N^o d'ordre : ANNÉE 2014

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du signal et de l'image

Ecole doctorale Matisse

présentée par

Safa Chérigui

préparée à l'unité de recherche UMR 6074
IRISA/INRIA

Techniques de codage d'images basées représentations parcimonieuses de scènes et prédiction spatiale multi-patches

Thèse soutenue à (lieu)
le (date)

devant le jury composé de :

François-Xavier COUDOUX
Professeur à l'Université de Valencienne / Rapporteur

Marco CAGNAZZO
Professeur à Télécom ParisTech / Rapporteur

Luce MORIN
Professeur à l'INSA de Rennes / Examinateur

Patrick LE CALLET
Professeur à l'Université de Nantes/ Examinateur

Christine GUILLEMOT
Directeur de recherche à l'INRIA / Directeur de thèse

Dominique THOREAU
Ingénieur de recherche à Technicolor / Co-directeur de thèse

Philippe GUILLOTEL
Distinguished Scientist à Technicolor / Membre invité

Table des matières

Remerciements

J'adresse tous mes remerciements à l'ensemble des personnes qui ont contribué à la réalisation et l'amélioration de mes travaux de thèse ainsi qu'à la rédaction de mon manuscrit.

En premier lieu, je tiens à exprimer ma reconnaissance et ma gratitude à ma directrice de thèse, Christine Guillemot, Directrice de recherche à l'INRIA et responsable du projet SIROCCO. Ses conseils avisés et sa capacité à guider mes recherches ont fortement participé à la réussite de ces travaux.

Je tiens à remercier également mon co-directeur de thèse, Dominique Thoreau, ingénieur de recherche à Technicolor, pour m'avoir guidé avec attention durant mes travaux. Ses qualités scientifiques et humaines, ses remarques ont largement contribué à l'aboutissement de cette thèse. Qu'il trouve ici l'expression de ma grande gratitude.

Je remercie également Philippe Guillotel, Distinguished Scientist à Technicolor, pour ses collaborations, à ce travail de recherche, pour son soutien et les nombreux conseils qu'il m'a prodigués. Je tiens également à remercier Patrick Pérez, Distinguished Scientist à Technicolor, pour avoir suivi l'évolution de ma thèse et pour son aide.

Je tiens à remercier particulièrement MM. François-Xavier Coudoux et Marco Cognazzo qui ont accepté de juger ce travail et d'en être les rapporteurs. Mes remerciements s'adressent également aux autres membres du Jury qui me font l'honneur de participer à la soutenance : Mme. Luce Morin et M. Patrick Le Callet.

Introduction générale

Présentation générale

Pourquoi comprimer les images et la vidéo quand le débit maximal d'information disponible sur Internet suit une courbe exponentielle dont la pente est encore plus forte que celle de la loi de Moore, et que la capacité des disques durs augmente d'environ 40% chaque année ? Le problème est que cette augmentation de ressource ne résout pas le problème, voire au contraire l'accroît car elle favorise l'arrivée de nouveaux services et de nouveaux usages qui font que plus de données transitent sur les réseaux. Ainsi l'ADSL a permis le déploiement du streaming et d'applications telles que YouTube, la 4G et ses débits supérieurs à 20 Mbit/s vont aussi amener à d'autres consommations. De façon similaire, la disponibilité de bande passante permet aussi d'accroître la qualité des vidéos ou d'améliorer l'expérience de l'utilisateur au travers de la 3D, de l'Ultra-HD, de fréquence d'images augmentée, ou de vidéo à dynamique étendue (proche de la vision humaine). Ainsi on estime que le trafic vidéo représentera 2/3 du trafic internet mondial en 2017, avec une augmentation considérable dans les prochaines années :

- Pour le fixe, augmentation de 30 à 35% par an ;
- Pour le mobile, évolution de 60% par an.

Et le total du trafic pourrait aller jusqu'à 9 térabits/s à l'heure chargée d'ici à 2015. Il est donc essentiel de prolonger la progression de l'efficacité des outils de compression, et même si le récent format MPEG HEVC a permis de gagner un facteur 2 sur son prédécesseur MPEG AVC, ce ne sera pas suffisant pour absorber les demandes à venir.

Les outils de codage vidéo actuels sont tous issus du même principe de base, à savoir un découpage en blocs de l'image, l'utilisation de la corrélation spatiale et temporelle entre blocs, une décomposition fréquentielle pour décorréler l'information du bloc, une quantification psychovisuelle et un codage entropique pour tenir compte des propriétés statistiques du signal. On distingue deux catégories de compression : le codage sans pertes et le codage avec pertes. Le premier ne spécifie pas le débit que l'on cherche à atteindre mais retourne un signal strictement identique au signal original. Ce type de compression est notamment utile dans le domaine médical où les données sont précieuses pour établir un diagnostic correct. Cependant, ce type de codage ne permet pas d'atteindre des forts taux de compression. Quant au second, il accepte une dégradation du signal afin de satisfaire une contrainte de débit fixé. On cherche néanmoins à minimiser cette dégradation. Ce type de codage est davantage appliqué dans le domaine du multimédia où l'on peut tolérer un certain niveau de dégradation sans que l'oeil humain ne discerne une grande modification. Plus récem-

ment une voie intermédiaire focalise l'intérêt, il s'agit de compression visuellement sans perte. Ceci signifie qu'une certaine dégradation est acceptée à condition qu'elle ne soit pas perceptible par l'oeil d'un expert, même dans des conditions de visualisation poussées.

Le savoir-faire des systèmes de codage avec pertes réside dans la nature même des données qui sont encodées. Les méthodes les plus efficaces et les plus utilisées se basent principalement sur le codage prédictif. L'idée est de générer une image, dite de prédiction, la plus similaire à l'image source via divers outils de prédiction afin de venir encoder uniquement l'image résiduelle, beaucoup moins coûteuse que l'image source. Cette prédiction exploite la corrélation très importante qui existe entre pixels voisins et images voisines. Parmi les techniques permettant de générer une image de prédiction, il existe la prédiction spatiale, également appelée "prédiction intra" où le codeur s'appuie sur un voisinage local connu pour prédire un bloc de l'image. La qualité de la prédiction est fondamentale dans un schéma de compression car elle conditionne directement les performances de compression. En effet, plus la prédiction est efficace, moins la quantité de données résiduelles à encoder est importante et meilleurs seront les taux de compression.

Le standard de compression le plus couramment utilisé aujourd'hui dans les applications multimédia est le format MPEG AVC, il sera remplacé dans les années à venir par le format MPEG HEVC. La prédiction spatiale de ces standards repose sur la propagation unidirectionnelle du signal image issu du voisinage local du bloc considéré (cf. figure 1−gauche). Bien que les outils de prédiction spatiale utilisés dans ces codeurs fournissent de bonnes performances de codage, ils ne permettent pas de prédire des textures complexes. Par ailleurs, ces outils de prédiction n'exploitent que la corrélation locale entre les voisinages locaux pour réduire la redondance et minimiser l'information à encoder d'une image mais n'exploitent pas la corrélation de patches à travers des voisinages non locaux. Cette thèse vise donc à explorer de nouveaux schémas de compression d'images fixes afin d'améliorer les techniques actuelles de prédiction intra, en étendant ces schémas locaux et monodimensionnels à des schémas globaux, multidimensionnels et multi-patches.

Contributions

Dans ce contexte, plusieurs techniques sont proposées dans ce manuscrit pour apporter ce nouveau aspect en codage d'image, à savoir la multi-dimensionnalité, comme illustré Figure 1. La prédiction locale actuelle (Figure 1−gauche) se contente des pixels voisins du bloc considéré alors que nous souhaitons prendre en compte tout le voisinage causal du bloc (Figure 1−droite).

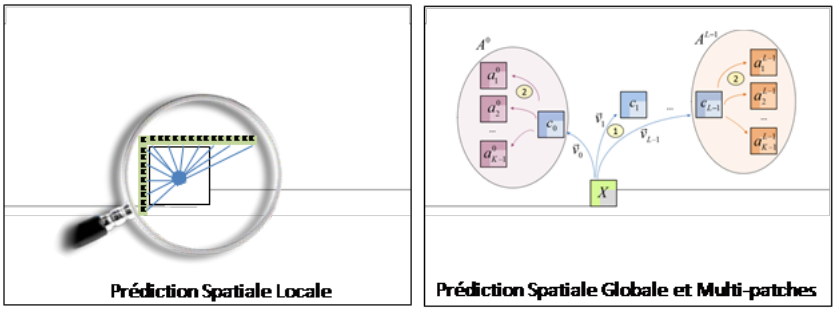

Figure 1 – *Méthodes de prédiction spatiales locales (à gauche) et globales (à droite)*

Dans une première partie des travaux nous avons donc abordé les techniques de prédiction spatiale basées bloc et nous sommes orientés vers l'étude d'un schéma de compression hybride intégrant correspondance de bloc et correspondance de gabarit (*Block Matching* et *Template Matching*). Le gabarit utilisé ici correspond à l'ensemble des pixels du voisinage causal du bloc considéré (en forme de L inversé). Ces méthodes de prédiction ont été intégrées dans un encodeur de type MPEG AVC (KTA software, Key Technical Area) et ont montré une certaine efficacité pour les zones d'images homogènes. De plus, le gabarit (*template*) est apparu une solution très efficace pour éviter le coût de codage d'un vecteur d'indice.

Malheureusement, les textures complexes sont beaucoup plus difficiles à prédire à partir d'un seul bloc, même similaire, de l'image. Nous avons donc étudié une autre approche hybride s'appuyant sur des techniques de réduction de dimensionnalité (LLE, NMF). Ces techniques de prédiction vont au-delà de la simple mise en correspondance de blocs et sont basées sur une combinaison linéaire de plusieurs patches. L'idée sous-jacente est qu'une texture complexe peut être interpolée par une combinaison linéaire de plusieurs textures similaires. L'amélioration apportée à ces outils repose principalement sur l'utilisation d'un vecteur afin de sélectionner plus judicieusement les patches utilisés lors de l'approximation linéaire. Plusieurs alternatives ont été proposées pour définir ce vecteur et ainsi optimiser la sélection des patches. De plus, la définition d'un dictionnaire de taille "réduite" contenant un sous ensemble de patches issus de la fenêtre causale a également été effectuée afin d'une part d'accélérer la sélection des patches dans les outils de prédiction développés, et d'autre part de réduire le coût de codage du vecteur.

Reste que le coût de ce vecteur réduit l'efficacité globale de la méthode. L'utilisation du gabarit pour trouver les patches permet de compenser cette perte, mais avec la contrainte que le gabarit doit correspondre à la texture du bloc à coder, ce qui n'est pas toujours le cas. D'où l'idée de considérer un dictionnaire de patches représentatifs de l'image. La notion d'épitome correspond exactement à cette approche. Nous avons donc étudié comment cette forme condensée d'une image pourrait être utilisée dans un contexte de codage multi-patches. Dans un premier temps, un nouvel algorithme permettant d'analyser les structures répétitives d'une image source a donc été étudié afin d'extraire une représentation compactée de cette image. Pour ce faire, le concept d'épitome relatif à la factorisation de textures a été retenu pour notre étude. En effet, un épitome condensé et une carte d'assignation (ensemble de vecteurs) sont créés itérativement de façon à ce que l'ensemble des blocs constituant l'image puissent être reconstruits à partir des patches de texture issus de l'épitome.

Par la suite, plusieurs schémas de codage ont été explorés afin d'encoder l'image d'origine à partir de cette représentation compacte. Les travaux menés ont permis la définition de deux types de schéma de codage basés sur cette notion d'épitome. Dans le premier schéma de codage proposé, une image de prédiction générée à partir de l'épitome codé/décodé et de la carte d'assignation associée est utilisée comme image de référence pour encoder l'image d'origine. Cette prédiction étant mise ensuite en concurrence avec les modes classiques pour choisir le mode de prédiction le plus efficace au sens d'un critère débit-distorsion. La texture de l'épitome et la carte d'assignation associée doivent également être transmises au décodeur afin que ce dernier puisse reconstruire la même image de référence. Dans une deuxième approche, afin de limiter l'utilisation des vecteurs d'assignation et surtout de propager la texture selon les isophotes de plus grande énergie, un second schéma basé épitome pure a été exploré. Il s'agit de reconstruire l'image à partir de l'épitome codé/décodé et de techniques proches de la synthèse de texture pour propager l'épitome.

Cette approche innovante permet de rompre avec le traditionnel balayage par bloc d'une image dans un ordre ligne-colonne et de proposer un ordre adaptatif fonction du contenu, on notera que le décodeur détermine (évidemment) de façon duale le même ordre de balayage que le codeur. Cet ordre de balayage adaptatif permet de propager la texture dans le sens des contours et autres textures. L'image est alors reconstruite de façon progressive, suivant les deux dimensions X et Y, et dans un ordre déterministe lié au signal.

Finalement les techniques multi-patches et épitome ont été fusionnées pour tirer la quintessence des deux algorithmes.

Organisation du document

Ce manuscrit s'organise en deux parties. La première partie présente l'état de l'art de deux domaines abordés par ces travaux de thèse, à savoir la compression vidéo et les techniques de construction d'image résumé. La deuxième partie détaille les contributions des travaux effectués en abordant successivement les solutions de codage multi-patches, puis la notion d'épitome, pour finir sur la fusion des différentes techniques. Le manuscrit est organisé comme suit :

Chapitre 1. : Cette première partie permet de rappeler les notions de codage prédictif, base des algorithmes de codage utilisés aujourd'hui dans les formats de référence MPEG (AVC et HEVC). Nous focaliserons sur la prédiction intra (spatiale) qui nous intéresse plus particulièrement ici. Nous aborderons aussi les améliorations apportées par les techniques dites de correspondance par gabarit par opposition au bloc (plus communément appelées "Template Matching"), et les récentes avancées sur la compression d'images fixes basées sur les techniques de synthèse de texture.

Chapitre 2. : Nous présenterons ici la notion d'épitome. A savoir qu'est-ce qu'un épitome et quelles sont les différentes techniques permettant d'extraire un épitome d'une image. Nous verrons que la façon dont un épitome est généré n'est pas anodine, elle aura un impact sur les performances de reconstruction ou de codage utilisant cette forme de dictionnaire de patches. Ensuite, nous passerons en revue les très récentes techniques existantes, à ce jour, de prédiction intra basées sur l'utilisation d'une image résumée de type épitome. Nous montrerons leurs limites ouvrant ainsi des perspectives de recherche pertinentes au regard de cette thèse.

Chapitre 3. : Avant d'aborder la notion multi-patches en prédiction intra-image, il est important de considérer l'apport des techniques de mises en correspondance de bloc ou de gabarit. La première justification est le fait que ces techniques sont les plus simples permettant d'étendre la prédiction classique monodirectionnelle, aux deux dimensions d'un bloc. Les textures générées pouvant alors être plus complexes que suivant une seule direction. La seconde justification réside dans le fait que ces méthodes sont le cas particulier des solutions multi-patches et doivent donc servir de référence pour évaluer l'efficacité des solutions multi-patches.

Chapitre 4. : Dans ce chapitre nous aborderons les techniques de prédiction multi-patches s'appuyant sur des outils de réduction de dimensionnalité et intégrant le voisinage local du bloc courant. Après avoir présenté les principes de base des méthodes de réduction de dimensionnalité qui seront utilisées dans ce chapitre, nous rappellerons les méthodes de prédiction intra déjà explorées dans la littérature basées sur les techniques de réduction de dimensionnalité de type NMF (Non Negative Matrix Factorization) et LLE (Locally Linear Embedding) plus particulièrement. Un algorithme innovant, dénommé MANE, sera

proposé. Une autre variante optimisée de cette approche, appelée oMANE, sera également détaillée. Elle améliore les performances des méthodes multi-patches en proposant l'utilisation d'un vecteur de correspondance pour optimiser le choix des patches. Nous montrerons comment choisir au mieux ces vecteurs et comment ce choix impacte la qualité de la prédiction et donc les performances de compression.

Chapitre 5. : Nous décrirons d'abord l'algorithme d'extraction d'épitome proposé. Il est basé sur un modèle simple de translation pure, mais prend en compte le codage postérieur pour optimiser le contenu de cet épitome. Deux schémas de compression d'images fixes basés sur l'épitome seront ensuite détaillés. Le premier utilise l'épitome pour reconstruire les blocs de l'image via un vecteur d'assignation, le second part de l'épitome pour ensuite récursivement reconstruire l'image entière via des techniques de synthèse de texture avec l'utilisation avantageuse d'un ordre de parcours des blocs adaptatif au contenu en termes de structures. La méthode de codage de l'épitome proprement dit sera aussi décrite. Nous montrerons enfin comment adapter les outils de prédiction intra H.264 pour coder une image selon cette façon.

Le manuscrit conclut finalement par un résumé des principales contributions de la thèse et présente des perspectives qui étendent le champ des possibilités apporté par ce travail.

Première partie

Contexte et état de l'art

CHAPITRE 1

Compression vidéo

1.1 Les principes généraux de compression d'images fixes et animées

Un schéma de compression d'images est un procédé permettant de transformer un signal numérique en un train binaire, moins volumineux que le signal d'origine. Ce type de schéma est principalement modélisé par trois opérations indépendantes appliquées au signal (cf. figure 1.1), chacune des opérations visant à réduire un type de redondance spécifique. En effet, en compression d'images, trois types de redondance sont fréquemment identifiés :

- la **redondance interpixel** : les images contiennent de fortes corrélations au niveau spatial, temporel et fréquentiel. La valeur d'un pixel de l'image peut donc être raisonnablement prédite à partir des valeurs des pixels environnants. Ce type de redondance est exploité à l'encodeur lors de la phase de **décorrélation** du signal d'entrée. Deux techniques sont essentiellement utilisées en vue de décorréler le signal :
  - la *prédiction* : on cherche à travers ce traitement à réduire les redondances spatiales et temporelles en ne venant transmettre uniquement au décodeur la différence entre le signal d'entrée et la prédiction.
  - la *transformation* : ce procédé consiste à représenter le signal dans un format plus efficace que le format "2D-pixel". Cette transformation est réalisée afin de mettre en évidence les informations pertinentes.
- la **redondance psychovisuelle** : le système visuel humain (SVH) est capable de détecter dans l'image des informations qui sont moins importantes que d'autres. Ces informations sont dites être psychovisuellement redondantes. Par conséquent, on peut retirer des détails qui ont très peu d'importance vis à vis du système visuel humain sans dégrader de façon significative la qualité de l'image. Ce type de redondance est exploré à l'encodeur lors de la phase de **quantification**, cette phase étant irréversible.
- la **redondance de codage** : une image contient de la redondance de codage lorsque les codes associés aux valeurs des niveaux de gris n'ont pas été sélectionnés en tirant parti des probabilités des événements. Afin de prendre en compte ce type de redondance, l'encodeur applique un codage à longueur variable (i.e. le **codage entropique**) juste après la phase de quantification.

Nous allons détailler plus précisément dans les sections suivantes les différents traitements effectués dans un schéma de compression typique en commençant par aborder le

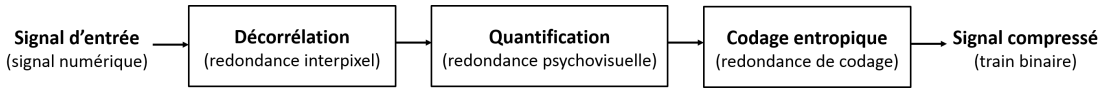

FIGURE 1.1 – *Structure d'un schéma de compression d'images.*

processus de prédiction.

1.1.1 la prédiction

Le procédé de prédiction consiste à approximer un pixel ou un bloc de l'image à partir d'informations causales, c'est à dire à partir de données issues d'une partie de l'image courante déjà encodée ou bien d'une image précédemment décodée de la séquence. L'idée est que le décodeur puisse reconstituer la même prédiction qu'à l'encodeur, sans que l'on ait besoin de transmettre des informations supplémentaires.

En codage vidéo, on distingue deux types de prédiction, **Intra** et **Inter**, selon que l'on exploite respectivement les redondances spatiales ou bien temporelles. Nous reviendrons plus en détail sur la description de ces différents types de prédiction à travers la présentation des standards usuels de compression. Cette étape de prédiction génère ensuite un résidu obtenu par la différence entre les valeurs sources et la prédiction. L'enjeu de l'étape de prédiction est majeur dans les schémas de compression puisqu'une prédiction efficace réduira de façon considérable la quantité d'informations à transmettre.

La section suivante présente les transformations les plus répandues en compression d'images et de vidéos.

1.1.2 la transformation

La transformation est un autre outil qui permet de décorréler le signal. Elle peut être opérée directement sur les données sources ou sur le résidu de prédiction. Cette opération consiste à projeter un signal sur des fonctions de base orthogonales de façon à obtenir une représentation plus compacte de l'information. En effet, l'idée est que l'énergie du signal transformé soit distribuée sur un nombre réduit de coefficients décorrélés. Parmi les transformations les plus répandues, on trouve :

- la **transformée de Karhunen-loève (KLT)** : cette transformée consiste à déterminer les composantes principales du signal en diagonalisant sa matrice de covariance. La transformation est optimale car l'énergie est distribuée sur les composantes principales du signal. La particularité de cette transformée est que les fonctions de base s'adaptent au signal source. Cependant, cet outil est très peu utilisé en pratique car trop complexe à mettre en oeuvre.
- la **transformée de Fourier** : contrairement à la KLT, la base d'arrivée est fixe et des algorithmes rapides tels que la FFT (Fast Fourier Transform) permettent d'en faire un outil plus léger à utiliser. La transformée discrète ou DFT est donnée par :

$$F(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m,n)\, \mathrm{e}^{-\frac{2i\Pi km}{M}}\, \mathrm{e}^{-\frac{2i\Pi ln}{N}} \tag{1.1}$$

  pour une image I ou un bloc de taille $M \times N$. Cette transformée présente cependant l'inconvénient de fournir un signal complexe.
- la **transformée en cosinus discrète (DCT)** : outil de transformation le plus souvent utilisé dans les standards de compression d'images et de vidéos, cette DCT

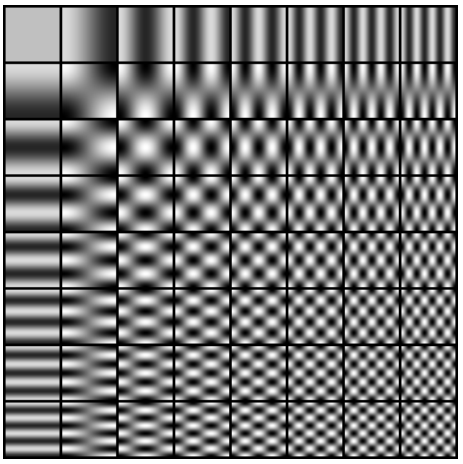

Figure 1.2 – *Fonctions de base 2D de la transformée en cosinus discrète.*

Figure 1.3 – *Exemple de décomposition en ondelettes.*

en 2D est construite en calculant séparément les DCT 1D en horizontal puis en vertical. L'expression de la DCT est la suivante :

$$F(u,v) = \frac{1}{4} C_u C_v \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_{mn} \cos(u\pi \frac{2m+1}{2M}) \cos(v\pi \frac{2n+1}{2N}) \tag{1.2}$$

où

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } u = 0, \\ 1 & \text{sinon,} \end{cases} \quad \text{et } C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } v = 0, \\ 1 & \text{sinon.} \end{cases} \tag{1.3}$$

Cette transformée est composée d'un ensemble de fonctions cosinus correspondant à différentes fréquences. La figure 1.2 indique les fonctions de base élémentaires de la DCT 2D pour un bloc de taille 8x8. L'intérêt de la DCT est qu'elle permet de concentrer l'énergie du signal principalement sur les basses fréquences alors que les autres coefficients correspondant aux hautes fréquences sont d'amplitudes plus faibles, voire nulles.

- la **transformée en ondelettes** : cette transformée est principalement utilisée dans le standard de compression d'images fixes JPEG2000. Elle permet une décomposition hiérarchique de l'image (figure 1.3). A la différence de la DCT, cette décomposition permet une localisation spatiale.

Une fois le signal transformé, une opération de quantification est ensuite appliquée afin de conserver l'information la plus importante pour l'oeil humain.

1.1.3 la quantification

Les propriétés de l'oeil sont prises en compte à travers l'opération de quantification afin de supprimer l'information là où elle n'est pas visible. En effet, la sensibilité du système visuel humain (SVH) étant plus forte dans les basses fréquences que dans les hautes fréquences, la quantification utilisera cette propriété au travers des coefficients de pondération appliqués en fonction de la position des coefficients DCT.

La quantification consiste à proprement parler à réduire la précision du signal. En effet, un signal prenant initialement des valeurs dans un dictionnaire de taille n est représenté dans un nouveau dictionnaire de taille inférieure m. Le signal quantifié pourra ainsi être décrit avec moins de bits que la version initiale. La quantification est une opération irréversible car elle introduit une erreur de quantification. On parle ainsi de compression avec pertes. On distingue plusieurs types de quantification : la quantification scalaire, vectorielle et la quantification en treillis.

Les standards de compression usuels utilisent principalement la quantification scalaire dans laquelle un pas de quantification q_p est défini. Ce pas de quantification indique le niveau de dégradation appliqué au signal. Soit $Y = \{y_0, ..., y_{N-1}\}$ l'ensemble des niveaux de reconstruction possibles. L'opérateur de quantification Q associe un sous-ensemble de l'espace d'entrée à une unique valeur y_i comme suit :

$$Q(x) = y_i \text{ si x } \in \left[y_i - \frac{q_p}{2}, y_i + \frac{q_p}{2}\right[\tag{1.4}$$

Après avoir décorrélé et quantifié le signal, celui-ci est transmis au codeur entropique afin de réduire la quantité de bits nécessaire pour décrire le signal à transmettre.

1.1.4 Codage entropique réversible

Un codeur entropique exploite les statistiques du signal à coder pour construire un code composé de mots de code de différentes tailles. On parle ainsi de codes à longueurs variables (VLC pour Variable Length Code). Parmi les codeurs les plus connus, on peut citer le codage de Huffman, le codage arithmétique ou encore le CAVLC (Context Adaptive Variable Length Coding) qui est utilisé dans H.264/AVC et dans lequel les mots de code les plus courts possibles sont attribués aux symboles les plus fréquents, réduisant ainsi le coût de codage de l'information à coder. Le standard H.264 possède également un second codeur entropique le CABAC (Context Adaptive Binary Arithmetic Coding) qui est plus efficace en termes de compression par rapport au CAVLC (de l'ordre de 10% à 20%) mais au détriment dune augmentation de la complexité. Le CABAC est basé sur l'exploitation de différents modèles en fonction de l'information à coder et de probabilités mises à jour au cours du codage.

1.2 Les normes H.264/AVC et HEVC

Cette section présente les standards les plus utilisés dans le domaine de la compression vidéo, à savoir H.264/AVC ainsi que son nouveau successeur HEVC.

1.2.1 Le standard H.264/AVC

1.2.1.1 Préambule

Le standard H.264/AVC (AVC pour Advanced Video Coding), aussi connu sous le nom MPEG-4 Part 10, a été publié dans sa première version en 2003 par les groupes VCEG (Video Coding Experts Group) ITU-T SG16-Q6 et MPEG (Moving Picture Expert Group) ISO/IEC JTC 1/SC 29/WG regroupés au sein du groupe JVT (Joint Video Team). En 2005, l'extension H.264/AVC FREXt (Fidelity Range Extensions) apportant de nouveaux outils de codage fût finalisée et constitue le High Profile de cette norme qui est maintenant l'un des profils le plus utilisé.

