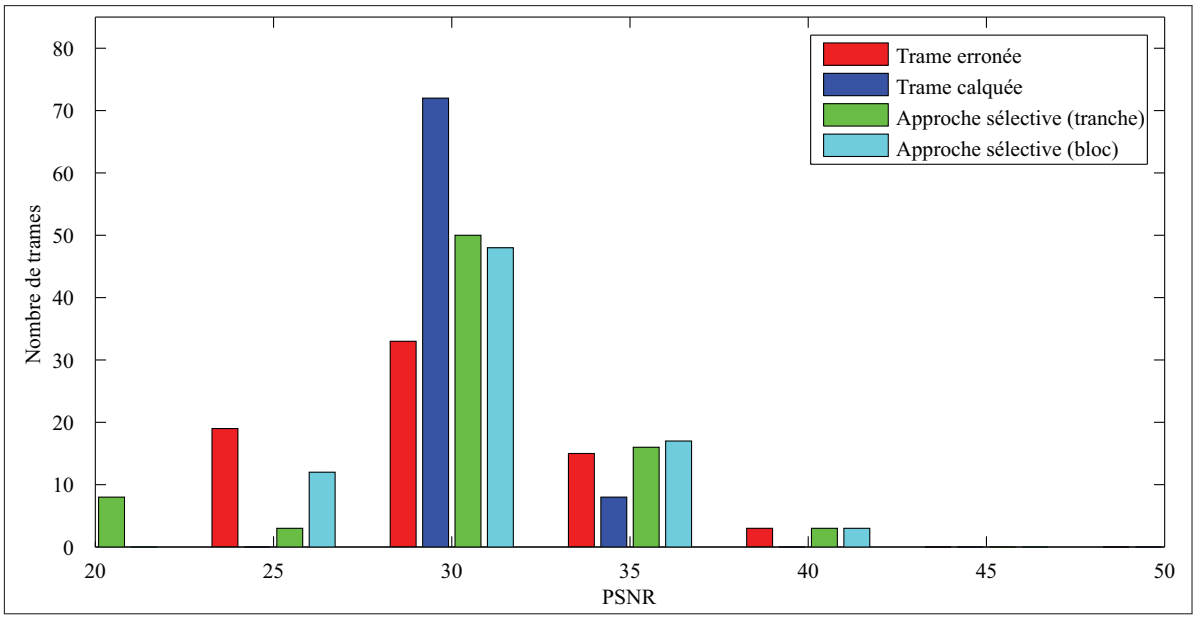


Figure II.24 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (Séquence=*Foreman*, FMO = Entrelacé)

2 Résultats pour l'ensemble des séquences

Dans cette section, nous présentons les résultats de la qualité visuelle, mesurée avec le PSNR, obtenus pour chacune des différentes approches de dissimulation. Ceux-ci sont mesurés en fonction des taux d'erreurs binaires (BER) 0.0004, 0.0008, 0.0016 et 0.0032 représentant différentes conditions de diffusion sur un réseau peu fiable. Tout d'abord, pour chaque taux d'erreurs, nous présentons un tableau qui résume les valeurs moyennes du PSNR de chaque approche. Par la suite, nous présentons les histogrammes de la distribution du PSNR pour chaque QP (16, 20, 24, 28). Ces taux et ces QP sont les mêmes que ceux utilisés au chapitre 6 « Résultats de simulations et analyse ».

2.1 Taux d'erreurs binaires (BER) 0.0004

Tableau II.5 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0004 (dispersé).

Approche	**PSNR Moyen dispersé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.57	39.59	36.88
Sélective par bloc (avec référence)	40.53	39.20	38.15	36.15
Sélective par tranche (avec référence)	39.64	38.39	37.59	35.83
Sélective par bloc	**39.31**	**38.06**	**37.22**	**35.31**
Sélective par tranche	**38.94**	**37.81**	**37.37**	**35.68**
Dissimulation tranche calquée	37.80	36.87	35.78	34.27
Trame corrompue	37.80	36.87	35.78	34.27

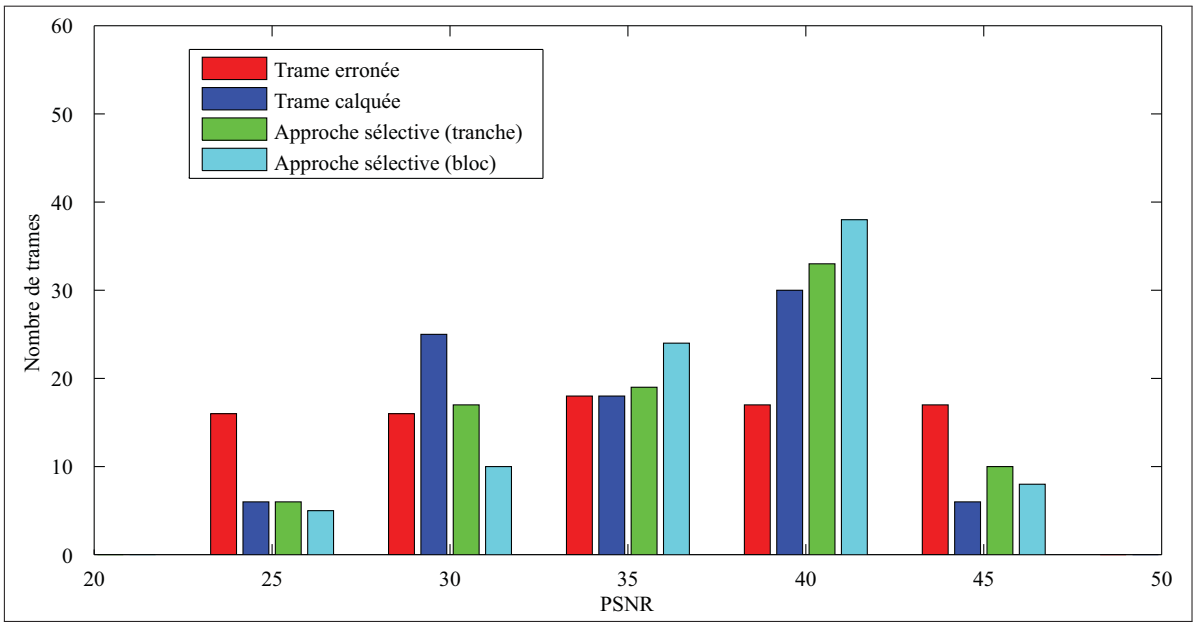


Figure II.25 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0004, FMO = Dispersé)

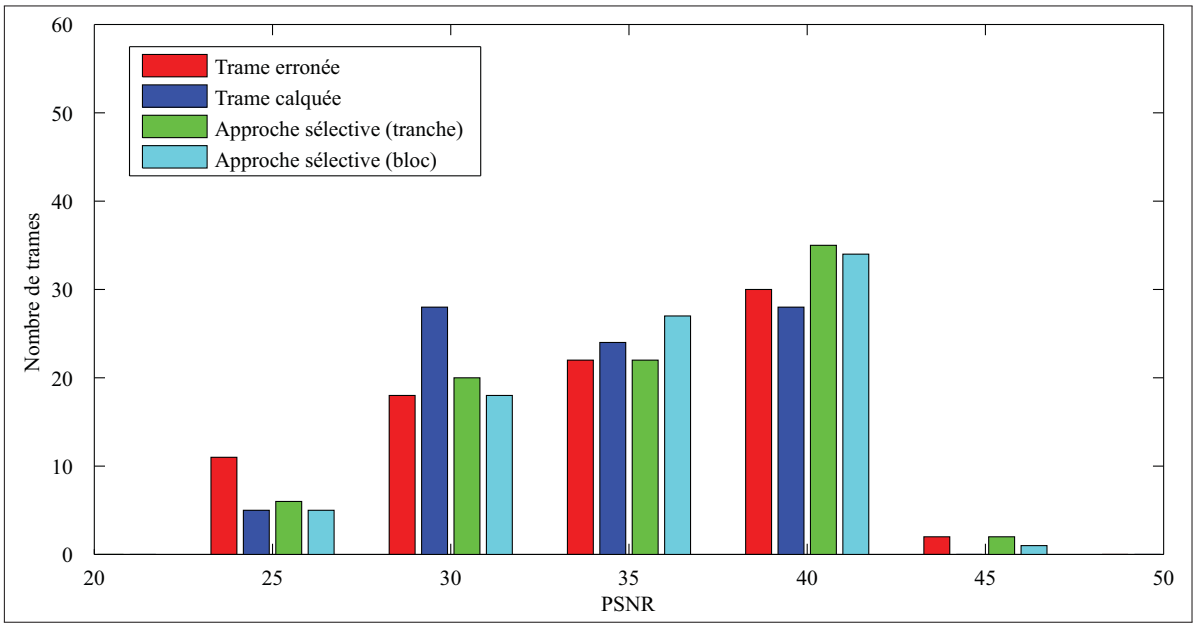


Figure II.26 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0004, FMO = Dispersé)

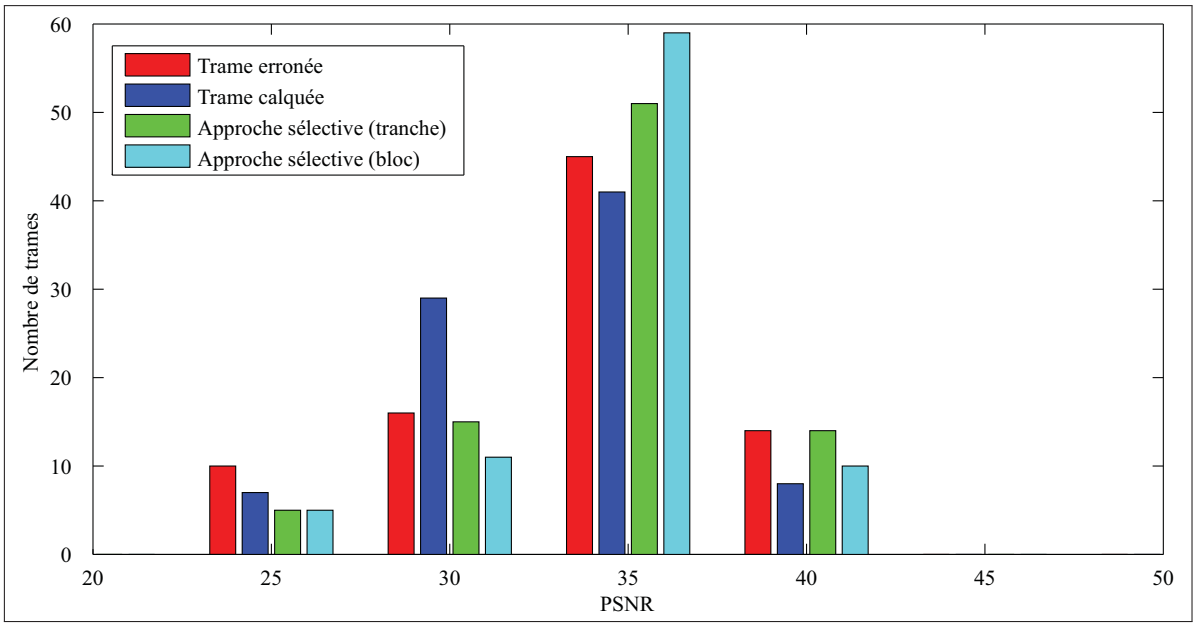


Figure II.27 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0004, FMO = Dispersé)

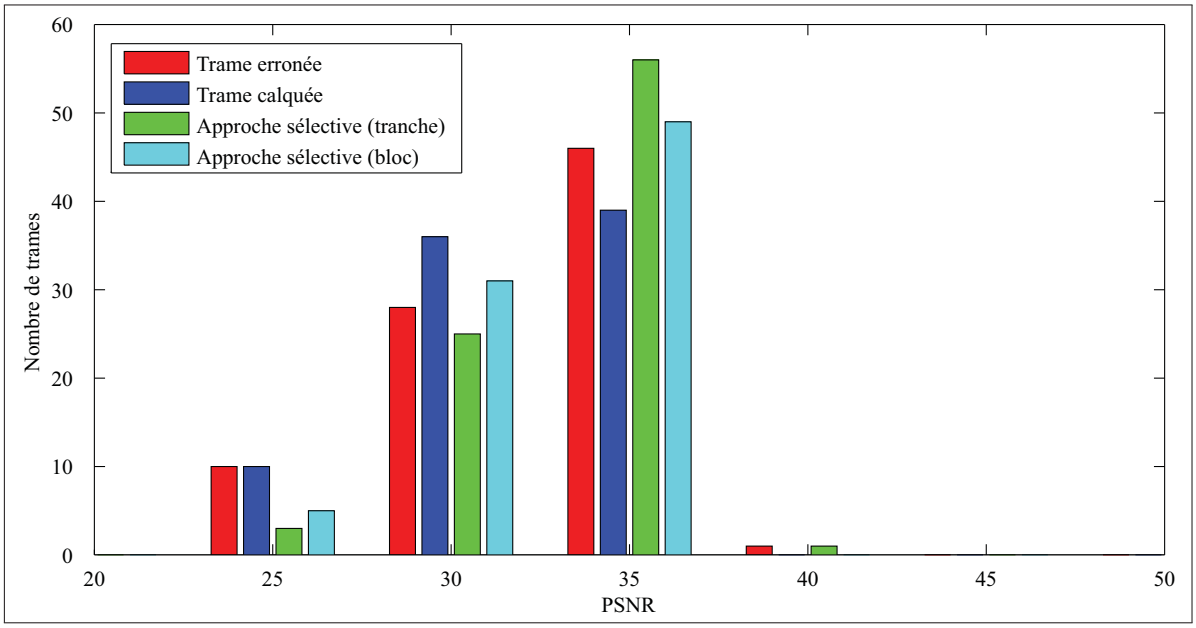


Figure II.28 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0004, FMO = Dispersé)

Tableau II.6 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0004 (entrelacé).