1.2.1.2 Syntaxe hiérarchique

Les standards MPEG décrivent une séquence vidéo en utilisant différents niveaux de granularité. La figure 1.4, présentée dans [BD96], illustre l'imbrication des différents niveaux utilisés :

- La **séquence** qui comprend les paramètres globaux tels que le nombre d'images par seconde, la taille des images, le format de représentation des couleurs.
- Le **GOP** ou groupe d'images : c'est le schéma de codage minimal qui est répété périodiquement dans le temps. Il définit ainsi la période de codage de la séquence.
- L'**Image** est l'unité de la séquence sur l'axe temporel.
- Le **Slice** définit un groupe de macroblocs (voir ci-après). Un slice représente une partie de l'image ou l'image entière. C'est la partie élémentaire permettant la synchronisation du flux. Lorsqu'une partie du flux transmis est erronée, le décodeur passe au traitement du slice suivant.
- Le **Macrobloc** (MB) est un bloc de taille 16×16 pour la composante des luminances. La taille des blocs associés de chrominances dépend du format de couleur choisi : pour un format $4:2:0$ par exemple, la taille des blocs de chrominances sera 8×8.
- Le **Bloc** de taille 4×4 ou 8×8 généralement. C'est à ce niveau que s'opèrent les choix des modes de prédiction et de transformation quand le macrobloc est trop large. La taille du bloc élémentaire permet ainsi de s'adapter à l'activité locale du signal source.

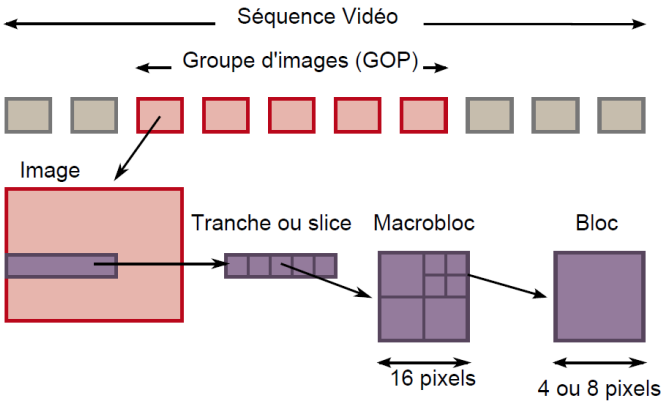

Figure 1.4 – *Les différents niveaux d'une séquence dans la syntaxe MPEG.*

1.2.1.3 Types d'images

A l'intérieur d'un GOP, on distingue trois types d'image :

- **Les images I** pour lesquelles tous les macroblocs sont codés en mode Intra : elles sont indépendantes des images voisines et constituent donc les images de références. Ce sont celles qui consomment le plus de ressources binaires. Elles permettent cependant de se resynchroniser sur le signal source car elles sont codées indépendamment des autres images.
- **Les images P** sont des images prédites temporellement à partir d'une image de référence dans le passé. Les macroblocs sont codés soit en Inter, soit en Intra. Ces images sont codées par différence avec une image passée de type I ou P. Elles bénéficient dune meilleure efficacité de codage que les images I, mais le calcul de la prédiction inter-images est plus coûteux.
- **Les images B** sont des images bi-prédites temporellement à partir de deux images de référence situées, généralement, respectivement dans le passé et dans le futur. Les macroblocs peuvent là encore être codés soit en Inter, soit en Intra. Ce type d'images permet non seulement de réduire significativement le coût de codage par rapport aux images I et P mais également d'introduire de la scalabilité temporelle, *i.e.* la possibilité de décoder la vidéo à différentes cadences. Cependant, elles nécessitent une gestion particulière des images puisque l'ordre de décodage des images n'est plus celui de l'affichage.

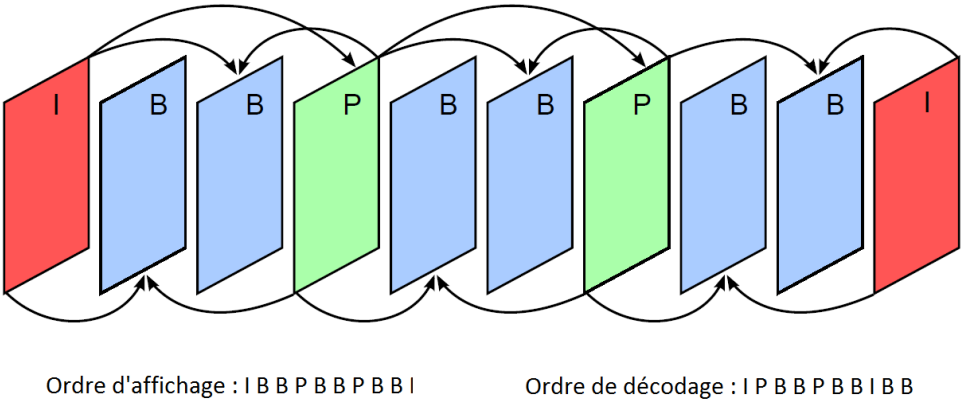

Figure 1.5 – *Exemple d'enchaînement des types d'images pour la prédiction* Inter.

Un exemple d'enchaînement de ces images dans un GOP est illustré en figure 1.5. Ainsi, le codage des images I ne contient qu'une étape de prédiction intra. Pour les autres, les modes de codage intra et inter seront en compétition.

1.2.1.4 Structure de l'encodeur

Le schéma d'encodage du standard H.264 est résumé dans la figure 1.6. Il fonctionne en boucle fermée, c'est à dire que les données déjà codés sont disponibles pour traiter la partie courante. Les données préalablement codés sont obtenus à partir de la boucle de décodage intégrée dans l'encodeur. Les images sont généralement traitées selon un parcours **raster scan**, débutant en haut à gauche de l'image pour terminer en bas à droite. Notons que le décodeur est normatif tandis que l'encodeur ne l'est pas. En effet, l'encodeur peut être modifié pour son optimisation ou l'adaptation à un contexte de transmission vidéo.

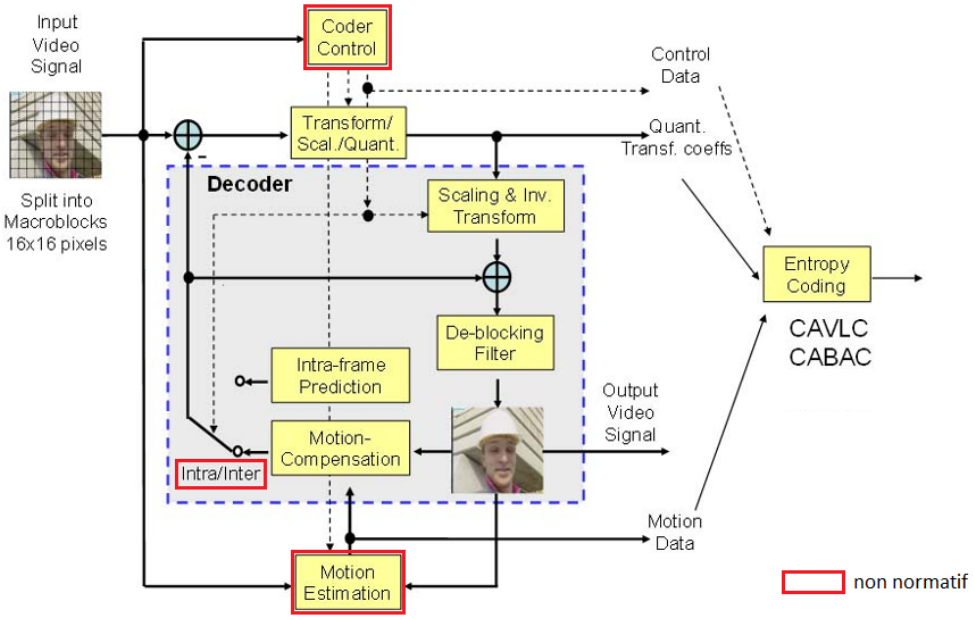

Figure 1.6 – *Schéma d'un encodeur vidéo H.264.*

Les sections suivantes décrivent plus en détails les modes de prédiction Intra et Inter.

1.2.1.5 Prédiction Intra

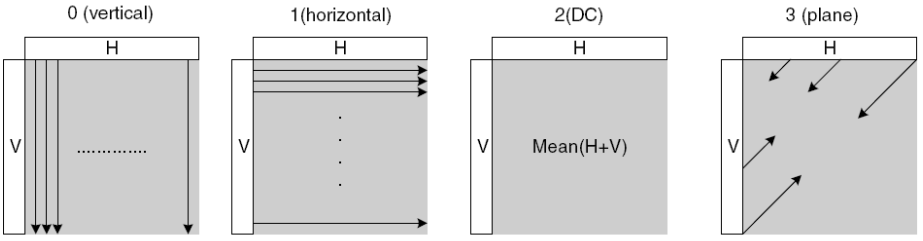

Figure 1.7 – *Modes de prédiction Intra* 16*x*16.

La prédiction Intra s'apparente au codage d'images fixes. En intra, la prédiction d'un bloc est faite à partir des blocs voisins précédemment encodés et reconstruits, les blocs voisins étant choisis parmi le bloc à gauche et les trois blocs au-dessus du bloc courant. La proximité entre le bloc courant et son voisinage laisse espérer une forte corrélation entre les deux, ce qui rend pertinent l'utilisation de l'information locale. Dans le standard H.264/AVC, trois tailles de blocs sont utilisées pour ce type de prédiction : 16x16, 8x8 et 4x4.

Ces partitionnements permettent de s'adapter à l'activité locale contenue dans le macrobloc. En effet, une zone homogène aura tendance à utiliser une prédiction 16x16 alors qu'une zone un peu plus complexe sera plutôt encodée en Intra 4x4. Différents prédicteurs sont définis pour chaque taille de bloc : quatre en Intra 16x16 et neuf en Intra 8x8 et 4x4. La figure 1.7 présente les prédicteurs utilisés pour le partitionnement 16x16 : un prédicteur horizontal, un prédicteur vertical, un prédicteur DC et un prédicteur nommé plane corres-

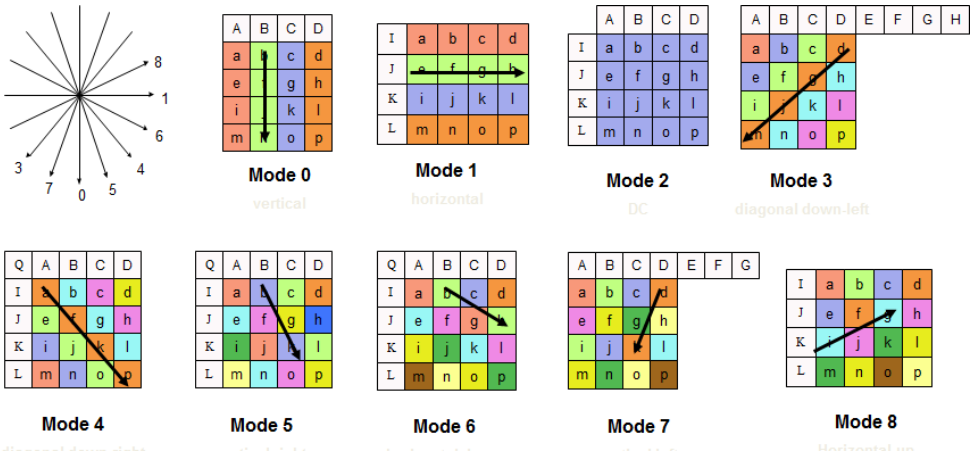

FIGURE 1.8 – *Modes de prédiction Intra 4x4.*

pondant à un dégradé oblique. La figure 1.8 indique l'ensemble des modes de prédiction utilisés pour un bloc 4×4 : le mode DC et huit modes directionnels. Parmi ces derniers, on peut citer les modes horizontal, vertical, diagonal gauche-bas, diagonal droite-bas. Les modes restants correspondent à des directions dont l'angle est supérieur ou inférieur à 45°. Il est à noter que les mêmes modes existent pour un bloc de taille 8×8.

1.2.1.6 Prédiction Inter

La prédiction Inter consiste à réaliser une compensation de mouvement à partir d'une image précédemment encodée sur laquelle une estimation de mouvement est effectuée. Un estimateur de mouvement consiste à déterminer le déplacement de la sous-partition courante (bloc, macrobloc) vis à vis d'une image de référence. En règle générale, les algorithmes d'estimation de mouvement sont de type *block matching*. Cette technique, illustrée par la figure 1.9, permet de trouver dans une image de référence le bloc maximisant une mesure de corrélation avec le bloc courant. Ensuite, le procédé de compensation de mouvement consiste à utiliser le bloc de l'image de référence retenu en tant que prédicteur temporelle.

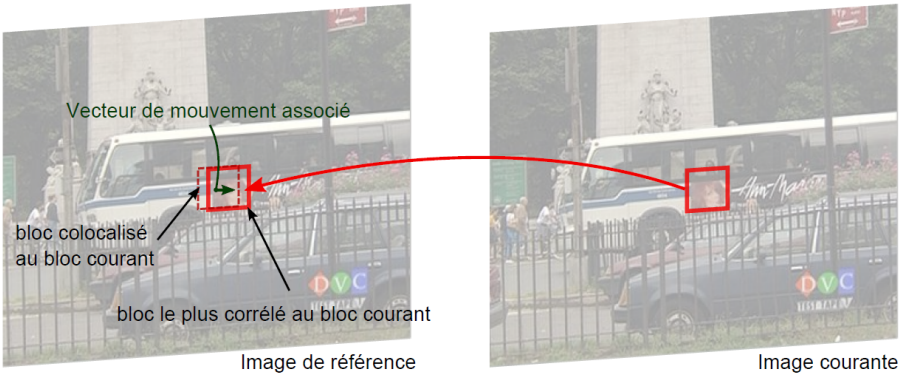

FIGURE 1.9 – Block matching *: recherche de corrélation dans une image de référence.*

Classiquement, on associe à chaque bloc un vecteur de mouvement. Ce vecteur est ensuite encodé et transmis au décodeur afin de réaliser la même compensation de mouvement

côté décodeur. Par conséquent, l'estimation de mouvement est réalisée uniquement à l'encodeur. Comme pour la prédiction spatiale, il est également nécessaire de transmettre l'erreur de prédiction. Comme en Intra, plusieurs partitionnements sont également définis en Inter avec l'addition de partitions rectangulaires qui permettent de segmenter efficacement le mouvement. Les tailles de blocs disponibles sont : 16x16, 16x8, 8x16 et 8x8 (Figure 1.10) pouvant lui-même être partitionné en 8x4, 4x8 et 4x4 (Figure (1.11)). L'intérêt d'une telle décomposition est de permettre des traitements à différentes échelles et par conséquent de s'adapter au contenu local de l'image. Le standard utilise de plus une représentation sous-pixellique du mouvement jusqu'au 1/4 de pixel. Enfin, un mode particulier est défini en Inter, il s'agit du mode Skip qui s'applique sur les blocs de taille 16x16 et a la particularité de ne nécessiter aucune information additionnelle (ni vecteur de mouvement, ni résidu de texture).

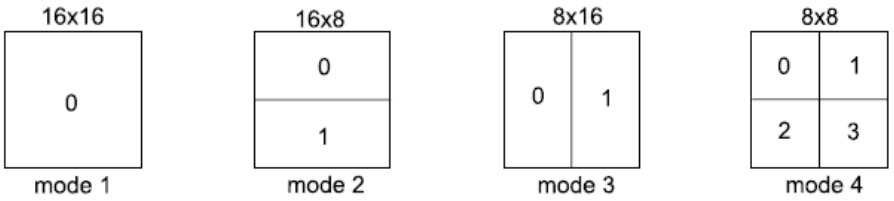

Figure 1.10 – *Partitions macrobloc :*16*x*16*,* 16*x*8*,* 8*x*16 *et* 8*x*8*.*

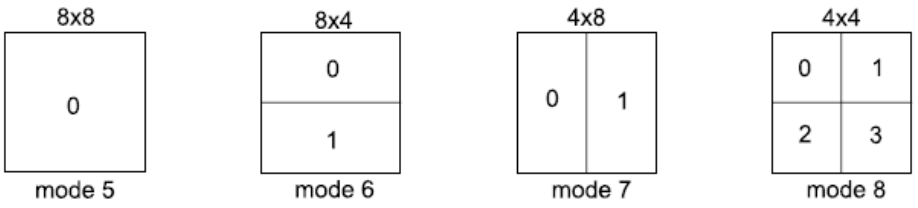

Figure 1.11 – *Partitions sous-macrobloc :* 8*x*8*,* 8*x*4*,* 4*x*8 *et* 4*x*4*.*

Par ailleurs, pour des images P et B, il est possible d'utiliser plusieurs macroblocs provenant de plusieurs images de référence, afin de prédire le macrobloc courant. Pour les images P, le processus reste monodirectionnel puisqu'un seul prédicteur est choisi, mais il peut maintenant être choisi dans plusieurs images de référence. Pour les images B, deux macroblocs sont utilisés, et il est possible de changer d'image de référence pendant le processus de prédiction de l'image en cours. La figure 1.12 montre ainsi que deux macroblocs d'une même image peuvent être prédits à partir d'images de référence différentes : la flexibilité de la prédiction est ainsi accrue.

Après avoir détaillé les deux grandes catégories de prédictions, la section suivante présente comment le codeur peut choisir le mode de codage approprié au MB en cours.

1.2.1.7 Choix des modes de codage

Dans H.264, le procédé de sélection des modes de codage à l'encodeur n'est pas normatif, les informations relatives aux modes choisis étant envoyées au décodeur. Le codeur est par conséquent libre dans le choix des modes de codage. Cependant, ce choix est conditionné

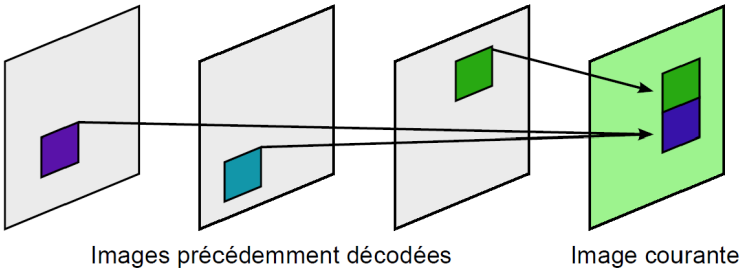

Figure 1.12 – *Références multiples pour la prédiction inter d'un bloc.*

par le type d'image traitée : seule la prédiction intra sera par exemple active dans le cas d'une image I. Un exemple de schéma de sélection des modes de codage est fourni dans la figure 1.13 dans le cas d'un macrobloc de type P ou B. Dans cet exemple, tous les modes de prédiction intra et inter sont dans un premier temps testés. Ensuite, on procède à la sélection du meilleur mode intra puis celle du meilleur mode inter. Parmi ces meilleurs modes, on vient ensuite choisir la meilleure prédiction pour le macrobloc.

La sélection des modes se fait en général par le biais de la minimisation d'une fonction de coût. Cette fonction peut être définie selon plusieurs critères. Deux types de critère sont en général utilisés : un critère basé sur une mesure de distorsion et un autre plus complexe basé sur une optimisation débit/distorsion, également appelé RDO (Rate/Distorsion Optimization). Dans la plupart des applications, le procédé de sélection vise plutôt à optimiser un critère de type RDO afin d'obtenir le meilleur compromis entre qualité de reconstruction et coût de codage selon des contraintes d'utilisations données. Dans cette section, nous nous focaliserons sur la description du second critère.

Afin de choisir le meilleur mode possible parmi les modes et sous-modes disponibles en intra et inter, on cherche soit à minimiser la distorsion sous la contrainte d'un certain débit ou minimiser le débit pour une distorsion donnée, comme détaillé dans [SW98, OR98]. Cette opération consiste à trouver pour un signal S et un mode de prédiction I :

$$\min_I D(S.I) \text{ avec } R(S,I) \leq R_c \tag{1.5}$$

ou encore :

$$\min_I R(S.I) \text{ avec } D(S,I) \leq D_c \tag{1.6}$$

avec $D(S.I)$ le critère de distorsion, $R(S.I)$ le débit et R_c et D_c les contraintes respectivement sur le débit et la distorsion. En pratique, cette opération revient à minimiser une fonction Lagrangienne donnée par :

$$J_\lambda(S,I) = D(S,I) + \lambda.R(S,I), \tag{1.7}$$

avec $\lambda \geq 0$ un paramètre permettant de pondérer la fonction entre ces deux critères. Le processus a donc deux options pour faire ce choix en tenant compte des deux paramètres :

- Méthode *a posteriori* : dans ce cas, les étapes suivantes de l'encodage d'un macrobloc *i.e.* la transformée spatiale, la quantification et le codage entropique, sont nécessaires pour connaître le débit. De plus, le calcul de la distorsion par rapport au signal source, nécessite d'appliquer les opérations inverses de décodage. De manière optimale, cette étape est appliquée pour tous les modes afin de trouver celui qui minimise la distorsion tout en atteignant la contrainte de débit simultanément. Cependant, ces

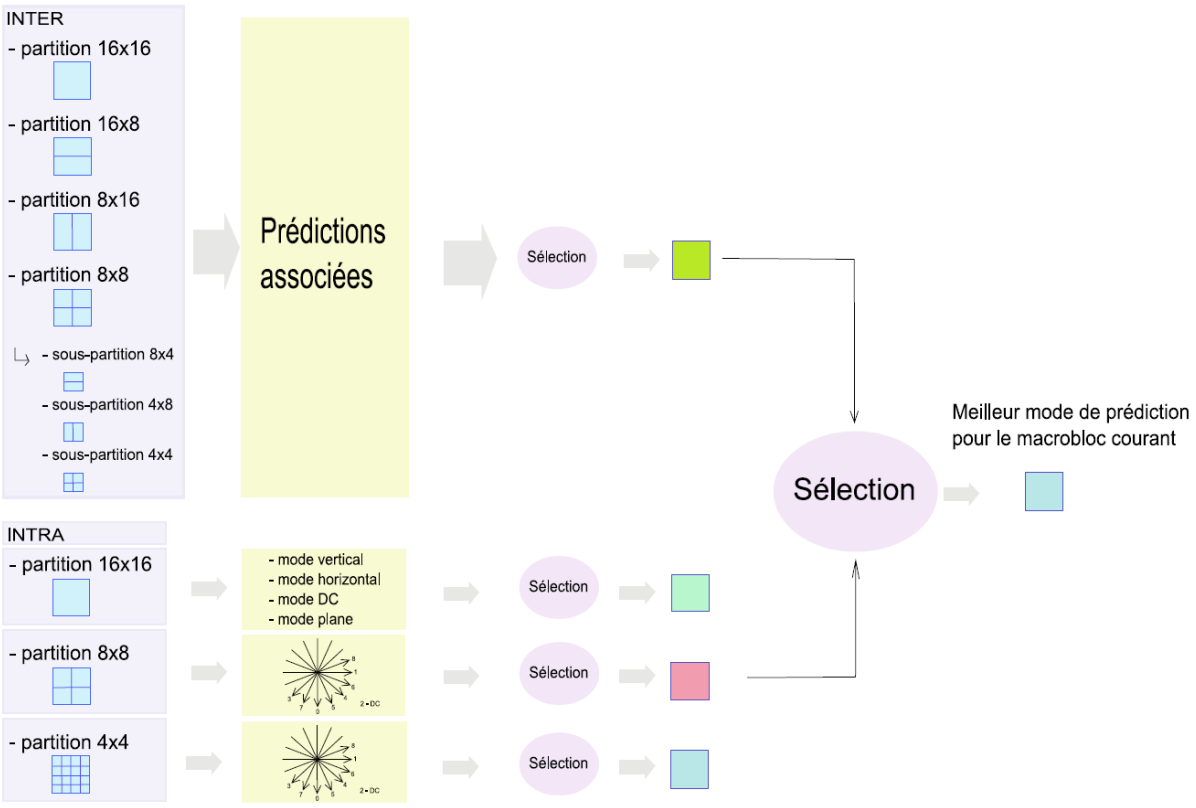

Figure 1.13 – *Exemple de sélection des modes de prédiction.*

opérations étant coûteuses, notamment les transformées, les codeurs utilisant la RDO *a posteriori* ne testent que les modes les plus probables.

- Méthode *a priori* : dans ce cas, un modèle est utilisé pour estimer le débit nécessaire et la distorsion correspondante. Ce modèle est empirique et permet d'approcher les performances des tests a posteriori. Cette méthode sous-optimale réduit cependant nettement la complexité du choix de mode de codage.

1.2.2 Vers le futur standard HEVC

1.2.2.1 Préambule

Le standard HEVC pour High Efficiency Video Coding [HEV11, HEV10b], est une nouvelle norme de codage vidéo développée par l'équipe Joint Collaborative Team on Video Coding (JCT-VC) [HEV10a] qui a été finalisée en janvier 2013. Ce nouveau schéma de codage (cf. figure 1.14) est conceptuellement proche de son prédécesseur H.264. Cependant, des outils existants dans le codeur H.264 ont été revisités. Ce nouveau standard peut apporter un taux de compression de l'ordre de 50% à qualité subjective équivalente par rapport à H.264 avec ses paramètres les plus performants. HEVC a été conçu afin de permettre l'accès aux contenus vidéo HD et ultra HD (7680×4320 pixels).

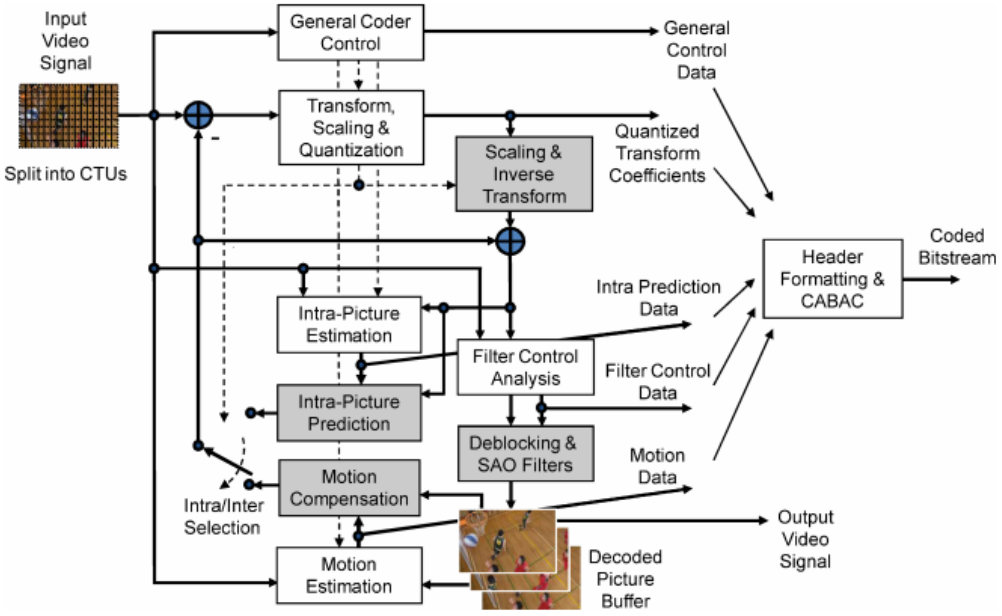

Figure 1.14 – *Schéma d'un encodeur vidéo HEVC (avec le décodeur représenté par les éléments en gris clair).*

1.2.2.2 Structure des données

HEVC a introduit un nouveau concept de partitionnement d'image basé sur une structure en arbre (en anglais *quadtree*) permettant des partitionnements en tailles de blocs variables qui s'adaptent au contenu dans l'image. Quatre structures de données sont principalement définies pour l'encodage d'une séquence [SOH12] :

- la structure **Coding Tree Unit (CTU)** : elle représente l'élément fondamental de codage. Une séquence est en effet divisée en une grille régulière de CTU (cf. figure 1.15.a). Sa taille est définie à l'encodeur et varie de 16x16 à 64x64 pixels selon l'application. Une grande taille de CTU permet ainsi de fournir une meilleure compression notamment pour des séquences de haute résolution pour lesquelles la corrélation entre pixels est plus importante mais ceci au prix d'une complexité de calcul plus élevée.
- la structure **Coding Unit (CU)** : Chaque CTU de l'image est ensuite divisée en Coding Unit (CU). La taille d'une CU varie de 8x8 à 64x64 pixels. La structure d'une CU est celle d'un quadtree. En effet, une CU peut être ensuite divisée en 4 CU de manière récursive. On peut souligner également que la décision entre le codage intra et inter est effectuée au niveau CU. La décomposition d'une CTU en un ensemble de CU est présentée dans la figure (cf. figure 1.15.b).
- la structure **Prediction Unit (PU)** : elle contient les informations relatives au processus de prédiction (direction de prédiction en intra et information de mouvement en inter). La taille maximale d'une PU est la taille de la CU dans laquelle elle est incorporée. Une PU peut également être décomposée en plusieurs PU. Cependant, la structure d'une PU diffère selon le type de prédiction (intra ou inter). En prédiction intra, une PU peut ne pas être partitionnée et correspondre à la CU ou bien être décomposée en quatre partitions de tailles égales. Tandis qu'en prédiction inter, une

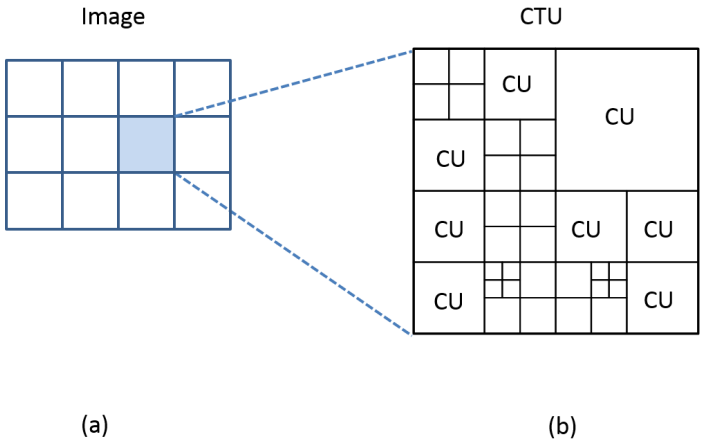

FIGURE 1.15 – *(a) décomposition d'une image en CTU ; (b) exemple d'une structure CTU.*

PU peut ne pas être partitionnée ou bien être partitionnée en 2 PU (6 choix possibles) ou être divisée en 4 PU. La figure 1.16 résume les différents partitionnements possibles d'une PU en intra et inter pour un niveau de décomposition de l'arbre.