Approche	**PSNR Moyen entrelacé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.56	39.56	36.85
Sélective par bloc (avec référence)	42.19	40.55	38.59	36.22
Sélective par tranche (avec référence)	41.63	39.91	38.20	35.91
Sélective par bloc	**40.86**	**39.51**	**37.98**	**35.79**
Sélective par tranche	**41.46**	**39.75**	**38.15**	**35.76**
Dissimulation tranche calquée	39.26	38.14	36.72	34.94
Trame corrompue	39.26	38.14	36.72	34.94

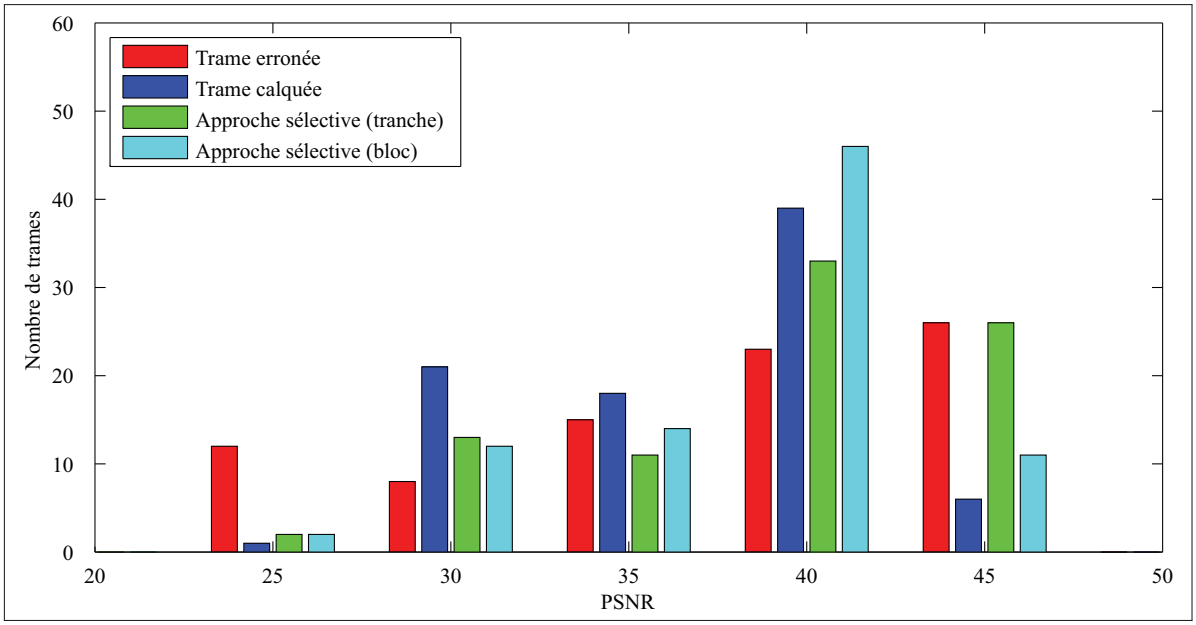


Figure II.29 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0004, FMO = Entrelacé)

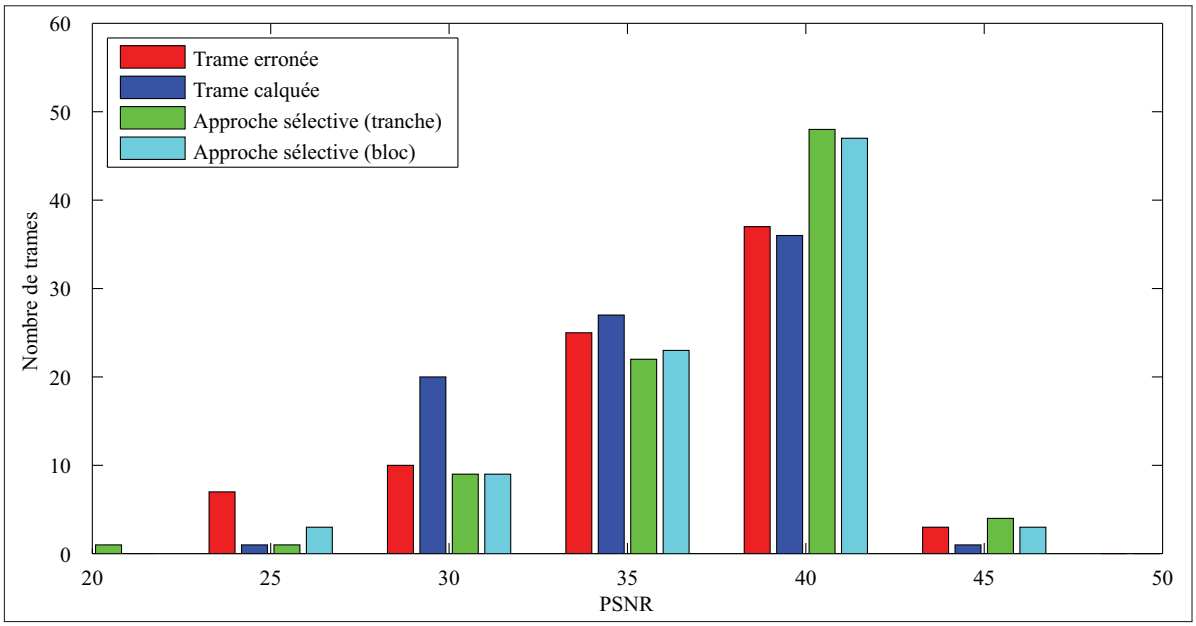


Figure II.30 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0004, FMO = Entrelacé)

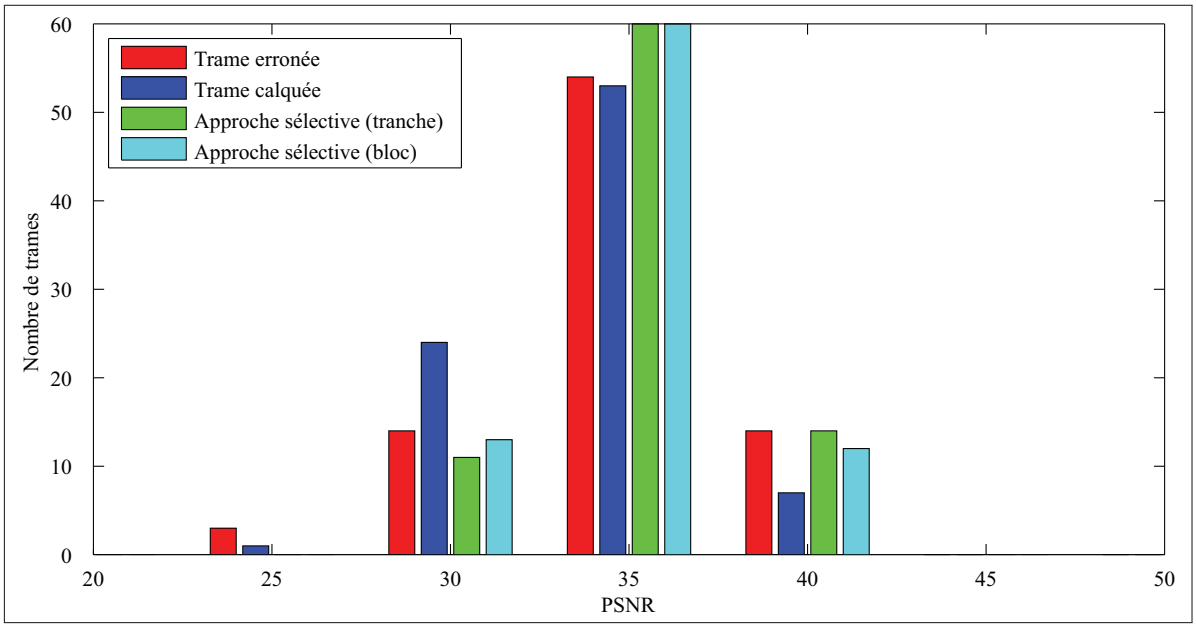


Figure II.31 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0004, FMO = Entrelacé)

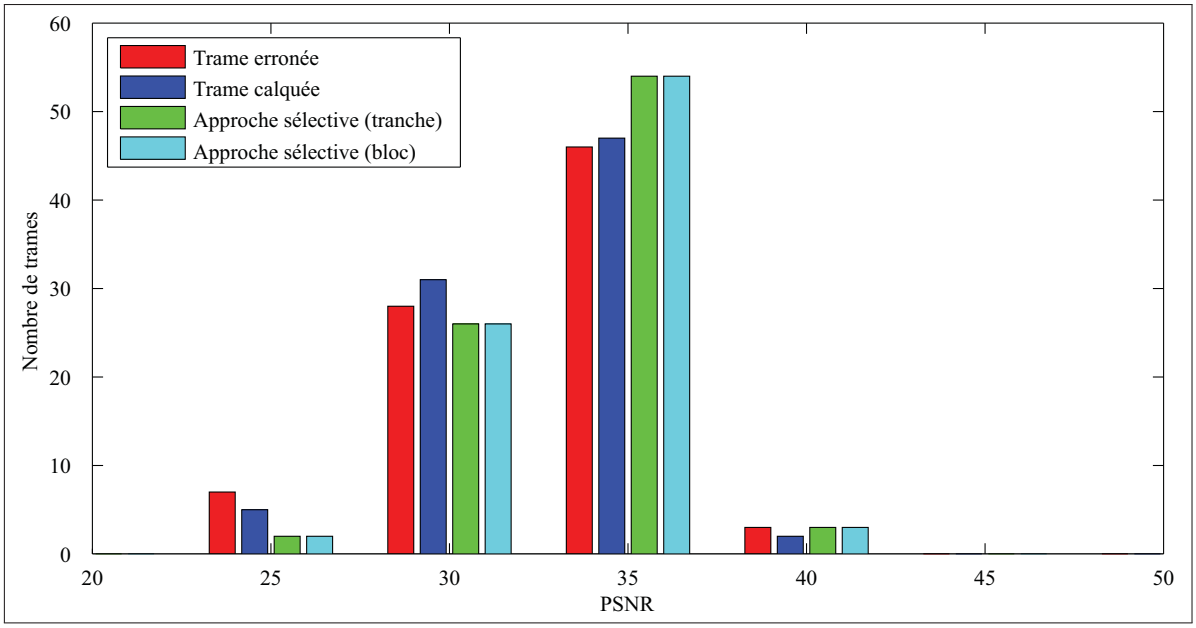


Figure II.32 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0004, FMO = Entrelacé)

2.2 Taux d'erreurs binaires (BER) 0.0008

Tableau II.7 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0008 (dispersé).

Approche	**PSNR Moyen dispersé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.57	39.59	36.88
Sélective par bloc (avec référence)	39.44	39.07	37.67	35.94
Sélective par tranche (avec référence)	38.75	38.27	36.99	35.42
Sélective par bloc	**38.75**	**38.19**	**36.83**	**35.20**
Sélective par tranche	**38.37**	**37.98**	**36.84**	**35.32**
Dissimulation tranche calquée	37.80	36.87	35.78	34.27
Trame corrompue	37.80	36.87	35.78	34.27

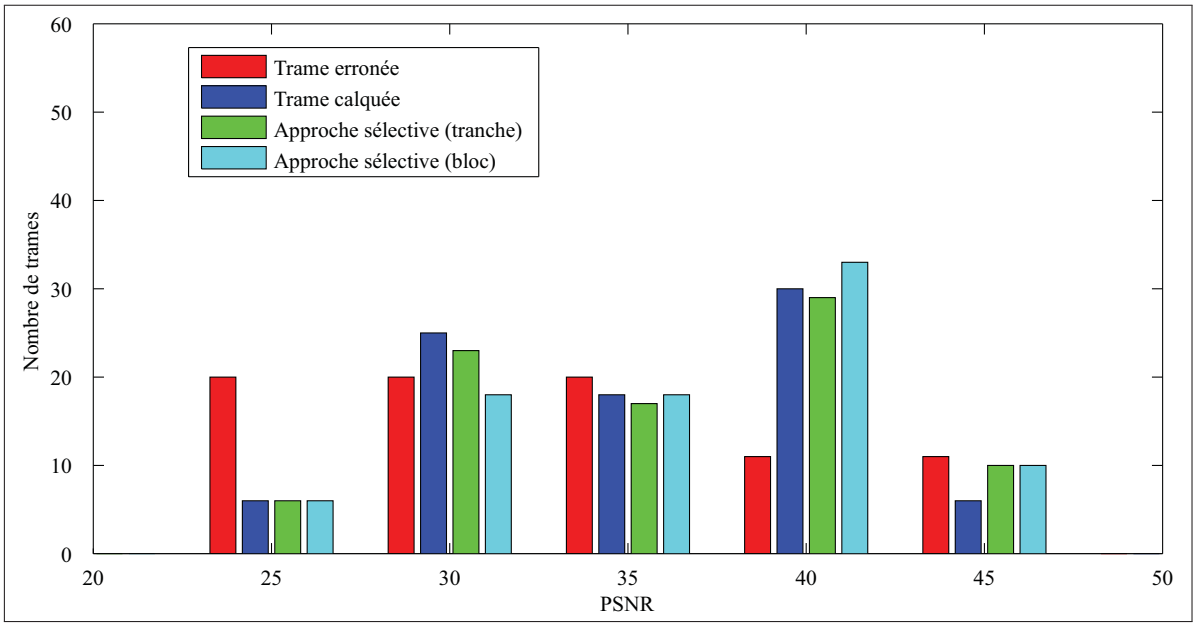


Figure II.33 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0008, FMO = Dispersé)