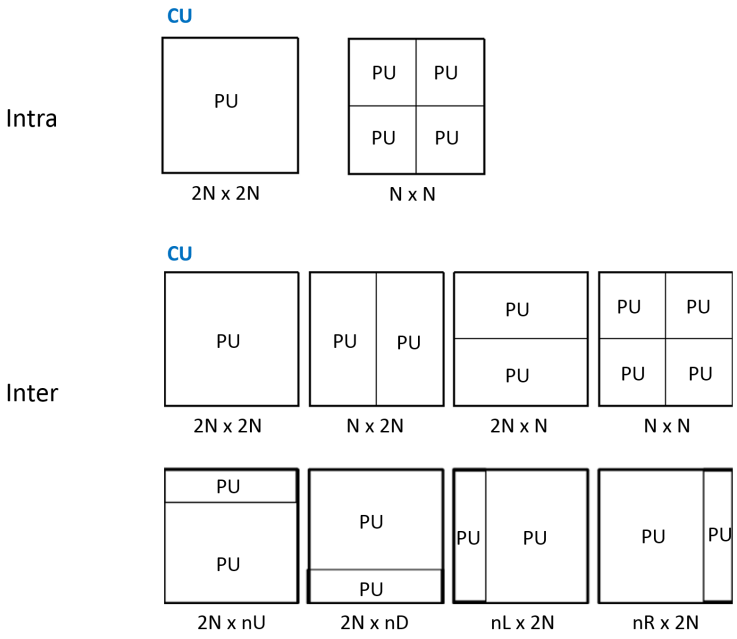

FIGURE 1.16 – *Exemples d'une structure PU pour une prédiction intra et une prédiction inter.*

- la structure **Transform Unit (TU)** : elle représente la structure de base pour le processus de transformation et de quantification. Elle est indépendante de la partition en PU. Une TU peut en effet s'étendre sur plusieurs PU ou ne correspondre qu'à une partie d'une PU. Une CU peut également contenir une ou plusieurs TU ordonnées selon un schéma en quadtree pouvant aller jusqu'à trois niveaux de décomposition. En intra, une TU est de forme carrée et peut avoir une taille allant de 4x4 à 32x32 pixels pour la composante de luminance. En inter, des formes rectangulaires peuvent également être utilisées (4x16, 16x4, 8x32 et 32x8 pixels) pour éviter qu'une transformée se trouve sur une frontière de bloc induisant ainsi des coefficients de hautes

fréquences.

Les sections suivantes décrivent plus en détails les améliorations apportées par HEVC au niveau de la prédiction Intra et la prédiction Inter.

1.2.2.3 Prédiction intra-images

L'une des améliorations apportées dans le nouveau standard concerne l'augmentation du nombre de prédicteurs intra mis en concurrence. En effet, 35 modes de prédiction sont désormais possibles pour la composante de luminance incluant 33 directions spatiales, un prédicteur DC (moyenne des pixels de référence) et un nouveau prédicteur nommé *Planar*. Néanmoins, le nombre de directions intra utilisées diffère selon les tailles des PU. En effet, l'encodeur teste 4 directions pour les PU de taille 64x64 pixels, 16 pour les PU 4x4 et 33 pour les autres tailles de PU (cf. figure 1.17). L'augmentation du nombre de modes permet ainsi d'exploiter plus efficacement les redondances pixelliques avec le voisinage local au détriment d'une signalisation plus accrue.

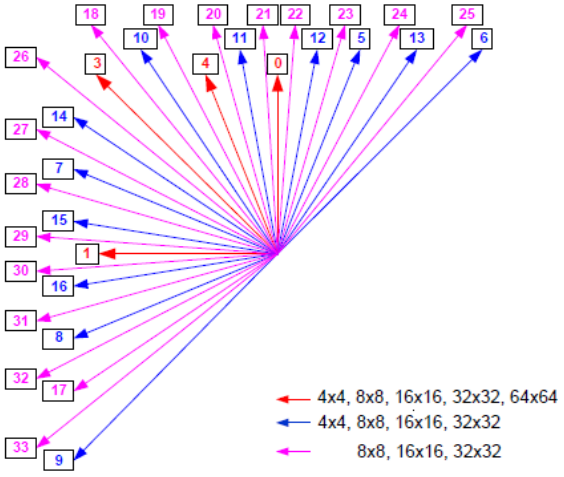

Figure 1.17 – *Modes directionnels en HEVC.*

Comme évoqué ci-dessus, une autre amélioration du processus de prédiction intra repose sur la définition d'un nouveau prédicteur *Planar* allant au-delà d'une propagation monodirectionnelle. En effet, ce nouveau mode de codage réalise une prédiction multidirectionnelle et permet ainsi d'être plus efficace pour prédire des zones texturées. Son fonctionnement est illustré dans la figure 1.18.

Notons que le nombre de modes est moindre (6 modes) pour la composante de chrominance. Par ailleurs, par rapport à H.264, HEVC effectue en plus un lissage sur les pixels de référence utilisés pour la prédiction permettant ainsi d'adoucir les contours et de réduire les artefacts. Pour finir, des efforts particuliers ont été menés lors du processus de transformation dans lequel deux transformées sont mises en concurrence (DCT et DST) et également au niveau du codage des modes de prédiction.

1.2.2.4 Prédiction inter-images

Comme dans H.264, la prédiction inter est faite en utilisant des références multiples dans une liste. Cependant, certains aspects ont été améliorés tel que le codage du vecteur de

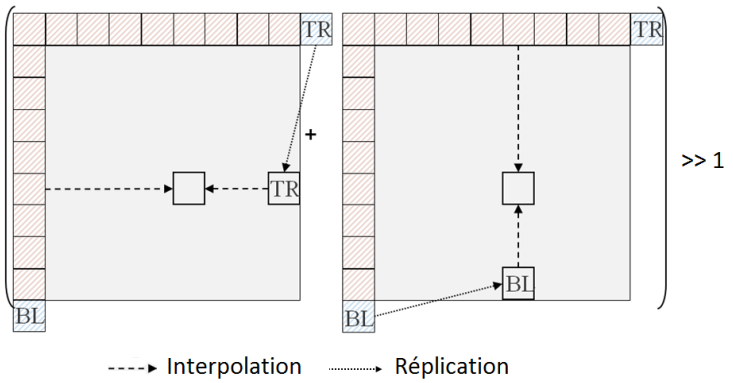

Figure 1.18 – *Fonctionnement d'un prédicteur Planar.*

mouvement. En effet, un nouveau mode nommé *Merge* a été introduit dans lequel plusieurs prédicteurs de vecteurs de mouvement sont mis en concurrence issus des PU spatialement voisins et du PU colocalisé temporellement. Ce mode s'apparente aux modes *Skip/Direct* de H.264 avec néanmoins deux différences : on transmet explicitement le prédicteur de mouvement retenu ainsi que l'image de référence utilisée. De plus, des filtres d'interpolation améliorés ont été définis pour l'estimation de mouvement sous-pixellique.

1.2.2.5 Autres outils

Les améliorations apportées dans ce nouveau standard concernent également les filtres intégrés dans la boucle de décodage ainsi que la parallélisation des traitements. En effet, après avoir utilisé le *déblocking filter* pour réduire les effets de bloc, un filtre supplémentaire est appliqué : le *Sample Adaptive Offset* (SAO). Ce filtre consiste à ajouter un offset en fonction de la valeur du pixel et des caractéristiques de la région (contour, région lisse ...). Par ailleurs, afin de permettre un codage plus efficace pour des applications qui exploitent le parallélisme, deux principaux outils ont été introduits : le *Wavefront parallel processing* (WPP) et les *Tiles*. Le premier outil permet de traiter les CTU en parallèle ligne par ligne avec un décalage du nombre de CTU nécessaire pour conserver les dépendances. En effet, l'idée est de s'assurer que les CTU voisins sur lesquels on s'appuie pour réaliser la prédiction intra ou la prédiction inter soient disponibles. Quant au second outil, il traite le problème différemment en divisant une image en plusieurs régions rectangulaires appelées *Tiles*, permettant ainsi de faciliter la parallélisation des traitements.

1.3 De nouveaux outils

1.3.1 Méthodes de prédiction intra basées Template Matching

D'autres méthodes de prédiction intra ont également été introduites dans la littérature permettant d'aller au-delà d'une prédiction spatiale mono-directionnelle. En effet, une nouvelle recherche de prédicteurs en mode intra appelée *template matching* est proposée dans [TBS06]. Ce mode de prédiction, illustré par la figure 1.19, cherche à faire correspondre une zone *template* constituée des pixels causaux issus d'un voisinage direct au bloc à prédire avec les pixels contenus dans une zone de même forme dans la partie causale de l'image. Le bloc candidat associé à la zone template la plus corrélée à celle du bloc courant sert ensuite de prédicteur pour coder une image en intra.

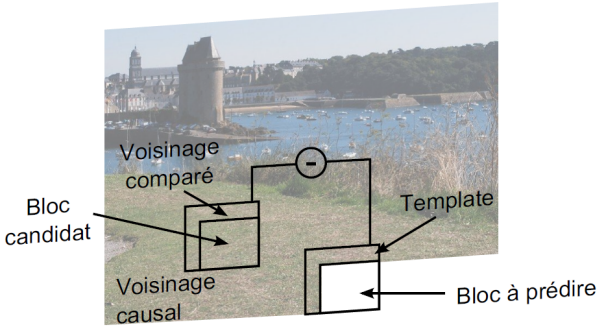

FIGURE 1.19 – Template matching *: Prédiction d'un bloc en fonction de son voisinage* template.

Le template matching a plus tard été amélioré dans [TBS07] où une moyenne de plusieurs templates est effectuée. De plus, une autre variante de la méthode de prédiction basée template matching a été présentée dans [ZYE+08] où une compensation d'illumination adaptative sur le template est réalisée.

1.3.2 Compression d'images fixes basée sur les techniques de synthèse

D'autres approches en rupture avec les schémas de codage conventionnels ont également été étudiées dans la littérature. Parmi ces approches, on peut énoncer les algorithmes de compression d'images basés sur des techniques de synthèse [DHIC03, DH04]. Ce type de schéma de codage consiste à analyser à l'encodeur le contenu d'une image afin de supprimer des régions considérées comme synthétisables, les autres régions étant codées classiquement en utilisant un codeur standard de l'état de l'art. Ensuite, les régions supprimées sont reconstruites lors du décodage à partir d'outils de synthèses de texture exploitant les données déjà décodées. D'autres travaux basés sur le même principe ont également été introduits dans [NNMS+03, NNMB+03] qui définissent un schéma de codage comportant des outils d'analyse de texture côté encodeur et de synthèse côté décodeur. L'analyseur de texture utilisé dans cette approche vient dans un premier temps diviser l'image selon une structure en quadtree par le biais d'une mesure de similarité puis ensuite fusionner les blocs considérés comme homogènes. Un algorithme de compression basé inpainting plus avancé a également été exploré dans les travaux de thèse de F. Racapé dans [RLF+11b, RLF+11a] où l'analyseur de texture comprend des étapes de segmentation et de caractérisation de textures. L'étape de caractérisation permet de détecter si une région texturée peut être synthétisable selon la taille des motifs trouvée. La particularité du schéma proposé dans [RLF+11a] repose aussi sur l'utilisation de deux méthodes de synthèse complémentaires selon le type de texture : une méthode de synthèse basée pixel de type [WL00] et une autre approche de synthèse basée patch inspirée de [KSE+03].

1.4 Conclusion

Dans ce chapitre, nous avons introduit les principes de base de la compression d'images fixes et vidéos. Ensuite, nous avons présenté le standard de compression H.264/AVC exploitant à la fois les redondances spatiales et temporelles du signal. Un intérêt particulier a également été accordé à son successeur HEVC qui à l'aide de nombreuses améliorations représente le codeur le plus performant à l'heure actuelle en termes d'efficacité de codage. En effet, il permet de réduire le taux de compression jusqu'à 50% par rapport à H.264. Ce

nouveau standard va vraisemblablement s'imposer dans les années à venir comme l'outil incontournable de transmission et de stockage de contenu vidéo. Pour finir, d'autres approches en rupture avec les schémas conventionnels telles que des techniques de codage basées synthèse ont brièvement été introduites. Ce type de schéma inspirera probablement les prochaines générations de standards dans les applications qui nécessiteront encore davantage de réduction de débit. Nous allons maintenant aborder dans le chapitre suivant un autre point fondamental en rapport avec les schémas de codage développés au cours de ces travaux de thèse : les épitomes.

CHAPITRE 2

Techniques de construction d'image "résumé"

Cette section vise à fournir un aperçu général des différentes techniques existantes pour construire une représentation épitomique d'une image. Après avoir défini la notion d'épitome, nous introduirons les différentes approches permettant d'extraire un épitome d'une image. La multitude des algorithmes d'extraction d'épitome démontre leur intérêt pour diverses applications allant du denoising à la synthèse de texture. La problématique de cette thèse étant la compression d'images, nous nous focaliserons sur les récentes techniques de prédiction intra existantes basées sur l'utilisation d'une image résumé de type épitome. Nous verrons que la manière dont un épitome est généré est importante. En effet, elle aura un impact sur les performances de codage utilisant cette forme de dictionnaire de patches. Nous montrerons également leurs limites ouvrant ainsi des perspectives de recherche pertinentes au regard de cette thèse.

2.1 Résumé d'image au sens de l'épitome

2.1.1 Analyse épitomique basée EM (Expectation Maximization) [Jogic, Cheung]

La théorie de l'analyse épitomique basée EM a été initialement introduite dans la littérature pour modéliser des images en vue de leur segmentation [JFK03] ou clusterisation [KWR06]. L'épitome d'une image est considéré ici comme une image "résumé" contenant les caractéristiques essentielles de texture et de forme de l'image pour sa reconstruction. Plus précisément, l'épitome est défini comme un modèle de probabilité basé patch qui est appris sur un nombre élevé de patches d'entrainement issus de l'image d'entrée. Ce modèle d'épitome a également été étendu au domaine de la vidéo afin de traiter des problèmes de super-résolution, d'interpolation vidéo, d'inpainting ou encore de denoising [CFJ05]. Cette représentation épitomique a aussi été exploitée pour des applications telles que la reconnaissance de localisation [NKCW08, NKCW09] et la reconnaissance faciale [CYL+10].

Nous allons maintenant décrire plus en détail le principe de fonctionnement de l'algorithme EM de façon générale et ensuite voir comment cet outil est appliqué pour l'extraction d'un épitome.

2.1.1.1 Principe de l'EM

L'EM [DLR77] est un algorithme d'apprentissage semi-supervisé qui permet d'estimer le maximum de vraisemblance des paramètres θ d'une loi de probabilité $p_\theta(z)$ associée à des données observées z_o et à des données manquantes z_m (également appelées des données cachées). Plus précisément, cette technique se base sur l'utilisation de la fonction de vraisemblance de l'ensemble complet des données $z = (z_o, z_m)$ pour déterminer des estimateurs en dépit du fait que les données observées sont incomplètes. L'expression de la fonction de vraisemblance de l'ensemble complet des données z est donnée par :

$$L(\theta|z) = L(\theta|z_o, z_m) = p(z_o, z_m|\theta) \tag{2.1}$$

Etant donné que les données cachées ne sont pas connues, l'algorithme EM se base sur l'hypothèse que les données manquantes z_m sont tirées à partir de l'estimation des paramètres du modèle de l'itération précédente. L'EM vient donc calculer l'espérance de la fonction de vraisemblance (ou du log-vraisemblance) des données complètes z par rapport aux données manquantes z_m en prenant en compte les données observés z_o et les paramètres courants du modèle θ_t :

$$Q(\theta, \theta_t) = E_{z_m}\left[\log(p_\theta(z_o, z_m))|\theta_t, z_o\right] \tag{2.2}$$

Cette espérance peut également s'écrire sous la forme :

$$Q(\theta, \theta_t) = \int_{z_c} \log(p_\theta(z_o, z_c)).p_{\theta_t}(z_c|z_o)\,\mathrm{d}z_c \tag{2.3}$$

Cette fonction étant déterministe, l'EM maximise cette espérance pour estimer de nouveau les paramètres du modèle θ_{t+1}.

L'EM opère donc par itérations successives de deux étapes :

- **l'étape d'estimation** (E-step) : qui consiste à évaluer l'espérance du log-vraisemblance $Q(\theta, \theta_t)$ à l'aide des données observées et des paramètres disponibles. Cette étape permet ainsi d'estimer les données manquantes.
- **l'étape de maximisation** (M-step) : qui consiste à estimer la maximisation de vraisemblance des paramètres en maximisant l'espérance déterminée à l'étape E :

$$\theta_{t+1} = \arg\max_\theta Q(\theta, \theta_t) \tag{2.4}$$

Les itérations successives de ces deux étapes font augmenter la vraisemblance des paramètres et permet à l'algorithme de converger vers un maximum local de la fonction $Q(\theta, \theta_t)$. On peut noter cependant que l'initialisation des paramètres du modèle pour amorcer l'algorithme a un impact direct sur le comportement de l'EM. Le processus de fonctionnement de l'EM est résumé dans l'algorithme 2.1. Notons que le critère d'arrêt de cet algorithme est en général soit la stationnarité de la vraisemblance soit un nombre maximal d'itérations fixé initialement ou bien les deux critères à la fois.

Nous allons maintenant présenter comment un modèle épitomique est appris à partir d'un algorithme de type EM dans [JFK03].

2.1.1.2 Description du modèle épitomique de type EM

Nous commençons par présenter les annotations utiles pour la bonne compréhension de l'algorithme d'extraction d'épitome à partir de l'outil EM. On suppose que l'image en entrée

Algorithme 2.1 : Principe de fonctionnement de l'algorithme EM.

Initialisation : $t = 0$, paramètres θ_0
while $Q(\theta_{t+1}, \theta_t) - Q(\theta_t, \theta_{t-1}) < \epsilon$ **do**
  E-step : calcul de $Q(\theta_t, \theta_{t-1})$
  M-step : $\theta^* = \arg\max_\theta Q(\theta, \theta_t)$
  $\theta_{t+1} = \theta^*$
  $t := t + 1$
end
Estimée du maximum de vraisemblance
$\theta_{\mathbf{mv}} = \theta_{\mathbf{t+1}}$

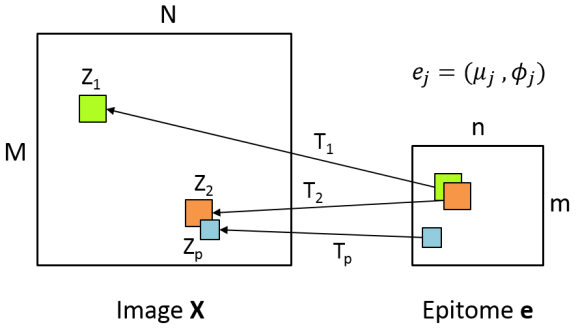

Figure 2.1 – *Modèle épitomique basé EM*

X de taille NxM pixels est représentée par un ensemble de patches $\{Z_k\}_{k=1}^P$ prédéfini. Chaque patch contient les pixels issus d'un sous-ensemble de coordonnées de l'image S_k. On suppose ici que les patches sont de formes carrées de taille KxK. Notons cependant que la forme des patches peut être arbitraire. Pour chaque patch Z_k, le modèle utilise un mapping caché T_k qui permet de faire la mise en correspondance entre les coordonnées $i \in S_k$ du patch et ceux de l'épitome $j \in E_k$. L'épitome e est de taille nxm pixels dont chaque élément contient deux paramètres : la moyenne et la variance d'un pixel de l'épitome. L'ensemble de ces variables sont illustrées dans la figure 2.1. On suppose que les patches sont générés indépendamment d'où l'expression de la distribution jointe suivante :

$$p\left(\left\{Z_k, T_k{}_{k=1}^P\right\}, e\right) = p(e) \prod_{k=1}^{P} p(T_k) p(Z_k | T_k, e) \tag{2.5}$$

Soit l'épitome $e = (\mu, \phi)$ et le mapping T_k, le patch Z_k est généré en copiant les pixels appropriés issus de la carte de la moyenne de l'épitome auxquels est ajouté un bruit gaussien dont le niveau est fourni par la carte de variance. La vraisemblance est donc modélisée ici comme étant normalement distribuée selon les pixels $z_{i,k}$ avec une moyenne $\mu_{T_k(i)}$ et une variance $\phi_{T_k(i)}$:

$$p(Z_k | T_k, e) = \prod_{i \in S_k} \mathcal{N}(z_{i,k}; \mu_{T_k(i)}; \phi_{T_k(i)}) \tag{2.6}$$

Notons que, dans [JFK03], les estimations des données cachées $\{T_k\}_{k=1}^P$ et de l'ensemble des paramètres de l'épitome e sont plutôt calculées à partir d'une variante de l'algorithme EM dans lequel le log-vraisemblance des données $\{Z_k\}_{k=1}^P$ est majoré par une fonction d'énergie libre de Helmholtz négative (voir [JF03, JGJS98] pour plus de détails). Cette

borne inférieure du log-vraisemblance rend plus facile la maximisation par rapport à la distribution a posteriori du mapping et aux paramètres de l'épitome.

Dans l'étape d'estimation (E-step), le mapping est décrit par la distribution :

$$q(T_k) \propto \prod_{i \in S_k} \mathcal{N}(z_{i,k}; \mu_{T_k(i)}; \phi_{T_k(i)}) \tag{2.7}$$

Dans l'étape de maximisation (M-step), les équations de mise à jour des paramètres de l'épitome sont fournies par :

$$\widehat{\mu_j} = \frac{\sum_k \sum_{i \in S_k} \sum_{T_k, T_k(i)=j} q(T_k) z_{i,k}}{\sum_k \sum_{i \in S_k} \sum_{T_k, T_k(i)=j} q(T_k)} \tag{2.8}$$

$$\widehat{\phi_j} = \frac{\sum_k \sum_{i \in S_k} \sum_{T_k, T_k(i)=j} q(T_k)(z_{i,k} - \widehat{\mu_j})^2}{\sum_k \sum_{i \in S_k} \sum_{T_k, T_k(i)=j} q(T_k)} \tag{2.9}$$

La figure 2.2 fournit un exemple d'épitome obtenu avec la méthode d'apprentissage de type EM.

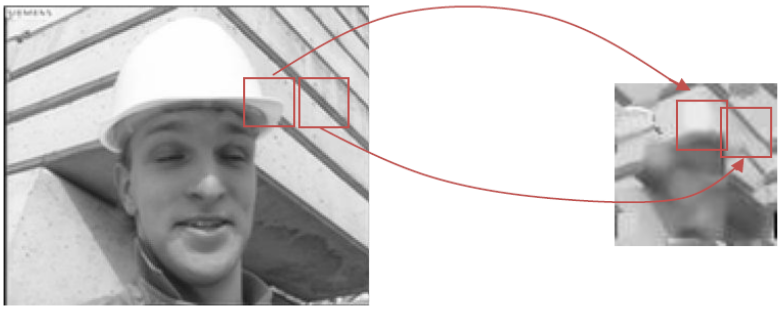

Figure 2.2 – *Exemple d'un épitome d'une image construit selon un algorithme de type EM (à droite)*

Certaines études ont été menées pour utiliser ce type d'épitome dans le cadre de la compression d'image. Nous reviendrons dessus dans la section 2.2.1 où nous présenterons les différentes méthodes de prédiction intra se basant sur ce type d'approche ainsi que leurs limitations.

2.1.2 Résumé d'image basé sur une mesure de similarité bidirectionnelle [Irani]

Une autre approche de construction d'une image "résumé" est explorée par M. Irani et al. dans [SCSI08] introduisant une métrique de similarité dite-bidirectionnelle qui contrôle à la fois les propriétés de "complétude" et de "cohérence" dans l'image résumé par rapport à l'image source. Une image "résumé" est dite "complète" si l'ensemble des patches issus de l'image source sont représentés par un patch de l'image "résumé" (cf. figure 2.3.a). Une image "résumé" est "cohérente" si l'ensemble des patches de l'image "résumé" se trouve dans l'image source (cf. figure 2.3.b). Ainsi, l'image "résumé" ne contient pas d'artefacts visuels non observés dans l'image source. Afin d'obtenir cette représentation condensée, l'algorithme tente ici de conserver les informations pertinentes de l'image source en éliminant progressivement les informations redondantes.

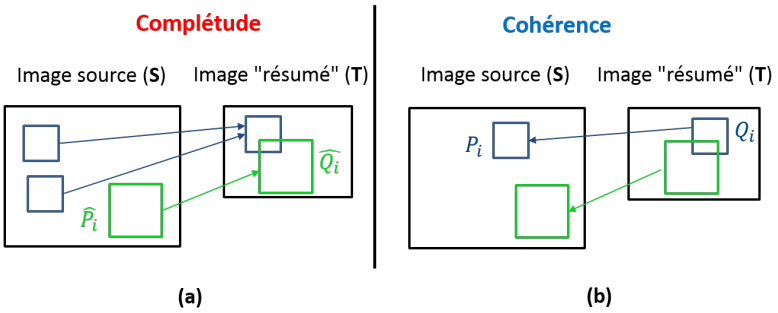

Figure 2.3 – *Similarité bidirectionnelle ((a) complétude + (b) cohérence)*

Soient S et T désignant respectivement l'image source et l'image "résumé". Soient P et Q les patches se trouvant respectivement dans S et T, et soient N_S et N_T le nombre de patches dans S et T respectivement. La mesure de dissimilarité proposée dans [SCSI08] est définie comme suit :

$$d(S,T) = \frac{1}{N_S}\sum_{P\subset S}\min_{Q\subset T} D(P,Q) + \frac{1}{N_T}\sum_{Q\subset T}\min_{P\subset S} D(Q,P) \tag{2.10}$$

Nous verrons plus tard que cette même mesure de dissimilarité peut également être utilisée pour quantifier l'efficacité d'une image "résumé" et comparer plusieurs types de résumés.

Afin de construire le résumé de l'image source, l'algorithme cherche à optimiser la mesure de dissimilarité de l'équation (2.10) en fonction de l'image "résumé" :

$$T_{out} = \arg\min_{T} d(S,T) \tag{2.11}$$

Pour résoudre ce problème d'optimisation, cet algorithme utilise un procédé de mise à jour itératif qui diminue progressivement la taille de l'image au fur et à mesure jusqu'à atteindre la taille de l'image "résumé" voulue. Le redimensionnement graduel de l'image va en effet permettre d'assurer une bonne convergence du processus de raffinement de l'image "résumé" vers un résultat parfait et cohérent.

La figure 2.4 montre les résultats obtenus pour l'ensemble des images résumé (intermédiaires + final) lorsqu'un redimensionnement graduel est appliqué sur l'image ainsi que la mesure de dissimilarité associée. Pour les premières images intermédiaires $T_1, ..., T_6$, on constate qu'il y'a chaque fois une légère perte d'information visuelle lorsque l'on réduit progressivement la taille de l'image, suivie d'une légère augmentation dans la mesure de dissimilarité $d(S,T)$. Cependant, à partir de l'image intermédiaire $T6$, on note une perte significative de l'information lorsque l'on passe d'une image à une autre ce qui est également confirmé par une forte augmentation de $d(S,T)$. En effet, lorsque la taille de l'image est trop petite, il n'y a pas assez d'espace pour représenter l'ensemble des patches de l'image source. Néanmoins, on peut remarquer que la cohérence est toujours préservée.

La figure 2.5 compare trois imagettes obtenues avec différents outils : une image croppée arbitrairement, un épitome obtenu via l'algorithme EM [JFK03] et un résumé par similarité bidirectionnelle (Eq. (2.10)). On constate que l'approche développée par M. Irani et al. permet d'obtenir une représentation condensée de meilleure qualité, sans perte d'information significative (i.e. résumé complet) et également sans ajout d'artefacts visuels non désirés (i.e. résumé cohérent).

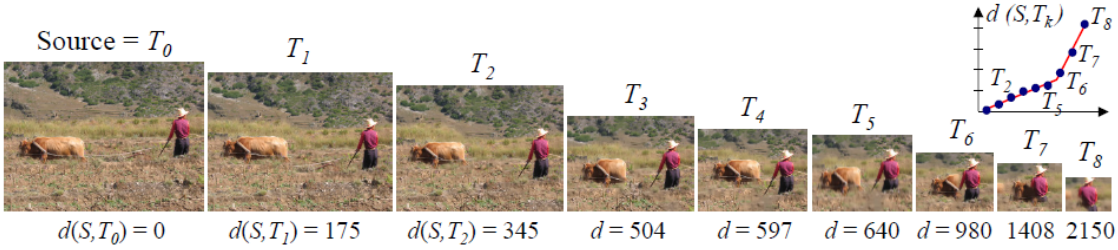

FIGURE 2.4 – *Résultats obtenus pour les différents images résumé (intermédiaires+final) lorsqu'un traitement de redimensionnement graduel est effectué ainsi que la mesure de similarité bidirectionnelle associée*

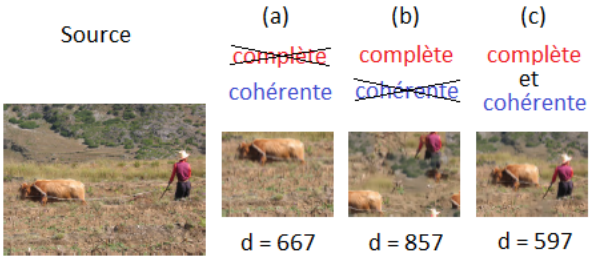

FIGURE 2.5 – *Comparaison de trois imagettes obtenues avec différents outils : (a) une image croppée arbitrairement, (b) un épitome obtenu via l'algorithme EM et (c) un résumé par similarité bidirectionnelle.*

Cette approche a également été étudiée pour des applications telles que le repurposing (le recadrage automatique), le montage d'image, l'édition, la synthèse d'images (dans laquelle l'image est agrandie au lieu d'être résumé), l'inpainting (où des parties manquantes dans l'image sont remplies) ou encore l'auto-cropping.

2.1.3 Synthèse inverse de texture

La synthèse inverse [WHZ+08] est un processus permettant de construire une petite image de texture à partir d'une grande en vue de la recréer à partir des outils de synthèse. Ce concentré d'image doit donc posséder les motifs et les variations essentielles contenues dans la grande texture de départ, afin de permettre à la synthèse, ensuite, de recréer une texture visuellement proche de l'originale. La figure 2.6 illustre l'avantage de cette technique pour une texture comportant un motif de premier plan devant un fond globalement variant du noir à l'orange. A travers cette illustration, on peut constater que le découpage d'un patch dans l'image source suivi d'une synthèse ne permet pas de recréer la variation de couleur du fond alors que la synthèse inverse recrée une image visuellement proche de la source. Bien que la reconstruction de cet exemple nécessite l'utilisation d'un champ d'orientation guidant les variations globales, cette technique est intéressante pour construire un *patch* de taille réduite possédant un maximum d'informations à propos des primitives et de leurs variations. Cependant, ce type d'outil peut être inefficace dans un contexte de compression lorsqu'il s'agit de traiter des variations difficilement modélisables.

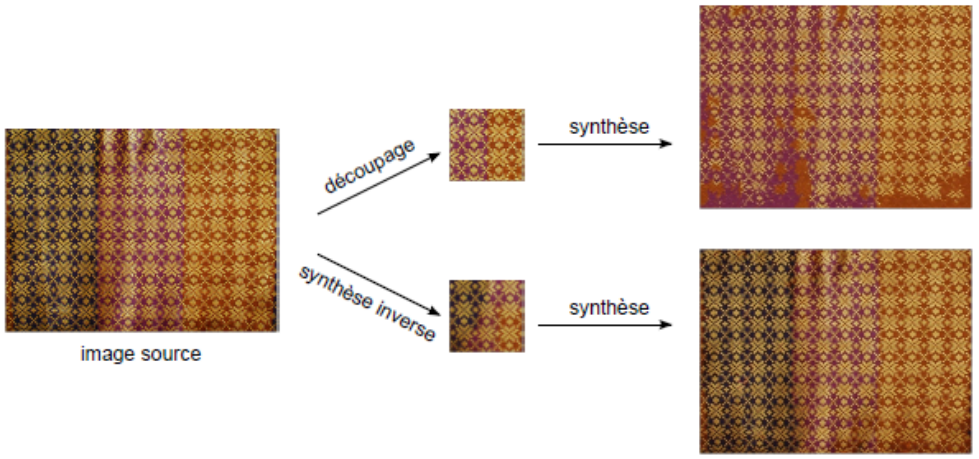

Figure 2.6 – *Synthèse inverse de texture, résultats avec découpage d'un* patch *en haut et avec la création du* patch *par synthèse inverse en bas.*

2.1.4 Modélisation parcimonieuse basée ISD

Une autre représentation épitomique appelée *image signature dictionary* (ISD) a également été introduite par [EA08] pour des applications de denoising. Cette représentation combine la notion d'épitome utilisée dans [JFK03] avec des algorithmes d'apprentissage de dictionnaire [EA06, OF97] afin de représenter un patch d'une image comme une combinaison linéaire sparse de plusieurs patches issus de l'épitome. Ce type de représentation a également prouvé son utilité dans d'autres domaines telles que la synthèse de texture [Pey09]. Par ailleurs, une extension de l'approche ISD a été explorée dans [BMBP11] pour apprendre une série d'épitomes de différentes formes. Nous proposons de présenter plus en détails le principe de l'ISD. Une autre représentation épitomique appelée *image signature dictionary* (ISD) a également été introduite par [EA08] pour des applications de denoising. Cette représentation combine la notion d'épitome utilisée dans [JFK03] avec des algorithmes d'apprentissage de dictionnaire [EA06, OF97] afin de représenter un patch d'une image comme une combinaison linéaire sparse de plusieurs patches issus de l'épitome. Ce type de représentation a également prouvé son utilité dans d'autres domaines telle que la synthèse de texture [Pey09]. Par ailleurs, une extension de l'approche ISD a été explorée dans [BMBP11] pour apprendre une série d'épitomes de différentes formes. Nous proposons de présenter plus en détails le principe de l'ISD.

2.1.4.1 Introduction sur les représentations parcimonieuses

Trouver la représentation parcimonieuse d'un signal, consiste à obtenir la meilleure représentation du signal, *i.e.* la plus parcimonieuse parmi celles ayant la même erreur de reconstruction. Cette représentation est constituée d'un faible nombre d'atomes qui ont été choisis parmi un vaste ensemble de signaux élémentaires, le dictionnaire [1]. Générer une représentation parcimonieuse ne correspond pas, à proprement parlé, à une projection sur une base.

Notons y le vecteur représentatif du signal source de dimension m et $A \in \mathbb{R}^{m \times n}$ le

1. Le dictionnaire est aussi appelé base *redondante* car le nombre d'atomes est supérieur à la dimension de l'espace formé par le signal et sont donc linéairement dépendants. Ainsi, utiliser le mot *base* est un abus de langage par définition d'une base dans un espace linéaire.

dictionnaire, avec $m << n$. Il existe une infinité de solutions quant au choix du vecteur x de dimension n, tel que :

$$y = Ax$$

Le but des représentations parcimonieuses est de trouver parmi l'ensemble des solutions possibles, celles qui sont parcimonieuses i.e. celles pour lesquelles le vecteur x a seulement un faible nombre de coefficients non nuls. Le problème à résoudre est donc le suivant :

$$\mathcal{P}_0 : \qquad \min_x \|x\|_0 \quad \text{sous} \quad y = Ax \tag{2.12}$$

où $\|.\|_0$ désigne le nombre de coefficients non nuls.

La minimisation exacte de la norme l_0 est un problème NP-complet qui n'a pas de solutions pratiques. Ceci signifie qu'obtenir la solution reviendrait à résoudre un problème combinatoire, *i.e.* tester toutes les combinaisons d'atomes possibles, méthode bien trop complexe dans un espace de grande dimension et en général, il faudrait de toute façon, m composantes, ce qui est trop dans ce contexte.

On considère donc plutôt le problème suivant : il s'agit de rechercher la solution la plus parcimonieuse tout en tolérant une erreur admissible de reconstruction, notée ρ :

$$\widetilde{\mathcal{P}}_0 : \qquad \min_x \|x\|_0 \quad \text{sous} \quad \|y - Ax\|_2 \leq \rho$$

où $\|.\|_2$ est la norme euclidienne l_2 : $\|x\|_2 = \sum_{i=1}^{N} \sqrt{|x_i|^2}$.

La solution de ce problème d'optimisation est le vecteur parcimonieux x, dont le nombre de coefficients non-nuls est minimal. Le vecteur x conduit à une approximation dont l'erreur de reconstruction est inférieure ou égale à ρ. Ce problème est cependant toujours trop difficile à résoudre et on ne le considère jamais.

2.1.4.2 Principe du Matching Pursuit (MP)

Le *Matching Pursuit* (MP) nommé ainsi par Mallat et Zhang en 1993 [MZ93] est un algorithme glouton, connu pour être une alternative à la recherche de la solution optimale. L'algorithme permet de trouver une approximation sous-optimale.

Le principe est donc de sélectionner pas à pas, les atomes les plus corrélés avec le signal.

- *La première approximation du signal* s'obtient en calculant le projeté orthogonal du signal observé y sur l'atome qui lui est le plus corrélé, notons le a_{γ_1}. Comme a_{γ_1} est normé, la pondération qui minimise la norme du résidu est $\langle y, a_{\gamma_1} \rangle$ et on a donc :

$$R_y^{(1)} = y - \langle y, a_{\gamma_1} \rangle a_{\gamma_1}$$

- *Lors des itérations suivantes*, on recherche l'atome le plus corrélé au résidu courant et on projette ce résidu sur l'atome sélectionné, noté a_{γ_k}. L'estimation obtenue par l'ajout de ce nouvel atome est ajoutée à l'estimation courante du résidu, pour former l'estimation courante du signal y. D'un point de vue plus formel, notons $R_y^{(k)}$ la valeur du résidu au pas k et $\widehat{y}^{(k)}$ l'approximation courante de y qui s'écrit alors :

$$\widehat{y}^{(k)} = \widehat{y}^{(k-1)} + \left\langle R_y^{(k-1)}, a_{\gamma_k} \right\rangle a_{\gamma_k}$$

Voici décrites ci-après, les étapes du *Matching Pursuit*.

Algorithme 2.2 : Algorithme du *Matching Pursuit*

Entrées : Le signal source y, le dictionnaire A et le seuil ρ

Initialisation : $\widehat{y}^{(0)} = 0, R_y^{(0)} = y$ et $k = 1$

while $\left\| R_y^{(k-1)} \right\|^2 \geq \rho$ **do**

- Recherche de l'atome le plus corrélé : $\gamma_k = \arg\max_{\gamma} \left| \left\langle R_y^{(k-1)}, a_\gamma \right\rangle \right|$
- Calcul du nouveau coefficient : $x_{\gamma_k} = \left\langle R_y^{(k-1)}, a_{\gamma_k} \right\rangle$
- Mise à jour des données :
  1. de l'estimée : $\widehat{y}^{(k)} = \widehat{y}^{(k-1)} + x_{\gamma_k} a_{\gamma_k}$
  2. du résidu : $R_y^{(k)} = y - \widehat{y}^{(k)} = R_y^{(k-1)} - x_{\gamma_k} a_{\gamma_k}$
  3. du vecteur parcimonieux : $x[\gamma_k] \leftarrow x_{\gamma_k}$

end

Dans la procédure algorithmique que nous venons de décrire, rien n'exclut le fait qu'un atome puisse être sélectionné plusieurs fois. Le nombre d'itérations k n'est donc pas nécessairement égal au nombre total d'atomes a_γ sélectionnés pour la représentation x. L'intérêt majeur du *Matching Pursuit* est sa grande simplicité d'implémentation et sa rapidité d'exécution. Contrairement à d'autres algorithmes que nous présentons par la suite, il ne nécessite aucune inversion matricielle. Cependant, cette simplicité a un inconvénient : il peut falloir un grand nombre d'itérations pour converger vers une solution, fait d'autant plus vrai que dans certains cas un atome déjà sélectionné peut à nouveau l'être.

2.1.4.3 Principe du Orthogonal Matching Pursuit (OMP)

L'*Orthogonal Matching Pursuit* (OMP) se base sur le même principe que le MP : sélectionner pas à pas les atomes les plus corrélés au signal pour tendre vers une approximation de la solution au problème $\widetilde{\mathcal{P}}_0$. La différence réside dans la mise à jour des coefficients.

L'objectif de l'OMP est de pallier la faille du MP, qui, comme nous venons de le voir, n'empêche pas la sélection multiple d'un même atome.

Afin d'y parvenir, l'OMP recalcule à chaque pas de l'algorithme la valeur de l'estimée $\widehat{y}$. Le MP se base sur la mise à jour du résidu : à chaque itération, on retire au résidu une contribution du *seul* nouvel atome sélectionné. Pour l'OMP, c'est différent : on évalue la reconstruction courante $\widehat{y}^{(k)}$ à chaque fois, ce qui signifie que l'on recalcule tous les coefficients jusqu'alors sélectionnés. L'entrée d'un nouvel atome dans la décomposition modifie l'espace engendré. Il est donc plus judicieux de projeter le signal y, non plus sur le seul nouvel atome, mais sur l'ensemble formé des atomes passés auquel s'ajoute le nouvel atome sélectionné.

Cette mise à jour de tous les coefficients pour chaque nouvel atome choisi pourrait se faire avec la procédure d'orthogonalisation de Gram-Schmidt. On génère donc avec cet algorithme une base orthogonale dont la dimension croît à chaque nouvelle sélection d'un atome. L'ensemble des atomes retenus formant une famille libre de l'espace, on exclut de fait la sélection d'un atome ayant déjà été ajouté à cette base.

Reprenons les mêmes notations que précédemment. Nous introduisons deux nouvelles variables :

- A_k, la matrice contenant l'ensemble des atomes sélectionnés à l'itération k : $A_k = [a_{\gamma_1} \dots a_{\gamma_k}]$.
- x_k, le vecteur contenant uniquement les coefficients non-nuls qui ont été calculés jusqu'au pas courant : $x_k = [x_{\gamma_1} \dots x_{\gamma_k}]$.

Algorithme 2.3 : Algorithme de l'*Orthogonal Matching Pursuit*

Entrées : Le signal source y, le dictionnaire A et le seuil ρ

Initialisation : $\widehat{y}^{(0)} = 0, R_y^{(0)} = y$, $A_0 = [.]$ et $k = 1$

while $\left\| R_y^{(k-1)} \right\|^2 \geq \rho$ **do**

- Recherche de l'atome le plus corrélé : $\gamma_k = \arg\max_{\gamma} \left| \left\langle R_y^{(k-1)}, a_\gamma \right\rangle \right|$
- Ajout du nouvel atome au sous-dictionnaire : $A_k = [A_{k-1} \quad a_{\gamma_k}]$
- Calcul des coefficients : $x_k = A_k^+ y \quad$ avec $\quad A_k^+ = \left(A_k^T A_k\right)^{-1} A_k^T$
- Mise à jour des données :
  1. de l'estimée : $\widehat{y}^{(k)} = A_k.x_k$
  2. du résidu : $R_k = y - \widehat{y}^{(k)}$

end

La projection du signal sur l'ensemble formé des atomes sélectionnés jusqu'au pas courant se fait via le calcul de la pseudo-inverse A_k^+. Comme nous l'avons précédemment remarqué, cette méthodologie empêche la sélection d'un atome qui aurait déjà été sélectionné. Cela présente l'avantage de surpasser le *Matching Pursuit* en terme de nombre d'itérations mais nécessite une inversion matricielle à chaque itération. Il existe toutefois des simplifications à l'algorithme présenté qui consistent à déterminer la valeur de A_k^+ de manière récursive, *i.e.* en utilisant la valeur de A_{k-1}^+.

2.1.4.4 Image Signature Dictionary (ISD)

Un ISD est une "imagette" qui s'apparente à un dictionnaire issu d'un apprentissage à partir de vecteurs d'entraînement provenant de l'image à modéliser. Chaque patch de l'ISD correspond à un atome du dictionnaire (à raison d'un patch par pixel). Un ISD peut être vu comme une généralisation de l'approche probabiliste des épitomes de Jojic et Cheung. En effet, dans [JFK03], une simple copie des patches de l'épitome est réalisée pour reconstruire une image. Tandis qu'avec les ISD, une combinaison linéaire de patches de l'ISD est utilisée.

Algorithme de base :

Le processus d'apprentissage de l'ISD est réalisé par itérations successives de deux étapes :

- L'étape de codage parcimonieux : pour chaque patch d'entraînement issu de l'image d'entrée, on détermine une représentation parcimonieuse des atomes x_i de l'ISD courant $\mathbf{D_k}$ à partir de l'OMP :

$$\arg\min_{x_i} \|y_i - \mathbf{D_k} x_i\|_2^2 \text{ avec } \|x\|_0 \leq L \tag{2.13}$$

- L'étape de mise à jour de l'ISD : la mise à jour de l'ISD est effectuée en considérant l'ensemble des vecteurs de représentation parcimonieux $X = x_1, ..., x_N$ déterminés à l'étape précédente permettant de minimiser l'erreur de représentation des patches d'entraînements $Y = y_1, ..., y_N$. L'ISD optimal est obtenu par :

$$\mathbf{D_{k+1}} = X^{-1}Y \tag{2.14}$$

Multiscale ISD :

Par rapport aux dictionnaires classiques, la structure de l'ISD présente une certaine flexibilité. En effet, l'apprentissage de l'ISD peut être réalisé en considérant des patches de tailles variables. De plus, le nombre d'atomes utilisés pour la reconstruction peut changer selon la taille des patches utilisés.

Stochastic Gradient :

Une approche *Stochastic Gradient* (SG) peut également être utilisée pour l'apprentissage des ISD. Cette méthode d'apprentissage consiste à mettre à jour l'ISD après le calcul de chaque représentation parcimonieuse x_i d' un patch d'entrainement y_i. L'apprentissage via cette approche permet non seulement de converger plus vite que la méthode classique utilisée pour obtenir des ISD mais également de reconstruire une image de meilleure qualité. La figure 2.7 compare deux exemples d'ISD obtenus réciproquement avec l'algorithme classique et la méthode SG.

Figure 2.7 – *Comparaison de deux exemples d'ISD de taille* 35x35 *pixels : (a) image d'entrée de taille* 130x130 *pixels, (b) ISD basé sur l'algorithme de base, (c) ISD basé sur la méthode SG*

2.1.5 Factorisation d'image [Hoppe]

Une autre approche a également été développée dans [WWOH08] pour construire une image résumé de l'image source. Cette approche repose sur la factorisation d'éléments répétitifs jugés redondants dans l'image tout en cherchant à contrôler la perte d'information introduite dans l'image. Seuls quelques patches représentatifs de l'image source sont stockés pour constituer l'épitome de celle-ci ainsi que les transformations permettant sa reconstruction. Ce procédé décompose ainsi une image source en un épitome condensé et

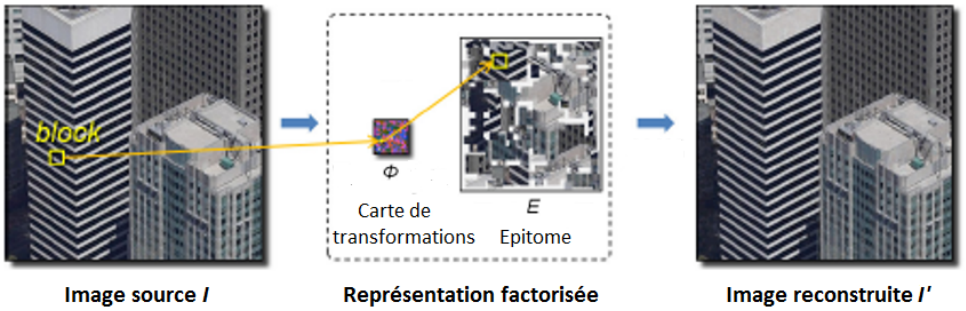

FIGURE 2.8 – *Factorisation d'une image en un épitome condensé et une carte de transformations.*

une carte de transformations qui permettra par la suite de restituer une approximation de l'image de départ (cf. figure 2.8).

L'approche décrite dans [WWOH08] se démarque des méthodes présentées précédemment [EA08, JFK03]. En effet, tout d'abord, l'image source est divisée ici en une grille régulière de blocs qui ne se chevauchent pas. La reconstruction des blocs de l'image peut ainsi être réalisée avec un accès aléatoire étant donné qu'il n'y a pas de dépendances avec les blocs voisins. De plus, l'épitome résultant de la factorisation ne comporte que des éléments de texture issus de l'image source. L'épitome est par conséquent parfaitement cohérent vis à vis de l'image de départ (i.e. il n'y a pas d'artéfacts dans l'épitome non observés dans l'image source).

Afin de construire une représentation factorisée, l'algorithme spécifie en premier lieu l'erreur de reconstruction maximale ϵ qui doit être satisfaite pour l'ensemble des blocs de l'image source. L'algorithme tente ensuite de déterminer la représentation la plus concise qui respecte le seuil d'erreur fixé ϵ :

$$\min_{E,\phi} |E| \text{ tel que } \forall B \in I,\ e(B) \leq \epsilon \tag{2.15}$$

Cette minimisation est approximée en utilisant une procédure de construction "glouton" dans lequel l'épitome est agrandi pas à pas par un élément de texture "optimum" issu de l'image source. La stratégie adoptée ici consiste à maximiser le nombre de blocs supplémentaires reconstruits dans l'image lors de l'agrandissement de l'épitome tout en minimisant l'accroissement de ce dernier.

L'algorithme décrit dans [WWOH08] contient quatre étapes principales :

1. **Recherche des auto-similarités dans l'image source** : cette étape détermine, pour chaque bloc dans l'image, l'ensemble des patches (régions transformées) dans l'image présentant un contenu similaire avec une erreur de reconstruction inférieure au seuil fixé ϵ. Elle permet de construire des listes d'appariements pour chaque bloc de l'image. Chaque transformation inclut des déformations affines et des variations de couleur (color scaling). La recherche des similarités est réalisée ici à partir de la fonction de tracking KLT (Kanade-Lucas-Tomasi) qui optimise l'alignement affine entre deux fenêtres. En effet, la KLT détecte des mises en correspondance de patches en cherchant à minimiser l'erreur quadratique suivante par rapport aux coefficients de transformation :

$$\min_{A,d,\alpha,\beta} \sum_{x} \left(P_1(Ax+d) - (\alpha P_2(x) + \beta)\right)^2 \tag{2.16}$$

où A correspond à la matrice de déformation, d représente les paramètres de translation et (α, β) constituent respectivement les paramètres de contraste et de luminosité. Le problème de minimisation est résolu ici en utilisant la méthode itérative Newton Raphson. Du fait que la KLT est utilisée uniquement pour faire correspondre des patches présentant de petites variations, les paramètres de déformation doivent bien être initialisés avant d'amorcer l'algorithme.

2. **Création des épitomes chart** : une fois que les listes d'appariements sont créées, l'étape suivante consiste à extraire dans l'image les éléments de texture les plus représentatifs de l'image, ce que l'on appelle ici les épitomes chart, par le biais d'un critère de sélection prédéfini. Un épitome chart est décrit ici comme une représentation condensée d'une zone spécifique dans l'image, l'ensemble des épitomes charts constituant l'intégralité de l'épitome. Soit I^E le sous-ensemble de l'image source I déjà représenté par l'épitome actuel E (qui regroupe l'ensemble des épitomes chart précédemment créés ainsi que celui en cours de construction). L'épitome chart en cours de construction est agrandi ici pas à pas par des éléments de texture $\Delta E \subset I$ qui maximisent la fonction suivante :

$$\max_{\Delta E} |I^{E+\Delta E} \backslash I^E| - |\Delta E| \tag{2.17}$$

   En effet, pour étendre un épitome chart, l'algorithme favorise les éléments de texture ΔE qui permettent de reconstruire le plus grand nombre de blocs dans l'image toujours non représentés par l'épitome actuel. Soulignant que les candidats ΔE utilisés pour étendre les épitomes charts sont directement fournis à partir des listes d'appariements dressées lors de l'étape 1. de l'algorithme.

3. **Optimisation de la carte de transformations** : cette étape consiste à rechercher pour chaque bloc de l'image le meilleur appariement dans l'épitome (une fois que ce dernier est totalement construit) puis à remettre ensuite à jour la carte de transformations. En effet, au fur et à mesure que l'épitome est agrandi, l'ajout d'un nouvel élément de texture dans l'épitome peut permettre une meilleure reconstruction à des blocs de l'image qui ont été précédemment reconstruits.

4. **Assemblage des épitomes chart en un épitome atlas E** : cette étape consiste à déplacer les épitomes charts de façon à ce que l'épitome puisse être représenté sur une image plus petite. Plus précisément, pour déterminer le meilleur emplacement des épitomes charts, les auteurs utilisent un algorithme heuristique de [FDK02] qui vient trier les épitomes chart par taille décroissante et dans lequel des transformations (telle qu'une rotation ou un effet de miroir) peuvent être appliquées pour déplacer les épitomes chart. La figure 2.9 fournit un exemple de construction d'un épitome atlas.

Afin d'accélérer l'algorithme d'extraction d'épitome, une approche hiérarchique a également été explorée. Cette approche consiste à partitionner l'image source en plusieurs sous-images et ensuite à construire un épitome pour chaque sous-image. Les différents épitomes obtenus sont ensuite fusionnés pour constituer un seul épitome. Une étape d'optimisation (étape 3. de l'algorithme de base) est ensuite appliquée pour raffiner la reconstruction de l'image source et voire même enlever certains contenus de l'épitome non utilisés. Ce même type d'approche a également été étendu au domaine temporel afin de factoriser une série d'images. De plus, une représentation progressive de l'image a été étudiée afin d'effectuer de la scalabilité de type SNR (Signal-to-Noise Ratio). En effet, une première approximation de l'image est fournie à partir d'un premier épitome et de la carte de transformation associée. Ensuite, une deuxième approximation plus précise de l'image est obtenue par le

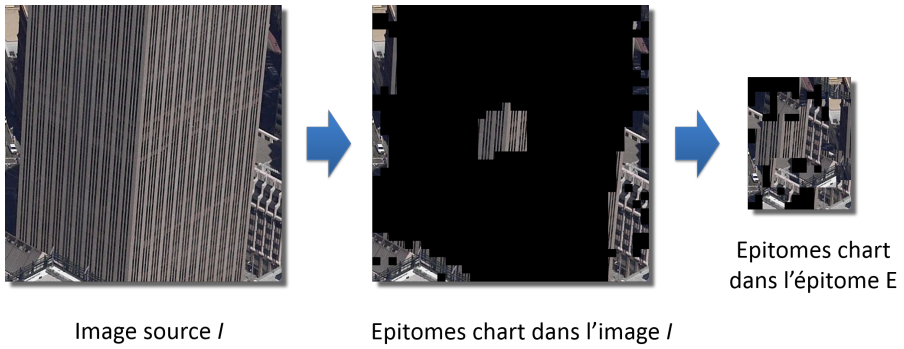

Figure 2.9 – *Exemple de construction d'un épitome atlas.*

biais d'un second épitome et de sa carte de transformation, le deuxième épitome étant un sur-ensemble du premier. Les auteurs ont également exploré cette représentation factorisée (épitome+map) dans un contexte de compression dans lequel l'épitome et la carte de transformation sont encodés respectivement avec un codeur JPEG2000 et une méthode de codage *lossless* de type PNG. Les auteurs ont montré que cette nouvelle approche permettait d'obtenir de meilleures performances de codage comparé à une image encodée uniquement en JPEG2000.