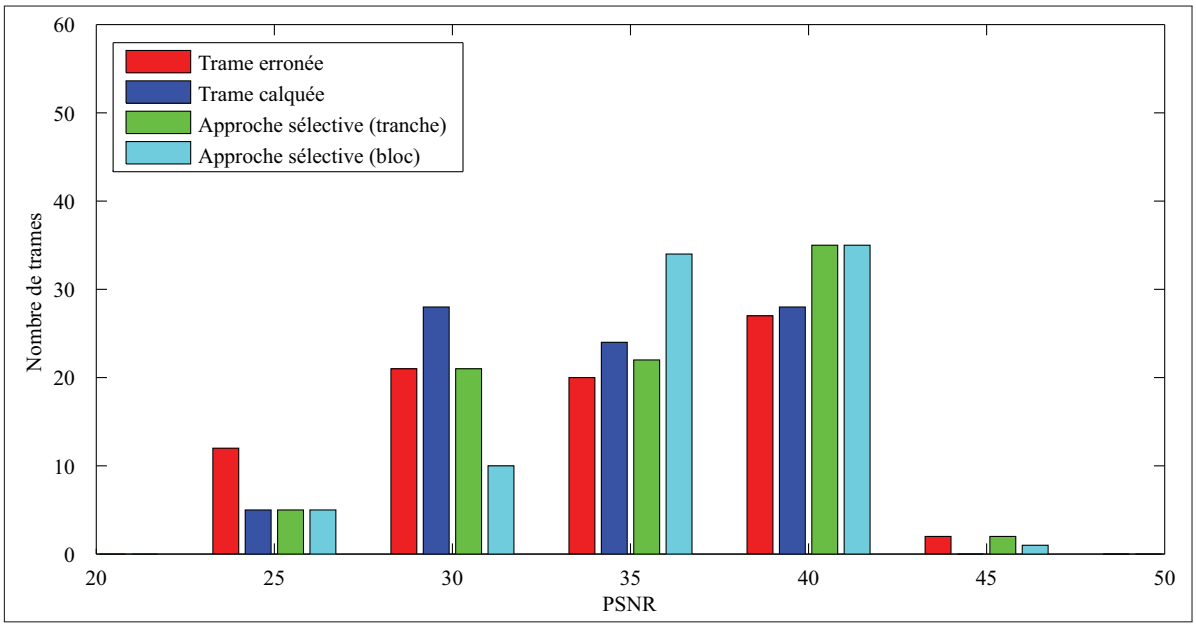


Figure II.34 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0008, FMO = Dispersé)

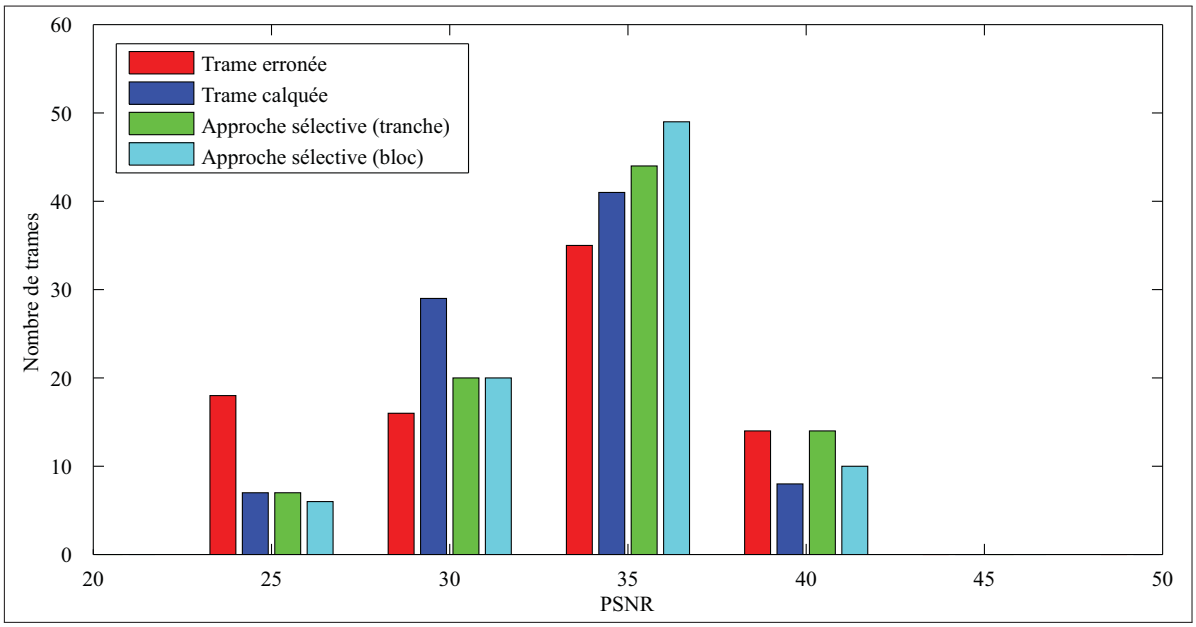


Figure II.35 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0008, FMO = Dispersé)

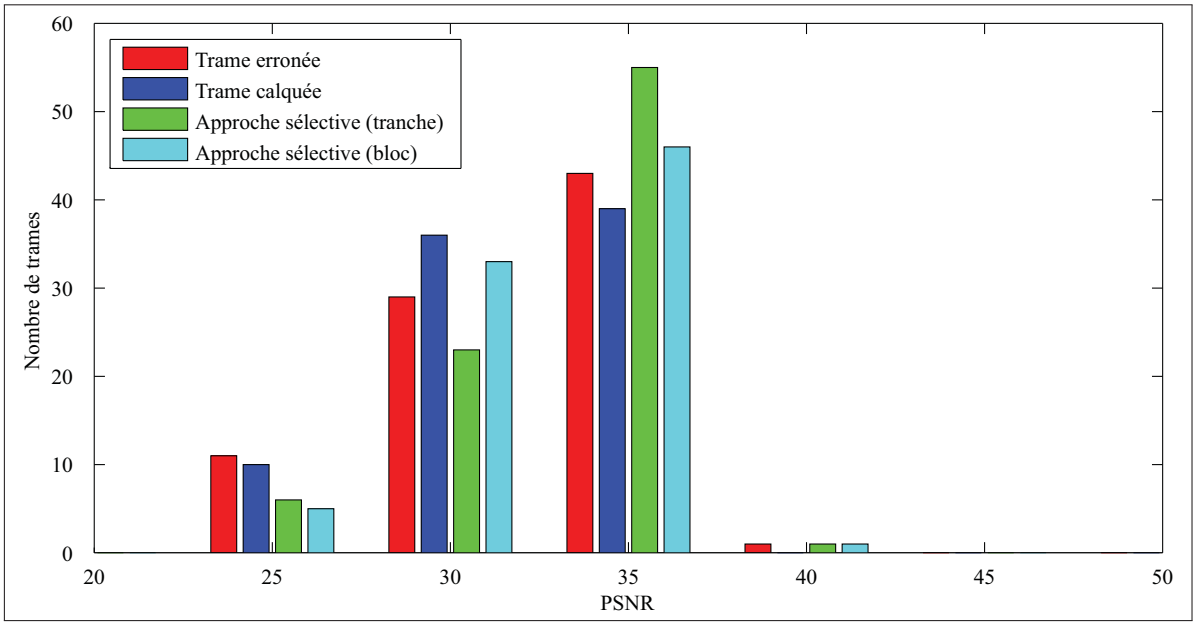


Figure II.36 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0008, FMO = Dispersé)

Tableau II.8 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0008 (entrelacé).

Approche	**PSNR Moyen entrelacé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.56	39.56	36.85
Sélective par bloc (avec référence)	40.78	39.60	38.10	36.13
Sélective par tranche (avec référence)	40.02	38.99	37.65	35.83
Sélective par bloc	**39.97**	**38.96**	**37.59**	**35.66**
Sélective par tranche	**39.78**	**38.83**	**37.47**	**35.69**
Dissimulation tranche calquée	39.26	38.14	36.72	34.94
Trame corrompue	39.26	38.14	36.72	34.94

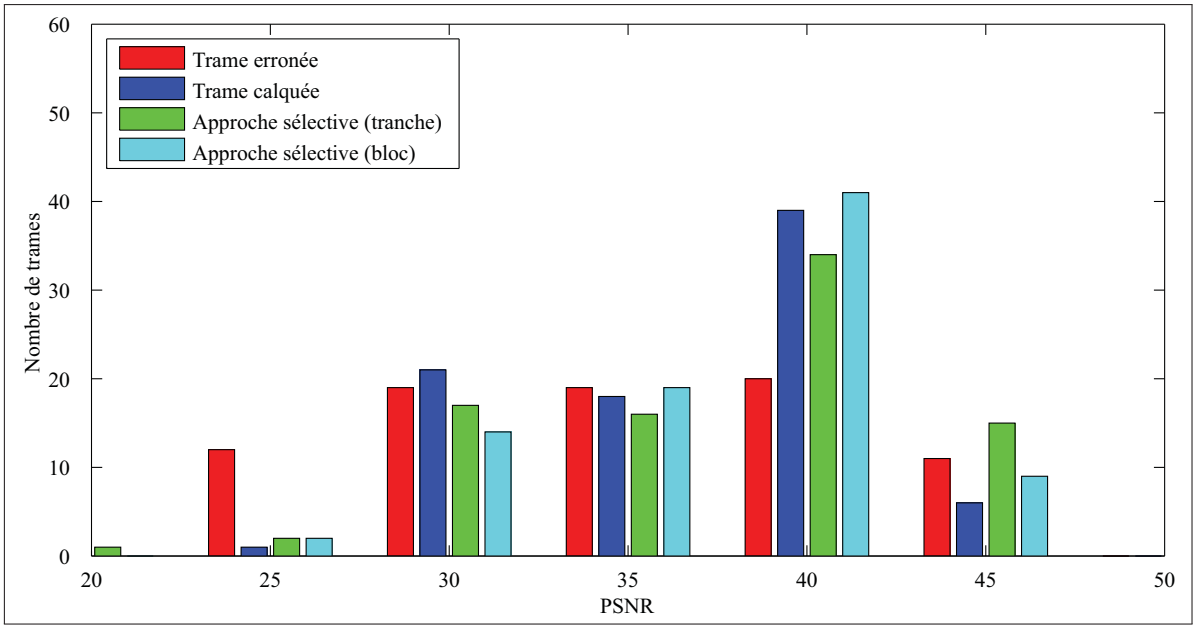


Figure II.37 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0008, FMO = Entrelacé)

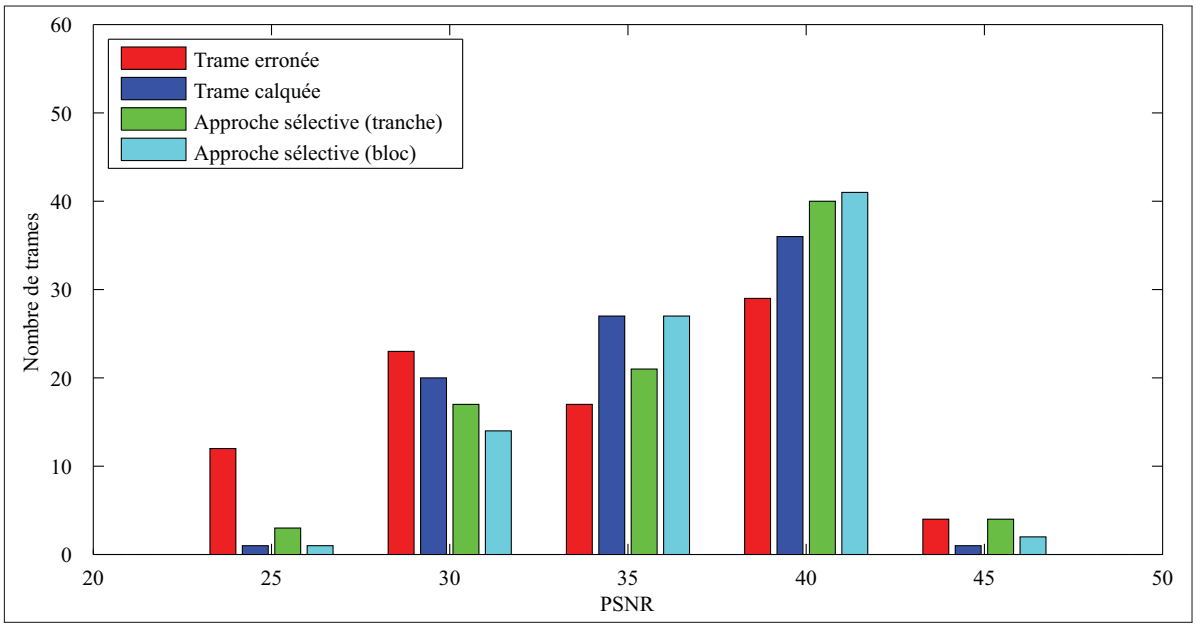


Figure II.38 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0008, FMO = Entrelacé)

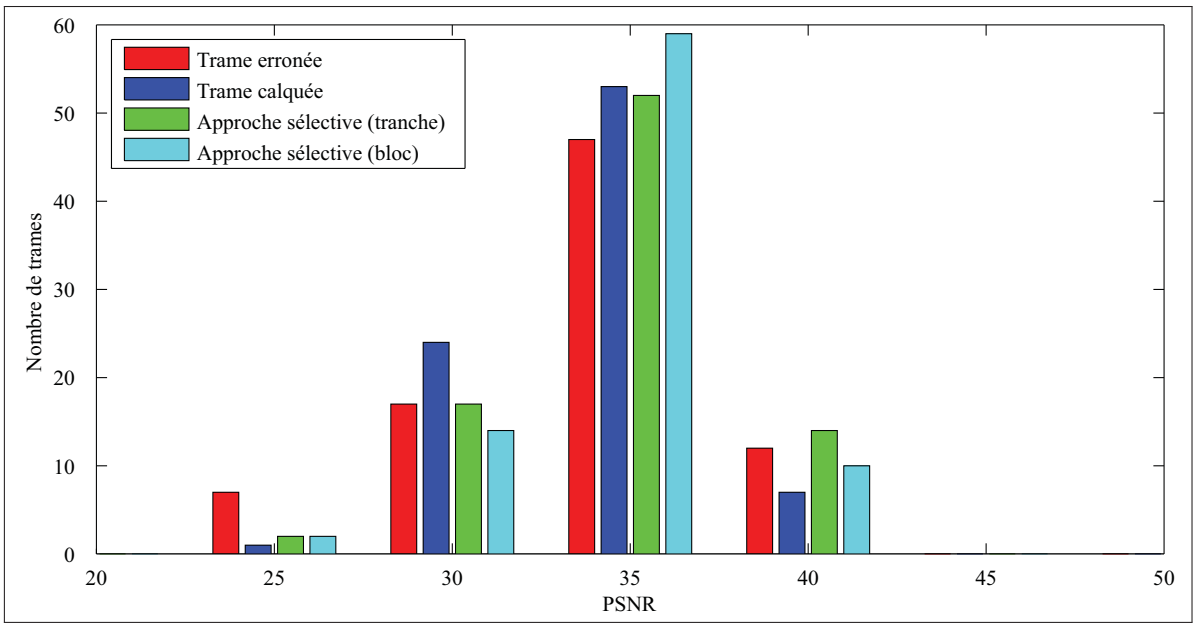


Figure II.39 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0008, FMO = Entrelacé)