2.2 Techniques de compression d'images fixes exploitant la redondance globale de l'image

Comme nous l'avons décrit dans la section 1.2.1, la prédiction intra de la norme H.264/AVC permet de réduire les redondances entre le bloc courant et son voisinage local (i.e. pixels voisins précédemment décodés localisés en haut et à gauche du bloc courant). Cependant, dans certaines images, le bloc courant peut présenter de fortes similitudes avec un patch distant qui peut se trouver aussi bien sur la zone de l'image reconstruite que sur la partie non reconstruite. Afin de gérer ce type de redondance, des méthodes de prédiction intra basées sur l'utilisation d'une image épitome ont été introduites dans la littérature [WHW09, WHWH10] pour encoder l'image courante. Ces techniques permettent, à travers l'utilisation de l'épitome, de réduire les corrélations du bloc courant avec l'ensemble de l'image (ie. la partie reconstruite et non reconstruite). Les méthodes de prédiction existantes se basent particulièrement sur l'utilisation d'un épitome construit selon l'algorithme EM [JFK03]. En effet, comme nous l'avons montré précédemment, l'algorithme EM est un des outils qui peut faire converger une image épitome vers une image contenant des caractéristiques de texture similaires à celles de l'image source.

2.2.1 Codage intra utilisant l'épitome basé EM [Jogic]

2.2.1.1 Prédiction par Template Matching [Q. Wang]

Le schéma de compression décrite dans [WHW09] est basé sur l'utilisation d'une image épitome choisie judicieusement pour encoder des images d'une même capture vidéo. Cette approche est dédiée en particulier à des applications telles que l'édition de vidéo, la transmission de vidéo ou encore la compression vidéo de films haute définition où les images intra apparaissent à de très hautes fréquences. Le schéma de codage présenté dans ces travaux est résumé dans la figure 2.10.

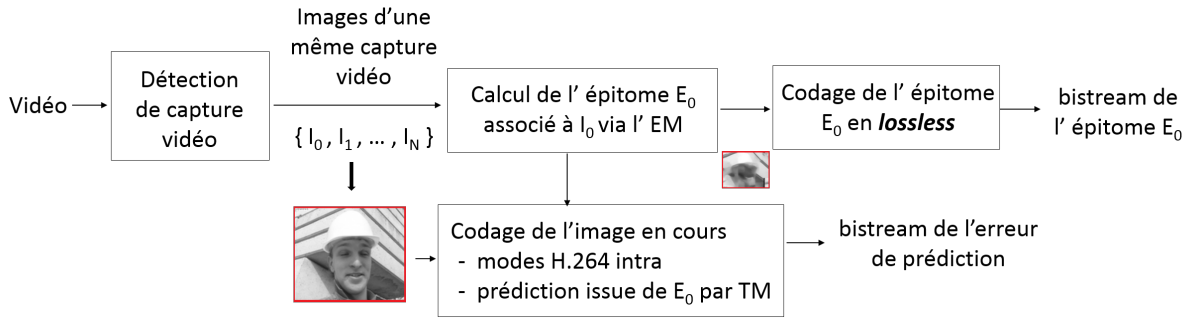

FIGURE 2.10 – *Schéma de codage présenté dans [WHW09].*

La nouvelle méthode de prédiction intra introduite ici consiste à générer une prédiction issue de l'image épitome par le biais du template matching. Ce mode de prédiction, illustré dans la figure 2.11, identifie dans l'image épitome, le template le plus proche du template courant au sens de la distance euclidienne. Le bloc associé au template le plus corrélé au template courant est ensuite utilisé pour prédire le bloc courant.

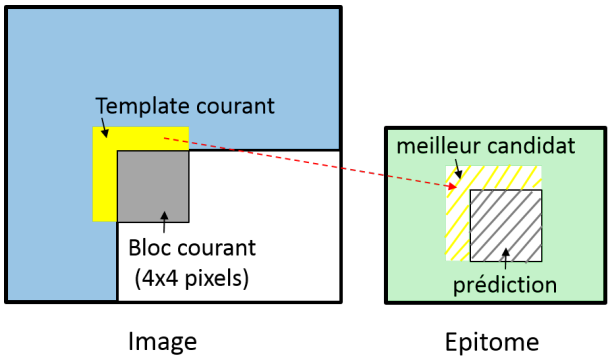

FIGURE 2.11 – *Prédiction d'un bloc 4x4 générée à partir de l'image épitome par template matching.*

Algorithme EM par Wang et al.

Comme dans la méthode proposée par N. Jojic [JFK03], l'image d'origine est divisée en un réseau de patches de tailles différentes dans lequel les patches peuvent également se chevaucher. Soient X et Z désignant respectivement l'image d'origine et l'image épitome. Soit X^+ représentant le réseau de patches issus de l'image d'origine. Au lieu de maximiser le log-vraisemblance d'une loi de densité de probabilité comme dans la théorie de l'EM, les auteurs choisissent plutôt d'effectuer la minimisation d'une fonction d'énergie définie par :

$$\varphi(X,Z) = \frac{1}{X^+} \sum_{p \in X^+} |x_p - z_{p^*}|^2 \tag{2.18}$$

où x_p représente un patch de X^+ et z_{p^*} représente la meilleure correspondance de x_p dans l'image épitome. Le problème de minimisation de la fonction d'énergie est également résolue en alternant successivement les étapes *E-step* et *M-step* jusqu'à convergence de l'algorithme EM.

L'algorithme EM procède ainsi de la manière suivante :

1. Initialisation aléatoire de l'épitome Z
2. Pour chaque itération :
   - *E-step* : recherche du meilleur représentant z_{p^*} pour chaque patch x_p de X^+.
   - *M-step* : mise à jour du contenu de l'épitome en minimisant la fonction d'énergie $\varphi(X, Z)$ selon Z :

$$Z_{best} = \arg\min_{Z} \varphi(X, Z) \tag{2.19}$$

   En d'autres termes, à chaque pixel de l'image épitome $Z(i, j)$, on affecte la moyenne des valeurs des pixels de l'ensemble des patches x_p issus de X_+ pour lesquels la correspondance du pixel se trouve à la position (i, j) de l'épitome Z
3. Répétition de l'étape 2 jusqu'à ce que les conditions d'arrêt soient atteintes. Par exemple, lorsque la fonction d'énergie $\varphi(X, Z)$ est inférieure à un seuil ϵ fixé par l'utilisateur.

Les auteurs ont également contribué à l'amélioration de l'outil d'extraction de l'épitome de Jogic en réalisant l'apprentissage de l'épitome de manière "hiérarchique" à partir d'une décomposition pyramidale de l'image source. Le but de cette manoeuvre est de mieux capter les motifs présents dans l'image source. L'épitome correspondant à l'image de résolution la plus faible est initialisé ici à partir d'une image sous-échantillonée de l'image source. L'épitome obtenu pour l'image de faible résolution via l'algorithme EM est ensuite interpolé pour initialiser l'image épitome correspondant à l'image de plus haute résolution. La figure 2.12 compare un épitome EM basé multi-résolution (b) [WHW09] avec un épitome EM basé simple résolution (a) [JFK03].

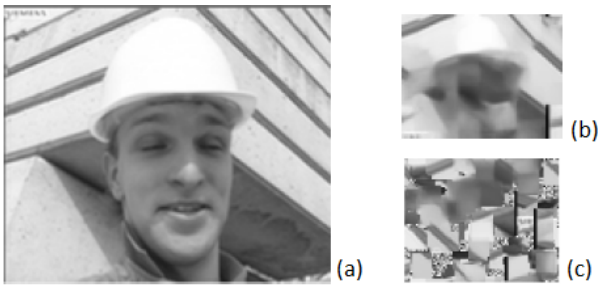

Figure 2.12 – *(a) Image source : Foreman, (b) Epitome EM de [WHW09] : approche multi-résolutions, (c) Epitome EM de [JFK03] : approche simple résolution*

Description du schéma de codage dans [WHW09]

Cet algorithme segmente dans un premier temps la séquence vidéo en une série de capture vidéo à partir d'une méthode de détection de changement de scène [GKS00]. Les auteurs prennent parti d'utiliser ici une seule image épitome pour l'ensemble des images de la capture vidéo. En effet, puisque les images au sein d'une même capture vidéo présentent de grandes similarités, seule la première image est utilisée pour générer une image épitome, le même épitome sert ensuite d'image épitome pour le reste des images de la capture. Cette approche permet non seulement d'éliminer les corrélations du bloc courant avec l'ensemble de l'image mais également de réduire en partie les redondances temporelles entre les images adjacentes de la capture vidéo tout en évitant des dépendances de codage entre les images. La méthode de prédiction intra basée épitome par template matching est intégrée ici directement dans le codeur H.264/AVC en substituant ce nouveau mode

de prédiction au mode de prédiction DC du codeur. Pour chaque capture vidéo, l'image épitome est ensuite encodée à partir d'une méthode de codage sans pertes et transmise au décodeur. Plus précisément, les auteurs encodent l'image épitome en utilisant le mode IPCM du codeur H.264/AVC.

Malgré l'amélioration de l'outil d'extraction de l'épitome, certains éléments de texture dans l'épitome demeurent incohérents vis-à-vis de l'image source. En effet, lors du processus de calcul de l'épitome, des artefacts assimilés à des effets de flou apparaissent. Ceci est principalement dû à la stratégie de moyennage effectué dans l'étape-M.

2.2.1.2 Prédiction par Block Matching [Q. Wang]

L'approche développée par Q. Wang et al dans [WHWH10] propose une nouvelle méthode de prédiction basée épitome pour améliorer non seulement le codage d'une image intra mais également pour améliorer la qualité des rafraîchissements intra. En effet, lors de la transmission d'un signal vidéo, afin d'éviter que des erreurs non corrigées se propagent d'image en image, des raffraichissements intra sont régulièrement appliqués sur des macroblocs appartenant à des images P. Cependant, si une erreur se produit sur des macroblocs adjacents, les performances des rafraîchissements intra se dégradent considérablement. La méthode de prédiction basée épitome proposée dans [WHWH10] permet de rompre la dépendance sur les macroblocs adjacents et d'améliorer ainsi la résistance aux erreurs des rafraîchissements intra. La méthode de prédiction introduite ici consiste à générer une prédiction issue de l'épitome par block matching. Ce mode de prédiction, illustré dans la figure 2.13, identifie cette fois-ci dans l'image épitome, le patch le plus proche au bloc courant au sens de la distance euclidienne. Ce patch est ensuite utilisé comme prédiction pour le bloc courant. Notons cependant que cette méthode de prédiction nécessite en plus la transmission des vecteurs matching au décodeur afin d'obtenir la même prédiction qu'à l'encodeur. Par ailleurs, le vecteur matching est encodé en utilisant un codeur à longueur fixe (FLC) qui varie en fonction de la longueur et la largeur de l'image épitome.

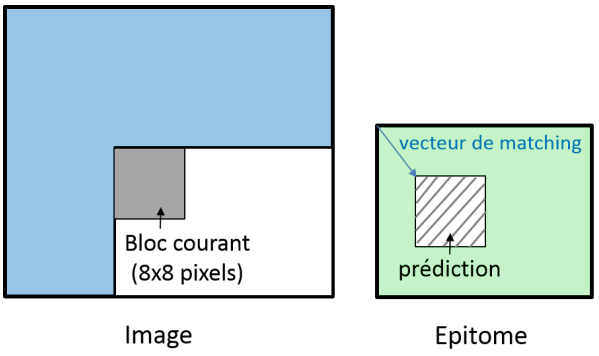

Figure 2.13 – *Prédiction d'un bloc 8x8 générée à partir de l'image épitome par block matching.*

Cette méthode de prédiction a été évaluée pour deux types de GOP : un GOP contenant uniquement des images I et une GOP de type IPP. Comme dans [WHW09], l'épitome EM n'est construit que sur la première image de la capture vidéo. Comme dans [WHW09], l'épitome EM est également généré en utilisant une approche multi-résolution.

Malgré l'amélioration apportée par l'approche multi-résolution, l'épitome EM reste tout de même peu adapté au contexte de la compression. En effet, en premier lieu, l'épitome est construit en prenant en compte toutes les régions de l'image source. Or certains blocs

de l'image peuvent déjà être bien prédits à partir des modes de prédiction directionnels classiques et par conséquent ne pas utiliser la prédiction provenant de l'épitome. De plus, le partitionnement des patches dans l'image pour construire l'épitome diffère de la partition bloc utilisée en compression. Par ailleurs, comme nous l'avons souligné dans la section précédente, la stratégie de moyennage adoptée pour remettre à jour le contenu de l'épitome (étape-M) introduit du flou dans l'épitome, ce qui limite la précision de la prédiction issue de l'épitome. Pour finir, le manque de cohérence dans l'épitome par rapport à l'image initiale peut également occasionner un surcoût de codage de l'épitome.

Afin d'obtenir un épitome plus approprié pour des applications de compression, Q. Wang et al ont étendu l'approche [WHWH10] en proposant une analyse épitomique orientée compression [WHW+12]. Les auteurs ont introduit en particulier une nouvelle fonction d'énergie **??** permettant de considérer certaines contraintes de compression.

$$\varphi(X,Z) = \frac{1}{X^+}\sum_{p\in X^+}\delta_p * |x_p - z_{p^*}|^2 + \frac{\alpha}{Z^+}\sum_{q\in Z^+}|x_{q^*} - z_p|^2 \tag{2.20}$$

Cette fonction d'énergie se rapproche de la mesure de similarité bidirectionnelle mise au point par D. Simakov et al.[SCSI08] dans laquelle des contraintes de complétude et de cohérence sont prises en compte. Notons que le terme de complétude seul (dans l'équation (2.20)) correspond à la fonction optimisée par N. Jojic. Deux améliorations sont apportées par cette nouvelle fonction d'énergie :

- Utilisation d'une pondération adaptative δ_p sur le terme de complétude. L'introduction de ce paramètre de pondération, relatif à chaque patch x_p considéré dans l'image source, permet de rendre la méthode de prédiction proposée dans [WHW+12] complémentaire aux modes de prédiction intra existants. En effet, les régions, qui ne sont pas correctement prédites par les modes de prédiction intra directionnels, auront une plus grande probabilité d'apparaître dans l'épitome. Afin d'éviter d'effectuer deux passes de codage pour déterminer les pondérations à appliquer sur les patches de l'image, les auteurs choisissent d'estimer grossièrement l'énergie résiduelle en utilisant une carte de contours associée à l'image source.
- Prise en compte d'une contrainte de cohérence. Afin de limiter les artefacts visuels introduits dans l'épitome, engendrant un coût de codage plus important dû à la faible corrélation des patches adjacents, le terme de cohérence $\frac{\alpha}{Z^+}\sum_{q\in Z^+}|x_{q^*} - z_p|^2$ a été ajouté dans (2.20).

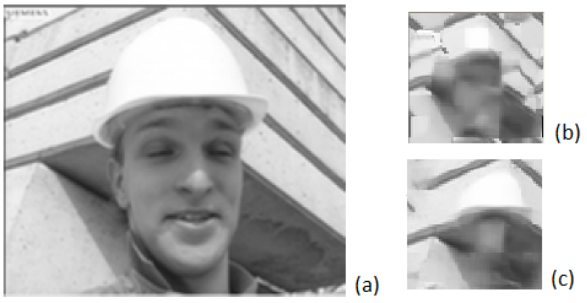

Figure 2.14 – *(a) Image source : Foreman, (b) Epitome EM de N. Jojic (analyse épitomique de base avec en plus une approche multi-résolution), (c) Epitome EM de [WHW+12] (analyse épitomique orientée compression avec un redimensionnement graduel de l'image*

La méthode de prédiction proposée dans [WHW+12] consiste à utiliser conjointement deux modes de prédiction basés épitome : une prédiction par template matching pour les blocs 4x4 et une prédiction par block matching pour les blocs 8x8. La sélection du meilleur mode est effectuée à partir du critère d'optimisation débit/distorsion (RDO). Les auteurs adoptent également la méthode de redimensionnement graduel de [SCSI08] pour approcher la meilleure solution dans l'algorithme EM. En effet, l'épitome final est obtenu ici en passant par une séquence d'épitomes intermédiaires de tailles décroissantes calculés à partir de l'algorithme EM. Le fait de réduire graduellement l'échelle de l'image va permettre ainsi d'obtenir une bonne initialisation pour chaque épitome qui facilitera la convergence globale de l'algorithme EM. La figure 2.14 montre le résultat de deux épitomes générés respectivement avec la méthode de N. Jojic (approche multirésolution) et la méthode proposée dans [WHW+12]. On constate que l'épitome utilisant une métrique bidirectionnelle est visuellement plus cohérent que celui obtenu à partir de la méthode de N. Jojic. Les auteurs montrent également que l'analyse épitomique orientée compression permet de fournir de meilleures performances de codage.

2.3 Conclusion

Dans ce chapitre, nous avons introduit tout d'abord les principales approches existantes dans la littérature permettant de construire une représentation condensée d'une image. En effet, plusieurs concepts ont été définis pour modéliser une image "résumé" de type épitome : un modèle probabiliste basé patch appris sur l'image d'entrée (i.e. modèle EM), un résumé par similarité bidirectionnelle, un modèle parcimonieux basé ISD ainsi qu'un modèle basé sur la factorisation d'éléments répétitifs dans l'image. L'outil de factorisation d'image se démarque des autres approches car seules des textures réelles issues de l'image source sont introduites dans l'épitome. Alors que pour les autres approches, l'épitome contient certaines textures/structures destructurées par rapport à l'image de départ. De plus, contrairement aux autres approches où la taille de l'épitome est préalablement fixée et un traitement de mise à jour itératif est opéré pour raffiner le contenu de l'épitome, la méthode de factorisation d'image permet de contrôler la perte d'information introduite dans l'image tout en essayant de construire un épitome le plus concis possible.

Nous avons ensuite passé en revue les récentes techniques de codage intra existantes à ce jour s'appuyant sur le concept d'épitome. Les principales techniques étudiées dans la littérature sont essentiellement basées sur des épitomes de type EM [JFK03] dans lesquelles de nombreuses améliorations ont été apportées afin d'obtenir une image "résumé" de meilleure qualité et visuellement plus cohérente. L'outil de factorisation de textures répétitives de H. Hoppe a également été exploré dans un contexte de compression où seule la représentation factorisée est encodée, à savoir la texture de l'épitome et la carte de transformation associée. Bien que ce type d'approche soit intéressant, des différences relativement importantes peuvent subsister entre l'image d'origine et l'image reconstruite dues à la tolérance d'erreur fixée lors de la procédure de génération de l'épitome. Nous avons donc pris parti d'orienter les travaux de cette thèse vers l'amélioration de ce type de schéma en explorant de nouveaux schémas allant au delà de l'approche proposée par [WWOH08]. Les nouvelles méthodes développées seront abordées dans le chapitre 5. Le chapitre suivant décrit les évolutions apportées à une technique de prédiction spatiale bien connue de la littérature, à savoir le Template Matching (TM).

Deuxième partie

Contributions

CHAPITRE 3

Prédiction intra image basée sur une méthode hybride TM/BM

La méthode de prédiction basée template matching (TM) a montré, dans bon nombres de cas, sa supériorité sur les modes de prédiction directionnels H.264 pour le codage Intra grâce à une meilleure prédiction spatiale sans l'adjonction de données auxiliaires supplémentaires à transmettre. En effet cette méthode fonctionne bien lorsque le template et le bloc à prédire sont fortement corrélés par exemple, dans les zones de l'image homogènes, cette méthode étant mise cependant en échec dans le cas contraire.

Ce chapitre explore l'utilisation d'un algorithme de prédiction spatiale basé block matching (BM) en plus du TM, ainsi grâce à un algorithme de décision de type débit-distorsion (RD) ce mode de prédiction va naturellement être utilisé dans les zones de l'image lorsque le mode template matching échoue, les modes classiques de prédiction étant par ailleurs toujours permis. Ce chapitre est donc organisé comme suit. Dans un premier temps, nous venons poser le problème de la prédiction intra et examiner les principales approches concernant cette problématique. Ensuite, nous décrirons le modèle hybride proposé et l'algorithme basé block matching spatial dédiés à la prédiction intra. Pour finir, nous fournirons les performances avec illustrations dans le contexte de la prédiction et de la compression. Nous verrons que cette approche hybride BM/TM permet d'offrir un gain de codage significatif par rapport aux performances des modes de prédiction intra de H.264. En effet, l'algorithme basé TM et l'approche hybride TM/BM montrent, avec la mesure de Bjontergaard, des gains en débits allant jusqu'à respectivement 38.02% et 48.38% à bas débits lorsqu'on les compare avec H.264 Intra.

3.1 Prédiction d'image (background)

Soit X un patch de texture qui comprend une partie connue $X(k)$ (d'une forme donnée) formée par les pixels situés dans un voisinage causal du bloc courant, appelé le template, et d'une partie inconnue $X(u)$ formée par le bloc courant à prédire (voir Figure 3.1). Cette section rappelle brièvement les approches les plus répandues pour prédire les échantillons inconnus $X(u)$ connaissant les échantillons précédemment codés et décodés $X(k)$.

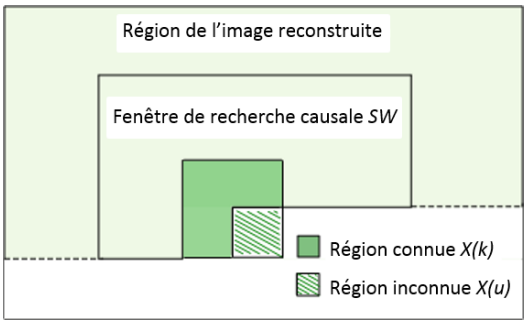

FIGURE 3.1 – *Formulation du problème de prédiction et notations : $X(k)$ désigne les pixels connus (ou template), $X(u)$ est le bloc courant à prédire, et SW est la fenêtre dans laquelle le prédicteur sera recherché.*

3.1.1 Modes de prédiction H.264

La norme H.264/AVC définit trois types de modes de prédiction intra en fonction de la taille de bloc : Intra-4x4, intra-8x8 et intra-16x16 [WSBL03]. Pour chaque mode de prédiction intra, un bloc est prédit à partir des échantillons antérieurs codés/décodés de blocs spatialement voisins. Pour la prédiction Intra des blocs 16x16, quatre modes de prédiction peuvent être utilisés tandis que pour la prédiction Intra des blocs 4x4 et 8x8, neuf modes ont été définis comme l'illustre la Figure 3.2. En plus du mode dit "DC" qui consiste à prédire l'ensemble du bloc à l'aide de la moyenne des pixels voisins, huit modes de prédiction directionnels sont spécifiés. La prédiction est alors réalisée par simple propagation (ou interpolation) des valeurs de pixels le long de la direction spécifiée. Cette approche est particulièrement adaptée en présence de contours, lorsque la direction du mode choisi correspond à l'orientation du contour.

Cependant, cette technique de propagation mono-directionnelle échoue dans les zones texturées plus complexes. Par ailleurs, dans H.264/AVC, la prédiction intra est utilisée pour réduire la corrélation entre le bloc courant et les pixels adjacents, mais pour certaines images, le bloc courant peut présenter une grande similitude avec des blocs provenant d'autres parties de l'image. Plusieurs nouvelles méthodes de prédiction intra ont donc été proposées en vue de mieux exploiter la redondance dans une image.

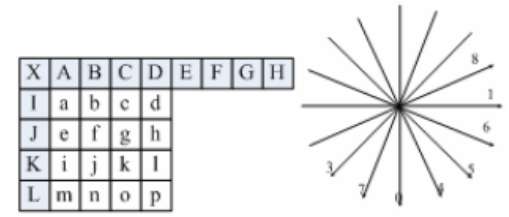

FIGURE 3.2 – *Un bloc de 4x4 pixels avec ses pixels limitrophes et ses huit modes de prédiction H.264 intra directionnels.*

3.1.2 Prédiction basée Block Matching (BM)

Tout d'abord, l'algorithme block matching, initialement utilisé pour la compensation de mouvement dans le cadre de la compression vidéo, a été exploré pour le codage intra

[YC02]. Cette méthode de prédiction intra basée bloc procède de la même façon que l'estimation et compensation de mouvement en codage prédictif inter-image, mais ici le bloc de référence utilisé pour la prédiction est recherché dans la partie causale de l'image courante. En effet, cet algorithme recherche, dans une fenêtre de recherche SW donnée dans l'image reconstruite, le meilleur bloc correspondant en minimisant la distance euclidienne entre le vecteur formé par les échantillons inconnus $X(u)$ du bloc à prédire et les blocs candidats provenant de SW. On notera que la région inconnue $X(u)$ est de taille n^2 pixels. Par ailleurs, les vecteurs associés aux blocs de prédiction doivent être codés car ils sont nécessaires au décodeur pour reconstruire ce type de prédiction.

3.1.3 Prédiction basée Template Matching (TM)

Un algorithme alternatif de prédiction spatiale basé sur le template matching a également été introduit dans [TBS06]. L'algorithme TM recherche, dans une fenêtre donnée SW située dans la partie causale de l'image, le patch présentant le "meilleur matching" dans le sens de la minimisation de la distance euclidienne entre le vecteur formé par les échantillons connus $X(k)$ du voisinage du bloc à prédire et les pixels co-localisés dans les patches candidats provenant de la fenêtre SW. En Figure 3.1, la région connue $X(k)$ est constituée par les valeurs des pixels de 3 blocs voisins, d'où, le vecteur $X(k)$ correspondant est de taille $\mathrm{N} = 3n^2$ pixels. On notera, cependant, que différentes formes de templates $X(k)$ peuvent être considérées [TG10]. Les pixels du patch candidat qui sont co-localisés aux pixels du bloc à prédire sont ensuite utilisés en tant que pixels de prédiction pour le bloc courant à coder.

Cependant, une bonne adéquation du template ne conduit pas nécessairement à une bonne prédiction pour le bloc à prédire, en particulier dans les zones de textures non homogènes de l'image ou quand il y a des discontinuités entre le template et le bloc à prédire. Les limitations du template matching nous ont ainsi conduits à nous orienter vers un autre algorithme de prédiction décrit dans la section suivante.

3.2 Algorithme de prédiction hybride TM/BM

3.2.1 Motivations

Afin de pallier les limites du template matching, les blocs présentant des discontinuités avec le template seront mieux prédits si nous pouvons faire une correspondance directe avec le bloc à prédire. Mais cela nécessite un coût supplémentaire en débit qui représentera le codage du vecteur de correspondance entre le bloc courant et le bloc de prédiction. Il s'agit de l'algorithme de bloc matching qui est utilisé classiquement pour l'estimation de mouvement, mais qui cette fois-ci va être appliqué dans une fenêtre de recherche de la partie causale de l'image courante reconstruite. Afin d'éviter le fort impact du coût du vecteur de matching sur les performances débit-distorsion de l'algorithme de compression complet, ce mode doit être utilisé avec parcimonie c'est à dire pour les blocs où le TM ne convient pas.