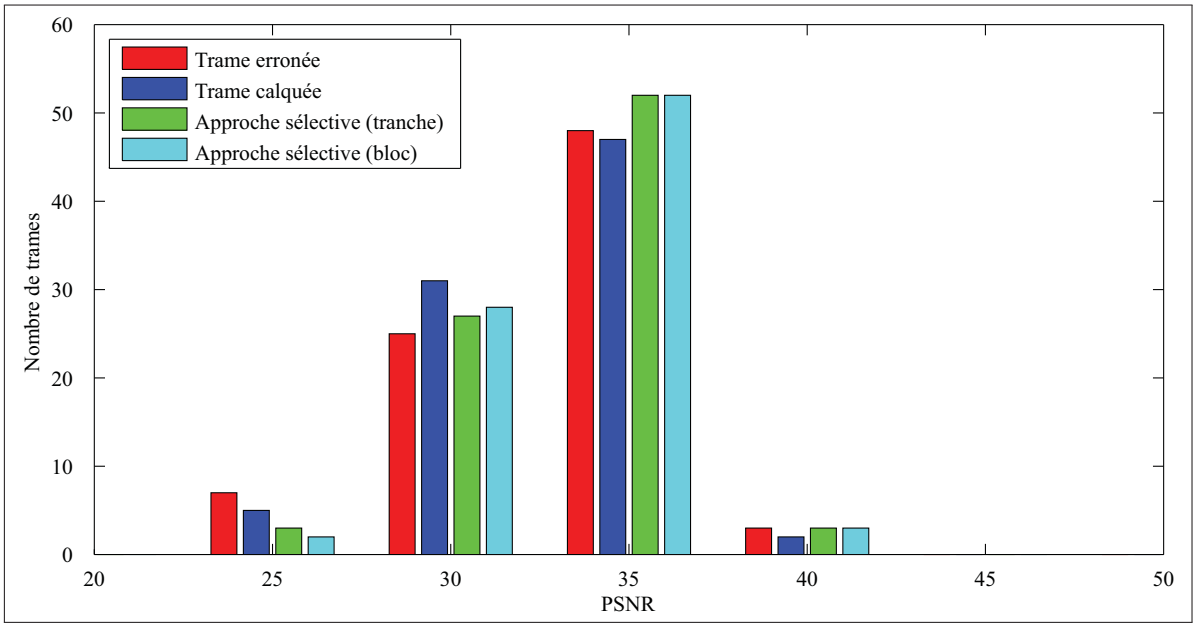


Figure II.40 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0008, FMO = Entrelacé)

2.3 Taux d'erreurs binaires (BER) 0.0016

Tableau II.9 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0016 (dispersé).

Approche	**PSNR Moyen dispersé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.57	39.59	36.88
Sélective par bloc (avec référence)	38.47	38.10	37.10	35.66
Sélective par tranche (avec référence)	37.96	37.46	36.45	35.28
Sélective par bloc	**38.12**	**37.61**	**36.53**	**35.13**
Sélective par tranche	**37.92**	**37.30**	**36.21**	**35.25**
Dissimulation tranche calquée	37.80	36.87	35.78	34.27
Trame corrompue	37.80	36.88	35.78	34.27

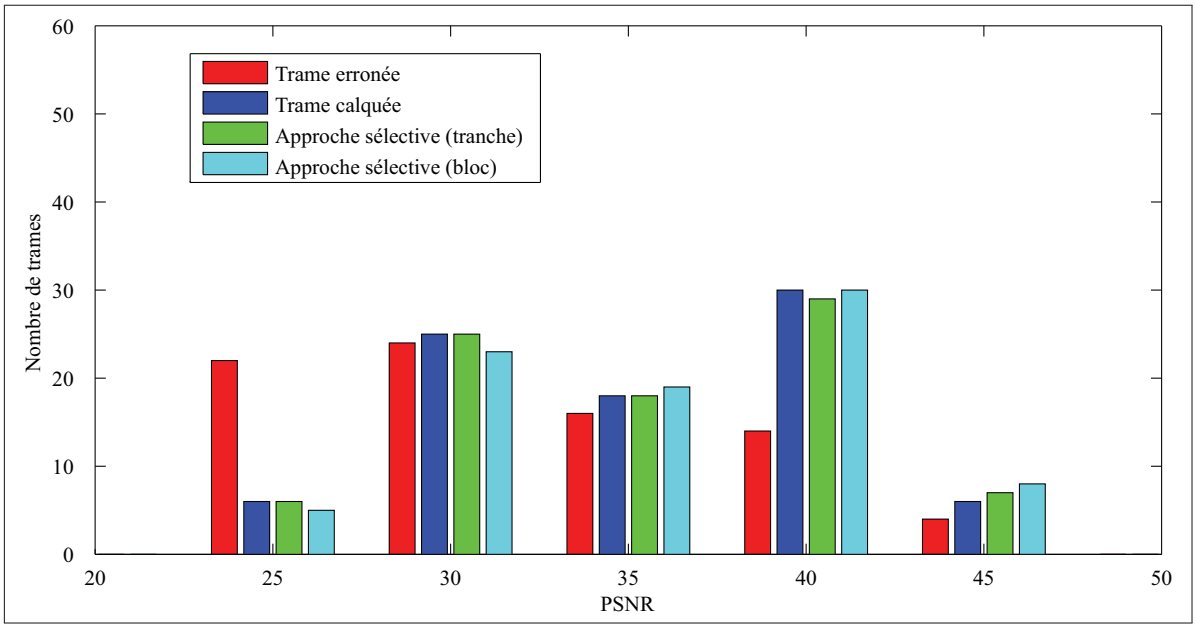


Figure II.41 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0016, FMO = Dispersé)

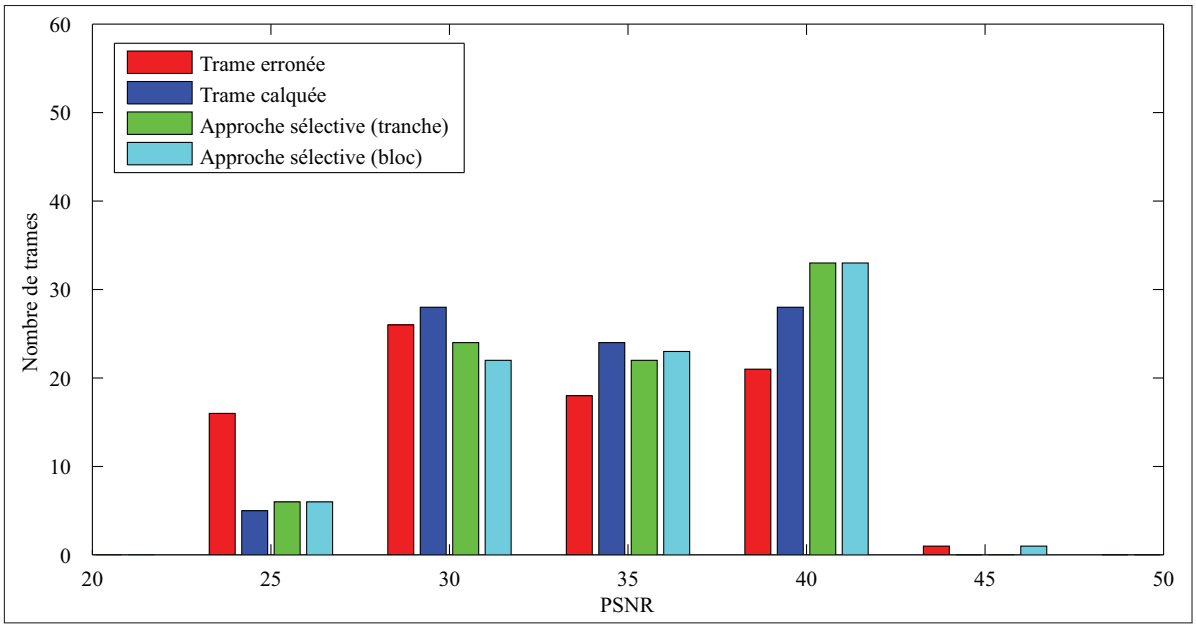


Figure II.42 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0016, FMO = Dispersé)

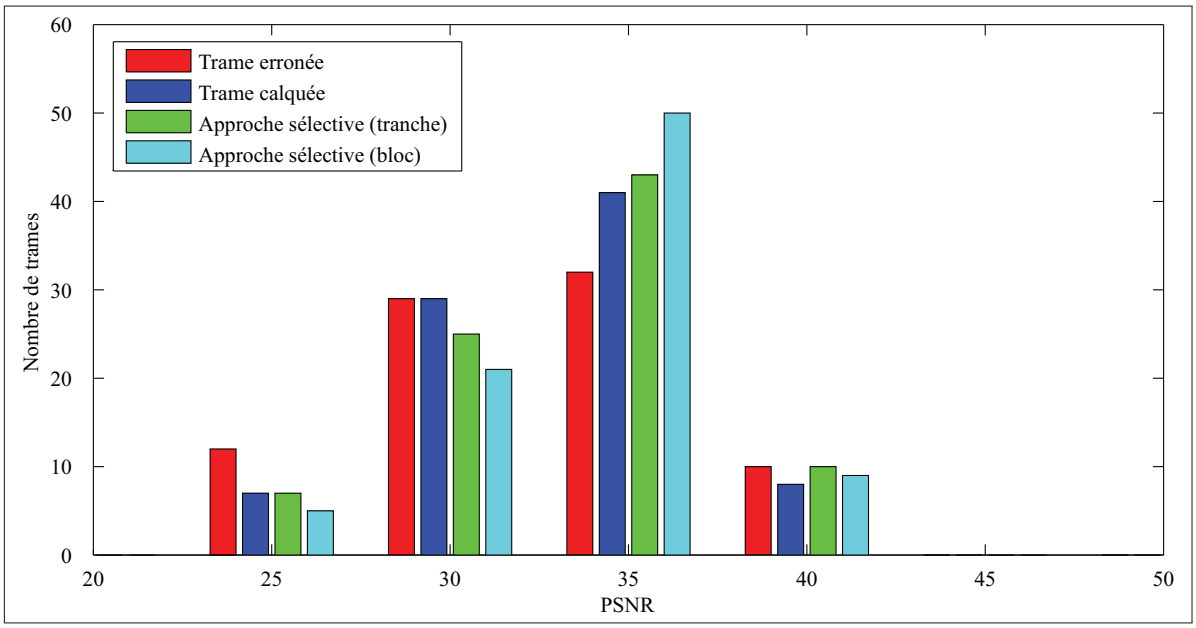


Figure II.43 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0016, FMO = Dispersé)

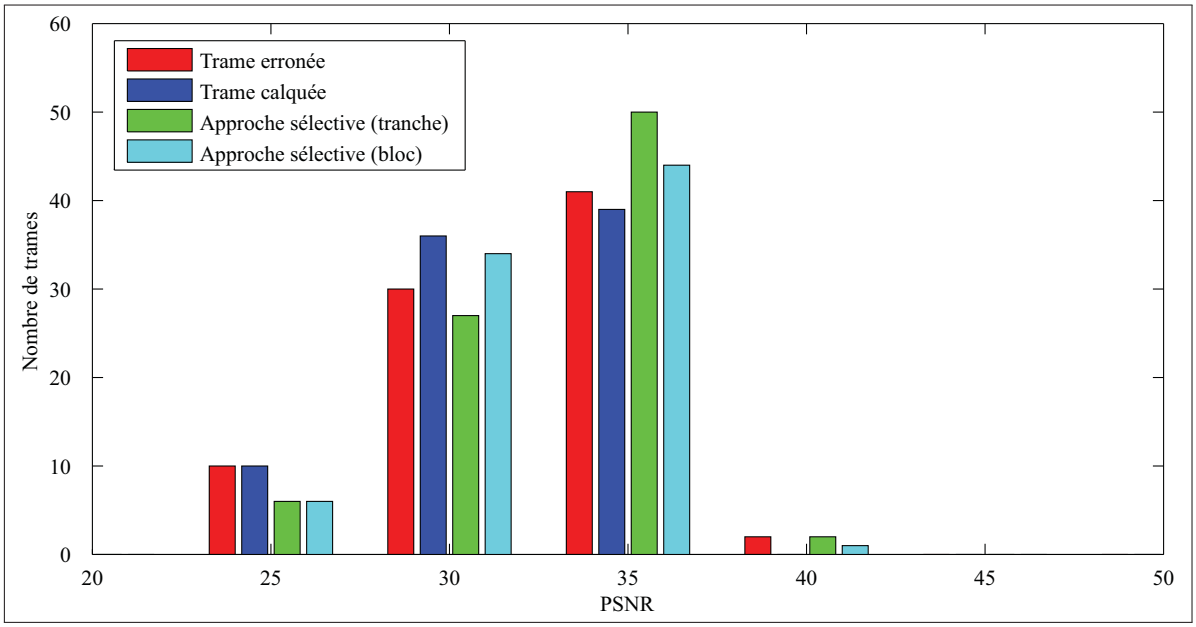


Figure II.44 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0016, FMO = Dispersé)

Tableau II.10 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0016 (entrelacé).