3.2.2 Structure de l'encodeur

La méthode de prédiction intra proposée consiste donc à utiliser le meilleur des algorithmes TM et BM. La prédiction spatiale BM est utilisée lorsque de la prédiction spatiale TM échoue et bien évidemment si les modes de prédiction classiques ne conviennent pas non plus. L'utilisation conjointe des deux modes de prédiction intra BM et TM permet

ainsi de tirer avantage des deux méthodes. En effet, l'approche template matching fonctionne bien lorsque le bloc inconnu à prédire et son template sont corrélés à ceux obtenus par la mise en correspondance du template. Par ailleurs, la prédiction par TM ne nécessite pas l'envoi de vecteurs de matching supplémentaires, mais le même processus de mise en correspondance de template doit être effectué côté décodeur ce qui engendre inévitablement une augmentation de la complexité. En revanche, la méthode de prédiction basée BM peut mieux décorréler le bloc à prédire avec la région d'image reconstruite et ne nécessite pas d'effectuer le même procédé de mise en correspondance de bloc côté décodeur, mais les vecteurs de matching doivent être transmis au décodeur. Par conséquent, la méthode hybride TM/BM peut non seulement réduire la complexité de calcul au niveau du décodeur par rapport au TM seul mais également réduire l'impact du coût de débit relatif aux vecteurs de matching par rapport au BM seul.

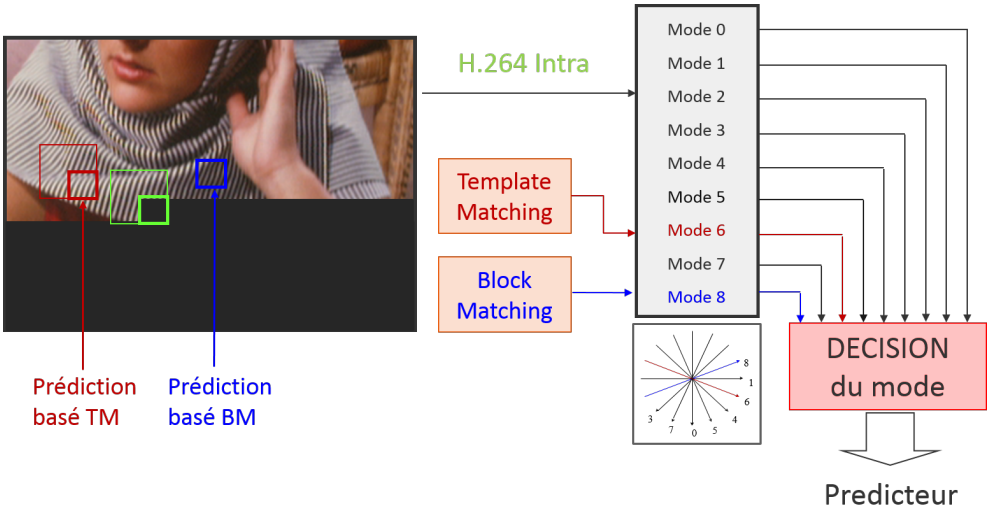

Figure 3.3 – *Synoptique du procédé de sélection des modes de prédiction intra hybride TM/BM pour des blocs* 4*x*4 *ou* 8*x*8.

Nous avons introduit des modes de prédiction intra basés TM et BM dans le codec H.264 (logiciel KTA référence [sof]) en remplaçant deux modes existants de l'algorithme H.264 comme l'illustre la figure 3.3. Ici, les deux modes de prédiction intra, qui sont remplacés par des modes TM et BM, correspondent statistiquement aux modes les moins utilisés en Intra H.264. Cela facilite la mise en oeuvre en gardant la même syntaxe et permet d'économiser des bits de signalisation. Ici, l'approche hybride TM/BM proposée est appliquée aux modes intra-4x4 et intra-8x8 et les vecteurs de matching sont limités à la précision pel entier. Le choix du meilleur mode de codage pour un bloc donné est effectué selon une minimisation d'un critère Lagrangien, également appelé optimisation d'un critère débit/distorsion (RDO) :

$$\min\left(D + \lambda \times R\right) \tag{3.1}$$

où D représente la distorsion entre le bloc original et le bloc reconstruit en utilisant la somme des erreurs au carrée (SSE) comme métrique, λ représente un coefficient permettant d'avoir le meilleur compromis entre la distorsion et les bits nécessaires pour coder le bloc. Le paramètre λ est déterminé comme dans le logiciel de référence KTA ($\lambda = 0.65 \times 2^{QP/3}$), avec QP correspondant au paramètre de quantification. Pour le mode de codage intra basé

BM, le coût R est calculé selon la formule suivante :

$$R = R_{mode} + R_{res} + R_{mvector} \tag{3.2}$$

où R_{mode}, R_{res} et $R_{mvector}$ représentent les bits nécessaires pour encoder respectivement le mode de codage du bloc, l'erreur résiduelle de prédiction et le vecteur de matching. Notons que R_{mode} et R_{res} sont encodés avec l'algorithme de codage entropique KTA (CABAC). Les vecteurs de matching sont encodés en utilisant un code à longueur fixe (FLC), qui varie toutefois selon la taille de la fenêtre de recherche causale SW. Pour le mode de codage intra basé sur TM, l'équation (3.2) est également utilisée avec $R_{mvector} = 0$.

3.3 Résultats expérimentaux

3.3.1 Performances en termes de débit/distorsion

Trois schémas de codage : l'algorithme basé TM, l'algorithme basé BM et l'algorithme basé sur l'approche hybride TM/BM ont été évalués relativement aux modes de prédiction intra H.264. Plusieurs images à différentes résolutions ont été testées et présentées dans les tableaux 3.1 et 3.2, où les efficacités de codage ont été comparées en utilisant la métrique de Bjontegaard [Bjo01] (appelée BD-rate). Ici, nous avons considéré une fenêtre de recherche de taille de 64x128 pixels et des vecteurs de prédiction à la précision pixellique. En référence à H.264 Intra, les gains BD-rate moyens obtenus avec la méthode de compression basée TM sont de 17.56% à bas débits et 13.64% à hauts débits, tandis que ceux obtenus avec la méthode de compression hybride TM/BM sont de 24.78% à bas débits et 16.74% à hauts débits.

Bas débits	TM		BM		TM + BM	
QP :26,31,36,41	dB	% débit	dB	% débit	dB	% débit
Barbara	0.45	-6.80	0.49	-7.41	0.83	-12.06
Wool	0.88	-13.81	1.13	-17.36	1.37	-20.93
Snook	1.08	-14.77	1.47	-20.25	1.58	-21.71
Zone1	5.87	-38.02	5.98	-38.76	6.60	-41.78
Pan0_qcif	1.21	-18.91	1.21	-19.16	1.64	-25.36
City2	3.32	-37.92	4.03	-44.21	4.42	-48.38
Matrix	0.85	-9.97	1.18	-15.33	1.78	-21.33
Spincalendar	0.56	-10.53	0.86	-15.74	1.13	-20.29
Foreman	0.38	-7.34	0.40	-7.67	0.58	-11.21
Moyenne	**1.62**	**-17.56**	**1.86**	**-20.65**	**2.21**	**-24.78**

Table 3.1 – *Gains en débit-distorsion (avec la mesure Bjontergaard) par rapport aux modes de prédiction intra H.264 à bas débits (calculés à partir de 4 mesures de débit : QP=26,31,36,41).*

La première série d'images (à partir de Barbara à City2) est partiellement composée de textures pseudo-régulières qui conviennent assez bien aux différents algorithmes testés. Les images Zone1 (image Fresnel) et Pan0_qcif (texture presque régulière) peuvent être considérées comme atypiques. L'image Matrix, composée de séquences de chiffres aléatoires, est une image de test (tirée du film Matrix) qui met en valeur les vertus des deux algorithmes. En effet, le TM ne permet pas une prédiction fiable d'un chiffre donné ne connaissant que son voisinage. Cependant, cette approche fonctionne bien dans la synthèse de texture

Hauts débits	TM		BM		TM + BM	
QP :16,21,26,31	dB	% débit	dB	% débit	dB	% débit
Barbara	0.45	-5.41	0.36	-4.38	0.61	-7.37
Wool	0.79	-8.57	0.75	-8.46	0.95	-10.53
Snook	1.31	-11.20	1.53	-13.13	1.65	-14.12
Zone1	6.47	-35.89	7.35	-38.83	7.66	-40.42
Pan0_qcif	1.44	-12.96	1.06	-10.00	1.60	-14.48
City2	4.66	-32.14	4.62	-33.05	5.20	-36.56
Matrix	0.75	-6.16	0.90	-7.52	1.42	-11.84
Spincalendar	0.56	-6.89	0.59	-7.52	0.86	-10.73
Foreman	0.25	-3.57	0.22	-3.31	0.32	-4.62
Moyenne	**1.85**	**-13.64**	**1.93**	**-14.02**	**2.25**	**-16.74**

Table 3.2 – *Gains en débit-distorsion (avec la mesure Bjontergaard) par rapport aux modes de prédiction intra H.264 à hauts débits (calculés à partir de 4 mesures de débit : QP=16,21,26,31).*

parce qu'elle est basée uniquement sur les aspects visuels au lieu d'une mesure objective telle que la SSE et le PSNR. En revanche, le BM permet d'obtenir la bonne prédiction d'un chiffre, si le même chiffre est présent dans la fenêtre de recherche causale. Nous avons obtenu des résultats aussi intéressants avec des images classiques comme Foreman (CIF) et Spincalendar (HD), en particulier à bas débit.

Suite aux simulations réalisées sur l'ensemble des séquences de test, nous pouvons remarquer que l'utilisation conjointe des prédictions TM et BM permet d'obtenir une meilleure efficacité du codage par rapport à H.264 Intra pour divers contenus dans toutes les résolutions. Cependant, la méthode hybride TM/BM proposée offre des performances de codage variables en raison du type de contenu dans les images. En effet, d'une part, nous pouvons voir que les gains de codage sont moins importants pour les séquences Barbara et Foreman par rapport à H.264 Intra puisque les modes de prédiction tant directionnels que du type DC de H.264 conviennent assez bien. D'autre part, nous pouvons observer que les gains de codage sont importants pour les images comme Zone1 et City2 qui présentent des propriétés stationnaires locales en termes de texture. En effet, un gain en débit jusqu'à 41.78% et 48.38% respectivement est réalisé à bas débits par rapport à H.264 Intra.

Les résultats expérimentaux montrent également que la méthode hybride proposée surpasse l'approche basée TM ainsi que celle basée BM. Ceci s'explique par le fait que ces deux outils algorithmiques de prédiction sont complémentaires. A partir des résultats de simulation, nous pouvons souligner que le BM fournit, sans grande surprise, une meilleure prédiction des variations locales du signal que le TM comme nous pouvons le voir avec les images et Barbara et Snook (cf. tableau 3.1).

3.3.2 Performances en termes de prédiction

La Figure 3.4 illustre visuellement les prédictions issues des modes TM et BM pour les modes Intra-8x8 seulement. Nous pouvons noter que l'image Barbara présente des structures pseudo-périodiques qui sont localement variables en raison de déformations spatiales du tissu. C'est la raison pour laquelle le mode TM est moins choisi que le mode BM par l'algorithme, basé RD, dédié au choix des modes de codage. L'image City2 montre que les deux méthodes TM et BM se complètent mutuellement puisque cette image présente localement des textures homogènes, tout en ayant en même temps un léger effet de perspective.

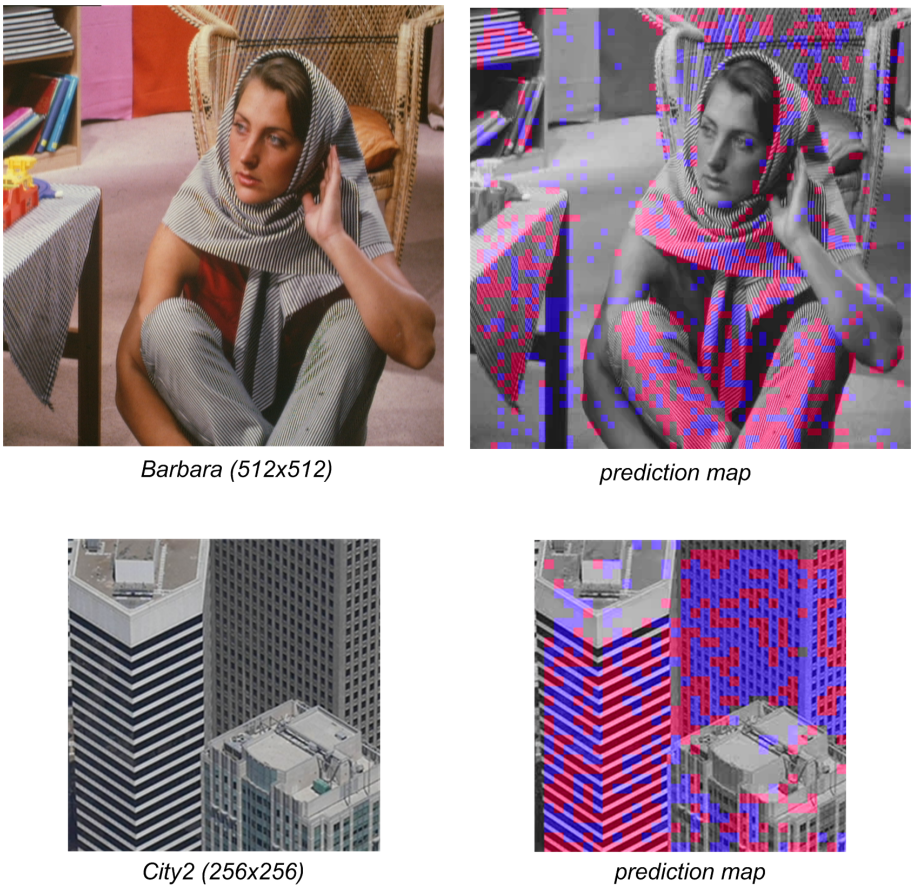

Figure 3.4 – *Illustration des modes utilisés en prédiction 8x8 : TM (en bleu) et BM (en rouge) (QP=31).*

3.4 Conclusion

Dans ce chapitre, un algorithme hybride de prédiction s'appuyant sur du template matching et du bloc matching a été introduit. Cette nouvelle approche de prédiction offre des résultats intéressants comparée à H.264 Intra mais également par rapport au mode TM introduit dans H.264 Intra. En outre, la méthode hybride TM/BM permet de réduire la complexité de calcul au niveau du décodeur par rapport au TM seul étant donné que le mode BM ne nécessite pas de recherche d'appariement au niveau du décodeur.

Malgré les premiers résultats prometteurs obtenus avec la méthode hybride TM/BM, les textures complexes demeurent néanmoins difficiles à prédire à partir d'un seul bloc, même similaire, de l'image. Dans le chapitre suivant, une autre approche hybride multi-patches de type "neighbor embedding" (NE) a été explorée. Les techniques de prédiction utilisées vont au-delà de la simple mise en correspondance de blocs et sont basées sur une combinaison linéaire de plusieurs patches. L'idée sous-jacente est qu'une texture complexe peut être interpolée par une combinaison linéaire de plusieurs éléments de texture similaire.

CHAPITRE 4

Prédiction intra image basée sur des méthodes hybrides multipatches de type neighbor embedding (NE)

Les méthodes de réduction de dimensionnalité de données sont largement utilisées pour une multitude d'applications telles que la fouille de données, l'apprentissage automatique, l'extraction de données, la vision par ordinateur et le traitement d'images (visualisation et compréhension de données, classification). Par ailleurs, des études récentes ont montré l'intérêt d'utiliser des techniques de type "neighbor embedding" pour la prédiction intra images [TG12]. Ces méthodes "neighbor embedding" sont utilisées en prédiction intra pour approximer un bloc d'entrée (le bloc à prédire) de l'image comme une combinaison linéaire de ses K plus proches voisins (K-NN). Afin que le décodeur puisse procéder de la même manière, les K plus proches voisins sont déterminés à partir des distances calculées entre les pixels connus dans le voisinage causal (appelé template) du bloc d'entrée et les pixels colocalisés dans les patches candidats issus de la fenêtre de recherche causale. Le template est donc approximé ici par une combinaison linéaire de ses K plus proches voisins situés dans la fenêtre de recherche et les mêmes coefficients de pondération de la combinaison linéaire sont ensuite utilisés pour estimer le bloc à prédire. Ces méthodes, bien qu'efficaces, présentent néanmoins certaines limitations lorsque le template est très peu corrélé avec le bloc à prédire, i.e. en présence de zones de texture non homogènes. En effet, dans ce cas, les poids menant à une bonne approximation du template risquent de ne pas être bien adaptés pour estimer le bloc à prédire. Ce problème est assez similaire à celui traité dans le chapitre précédent 3 dans lequel une méthode hybride basée sur l'utilisation conjointe du template matching et bloc matching a été envisagée pour pallier les limitations du template matching.

Ce chapitre vise à introduire de nouvelles méthodes de prédiction d'images basées sur les outils "neighbor embedding". Dans un premier temps, nous décrivons les principes de base des méthodes de réduction de dimensionnalité ainsi que certaines solutions fournies dans la littérature. Ensuite, nous rappellerons les méthodes de prédictions intra existantes [TG12] basées sur deux techniques de réduction de dimensionnalité : NMF (Non Negative Matrix Factorization) et LLE (Locally Linear Embedding) qui peuvent se référer à des outils de type "neighbor embedding". Nous introduirons ensuite de nouvelles méthodes de prédiction d'images basées sur les mêmes outils "neighbor embedding", nommées "Map-Aided Neighbor Embedding" (MANE) dans lesquelles la recherche des K-NN est effectuée ici en deux

étapes et est assistée, au décodeur, par un vecteur de matching. Une autre variante optimisée de notre approche, appelée "optimized Map-Aided Neighbor Embedding" (oMANE) sera également détaillée. Dans ces méthodes, plusieurs alternatives seront proposées pour améliorer la recherche des K-NN. Ensuite, un schéma complet de compression sera décrit dans lequel les performances de différentes méthodes ont été évaluées par rapport à celles obtenues avec les modes de prédiction intra H.264. Nous montrerons que des méthodes de prédiction hybrides proposées permettent de fournir des améliorations significatives en termes de débit/distorsion par rapport aux modes de prédiction Intra H.264 (jusqu'à une réduction de débit de 44.75 % à bas débits).

4.1 Méthodes de réduction de dimension

Depuis plusieurs décennies, le volume des données disponibles ne cesse de croître allant du méga-octets dans les débuts des années 80 jusqu'au tera-octets et au peta-octets aujourd'hui. La taille des données dépend essentiellement de la taille des variables utilisées (i.e. nombre d'attributs) ainsi que le nombre d'exemples. Ces derniers peuvent prendre des valeurs très élevées et par conséquent poser problème lors de l'exploration et l'analyse des données. Afin de pallier ce problème, la réduction des dimensions a été l'une des voies de recherche la plus explorée ces dernières années, permettant ainsi une meilleure analyse, régression, présentation et visualisation des données. L'objectif de ces méthodes est de sélectionner les caractéristiques les plus pertinentes afin d'éliminer les informations non pertinentes et redondantes. Une méthode très connue et utilisée pour la réduction des dimensionnalités des données linéaires est l'analyse en composantes principales (PCA) [Jol02]. La PCA consiste à effectuer un changement de repère qui vise à privilégier les axes de variance maximale par rapport à un ensemble de données. Les axes où la variance des données est réduite peuvent être éliminés pour atteindre une réduction de la dimensionnalité avec une perte minimale d'information. Formellement, cette méthode effectue une réduction de dimension en projetant L points de l'espace original $\mathbf{X}_i$ de dimension N sur un sous-espace vectoriel de dimension plus réduite M, qui est couvert par des fonctions de base orthogonales $\mathbf{A}_m, m = 1...M$ et $N \geq M$, en cherchant à minimiser l'erreur de reconstruction suivante :

$$\sum_i \left\| \mathbf{X}_i - \sum_m \left({\mathbf{X}_i}^T \mathbf{A}_m \right) \mathbf{A}_m \right\|_2^2 . \tag{4.1}$$

Pour ce faire, après avoir centré les données autour de l'origine, la PCA va calculer la décomposition en valeurs propres de la matrice de covariance de ces données. Le sous-espace de M dimensions est défini par les M vecteurs propres associés aux M plus grandes valeurs propres de la matrice de covariance. Ces M vecteurs propres seront les axes du nouveau système de coordonnées et nous obtiendrons une représentation des données en plus faible dimension en les projetant sur ces vecteurs. Notons que dans l'équation (4.1) le signal $\mathbf{X}_i$ est approximé par une combinaison linéaire des fonctions de base $\mathbf{A_m}$, où les coefficients de pondération correspondent aux projections orthonormales de $\mathbf{X}_i$ sur les directions définies par $\mathbf{A}_m, \forall m$.

Une autre méthode de réduction de dimensionnalité linéaire, appelée NMF (Non Negative Matrix Factorization) [LS00], a été introduite dans les domaines du traitement de signal et des images. Cette méthode peut être vue comme une généralisation du PCA avec des contraintes de non négativité, où uniquement des combinaisons linéaires additives sont

permises. En effet, l'algorithme de la NMF représente chaque donnée comme une combinaison linéaire de vecteurs de la base de l'espace réduit mais n'autorise pas les éléments négatifs dans les vecteurs de la base ni dans les poids de la combinaison linéaire. La NMF permet de fournir une représentation en "parties" d'un ensemble. L'article de référence de la NMF [LS00] montre que l'utilisation de celle-ci sur une base de données d'images de visages permet d'extraire les différentes parties du visage (nez, yeux, bouche ...). La NMF permet ainsi de fournir une interprétation physique des résultats et une décomposition plus proche de ce qu'on perçoit.

Les méthodes linéaires telles que le PCA, la NMF, donnent des résultats intéressants à condition que les données soient situées dans un sous-espace linéaire mais ces méthodes ont tendance à échouer dès que les données se trouvent sur une variété non linéaire. Or les données observées dans la nature présentent souvent des caractéristiques non linéaires. Par conséquent, la plupart des méthodes linéaires ont été étendues à une version non-linéaire, permettant ainsi de gérer les caractéristiques non linéaires des données.

Par exemple, l'approche de l'ACP a été étendue à une version non linéaire en introduisant des fonctions à noyaux non linéaires, appelée "Kernel Principal Component Analysis" (KPCA) [SSM98]. L'idée du Kernel PCA est de remplacer les produits scalaires par une fonction à noyau, qui peut être vue comme une mesure de similarité non linéaire. Pour le KPCA, les noyaux fréquemment rencontrés dans la littérature sont le noyau gaussien (Equation (4.2)) et le noyau polynomial (Equation (4.3)) :

$$K(x,y) = \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right). \tag{4.2}$$

$$K(x,y) = (xy+b)^c. \tag{4.3}$$

Cependant, ces noyaux sont mal adaptés pour l'apprentissage des variétés (ou manifolds).

Les méthodes ISOMAP [TdSL00] et LLE [RS00] (Locally Linear Embedding ou plongement localement linéaire) sont des méthodes d'apprentissage de variétés basées graphes. Ces méthodes peuvent être vues comme une extension de la technique KPCA puisque les noyaux utilisés résultent directement des données et non d'une fonction à noyau prédéfinie. L'ISOMAP est une méthode de réduction des dimensions qui préserve les distances

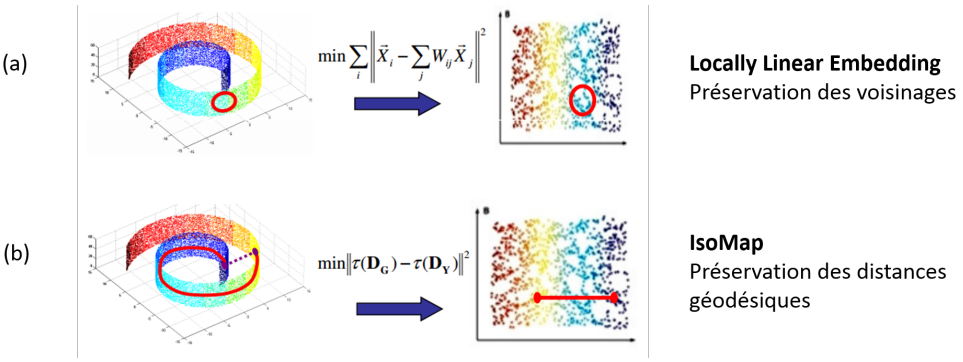

Figure 4.1 – *Deux méthodes de réducion de dimensionnalité non linéaires : (a) LLE [RS00] ; (b) IsoMap [TdSL00]).*

géodésiques entre les paires de points. En effet, l'objectif de l'algorithme consiste à projeter les points dans un espace de plus faible dimension de manière à ce que les distances euclidiennes dans cet espace deviennent approximativement égales aux distances des géodésiques dans l'espace original. Alors que la méthode LLE construit une projection vers un espace linéaire de faible dimension en préservant la structure locale des voisinages, chaque point étant ici caractérisé par sa reconstruction à partir de ses plus proches voisins. La figure 4.1 illustre les deux méthodes : LLE et ISOMAP. La suite des travaux présentés dans ce chapitre se focalisera principalement sur les outils de réduction de dimensionnalité de type LLE et NMF.

4.1.1 Principe du Locally Linear Embedding (LLE)

La méthode LLE suppose que chaque point et ses plus proches voisins dans l'espace d'origine sont approximativement situés sur une partie localement linéaire. L'algorithme vient donc exploiter la géométrie locale des points dans l'espace original pour le reproduire dans un espace de plus faible dimension. Chacun des points possédant un voisinage, l'idée consiste à exprimer chaque point comme une combinaison linéaire de ses voisins et de construire son image dans le nouvel espace en respectant cette relation géométrique, comme illustré en figure 4.2.