Approche	**PSNR Moyen entrelacé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.56	39.56	36.85
Sélective par bloc (avec référence)	40.22	39.13	37.68	35.94
Sélective par tranche (avec référence)	39.73	38.62	37.31	35.52
Sélective par bloc	**39.67**	**38.59**	**37.24**	**35.54**
Sélective par tranche	**39.57**	**38.43**	**37.13**	**35.27**
Dissimulation tranche calquée	39.26	38.14	36.72	34.94
Trame corrompue	39.26	38.14	36.72	34.94

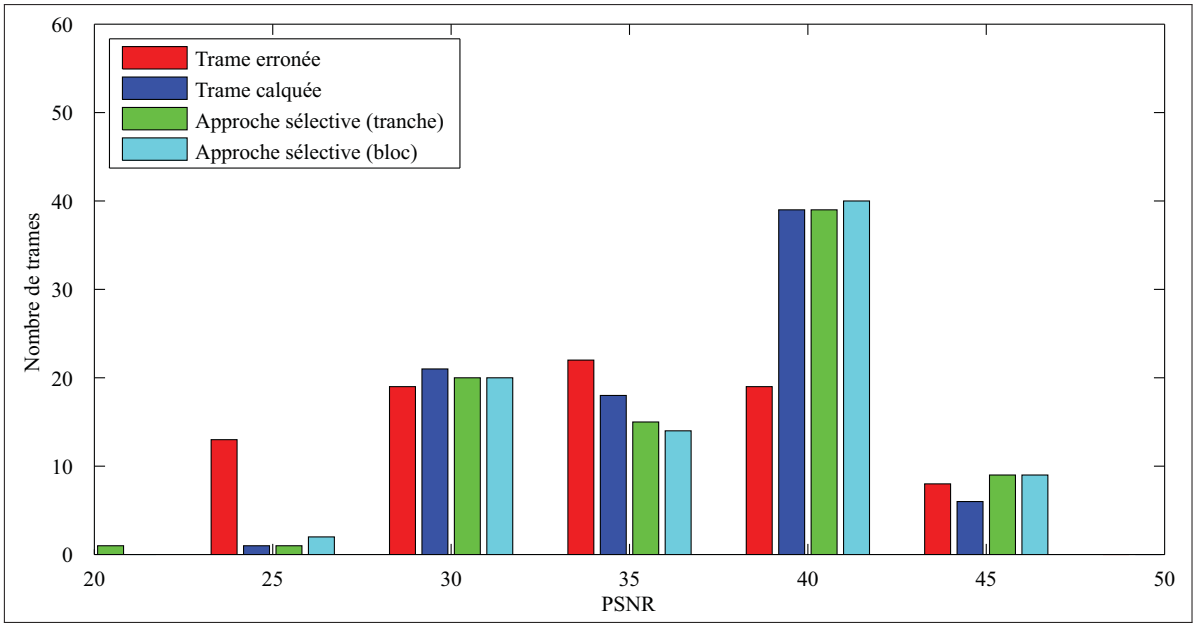


Figure II.45 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0016, FMO = Entrelacé)

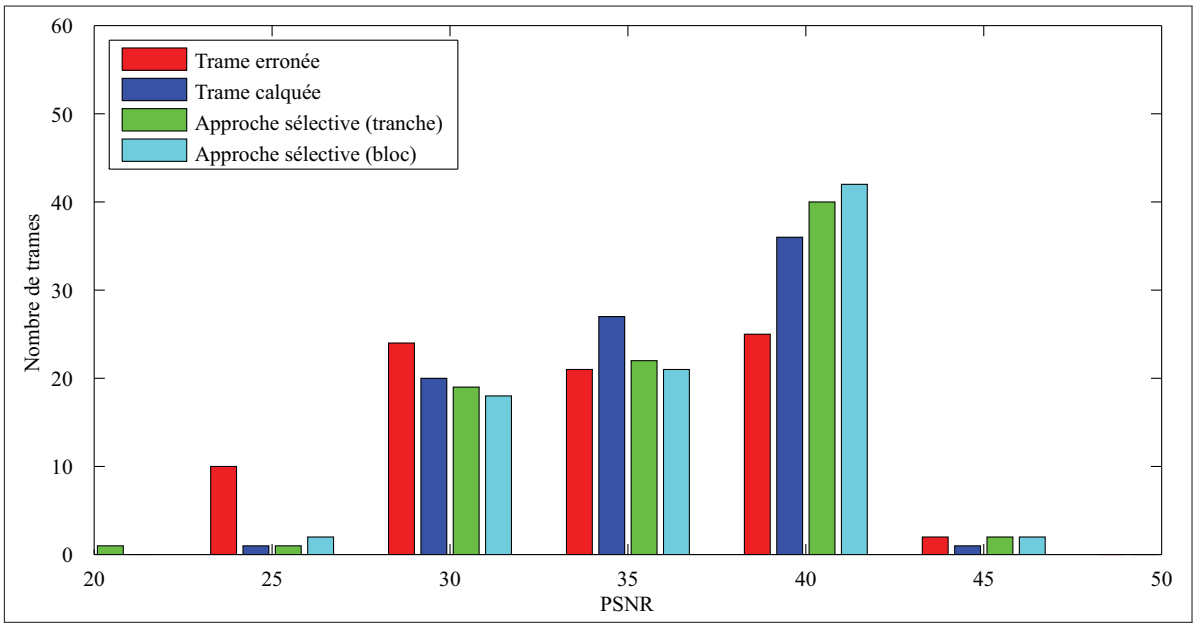


Figure II.46 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0016, FMO = Entrelacé)

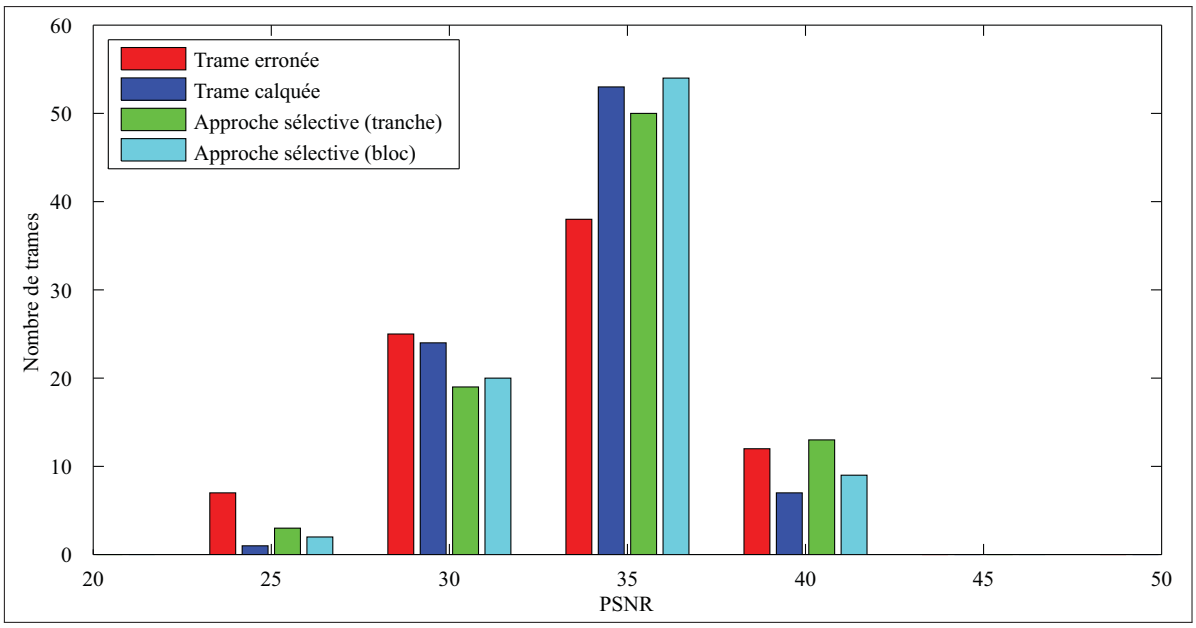


Figure II.47 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0016, FMO = Entrelacé)

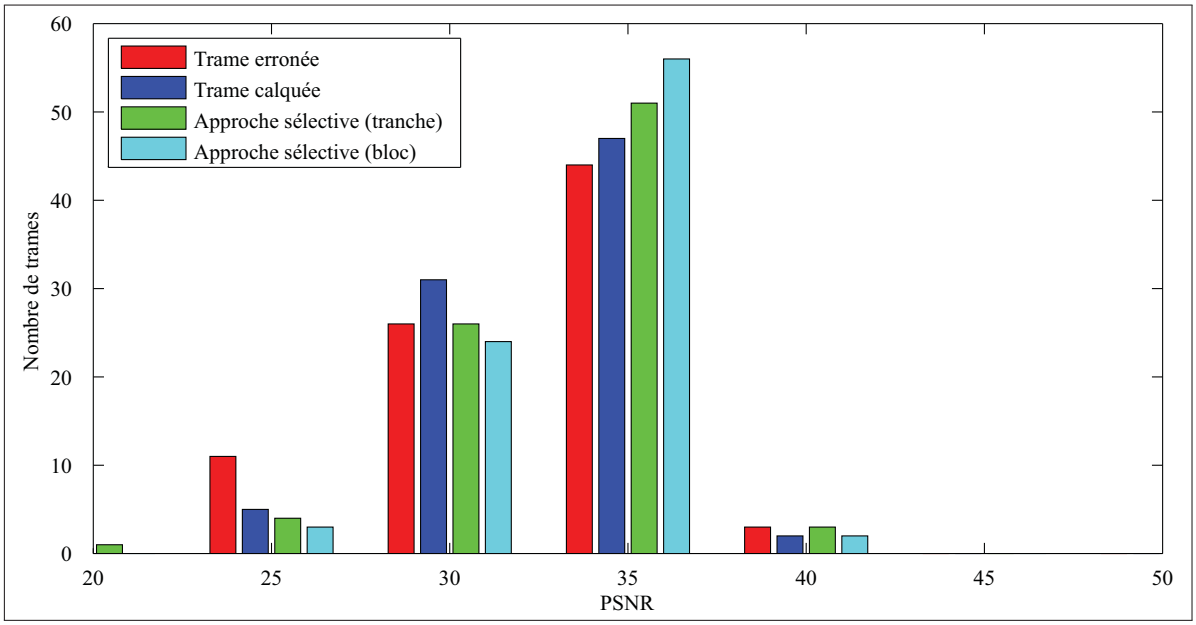


Figure II.48 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0016, FMO = Entrelacé)

2.4 Taux d'erreurs binaires (BER) 0.0032

Tableau II.11 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0032 (dispersé).

Approche	**PSNR Moyen dispersé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.57	39.59	36.88
Sélective par bloc (avec référence)	38.32	37.50	36.68	35.07
Sélective par tranche (avec référence)	37.95	37.01	36.24	34.66
Sélective par bloc	**38.10**	**37.22**	**36.23**	**34.69**
Sélective par tranche	**37.90**	**36.95**	**36.04**	**34.55**
Dissimulation tranche calquée	37.80	36.87	35.78	34.27
Trame corrompue	37.80	36.87	35.78	34.27

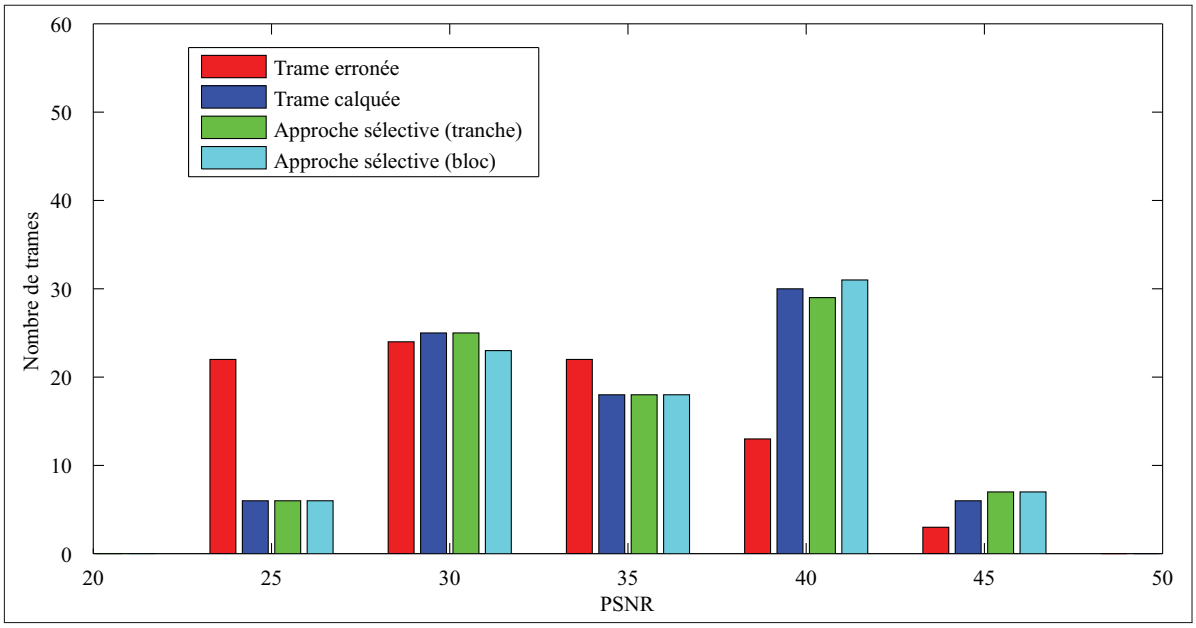


Figure II.49 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0032, FMO = Dispersé)

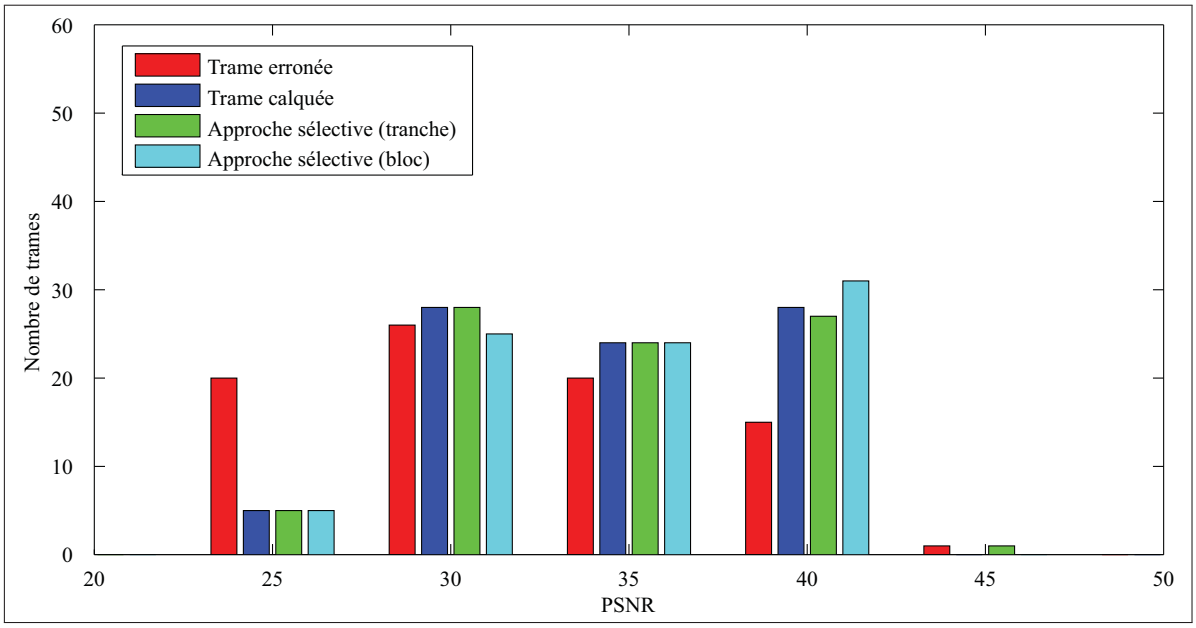


Figure II.50 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0032, FMO = Dispersé)

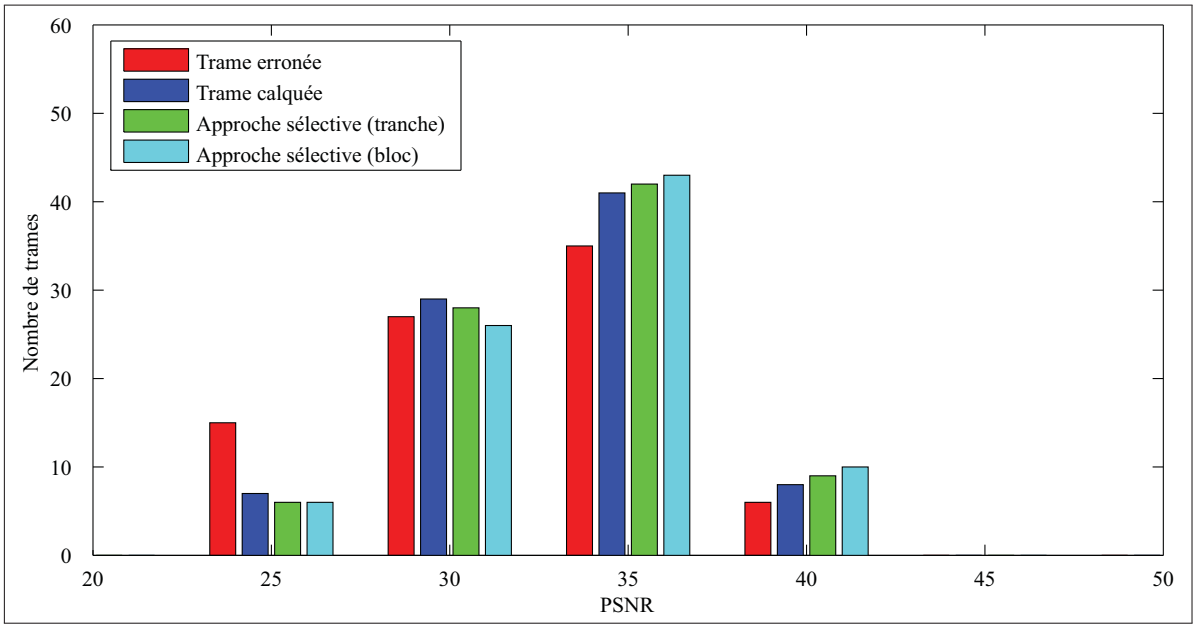


Figure II.51 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0032, FMO = Dispersé)

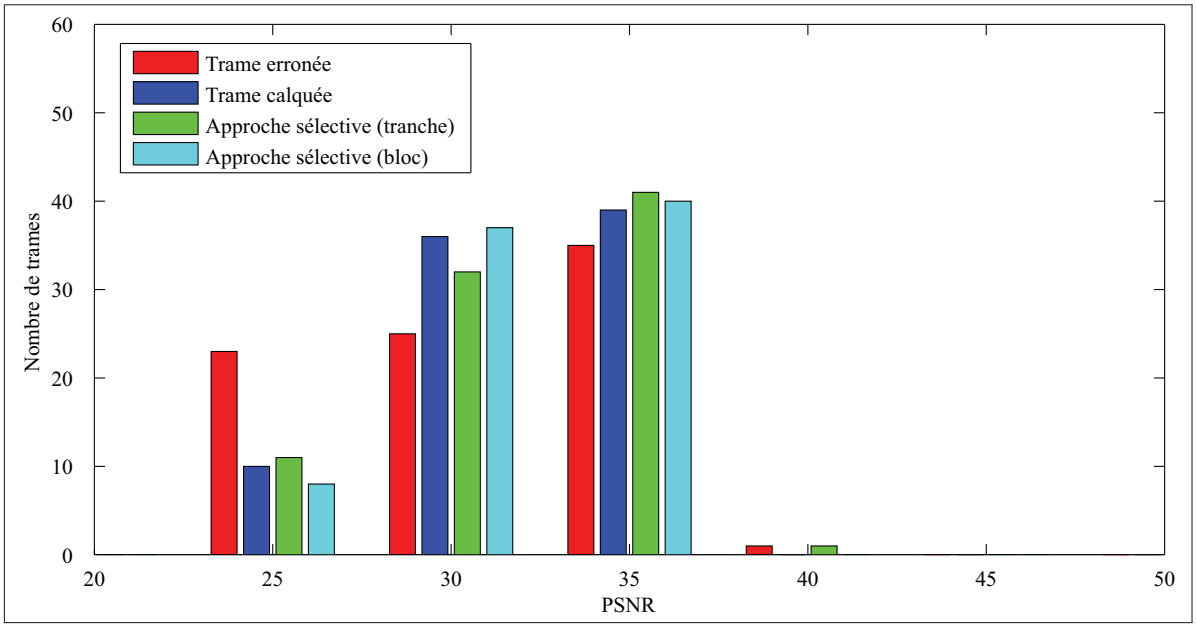


Figure II.52 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0032, FMO = Dispersé)

Tableau II.12 Résumé des résultats obtenus sur l'ensemble des séquences pour un taux d'erreurs de 0.0032 (entrelacé).

Approche	**PSNR Moyen entrelacé (dB)**			
	QP 16	QP 20	QP 24	QP 28
Encodée (sans erreur)	45.83	42.56	39.56	36.85
Sélective par bloc (avec référence)	39.50	38.69	37.51	35.63
Sélective par tranche (avec référence)	39.27	38.36	37.07	35.30
Sélective par bloc	**39.14**	**38.33**	**37.09**	**35.29**
Sélective par tranche	**39.13**	**38.18**	**36.85**	**35.16**
Dissimulation tranche calquée	39.26	38.14	36.72	34.94
Trame corrompue	39.26	38.14	36.72	34.94

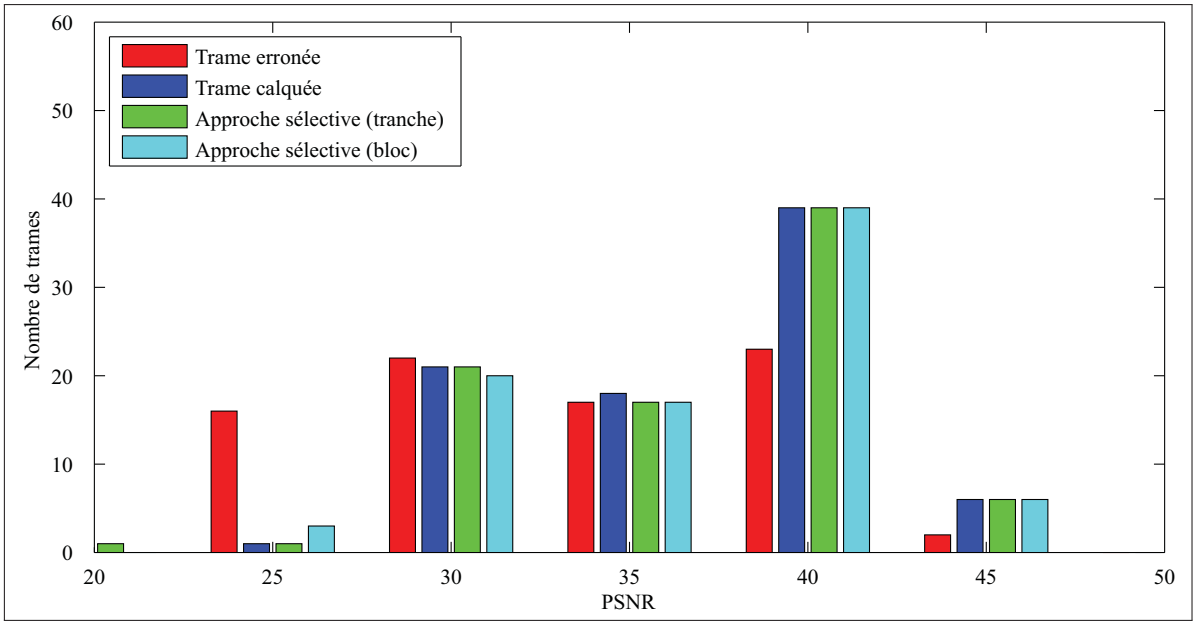


Figure II.53 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 16, BER = 0.0032, FMO = Entrelacé)

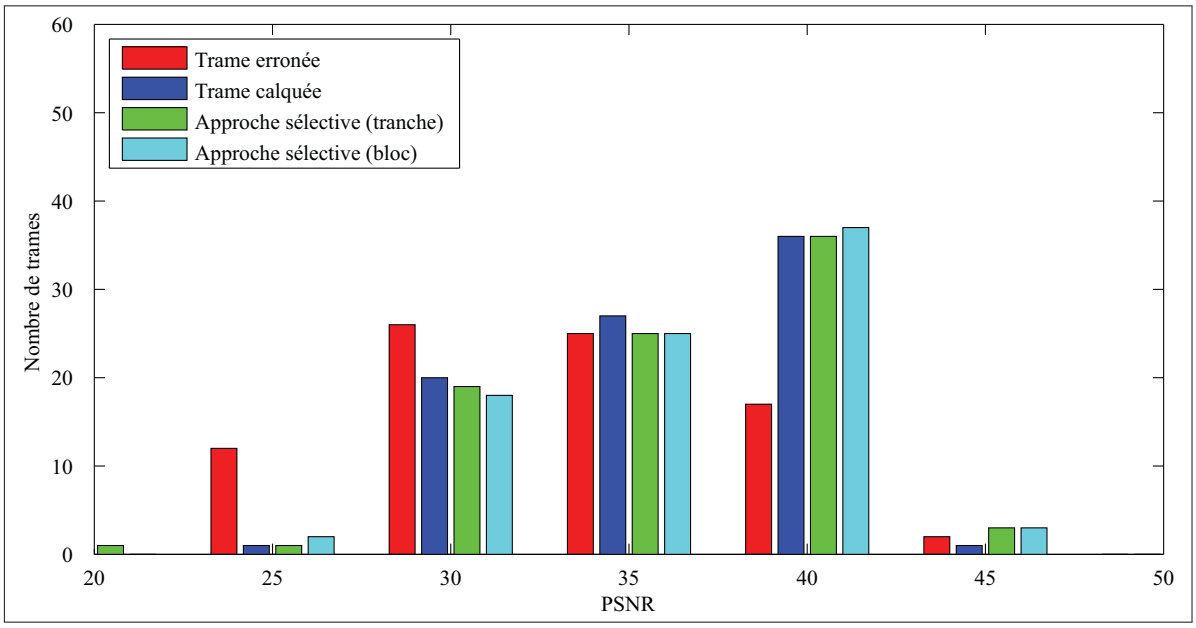


Figure II.54 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 20, BER = 0.0032, FMO = Entrelacé)

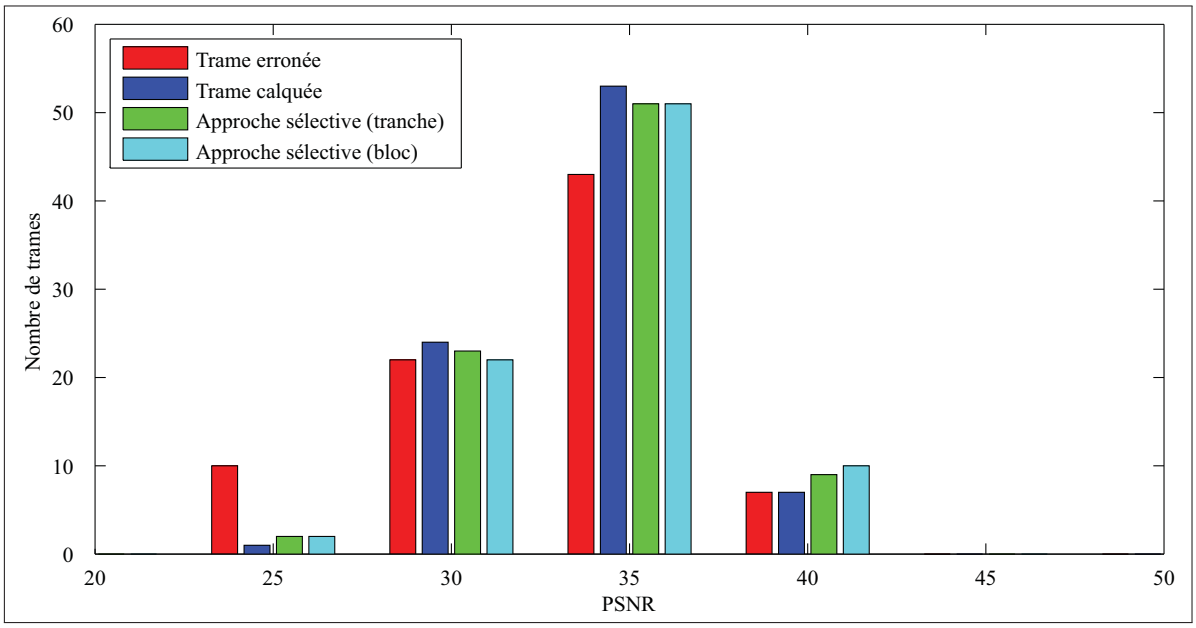


Figure II.55 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 24, BER = 0.0032, FMO = Entrelacé)