Soit L le nombre de points dans l'espace d'origine $\mathbf{X}_i$ de dimension N, la méthode LLE [RS00] procède de la manière suivante :

1. On commence par identifier les K plus proches voisins $\mathbf{X}_j$ pour chaque point $\mathbf{X}_i$ de l'espace d'origine $(i \neq j)$ au sens de la distance euclidienne.
2. Ensuite, on détermine les poids de reconstruction $\mathbf{W}_{i,j}$ de façon à ce que chaque point $\mathbf{W}_{i,j}$ soit approximé par ses voisins. L'erreur de reconstruction d'un point par ses voisins est mesurée à l'aide de la fonction :
$$E = \sum_i \left\| \mathbf{X}_i - \sum_j \mathbf{W}_{i,j} \mathbf{X}_j \right\|_2^2 . \tag{4.4}$$
où $\mathbf{W}_{i,j}$ représente la contribution du point j^{th} pour la reconstruction du point i^{th}. Afin d'estimer les poids de la reconstruction $\mathbf{W}_{i,j}$, on minimise la fonction de coût ci-dessus avec deux contraintes. Tout d'abord, chaque point n'est reconstruit qu'à partir de ses plus proches voisins (i.e. $\mathbf{W}_{i,j} = 0$ si le point $\mathbf{X}_j$ n'appartient pas aux voisins de $\mathbf{X}_i$). La seconde contrainte consiste à exiger que $\sum_j \mathbf{W}_{i,j} = 1, \forall i$. Ainsi, la contrainte sur les poids de reconstruction $\sum_j \mathbf{W}_{i,j} = 1, \forall i$ permet d'assurer l'invariance par translation des points et de ses voisins. Les poids optimaux satisfaisant ces contraintes sont donc obtenus en résolvant le problème des moindres carrés sous contraintes pour chaque point $\mathbf{X}_i$ comme ci-dessous :
$$E_i = \left\| \sum_j \mathbf{W}_{i,j} \left(\mathbf{X}_i - \mathbf{X}_j \right) \right\|_2^2 \quad \text{s.t.} \quad \sum_j \mathbf{W}_{i,j} = 1; \tag{4.5}$$
3. Pour finir, lorsque la LLE est utilisée à des fins de réduction de dimensionnalité de données, c'est à dire que l'on calcule les points dans l'espace de plus faible dimension, une fonction "embedding" est minimisée afin d'obtenir le nouveau système de coordonnées $\mathbf{Y}_i$ des points dans l'espace de dimension réduite comme suit :
$$\zeta\left(\mathbf{Y}\right) = \sum_i \left\| \mathbf{Y}_i - \sum_j \mathbf{W}_{i,j} \mathbf{Y}_j \right\|_2^2 . \tag{4.6}$$

les poids de reconstruction $\mathbf{W}_{i,j}$ étant fixés. Afin d'estimer les points $\mathbf{Y}_i$ dans l'espace de dimension réduite, on minimise également la fonction de coût embedding avec deux contraintes. Tout d'abord, les points $\mathbf{Y}_i$ doivent pouvoir être translatés sans affecter la fonction de coût (i.e. $\sum_j \mathbf{W}_{i,j} = 0$). Par ailleurs, l'algorithme impose que $\frac{1}{L}\sum_i \mathbf{Y}_i \mathbf{Y}_i^T = I$ afin d'éviter des solutions dégénérées. (voir [RS00] pour des informations supplémentaires sur la méthode)

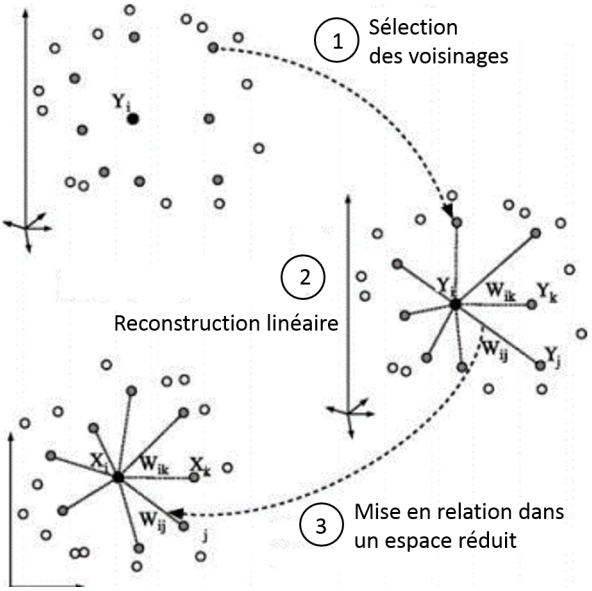

Figure 4.2 – *Principe de la LLE [RS00]).*

4.1.2 Principe du Non Negative Matrix Factorization (NMF)

La NMF est une méthode dédiée à la réduction de rang de données non négatives. Etant donnée une matrice d'observations $\mathbf{X} \in \mathbb{R}^{N \times L}$, à composantes non négatives, dans laquelle chaque colonne représente un point issue de l'espace d'origine, la méthode consiste à rechercher une factorisation de la forme $\mathbf{X} \approx \mathbf{AV}$ où $\mathbf{A} \in \mathbb{R}^{N \times M}$ et $\mathbf{V} \in \mathbb{R}^{M \times L}$ sont également à composantes non négatives. La dimension de M est choisie de sorte que $M \prec min\{N, L\}$. Déterminer $\mathbf{A}$ et $\underline{\mathrm{V}}$ revient à minimiser la distance entre la matrice d'observations X et son estimation $\widehat{\mathbf{X}} = \mathbf{AV}$ par rapport à $\mathbf{A}$ et $\underline{\mathrm{V}}$ sous les contraintes de non négativité :

$$\min_V ||X - \mathbf{A}V||_F^2 \text{ s.t. } \mathbf{A} \geq 0 \text{ and } V \geq 0 \tag{4.7}$$

où l'indice F désigne la norme matricielle de Frobenius. Notons que d'autres fonctions de coût sont également considérées dans la littérature [LS00] en particulier celles utilisant la divergence de Kullback-Leibler mais nous nous limiterons ici à cette dernière (Eq. (4.7)). La matrice $\mathbf{A}$ peut être vue comme une matrice contenant les vecteurs de base et $\mathbf{V}$ comme la matrice contenant les coefficients de l'approximation linéaire.

Une des méthodes les plus utilisées pour résoudre le problème de la NMF est une technique de mise à jour multiplicative proposée dans [LS00]. Avec cet algorithme, le problème de la NMF est résolu en mettant à jour itérativement les éléments de chaque matrice $\mathbf{A}$ et $\mathbf{V}$ comme suit :

$$V_{a\mu} \leftarrow V_{a\mu} \frac{(\mathbf{A}^{\mathrm{T}} X)_{a\mu}}{(\mathbf{A}^{\mathrm{T}} \mathbf{A} V)_{a\mu}} \qquad \mathbf{A}_{ia} \leftarrow \mathbf{A}_{ia} \frac{(XV^{\mathrm{T}})_{ia}}{(\mathbf{A} V V^{\mathrm{T}})_{ia}}. \tag{4.8}$$

Ici, $V_{a\mu}$ (ou A_{ia}) désigne le a^{th} (or i^{th}) élément ligne et le μ^{th} (ou a^{th}) élément colonne des matrices correspondants, $\mathbf{V}$ et $\mathbf{A}$, respectivement. Notons qu'au départ, les matrices $\mathbf{A}$ et $\mathbf{V}$ sont initialisées avec des valeurs non-négatives aléatoires. Ensuite, la mise à jour de $\mathbf{A}$ (récipr. $\mathbf{V}$) s'opère à partir d'un coefficient multiplicatif dépendant uniquement des données $\mathbf{X}$ et de la matrice $\mathbf{A}$ (récipr. $\mathbf{V}$). Par ailleurs, la norme de Frobenius est décroissante et devient invariante lorsque l'on a atteint un point stationnaire. Cependant, un point stationnaire ne garantit pas forcément un minimum global du critère. On peut interpréter que chaque vecteur colonne X_i de $\mathbf{X}$ est approximé par une combinaison linéaire des colonnes du dictionnaire $\mathbf{A}$, les poids étant fournis par les vecteurs colonnes de $\mathbf{V}$.

Nous allons maintenant voir dans la section suivante comment les méthodes de réduction de dimensionnalité de données sont appliquées au problème de prédiction d'image. Il est à noter qu'ici il n'est pas nécessaire de procéder au calcul des points dans l'espace de dimension réduite mais on doit uniquement établir ce que l'on appelle le "neighbor embedding" (NE) des points de données en entrée (i.e. l'espace de dimension original).

4.2 Prédiction intra image basée sur des techniques NE (background)

Cette section a pour but d'expliquer comment les méthodes "neighbor embedding" peuvent être utilisées dans le contexte de prédiction intra image [TG12]. Les outils "neighbor embedding" sont appliqués ici pour approximer les pixels du template en utilisant soit la LLE soit la NMF. Cela revient, pour les deux méthodes, à chercher dans un premier temps les K plus proches voisins (K-NN) du template dans une fenêtre de recherche située dans la partie causale de l'image (cf. figure 4.3.a) puis résoudre dans un second temps l'approximation des moindres carrés avec contraintes pour les pixels du template. Le template est ainsi approximé par une combinaison linéaire de ses K-NN situés dans la fenêtre de recherche et les mêmes poids de la combinaison linéaire des pixels colocalisés dans ses K-NN patches sont ensuite conservés pour estimer les pixels inconnus du bloc courant.

Soit $\mathbf{A} = [\frac{\mathbf{A}(k)}{\mathbf{A}(u)}]$ désignant une matrice de dimension $N \times K$, appelé dictionnaire, les colonnes du dictionnaire $\mathbf{A}$ sont construites en vectorisant les valeurs de luminance des K patches de texture les plus proches issus de la fenêtre de recherche SW. Les sous-matrices $\mathbf{A}(k)$ et $\mathbf{A}(u)$ contiennent respectivement les valeurs des pixels colocalisés aux pixels du template et aux pixels du bloc courant. Soit $X = [\frac{X(k)}{X(u)}]$ le vecteur composé des pixels connus du template $X(k)$ et des pixels inconnus du bloc courant $X(u)$.

En utilisant la méthode LLE, le problème de prédiction peut être réécrit de la manière suivante :

$$\min_{V} \|X(k) - \mathbf{A}(k)V\|_2^2 \text{ s.t. } \sum_m V_m = 1 \tag{4.9}$$

où V indique les coefficients de pondération optimaux, calculés comme suit :

$$V = \frac{\mathbf{D}^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{1}} \tag{4.10}$$

où $\mathbf{D}$ désige la matrice de covariance locale (i.e., en référence à $X(k)$) des K-NN patches sélectionnés et stockés dans $\mathbf{A}(k)$, et $\mathbf{1}$ indique le vecteur colonne composé de 1. En pratique, au lieu d'effectuer une inversion explicite de la matrice $\mathbf{D}$, le système linéaire des

équations $\mathbf{D}V = \mathbf{1}$ est résolu, ensuite les poids sont réajustés afin que la somme soit égale à un.

Utiliser la méthode NMF pour calculer les poids de l'approximation linéaire du template du patch en entrée à partir de ses K-NN, signifie que, pour chaque bloc en entrée exprimé sous forme d'un vecteur X, on vient fixer le dictionnaire non négative $\mathbf{A} \in \mathbb{R}^{N \times K}$ et déterminer uniquement l'autre élément de la factorisation soit le vecteur V. La mesure de distance pour déterminer les patches les plus proches est calculée entre les pixels connus $X(k)$ du patch en entrée à compléter et les pixels colocalisés dans les patches issus de la fenêtre de recherche SW. L'algorithme de prédiction s'effectue ensuite en résolvant le problème des moindres carrées avec contraintes :

$$\min_{V} ||X(k) - \mathbf{A}(k)V||_F^2 \text{ s.t. } \mathbf{A}(k) \geq 0 \text{ and } V \geq 0 \tag{4.11}$$

Une fois que la représentation LLE ou NMF a été déterminée pour l'ensemble des pixels connus $X(k)$ (dans le template), nous conservons les mêmes paramètres de la représentation pour approximer les valeurs des pixels inconnus $X(u)$ du bloc à prédire tel que $\hat{X}(u) = \mathbf{A}(u)V_{opt}$. Notons, que dans le cas de la méthode NMF, seule la règle de mise à jour de la matrice des coefficients V (dans l'équation (4.8)) est appliquée, la matrice de dictionnaire A restant fixée pour toutes les itérations.

Cependant, les méthodes neighbors embedding, décrites dans [TG12], présentent certaines limitations, lorsque le template et le bloc à prédire sont très peu corrélés, c'est à dire, que les K-NN déterminés pour le template ne sont pas bien corrélés avec les K-NN du bloc courant à prédire. Par conséquent, les poids pour approximer le template courant ne sont pas bien adaptés pour prédire le bloc courant.

4.3 Prédiction intra image basée sur des techniques NE assistées par un vecteur de matching

4.3.1 Algorithme MANE (map-aided neighbor embedding)

Les méthodes neighbor embedding proposées consistent à utiliser un vecteur de matching en vue d'améliorer les méthodes neighbor embedding actuelles (LLE, NMF) pour la prédiction intra image. En effet, l'idée est de prendre en compte (à l'encodeur) l'information de texture issue du bloc courant à prédire $X(u)$ afin d'obtenir de meilleurs candidats pour les K patches stockés dans $\mathbf{A}(k) = [\mathbf{a}_0(k), ..., \mathbf{a}_{K-1}(k)]$. L'objectif est que les patches choisis pour approximer le template $X(k)$, soient mieux adaptés pour estimer les pixels inconnus du bloc à prédire $X(u)$.

Les algorithmes MANE procèdent d'abord par une recherche en 2 étapes des K plus proches voisins (K-NN) associés au bloc d'entrée à prédire :

- La première étape consiste à chercher, à l'aide d'un algorithme de block matching classique (BM), le patch $\mathbf{a}_0$ le plus proche du bloc courant entier X (cf. figure 4.3.b), plutôt que de calculer directement les K plus proches voisins de son template comme effectué dans [TG12] (cf. figure 4.3.a).
- Dans une seconde étape, l'algorithme vient chercher les $K-1$ patches les plus proches du patch $\mathbf{a}_0$ trouvé à l'étape précédente, comme montré en figure 4.3.b. Le patch a_0 et ses $K-1$ voisins sont ensuite utilisés pour former $\mathbf{A} = [\mathbf{a}_0, ..., \mathbf{a}_{K-1}]$.

Pour le moment, la distance utilisée pour trouver les plus proches voisins est la distance Euclidienne (ou l'erreur quadratique moyenne) entre les vecteurs formés par les pixels du bloc courant avec son template et les pixels des candidats correspondants. Toutefois, dans

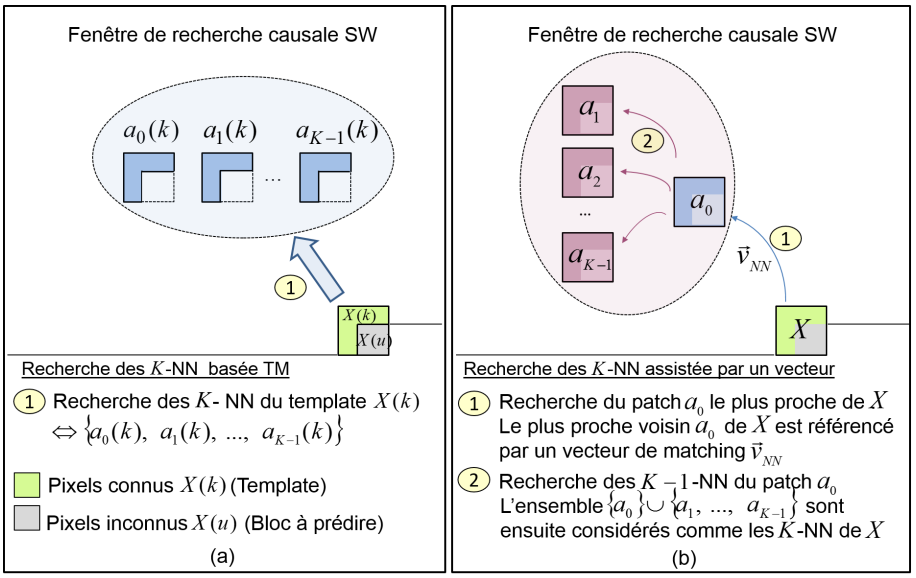

FIGURE 4.3 – *Deux méthodes de recherche des K-NN : (a) avec une distance calculée sur les pixels du template [TG12] ; (b) avec une distance calculée sur le patch entier X (méthode MANE).*

la section 4.3.2, nous étudierons également une autre mesure de distance basée sur l'erreur de prédiction embedding ainsi qu'une mesure de distance basée sur le coût débit/distorsion embedding.

Cette recherche en deux étapes permet la sélection de K-NN patches considérés ici comme les plus proches voisins du patch entier X. Le vecteur de matching résultant de l'algorithme BM nécessite d'être encodé et transmis afin que le décodeur puisse être capable de retrouver le même ensemble des K templates voisins $\mathbf{A}(k)$ à utiliser dans l'approximation linéaire du template et le même ensemble des K blocs voisins $\mathbf{A}(u)$ utilisés pour estimer le bloc courant $X(u)$. En effet, une fois que les K patches ont été déterminés, on peut procéder à la résolution du problème des moindres carrés fournie par les équations (4.9) et (4.11) lorsque l'on utilise les méthodes LLE et NMF respectivement, afin de déterminer les poids à appliquer dans l'approximation linéaire des valeurs de pixels du template. Les coefficients de pondération optimaux seront par la suite utilisés pour approximer les valeurs des pixels inconnus $X(u)$.

4.3.2 Algorithme oMANE (optimized map-aided neighbor embedding

4.3.2.1 Description de l'algorithme oMANE

Cette section décrit une extension des méthodes ci-dessus dans laquelle deux méthodes de recherche de K-NN ont été explorées : une première méthode basée sur l'erreur de prédiction "embedding" et une autre basée sur le coût débit/distorsion "embedding". L'idée de base sous-jacente de cet algorithme est de sélectionner les K-NN patches via un vecteur de matching "optimum" au sens d'un critère donné fourni par une méthode "neighbor embedding" (LLE, NMF). En d'autres termes, au lieu de travailler uniquement avec un seul patch candidat et ses $K-1$ voisins comme dans MANE, l'algorithme oMANE optimisé vient considérer plusieurs patches candidats $\mathbf{c}_0, ..., \mathbf{c}_{L-1}$ obtenus via un algorithme de Block Matching. Ces blocs et leurs voisins sont ensuite stockés dans différents dictionnaires $\mathbf{A}^0, ..., \mathbf{A}^{L-1}$. Les méthodes "neighbor embedding" (LLE ou NMF) sont ensuite appliquées

Algorithme 4.1 : Pseudocode de l'algorithme MANE

Entrées : SW, X, K
Sorties : valeurs prédites des pixels inconnus $\hat{X}_u$

1. Déterminer le patch le plus proche $\mathbf{a}_0$ du patch entier $X = [\frac{X(k)}{X(u)}]$ dans SW
2. Déterminer les $K-1$ plus proches voisins de $\mathbf{a}_0$, i.e., les patches $[\mathbf{a}_1, ..., \mathbf{a}_{K-1}]$
   - $\mathbf{A} = [\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_{K-1}]$
   - Récupérer $\mathbf{A}(k) = [\mathbf{a}_0(k), ..., \mathbf{a}_{K-1}(k)]$ et $\mathbf{A}(u) = [\mathbf{a}_0(u), ..., \mathbf{a}_{K-1}(u)]$
3. Résoudre le problème des moindres carrés sous contraintes :
   - $\hat{V} = \arg\min_V ||X(k) - \mathbf{A}(k)V||_2^2$ s.t. $\begin{cases} \sum_m V_m = 1 \text{ (LLE)} \\ \text{ou} \\ V \geq 0 \text{ (NMF)} \end{cases}$
   - $\hat{X}(u) = \mathbf{A}(u)\hat{V}$

pour chaque dictionnaire $\mathbf{A}^l$, $l = 0, ..., L-1$. Le dictionnaire conduisant au minimum d'une fonction de coût (erreur de prédiction ou coût RD) est ensuite retenu.

Soit $\mathbf{A}^l = [\frac{\mathbf{A}^l(k)}{\mathbf{A}^l(u)}]$ un dictionnaire de dimension $N \times K$ (cf. figure 4.4). Le dictionnaire $\mathbf{A}^l$ contient K patches de texture sélectionnés à partir de la fenêtre de recherche SW qui vont servir par la suite à calculer les coefficients de pondération pour l'approximation linéaire du template $X(k)$ et à fournir une prédiction possible pour le bloc courant. Les sous-matrices $\mathbf{A}^l(k)$ et $\mathbf{A}^l(u)$ comprennent respectivement les valeurs de pixels colocalisés aux pixels du template et aux pixels du bloc à prédire.

L'algorithme optimized map-aided neighbor embedding (oMANE) procède à la recherche des K-NN de la manière suivante :

1) Dans un premier temps, on vient chercher dans la fenêtre de recherche SW les L plus proches voisins du patch entier X, contenant le template $X(k)$ et le bloc original $X(u)$ (qui est connu uniquement à l'encodeur) comme présenté en figure 4.4. Ces L voisins $[\mathbf{c}_0, ..., \mathbf{c}_{L-1}]$ sont proches du patch X au sens de la distance euclidienne. Chaque voisin $\mathbf{c}_l, l = 0, ..., L-1$ est référencé par un vecteur de matching $\vec{v}_l$.

2) Un dictionnaire $\mathbf{A}^l$ est ensuite défini à partir de chaque voisin $\mathbf{c}_l$ de X. En effet, pour chaque vecteur de matching candidat $\vec{v}_l$, l'algorithme identifie les $K-1$ patches $[\mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l]$ dans la fenêtre de recherche SW qui sont proches du patch $\mathbf{c}_l$ au sens de la distance euclidienne. Ainsi, le dictionnaire $\mathbf{A}^l$ contient le patch $\mathbf{c}_l$ pointé par le vecteur $\vec{v}_l$ et les $K-1$ plus proches voisins du patch $\mathbf{c}_l$, i.e., $\mathbf{A}^l = [\mathbf{c}_l, \mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l]$. La figure 4.4 illustre la procédure de construction d'un dictionnaire $\mathbf{A}^l$ à partir d'un voisin $\mathbf{c}_l$ du patch X. Ensuite, on cherche une approximation du template $X(k)$ par une combinaison linéaire de K-NN templates identifiés dans les sous-matrices $\mathbf{A}^l(k)$. Pour chaque vecteur de matching candidat $\vec{v}_l$, on vient résoudre le problème d'approximation suivant :

$$\min_{V^l} ||X(k) - \mathbf{A}^l(k)V^l||_2^2 \text{ s.t. } \begin{cases} \sum_m V_m^l = 1 \text{ (LLE)} \\ \text{ou} \\ V^l \geq 0 \text{ (NMF)} \end{cases} \quad (4.12)$$

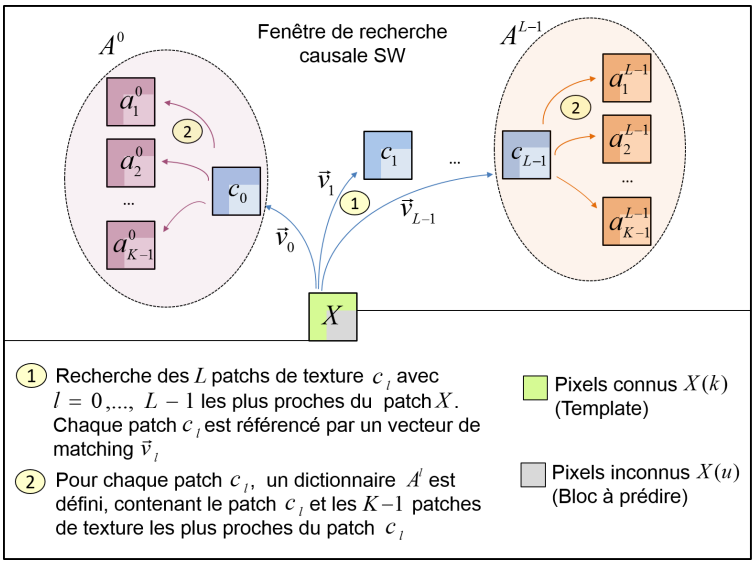

FIGURE 4.4 – *Procédure de construction d'un dictionnaire A^l à partir d'un voisin c_l du patch X (méthode oMANE).*

où V^l représente les coefficients de pondération des templates sélectionnés $\mathbf{A}^l(k)$. A l'encodeur, l'algorithme itère le processus jusqu'à ce que l'ensemble des vecteurs de matching candidats aient été testés, c'est à dire, jusqu'à ce qu'on atteigne le nombre maximum L d'itérations permises, conduisant ainsi à L vecteurs de pondérations possibles $\hat{V}^l, l = 0, ..., L-1$, chacun étant produit à chaque itération de l'algorithme.

3) Dans l'optique d'optimiser la prédiction selon un critère donné, on garde la trace des coefficients de pondération $\hat{V}^l, l = 0, ..., L-1$ générés par la méthode "neighbor embedding" sélectionné (LLE, NMF) à l'aide d'un vecteur de matching donné $\vec{v}_l$ à chaque itération de l'algorithme. Puis, le choix du vecteur de pondération "optimum" $\hat{V}^{l_{opt}}$ (ou le vecteur de matching "optimum" $\vec{v}_{l_{opt}}$) s'effectue par le biais d'un critère de décision donné. Les critères explorés ici pour déterminer le vecteur de matching "optimum" $\vec{v}_{l_{opt}}$ lors du calcul de l'approximation LLE ou NMF sont les suivants :

- Minimisation de la somme des erreurs aux carrés (SSE) dans la zone à prédire $X(u)$ afin d'obtenir la meilleure prédiction $\hat{X}(u)$, i.e.,

$$\arg\min_{0 \leq l \leq L-1} ||X(u) - \mathbf{A}^l(u)\hat{V}^l||_2^2 \tag{4.13}$$

- Minimisation lagrangienne d'une fonction de coût débit/distorsion afin de choisir la prédiction $\hat{X}(u)$ qui fournit le meilleur compromis entre la qualité du bloc reconstruit (ou décodé) et le coût de codage de l'erreur de prédiction, i.e.,

$$\arg\min_{0 \leq l \leq L-1} (D^l + \lambda R^l) \tag{4.14}$$

où D^l représente la distorsion entre le bloc original et le bloc reconstruit en utilisant la métrique de distance SSE, et R^l correspond au coût réel de codage. Ce coût inclut non seulement le coût de codage relatif à l'erreur résiduelle et le coût de la signalisation des modes de codage mais également le coût de codage du vecteur utilisé, comme nous le verrons plus en détail dans la section 4.3.3.

Le vecteur $\vec{v}_{l_{opt}}$ est ensuite transmis au décodeur afin d'utiliser les mêmes voisins pour calculer l'approximation LLE ou NMF. Le signal prédit final $\hat{X}(u)$ est ensuite obtenu en multipliant le dictionnaire $\mathbf{A}^{l_{opt}}(u)$ par $\hat{V}^{l_{opt}}$ comme $\hat{X}(u) = \mathbf{A}^{l_{opt}}(u)\hat{V}^{l_{opt}}$. L'algorithme 4.2 résume l'approche oMANE.

Algorithme 4.2 : Pseudocode de l'algorithme oMANE.

Entrées : SW, X, K, L
Sorties : valeurs prédites des pixels inconnus $\hat{X}(u)$

1. Déterminer les L plus proches voisins du patch entier $X = [\frac{X(k)}{X(u)}]$, i.e. les patches $[\mathbf{c}_0, ..., \mathbf{c}_{L-1}]$ dans SW t.q. $d_0 \leq ... \leq d_{L-1}$ avec $d_i = ||X - c_i||_2^2$

2. Initialisation : $l = 0$
   do until $l = L - 1$:
   - Déterminer les $K-1$ plus proches voisins de $\mathbf{c}_l$, i.e., les patches $\left[\mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l\right]$
   - $\mathbf{A}^l = \left[\mathbf{c}_l, \mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l\right]$
   - Récupérer $\mathbf{A}^l(k) = \left[\mathbf{a}_0^l(k), ..., \mathbf{a}_{K-1}^l(k)\right]$ de $\mathbf{A}^l$
   - Récupérer $\mathbf{A}^l(u) = \left[\mathbf{a}_0^l(u), ..., \mathbf{a}_{K-1}^l(u)\right]$ de $\mathbf{A}^l$
   - Résoudre le problème des moindres carrés sous contraintes :
     - $\hat{V}^l = \arg\min_{V^l} ||X(k) - \mathbf{A}^l(k)V^l||_2^2$ t.q. $\begin{cases} \sum_m V_m^l = 1 \text{ (LLE)} \\ \text{ou} \\ V^l \geq 0 \text{ (NMF)} \end{cases}$
     - $\hat{X}(u)^l = \mathbf{A}^l(u)\hat{V}^l$

   end do

3. Sélectionner l'optimum l_{opt} minimisant le critère choisi
   Mettre $\hat{X}(u) = \mathbf{A}^{l_{opt}}(u)\hat{V}^{l_{opt}}$

4.3.2.2 Dictionnaire réduit

Le processus de recherche des K-NN dans l'algorithme oMANE présente une complexité de calcul importante. Cette complexité est d'autant plus accrue que la fenêtre de recherche SW est grande. Il est néanmoins intéressant de noter que pour le décodeur, la complexité de l'algorithme oMANE est la même que l'algorithme MANE.

Afin d'accélérer le processus de recherche des $K - NN$ dans cet algorithme, nous allons considérer uniquement un sous ensemble de patches issus de la fenêtre de recherche SW. Ce sous ensemble de patches va former ce que l'on appelle un dictionnaire réduit. Soit $S = [\frac{S(k)}{S(u)}]$ désignant un dictionnaire contenant M' patches sélectionnés à partir de la fenêtre de recherche SW. Les patches $s_0, ..., s_{M'-1}$ sont sélectionnés en minimisant la distance euclidienne uniquement sur le template du bloc à prédire afin que le décodeur puisse être capable de reconstruire le même dictionnaire sans aucune signalisation supplémentaire. Ainsi, une fois que le dictionnaire S est construit, la méthode oMANE basée LLE ou NMF peut être appliquée directement sur le dictionnaire pré-défini S. Les résultats expérimentaux, décrits dans la section 4.3.4, montreront qu'utiliser un dictionnaire de taille réduite permet également d'améliorer l'algorithme en terme d'efficacité de codage, en réduisant le coût de codage d'un vecteur de matching.