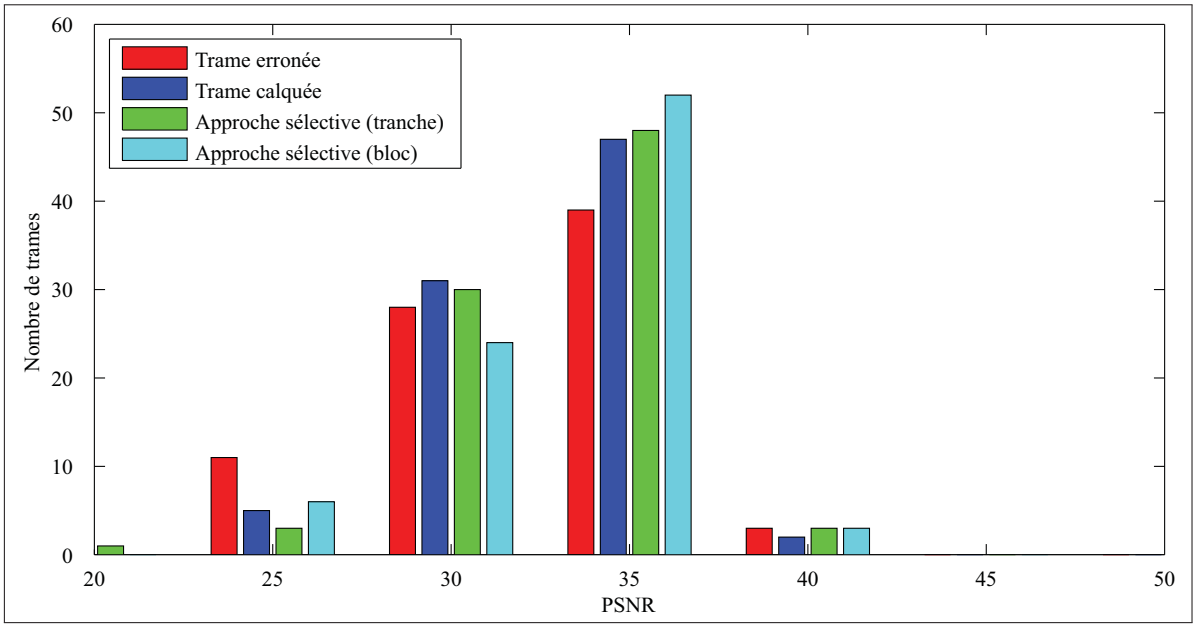


Figure II.56 Histogrammes des PSNR des trames, selon l'approche de dissimulation utilisée. (QP = 28, BER = 0.0032, FMO = Entrelacé)

ANNEXE III

CODE SOURCE

Cette annexe présente le code source des scripts Matlab implémentant les concepts présentés dans cet ouvrage, soit l'arrimage de blocs, la mesure des effets de bloc compensés par le mouvement et la distribution de bordures. Au besoin, un diagramme de classe UML est utilisé pour présenter les scripts associés à un sujet et les liens entre ceux-ci.

1 L'arrimage de blocs (*Block Matching*)

L'arrimage de blocs se fait dans le script *blockMatching.m*. Cependant, afin de découpler ce script à l'algorithme de recherche de bloc candidat (*fullSearch.m*) ainsi qu'à l'évaluation de la corrélation entre les blocs (*SAD.m*), ses notions ont été développées dans des scripts à part, interchangeable.

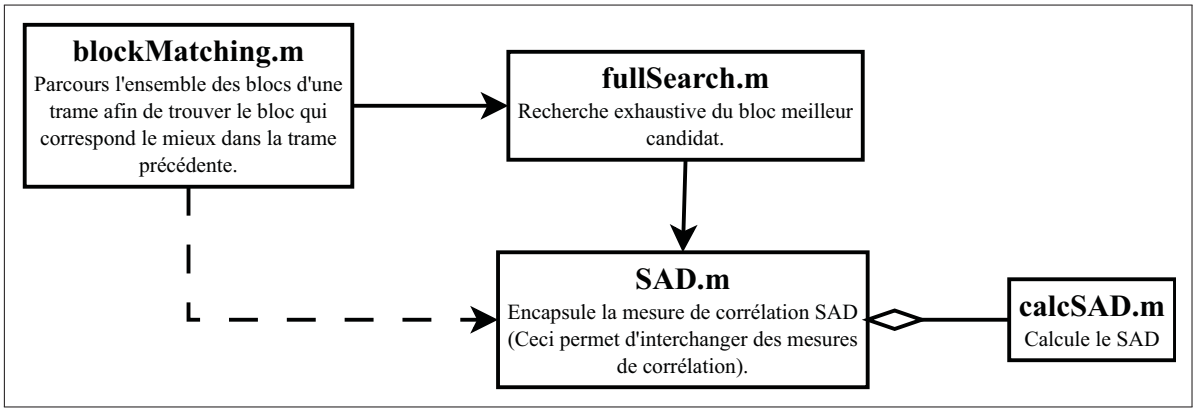


Figure III.1 Diagramme UML de classes des fichiers script utilisés pour l'arrimage de blocs.

Extrait III.1 blockMatching.m

```
function [results mvs] = blockMatching(prev, image, blockSize, searchRadius, searchFunction↩
    , correlationObject)
%BLOCKMATCHING matches the a block of the image with a block from the previous image.
%
%   [ result mvs ] = blockMatching( prev, image, blockSize, searchRadius, ↩
    correlationFunction ) returns the
```

```
%   value of the correlation between the blocks of the frame and the best match in the ↩
    previous image for
%   all blocks in the image based on the given blockSize.
%
% By Luc Trudeau, 25 August 2010.
[height width z] = size(image);
bHeight = height/blockSize;
bWidth = width/blockSize;

results = zeros(bHeight, bWidth);
mvs = zeros(bHeight, bWidth, 2);
bM1 = blockSize-1;

for y = 1 : blockSize : height
    yy = ceil(y/blockSize);
    yRange = y:y+bM1;
    searchTop = max(1, y-searchRadius);
    sTM1 = searchTop - 1;
    searchBottom = min(height, y+searchRadius);
    searchYRange = searchTop:searchBottom;
    for x = 1 : blockSize : width
        xx = ceil(x/blockSize);
        xRange = x:x+bM1;
        searchLeft = max(1, x-searchRadius);
        sLM1 = searchLeft - 1;
        searchRight = min(width, x+searchRadius);
        searchXRange = searchLeft:searchRight;

        block = image(yRange, xRange, :);
        searchArea = prev(searchYRange, searchXRange, :);

        [ results(yy,xx),  mV ] = searchFunction(searchArea, block, blockSize, ↩
            correlationObject);
        mvs(yy, xx, :) = [ mV(1) - (y - sTM1), mV(2) - (x - sLM1) ];
    end
end

end
```

Extrait III.2 fullSearch.m

```
function [optResult, mv] = fullSearch(searchArea, block, blockSize, correlationObject)
% FULLSEARCH finds the best candidat for a block in a given search area.
%
%   [minResult, mv] = matchBlock(searchArea, block, blockSize,
%   correlationFunction) returns the best match obtained with the
%   correlation function between the block and all possible blocks within the
%   searchArea.
%
% By Luc Trudeau, 25 August 2010.

bM1 = blockSize -1;
[height width z] = size(searchArea);
mid = [ceil(height / 2), ceil(width / 2)];

% TODO The starting position could be optimized to the center of the search area.
mv = [1, 1];
optResult = correlationObject.fun(block, searchArea(1:1+bM1, 1:1+bM1, :));
for y = 1: height - bM1;
    yRange = y:y+bM1;
    for x = 1: width - bM1;
        xRange = x:x+bM1;
        result = correlationObject.fun(block, searchArea(yRange, xRange, :));

        if correlationObject.optimisation(result, optResult)
            newMV = [y x];
            if result == optResult
                if sum(abs(mv - mid) - abs(newMV - mid)) > 0
                    optResult = result;
                    mv = newMV;
                end
            else
                optResult = result;
                mv = newMV;
            end
        end
    end
end
end
```

Extrait III.3 SAD.m

```
function [ sadObject ] = SAD
% SAD Creates a SAD correlation object.
```

```
%
%   The SAD correlation object encapsulates the information
%   related to using the SAD. Where fun is a pointer to the
%   function, optimisation is the optimisation function in
%   in this case, we want to minimize. And perfectCorrelation
%   indicates the value returned by fun when the correlation
%   is perfect.
%
% By Luc Trudeau, 25 August 2010.
    sadObject.fun = @calcSAD;
    sadObject.optimisation = @(x, y) x <= y;
    sadObject.perfectCorrelation = 0;

end
```

Extrait III.4 calcSAD.m

```
function [ result ] = calcSAD( A, B )
%CALCSAD Sum of absolute differences
%
%   result = calcSAD( A, B ) returns the sum of the absolute difference between
%   each value in the matrix A and the corresponding value in the matrix B.
%   These differences are summed to create a simple metric of block
%   similarity, the L1 norm of the difference image.
%
% By Luc Trudeau, 25 August 2010.

diff = imabsdiff(A, B);
result = sum(diff(:));

end
```

2 Évaluation des effets de blocs compensés par le mouvement (MCB).

À l'aide des vecteurs de mouvement issus de l'arrimage de blocs de la section précédente, nous sommes en mesure de calculer les effets de blocs compensés par le mouvement (*temporalBlockiness.m*). Notons que l'ancien nom du *motion compensated blockiness* était *temporalBlockiness*, ce qui explique les noms de fichiers. Le script *interpolateClasses.m* n'est pas requis, mais permet de mesurer le MCB au demi-pixel.

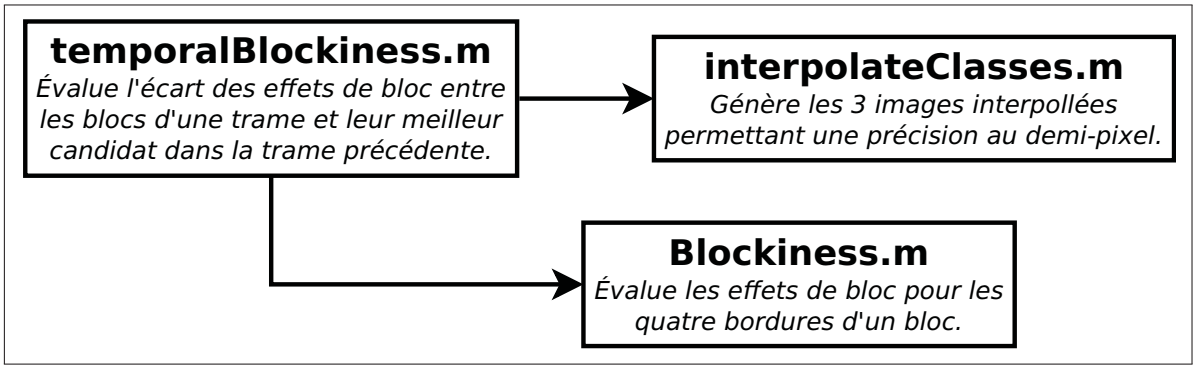


Figure III.2 Diagramme UML de classes des fichiers script utilisés pour mesurer les effets de bloc compensés par le mouvement.

Extrait III.5 temporalBlockiness.m

```
function [ tempBlockiness, diffTop, diffRight, diffBottom, diffLeft] = temporalBlockiness( ↩
    mVs, prev, bad, blockSize)
%TEMPORALBLOCKINESS measure de difference of blockiness between all the
%blocs in the frame and the blockiness of blocs candidate defined by there
%motion vectors.
%
%   [ tempBlockiness, diffTop, diffRight, diffBottom, diffLeft] =
%   temporalBlockiness( mVs, prev, bad, blockSize) returns the temporal
%   blockiness of each bloc in the frame. Difftop, diffRight, diffBottom
%   and diffLeft are the blockiness of each border of all blocks.
%
%
% By Luc Trudeau.
[class2, class3, class4] = interpolateClasses(prev);
[height width, z] = size(mVs);
diffTop = zeros(height, width);
diffRight = zeros(height, width);
diffBottom = zeros(height, width);
diffLeft = zeros(height, width);

for yy = 1: height
    y = (yy-1) * blockSize + 1;
    for xx = 1: width
        x = (xx-1) * blockSize + 1;

        vY = mVs(yy,xx,1);
        vX = mVs(yy,xx,2);

        if floor(vY) ~= vY && floor(vX) ~= vX
```

```
        pY = y + floor(vY);
        pX = x + floor(vX);
        class = class4;
    elseif floor(vY) ~= vY
        pY = y + floor(vY);
        pX = x + vX;
        class = class3;
    elseif floor(vX) ~= vX
        pY = y + vY;
        pX = x + floor(vX);
        class = class2;
    else
        pY = y + vY;
        pX = x + vX;
        class = prev;
    end

    [ top, right, bottom, left ] = blockiness( y, x, bad, blockSize);

    [ prevTop, prevRight, prevBottom, prevLeft ] = blockiness( ...
        pY, pX, class, blockSize);

    if yy > 1
        diffTop(yy,xx) = sum(abs(top - prevTop));
    else
        diffTop(yy,xx) = 0;
    end

    if xx < width
        diffRight(yy,xx) = sum(abs(right - prevRight));
    else
        diffRight(yy,xx) = 0;
    end

    if yy < height
        diffBottom(yy,xx) = sum(abs(bottom - prevBottom));
    else
        diffBottom(yy,xx) = 0;
    end

    if xx > 1
        diffLeft(yy,xx) = sum(abs(left - prevLeft));
    else
        diffLeft(yy,xx) = 0;
```

```
        end
    end
end

tempBlockiness = diffTop + diffRight + diffBottom + diffLeft;
end
```

Extrait III.6 interpolateClasses.m

```
function [ class2, class3, class4 ] = interpolateClasses( image )
%INTERPOLATECLASSES returns the 3 interpolation classes of an image for
%half pixel precision.
%
%   [ class2, class3, class4 ] = interpolateClasses( image ) returns 3
%   images that are interpolations of the pixels between the pixels of
%   image. This allows for half pixel precision.
%
% By Luc Trudeau.
hImage = image / 2;
qImage = image / 4;
[height width z] = size(image);

class2 = hImage(:, 1:end-1,:) + hImage(:, 2:end,:);
class2 = [class2, image(:,end, :)];
class3 = hImage(1:end-1, :,:) + hImage(2:end, :,:);
class3 = [class3; image(end,:, :)];

class4 = zeros(height, width, z);

for y = 1: height
    bottom = min(y + 1, height);
    for x = 1: width
        right = min(x + 1, width);
        class4(y,x, :) = qImage(y,x, :) + qImage(bottom,x, :) + ...
            qImage(y,right, :) + qImage(bottom,right, :);
    end
end
```

Extrait III.7 blockiness.m

```
function [ borderTop, borderRight, borderBottom, borderLeft ] =  blockiness(top, left, ↩
    image, blockSize)
%BLOCKINESS measures de blockiness of bloc borders.
%
%   [ borderTop, borderRight, borderBottom, borderLeft ] =  blockiness(top,
%   left, image, blockSize) returns the difference of the pixels along the
%   borders of a block.
%
% By Luc Trudeau.
bM1 = blockSize - 1;
[height width z] = size(image);
right = left + bM1;
bottom = top + bM1;

borderTop = zeros(1,blockSize,z);
borderRight = zeros(blockSize,1,z);
borderBottom = zeros(1,blockSize,z);
borderLeft = zeros(blockSize,1,z);

xRange = left:right;
yRange = top:bottom;

if top > 1
    borderTop = image(top-1, xRange, :) - image(top, xRange, :);
end

if left > 1
    borderLeft = image(yRange, left-1, :) - image(yRange, left, :);
end

if bottom < height
    borderBottom = image(bottom, xRange, :) - image(bottom+1, xRange, :);
end

if right < width
    borderRight = image(yRange, right, :) - image(yRange, right+1, :);
end

end
```

3 Distribution de bordures

Finalement, la distribution des bordures issue du MCB est accomplie à l'aide du script *enhanceBlockiness.m*.

Extrait III.8 enhanceBlockiness.m

```
function [ blocks ] = enhanceBlockiness( tops, rights, bottoms, lefts, threshold )
%ENHANCEBLOCKINESS Distributes a blocky border to the most blocky block.
%
%   [ blocks ] = enhanceBlockiness( tops, rights, bottoms, lefts, threshold
%   ) returns the distributed blockiness values of the blocks of a frame.
%
% By Luc Trudeau.
if nargin == 4;
    threshold = 0;
end

blocks = tops + rights + bottoms + lefts;
[height width z] = size(blocks);
map = true(height, width);
maxBlock = max(blocks(:));

while maxBlock > threshold;

    availableBlocks = blocks .* map;
    maxBlock = max(availableBlocks(:));
    [yPos, xPos] = find(availableBlocks == maxBlock,1);
    map(yPos, xPos) = 0;

    %Check top
    if yPos > 1
        if tops(yPos, xPos) > 0 && bottoms(yPos-1, xPos) > 0
            if blocks(yPos-1,xPos) > blocks(yPos, xPos)
                tops(yPos, xPos) = 0;
            else
                bottoms(yPos-1, xPos) = 0;
            end
        end
    end

    %Check left
    if xPos > 1
        if lefts(yPos, xPos) > 0 && rights(yPos, xPos-1) > 0
```

```
            if blocks(yPos,xPos-1) > blocks(yPos, xPos)
                lefts(yPos, xPos) = 0;
            else
                rights(yPos, xPos-1) = 0;
            end
        end
    end

    %Check bottom
    if yPos < height
        if bottoms(yPos, xPos) > 0 && tops(yPos+1, xPos) > 0
            if blocks(yPos+1,xPos) > blocks(yPos, xPos)
                bottoms(yPos, xPos) = 0;
            else
                tops(yPos+1, xPos) = 0;
            end
        end
    end

    %Check right
    if xPos < width
        if rights(yPos, xPos) > 0 && lefts(yPos, xPos+1)
            if blocks(yPos,xPos+1) > blocks(yPos, xPos)
                rights(yPos, xPos) = 0;
            else
                lefts(yPos, xPos+1) = 0;
            end
        end
    end

    blocks = tops + bottoms + lefts + rights;
end
end
```

BIBLIOGRAPHIE

Arizona State University, 2010. « YUV Video Sequences ». In *Video Trace Library*. En ligne. <http://trace.eas.asu.edu/yuv/>. Consulté le 17 mai 2011.

Cai, C., H. Zeng, et S. K. Mitra, 2009. Fast motion estimation for H.264. *Image Commun.*, 24 :630–636. ISSN 0923-5965. doi : 10.1016/j.image.2009.02.012.

comScore, 2011. « comScore Releases February 2011 U.S. Online Video Rankings ». In *comScore, Inc. - Measuring the Digital World*. En ligne. <http://www.comscore.com/Press_Events/Press_Releases/2011/3/comScore_Releases_February_2011_U.S._Online_Video_Rankings>. Consulté le 25 Mars 2011.

Cortes, C. et V. Vapnik, 1995. Support-vector networks. *Machine Learning*, 20 :273–297. ISSN 0885-6125. 10.1007/BF00994018.

Duhamel, P. et M. Kieffer, 2010. *Joint Source-Channel Decoding : A Cross-Layer Perspective with Applications in Video Broadcasting*. Academic Press, 1 edition.

Eastman Kodak Company, 1999. « Kodak Lossless True Color Image Suite ». In *RWF's Eclectic Miscellany*. En ligne. <http://r0k.us/graphics/kodak/>. Consulté le 25 Mars 2011.

Farrugia, R. A. et C. J. Debono, 2008. A Robust Error Detection Mechanism for H.264/AVC Coded Video Sequences Based on Support Vector Machines. *IEEE Trans. on Circuits and Systems for Video Technology*, 18(12) :1766–1770. ISSN 1051-8215. doi : 10.1109/TCSVT.2008.2004919.

Farrugia, R. A. et C. J. Debono, 2010. *Resilient Digital Video Transmission over Wireless Channel using Pixel-Level Artefact Detection Mechanisms*, pages 71 – 96. Intech. ISBN 978-953-7619-70-1.

Ghanbari, M., 1993. Cell loss concealment in ATM video codes. *IEEE Trans. Circuits Syst. Video Technol.*, 3 :238 – 247.

Girod, B. et N. Färber, 1999. Feedback-based error control for mobile video transmission. *Proceedings of the IEEE*, pages 1707–1723.

Haar, A., 1911. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 71 :38–53. ISSN 0025-5831. <http://dx.doi.org/10.1007/BF01456927>. 10.1007/BF01456927.

Hemami, S. et T.H.-Y. Meng, 1995. Transform coded image reconstruction exploiting interblock correlation. *Image Processing, IEEE Transactions on*, 4(7) :1023 –1027. ISSN 1057-7149. doi : 10.1109/83.392344.

Huffman, D., 1952. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9) :1098–1101. ISSN 0096-8390. doi : 10.1109/JRPROC.1952.273898. <http://dx.doi.org/10.1109/JRPROC.1952.273898>.

Ikuno, J. C., 2007. Performance of an Error Detection Mechanism for Damaged H.264/AVC Sequences. Mémoire de maîtrise, Technische Universitat Wien Institut fur Nachrichtentechnik und Hochfrequenztechnik Universitat Politècnica de Catalunya Escola Politècnica Superior de Castelldefels.

Joint Video Team (JVT) d'ISO/IEC MPEG et ITU-T VCEG, 2009. « 0000189 : Crash in JM 16.1 Decoder ». In *Mantis JVT JM H.264/AVC reference software project*. En ligne. <https://ipbt.hhi.fraunhofer.de/mantis/view.php?id=189>. Consulté le 17 mai 2011.

Joint Video Team (JVT) d'ISO/IEC MPEG et ITU-T VCEG, 2010. « H.264/AVC JM Reference Software ». In *Heinrich Hertz Institute*. En ligne. <http://iphome.hhi.de/suehring/tml/>. Consulté le 17 mai 2011.

Juniper Research, 2010. « Smartphone Video Call Users to reach 29 million by 2015 Globally, finds Juniper Research ». In *Telecoms Analysis Reports from Juniper Research*. En ligne. <http://juniperresearch.com/viewpressrelease.php?pr=209>. Consulté le 25 Mars 2011.

Li, Z. N. et M. S. Drew, 2004. *Fundamentals of Multimedia*. Pearson Prentice-Hall, Upper Saddle River, NJ. ISBN 0120618721.

List, P., A. Joch, J. Lainema, G. Bjontegaard, et M. Karczewicz, 2003. Adaptive deblocking filter. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7) :614–619. doi : 10.1109/TCSVT.2003.815175. <http://dx.doi.org/10.1109/TCSVT.2003.815175>.

Malvar, H. S., A. Hallapuro, M. Karczewicz, et L. Kerofsky, 2003. Low-complexity transform and quantization in H.264/AVC. *IEEE Trans. Circuits Systems Video Technology*, pages 598–603.

Modestino, J. et D. Daut, 1979. Combined Source-Channel Coding of Images. *Communications, IEEE Transactions on*, 27(11) :1644 – 1659. ISSN 0090-6778. doi : 10.1109/TCOM.1979.1094335.

Nielsen, 2011. « Apple Leads Smartphone Race, while Android Attracts Most Recent Customers ». In *Nielsen Wire*. En ligne. <http://blog.nielsen.com/nielsenwire/online_mobile/apple-leads-smartphone-race-while-android-attracts-most-recent-customers/>. Consulté le 25 Mars 2011.

Office québécois de la langue française, 2011. « Le grand dictionnaire terminologique ». In *Office québécois de la langue française*. En ligne. <http://www.granddictionnaire.com/>. Consulté le 25 Mars 2011.

Richardson, I. E., 2003. *H.264 and MPEG-4 Video Compression : Video Coding for Next Generation Multimedia*. Wiley, 1 edition. ISBN 9780470848371.

Schäfer, R., T. Wiegand, et H. Schwarz, 2003. The emerging H.264/AVC standard. *EBU Technical review*.

Shannon, C. E., 1948. A mathematical theory of communication. *Bell system technical journal*, 27.

Smith, C. et D. Collins, 2007. *3G Wireless Networks, Second Edition*. McGraw-Hill, Inc., New York, NY, USA, 2 edition. ISBN 007226344X, 9780072263442.

Stockhammer, T., Miska M. Hannuksela, et T. Wiegand, 2003. H.264/AVC in wireless environments. *IEEE Transactions on Circuits and Systems for Video Technology*, 13 (7) :657–673. ISSN 1051-8215. doi : 10.1109/TCSVT.2003.815167.

Sullivan, G. J. et T. Wiegand, 2005. Video Compression - From Concepts to the H.264/AVC Standard. *Proceedings of the IEEE*, 93(1) :18–31. ISSN 0018-9219. doi : 10.1109/JPROC.2004.839617. <http://dx.doi.org/10.1109/JPROC.2004.839617>.

Sun, H. et W. Kwok, 1995. Concealment of damaged block transform coded images using projections onto convex sets. *Image Processing, IEEE Transactions on*, 4(4) :470 –477. ISSN 1057-7149. doi : 10.1109/83.370675.

Sun, H., K. Challapali, et J. Zdepski, 1992. Error concealment in digital simulcast AD-HDTV decoder. *Consumer Electronics, IEEE Transactions on*, 38(3) :108 –118. ISSN 0098-3063. doi : 10.1109/30.156671.

Superiori, L. et O. Nemethova, 2006. Performance of a H.264/AVC Error Detection Algorithm Based on Syntax Analysis. *in Proc. of the Int. Conf. on Advances in Mobile Computing and Multimedia (MoMM 2006*, pages 49–58. Rinton Press.

Superiori, L., O. Nemethova, et M. Rupp, 2007. Detection of visual impairments in the pixel domain of corrupted H.264/AVC packets. *Procs. of the Picture Coding Symposium (PCS 2007), Lisboa, Portugal*, pages 7–9.

Talluri, R., 1998. Error-resilient video coding in the ISO MPEG-4 standard. *Communications Magazine, IEEE*, 36(6) :112 –119. ISSN 0163-6804. doi : 10.1109/35.685373.

Tourapis, A. M., O. C. Au, et M. L. Liou, 2001. Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation. *Proceedings of Visual Communications and Image Processing 2001 (VCIP'01)*.

Wang, L. et D.-C. He, 1990. Texture classification using texture spectrum. *Pattern Recognition*, 23(8) :905 – 910. ISSN 0031-3203. doi : DOI:10.1016/0031-3203(90)90135-8.

Wang, Y. et Q.-F. Zhu, 1998. Error control and concealment for video communication : a review. *Proceedings of the IEEE*, 86(5) :974 –997. ISSN 0018-9219. doi : 10.1109/5.664283.

Wang, Y., Q.-F. Zhu, et L. Shaw, 1993. Maximally smooth image recovery in transform coding. *Communications, IEEE Transactions on*, 41(10) :1544 –1551. ISSN 0090-6778. doi : 10.1109/26.237889.

Wang, Y., Zhang Y.-Q., et J. Osterman, 2001. *Video Processing and Communications*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition. ISBN 0130175471.

Wenger, S., 2003. H.264/AVC over IP. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7) :645–656. ISSN 1051-8215. doi : 10.1109/TCSVT.2003.814966. <http://dx.doi.org/10.1109/TCSVT.2003.814966>.

Wiegand, T., G. J. Sullivan, G. Bjontegaard, et A. Luthra, 2003. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7) :560–576. ISSN 1051-8215. doi : 10.1109/TCSVT.2003.815165. <http://dx.doi.org/10.1109/TCSVT.2003.815165>.

Wu, J. et J.M. Boyce, 2006. An error concealment scheme for entire frame losses based on h.264/avc. *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, page 4 pp. doi : 10.1109/ISCAS.2006.1693620.

Yan, B. et K. Wing, 2003. Analysis and detection of MPEG-4 visual transmission errors over error-prone channels. *Consumer Electronics, IEEE Transactions on*, 49(4) :1424 – 1430. ISSN 0098-3063. doi : 10.1109/TCE.2003.1261250.

Ye, S., X. Lin, et Q. Sun, 2003. Content based error detection and concealment for image transmission over wireless channel. *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 2, pages II–368 – II–371 vol.2. doi : 10.1109/ISCAS.2003.1205984.

Zimmermann, H., 1980. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4) :425–432.