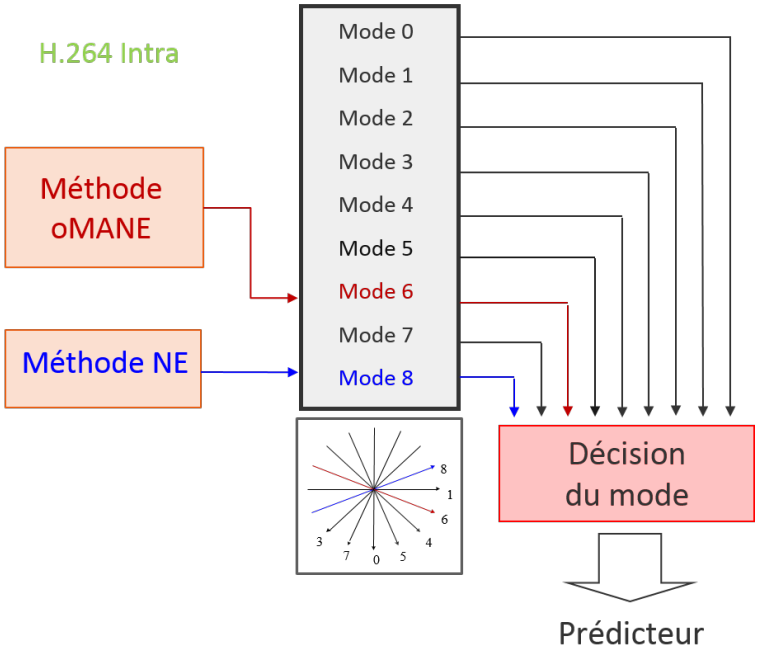

Figure 4.5 – *Synoptique du procédé de sélection des modes de prédiction intra basés NE et oMANE pour des blocs 4x4 ou 8x8.*

4.3.3 Algorithme de compression d'image (structure de l'encodeur)

La méthode de prédiction intra proposée consiste à utiliser conjointement un algorithme NE donné (tel que la LLE ou la NMF opérant uniquement sur le template) et un des algorithmes proposés : MANE ou oMANE (fonctionnant sur le patch entier). Dans cette section, la méthode oMANE sera considérée pour décrire l'algorithme de compression proposé, sachant que l'on procède de la même manière lorsque la méthode MANE est appliquée.

Les méthodes NE et oMANE sont donc utilisées ici comme deux modes concurrentiels, en plus des autres modes de prédiction intra H.264. L'idée est d'utiliser la prédiction basée oMANE dans les zones de l'image où la prédiction basée NE échouerait, ceci étant naturellement effectué grâce au mécanisme de décision basé Rate-Distorsion (RD). Les modes de prédiction intra basés NE et oMANE peuvent être vus comme deux méthodes complémentaires. En effet, la méthode de prédiction basée NE est efficace lorsque le bloc inconnu à prédire et son template sont fortement corrélés, c'est à dire, que les K-NN patches obtenus à partir du calcul des distances sur les pixels du template correspondent aux K-NN du bloc courant à prédire. Par ailleurs, la méthode NE ne nécessite aucune transmission de vecteur supplémentaire. Cependant, lorsque le bloc courant et son template sont très peu corrélés, la méthode de prédiction basé oMANE peut mener à une meilleure prédiction pour le bloc inconnu à prédire puisque la recherche des K-NN s'effectue en considérant le patch entier (i.e. le bloc courant et son template). Cependant, l'utilisation de ce mode de prédiction requiert l'envoi au décodeur d'un vecteur de correspondance entre le patch entier X et le voisin sélectionné pour déterminer les K-NN patches utilisés pour approximer le template (voir figure 4.4). Notons que l'utilisation simultanée des modes de prédiction NE et oMANE sera appelée par la suite méthode "hybride NE/oMANE".

Afin de s'affranchir de modifications fastidieuses, les modes de prédiction intra basés

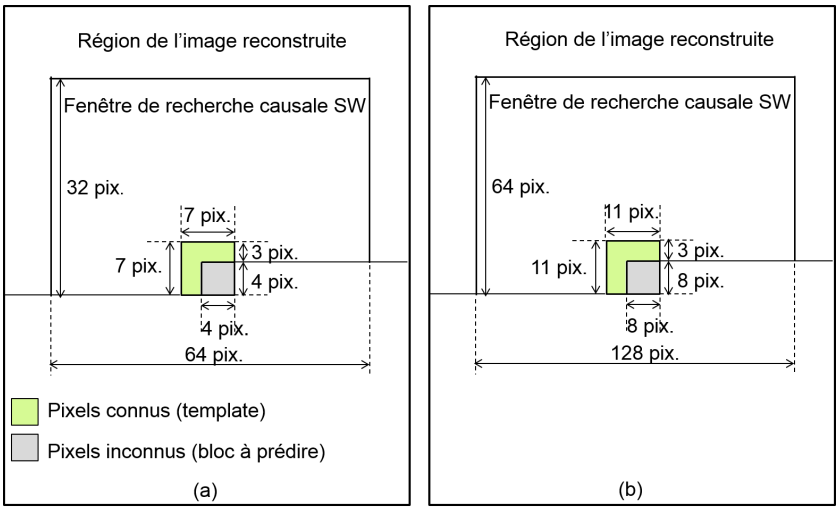

FIGURE 4.6 – *Configuration de la fenêtre de recherche et du template pour les blocs* 4*x*4 *(a) et les blocs* 8*x*8 *(b).*

NE et oMANE ont été intégrés dans le codeur H.264 (software de référence KTA [sof]) en remplaçant deux modes existants de l'algorithme H.264 comme décrit dans la figure 4.5. En effet, les deux modes de prédiction intra, qui ont été remplacés par les modes NE et oMANE, correspondent aux modes les moins utilisés statistiquement en H.264 Intra. Ceci facilite l'implémentation en conservant la même syntaxe et permet ainsi d'économiser des bits de signalisation. Dans le schéma de codage proposé, l'approche hybride NE/oMANE est implémentée en Intra-4x4 et Intra-8x8 et les vecteurs de matching sont limités à la précision pixellique. La sélection du meilleur mode de codage pour un bloc donné est ensuite opérée selon une minimisation d'un critère lagrangien, également appelé optimisation d'un critère débit/distorsion (RDO) :

$$\min\left(D + \lambda \times R\right) \tag{4.15}$$

où D représente la distorsion entre le bloc original et le bloc reconstruit en utilisant la métrique de distance SSE (Sum of Square Errors), λ est un coefficient permettant d'obtenir le meilleur compromis entre la distorsion et les bits nécessaires pour le codage du bloc. Le paramètre λ est identique à celui utilisé dans le software de référence KTA [sof] ($\lambda = 0.65 \times 2^{QP/3}$), avec QP correspondant au paramètre de quantification. Concernant le mode de codage intra basé sur la méthode oMANE, le coût débit R est calculé selon la formule suivante :

$$R = R_{mode} + R_{res} + R_{mvect} \tag{4.16}$$

où R_{mode}, R_{res} et R_{mvect} représentent les bits nécessaires pour encoder respectivement le mode de codage du bloc, le résidu et le vecteur de matching. Notons que R_{mode} et R_{res} sont encodés avec le schéma de codage entropique KTA (CABAC). Rappelons que le vecteur de matching correspond ici au vecteur retenu pour déterminer les K-NN patches utilisés pour réaliser une approximation basée NE sur le template courant et prédire le bloc à coder. Par ailleurs, le meilleur vecteur de correspondance est obtenu par le biais d'un critère de sélection basé SSE ou RDO. Dans les expériences reportées dans ce document, deux manières différentes ont néanmoins été considérées pour encoder le vecteur de matching :

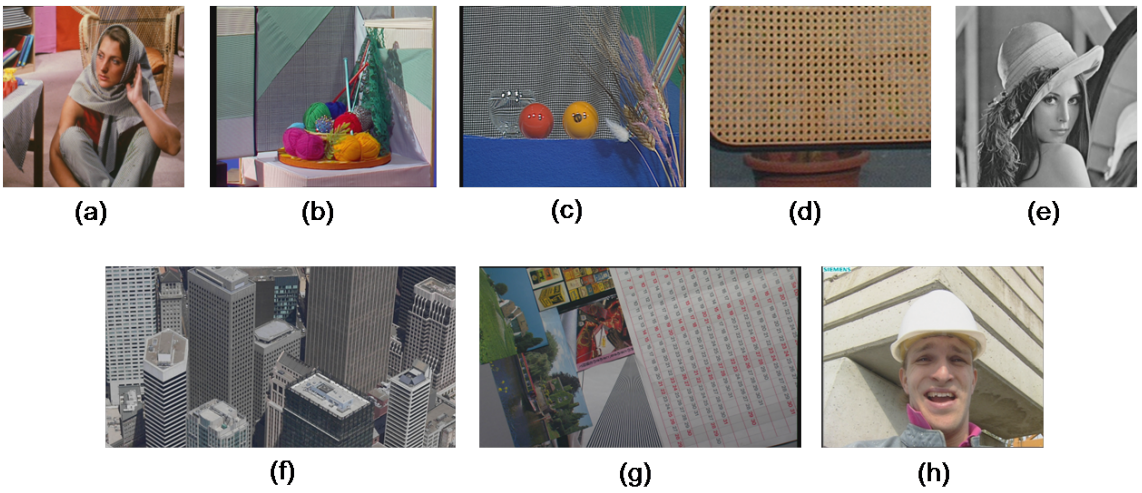

FIGURE 4.7 – *Les images test : (a) Barbara (512x512), (b) Wool (720x576), (c) Snook (720x576), (d) Pan0_qcif, (e) Lena (512x512), (f) Building (896x464), (g) Spincalendar (720p) and (h) Foreman (CIF).*

- Si l'algorithme oMANE travaille sur l'intégralité de la fenêtre de recherche causal SW, le vecteur de matching est l'index du patch (référencé par le vecteur de matching) dans le dictionnaire contenant l'ensemble des patches possibles contenus dans la fenêtre de recherche causal SW. Ainsi, l'index est encodé en utilisant un codeur à longueur fixe (FLC) qui varie en fonction du nombre de patches possibles dans SW.
- Si l'algorithme oMANE opère uniquement sur un dictionnaire réduit pré-défini (comme décrit dans la sous-section 4.3.2.2), c'est l'index du patch situé dans le dictionnaire réduit qui est encodé en utilisant également un codeur FLC qui varie selon la taille du dictionnaire.

A noter que pour le mode de codage intra basé sur la méthode NE, l'équation (4.16) est également utilisée avec $R_{mvect} = 0$.

4.3.4 Résultats expérimentaux

4.3.4.1 Performances en termes de débit/distorsion avec les différents algorithmes

Trois schémas de codage : l'algorithme basé NE [TG12], les algorithmes hybrides basés respectivement NE/MANE et NE/oMANE ont été évalués comparativement aux modes de prédiction Intra H.264 (High Profile). Notons que chaque méthode de prédiction hybride explorée est intégrée dans le codeur H.264. Par ailleurs, le mode de prédiction NE se réfère à une des techniques "neighbor embedding" actuelles (LLE, NMF) utilisées pour la prédiction intra image [TG12]. Les modes de prédiction MANE et oMANE, introduits précédemment, sont basés sur les techniques 'neighbor embedding" dans lesquelles la recherche des K-NN est assistée par un vecteur de matching. Pour la méthode basée MANE, le vecteur de matching correspond au patch le plus proche du bloc courant avec son template, tandis que pour la méthode basée oMANE, le vecteur de matching correspond à celui qui minimise un critère SSE ou RDO sur le bloc à prédire.

Les images test qui ont été considérées dans les expérimentations sont présentées dans

Bas débits	NE		NE + MANE		NE + oMANE	
QP=26,31,36,41	dB	% débit	dB	% débit	dB	% débit
Barbara	0.72	-10.49	0.76	-11.10	1.38	-19.30
Wool	1.55	-23.25	1.64	-24.36	2.10	-30.04
Snook	2.05	-28.02	2.10	-28.66	2.86	-37.15
Pan0_qcif	2.24	-33.19	2.38	-35.21	3.03	-43.56
Building	2.99	-34.76	3.14	-36.19	4.09	-44.75
Lena	0.06	-1.49	0.01	-0.26	0.11	-2.55
Spincalendar	0.81	-14.55	0.87	-15.56	1.47	-25.08
Foreman	0.26	-5.10	0.25	-4.89	0.60	-11.47
Moyenne	**1.34**	**-18.86**	**1.39**	**-19.53**	**1.96**	**-26.74**

Table 4.1 – *Gains en débit-distorsion (avec la mesure Bjontergaard et utilisant la LLE comme méthode NE) par rapport aux modes de prédiction intra H.264 à bas débits (calculés à partir de 4 mesures de débit : QP=26,31,36,41).*

Bas débits	NE		NE + MANE		NE + oMANE	
QP=26,31,36,41	dB	% débit	dB	% débit	dB	% débit
Barbara	1.06	-15.12	1.14	-16.27	1.54	-21.32
Wool	1.80	-26.22	1.92	-27.83	2.15	-30.54
Snook	2.69	-35.58	2.71	-35.74	3.02	-38.59
Pan0_qcif	2.79	-40.88	2.87	-41.74	3.08	-43.67
Building	3.19	-36.94	3.30	-38.06	3.66	-41.29
Lena	0.13	-3.15	0.12	-2.82	0.18	-4.27
Spincalendar	0.9	-16.12	0.97	-17.20	1.25	-21.67
Foreman	0.51	-9.68	0.57	-10.91	0.70	-13.23
Moyenne	**1.63**	**-22.96**	**1.70**	**-23.82**	**1.95**	**-26.82**

Table 4.2 – *Gains en débit-distorsion (utilisant la NMF comme méthode NE) par rapport aux modes de prédiction intra H.264 à bas débits.*

la figure 4.7. Les résultats obtenus avec la méthode NE et les méthodes hybrides proposées NE/MANE et NE/oMANE basées LLE sont fournis dans les tableaux 4.1 et 4.3 à bas et haut débit respectivement, l'efficacité de codage étant comparée en utilisant la métrique de Bjontegaard [Bjo01]. Ceux obtenus en considérant la NMF comme méthode NE sont également reportés dans les tableaux 4.2 et 4.4. Ici, nous avons considéré une fenêtre de recherche de taille 64x128 et un template de taille 3x11 (profondeur x longueur) pour des blocs 8x8 tandis que nous avons utilisé une fenêtre de recherche de taille 32x64 et un template de taille 3x7 pour les blocs 4x4. La figure 4.6 montre la configuration de la fenêtre de recherche et celle du template qui sont utilisées dans les simulations pour prédire les blocs de tailles 4x4 et 8x8 pixels. De plus, le nombre K de voisins utilisés pour approximer le template est réglé à 20 (pour les méthodes NE, MANE et oMANE) et le nombre de vecteurs candidats testés, dans la méthode de prédiction oMANE, est fixé à 100, le vecteur de matching étant sélectionné selon un critère RDO. Par ailleurs, un dictionnaire réduit S de 512 patches vectorisés, est construit avant d'appliquer la méthode oMANE comme expliqué dans la sous-section 4.3.2.2. Nous montrerons plus tard qu'utiliser conjointement un critère RDO et un dictionnaire S de taille réduite, dans la méthode oMANE, permet

Hauts débits	NE		NE + MANE		NE + oMANE	
QP=16,21,26,31	dB	% débit	dB	% débit	dB	% débit
Barbara	0.73	-8.61	0.76	-8.91	1.27	-14.72
Wool	1.19	-13.06	1.22	-13.45	1.62	-17.60
Snook	1.84	-15.94	1.86	-16.17	2.59	-21.90
Pan0_qcif	1.89	-17.55	1.99	-18.53	2.58	-23.40
Building	3.09	-24.80	3.20	-25.65	3.96	-31.07
Lena	0.05	-0.80	0.02	-0.27	0.11	-1.77
Spincalendar	0.72	-9.01	0.77	-9.67	1.15	-14.25
Foreman	0.19	-2.80	0.14	-2.10	0.36	-5.35
Moyenne	**1.21**	**-11.57**	**1.25**	**-11.84**	**1.71**	**-16.26**

Table 4.3 – *Gains en débit-distorsion (avec la mesure Bjontergaard et utilisant la LLE comme méthode NE) par rapport aux modes de prédiction intra H.264 à hauts débits (calculés à partir de 4 mesures de débit : QP=16,21,26,31).*

Hauts débits	NE		NE + MANE		NE + oMANE	
QP=16,21,26,31	dB	% débit	dB	% débit	dB	% débit
Barbara	0.84	-10.05	0.87	-10.47	1.23	-14.40
Wool	1.39	-15.24	1.41	-15.43	1.66	-17.87
Snook	2.23	-19.47	2.24	-19.58	2.66	-22.67
Pan0_qcif	2.18	-20.60	2.22	-21.12	2.55	-23.54
Building	3.06	-24.74	3.18	-25.61	3.58	-28.28
Lena	0.12	-1.95	0.11	-1.70	0.19	-3.03
Spincalendar	0.79	-9.80	0.85	-10.51	1.10	-13.44
Foreman	0.33	-4.78	0.32	-4.79	0.44	-6.42
Moyenne	**1.37**	**-13.33**	**1.40**	**-13.65**	**1.68**	**-16.21**

Table 4.4 – *Gains en débit-distorsion (utilisant la NMF comme méthode NE) par rapport aux modes de prédiction intra H.264 à hauts débits.*

d'améliorer nettement les performances d'encodage de l'algorithme hybride NE/oMANE (Tableau 4.5).

Comparés à H.264 Intra, les gains BD-rate moyens obtenus avec la méthode NE basée LLE, sont de 18.86% à bas débits et 11.57% à hauts débits. La méthode hybride NE/MANE permet quant à elle d'atteindre en moyenne une réduction de débit de 19.53% et 11.84% à bas et hauts débits respectivement, alors que la méthode hybride NE/oMANE permet de réaliser en moyenne une réduction de débit moyen de 26.82% et 16.21% à bas et hauts débits respectivement. Par ailleurs, lorsque la méthode NE basée NMF est utilisée, les pourcentages de gains moyens obtenus avec la méthode NE sont de 22.96% à bas débits et 13.33% à hauts débits. Ceux obtenus avec la méthode hybride NE/MANE sont de 23.82% et 13.65% à bas et hauts débits respectivement, tandis que ceux obtenus avec la méthode hybride NE/oMANE sont de 26.82% et 16.21% à bas et hauts débits respectivement.

La première série d'images (à partir de Barbara à Building) est partiellement composée de textures pseudo-régulières qui conviennent assez bien aux différents algorithmes testés. Par ailleurs, nous obtenons des résultats intéressants avec des images plus conventionnelles

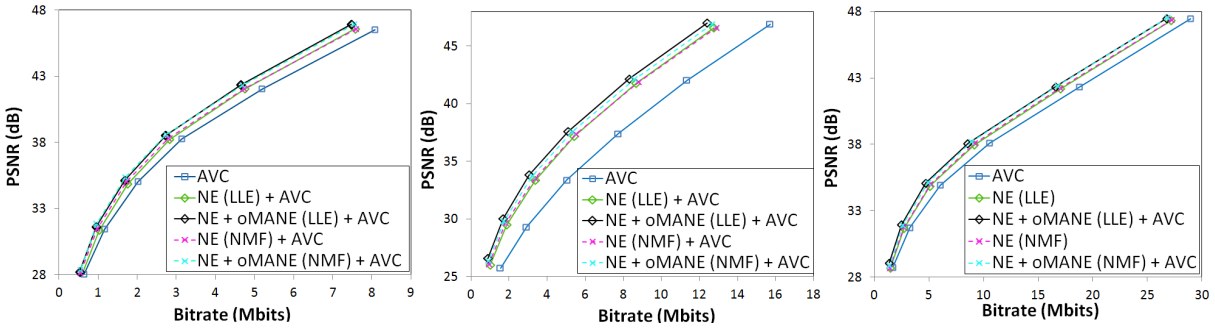

FIGURE 4.8 – *Evaluation des performances RD des images Barbara (à gauche), Building (au milieu) et Spincalendar (à droite) avec les méthodes conventionnelles basées NE (LLE, NMF), et les méthodes de prédiction hybrides basées NE/oMANE (chaque méthode de prédiction étant insérée dans le codeur H.264).*

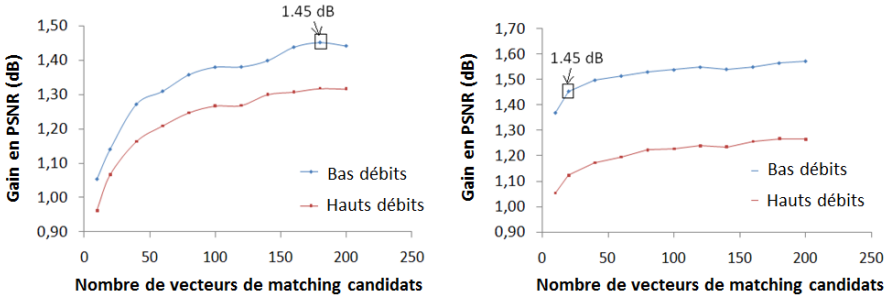

FIGURE 4.9 – *Gains en PSNR versus le nombre L de vecteurs de matching candidats pour l'image Barbara, avec la méthode de prédiction hybride NE/oMANE basée LLE (à gauche) et NMF (à droite).*

comme Foreman (CIF) et Spincalendar (HD) à bas et haut débit. Cependant, les pourcentages de gains obtenus avec Lena sont moins significatifs étant donné que les modes de prédiction Intra H.264 sont déjà très efficaces pour cette image.

Suite aux simulations réalisées sur l'ensemble des images test, nous pouvons remarquer que la méthode hybride NE/MANE basée LLE permet d'apporter des gains significatifs par rapport à H.264 Intra, mais des améliorations légères par rapport à la méthode NE basée LLE seule. La même tendance est également observée avec la NMF. De plus, les résultats expérimentaux montrent que la méthode hybride NE/oMANE surpasse les modes de prédiction intra H.264 et les techniques conventionnelles basées NE (LLE, NMF) en termes d'efficacité de codage pour divers contenus de résolutions différentes. Cependant, nous pouvons constater que les gains de codage sont moins importants pour l'image Foreman puisque les modes de prédiction tant directionnels que de type DC de H.264 s'avèrent assez bien adaptés pour ce type de contenu avec des grandes surfaces planes. Par ailleurs, nous pouvons observer que les gains de codage sont importants pour les images, telles que Pan0_qcif et Building qui présentent des propriétés stationnaires locales en termes de texture. En effet, un gain en débit jusqu'à 43.56% et 44.75% respectivement est réalisé à bas débits par rapport à H.264 Intra (cf. Tableau 4.1).

Pour $L = 100$, les résultats expérimentaux montrent que les gains de codage obtenus avec l'algorithme hybride basé NE/oMANE ne varient pas significativement entre les deux

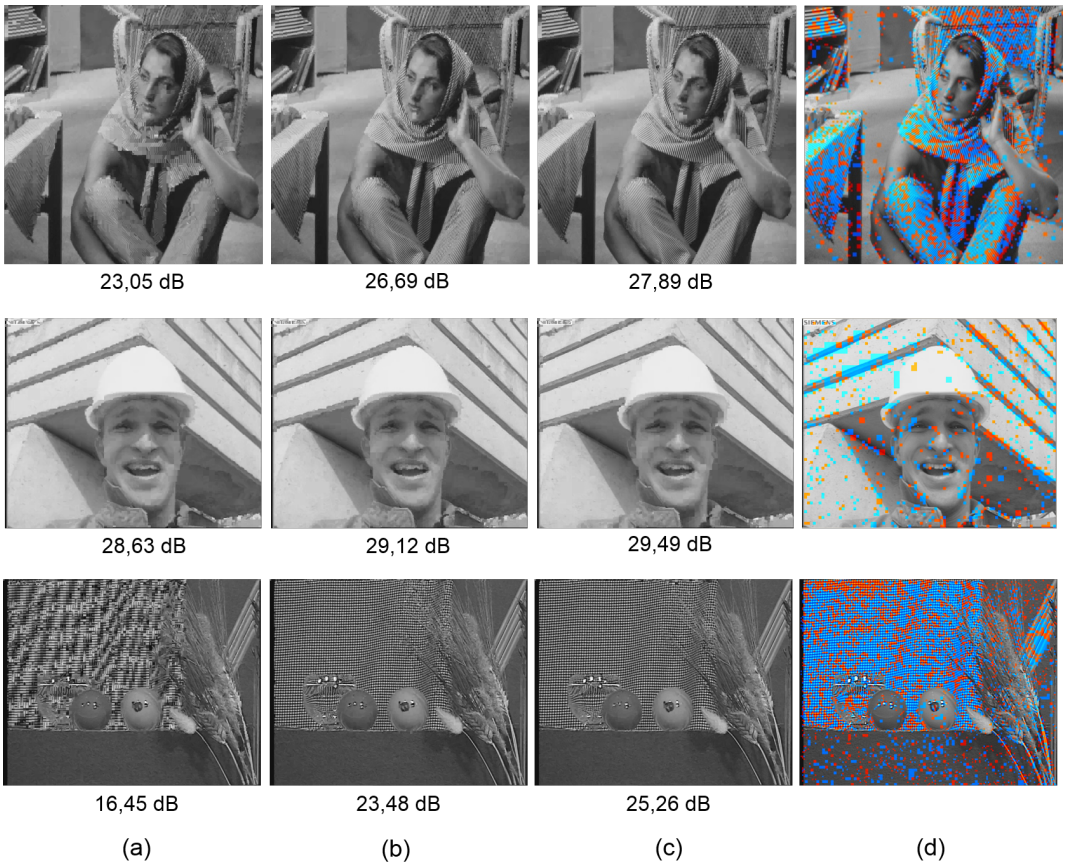

Figure 4.10 – *Résultats de prédiction spatiale pour les images Barbara, Foreman et Snook utilisant des tailles de bloc 4x4 et 8x8 dynamiquement choisies selon un critère RD (à QP=16) avec (a) les modes Intra H.264 (High Profile), (b) la méthode NE, (c) la méthode NE/oMANE et (d) sa carte de prédiction associée : NE (rouge) et oMANE (bleu).*

méthodes NE étudiées (LLE et NMF) comme présenté en figure 4.8. Par ailleurs, parmi ces deux méthodes NE, nous constatons que la NMF est celle qui requiert le plus de temps de calcul puisque cette dernière utilise une méthode de mise à jour multiplicative itérative. La figure 4.9 montre l'évolution des gains en PSNR (à bas et hauts débits) obtenus avec les méthodes hybrides NE/oMANE (LLE, NMF) par rapport à H.264 lorsque l'on fait varier le nombre L de vecteurs de matching pour l'image Barbara. A l'encodeur, nous pouvons souligner que pour atteindre un gain de 1.45 dB, la méthode hybride utilisant la LLE nécessite de tester 180 vecteurs de matching tandis que celle opérant avec la NMF requiert uniquement le traitement de 20 vecteurs de matching. Cependant, en termes de temps CPU, la méthode NE/oMANE utilisant la NMF nécessite presque cinq fois plus de CPU que celle utilisant la LLE. Par conséquent, nous avons choisi d'étudier pour la suite uniquement la méthode hybride NE/oMANE basée LLE.

4.3.4.2 Performances en termes de prédiction

Les performances de prédiction sont présentées pour les images Barbara, Foreman et Snook dans la figure 4.10 lorsque l'erreur de prédiction a été encodée avec un QP=16. Ces figures montrent également les zones de l'image où les modes de prédiction NE et oMANE basés LLE sont sélectionnés. On peut constater que la qualité du signal prédit, à

la fois visuellement et en termes de PSNR, est significativement améliorée par l'algorithme hybride proposé par rapport à H.264 Intra et à la méthode NE classique, en particulier pour les images Barbara et Snook qui contiennent des textures complexes. En effet, on peut observer que la méthode oMANE est beaucoup plus sélectionnée que la méthode NE principalement dans les régions de texture qui sont localement variables dû à la déformation spatiale du tissu (Barbara) ou de l'arrière-plan de l'image (Snook).

4.3.4.3 Critère de sélection du vecteur de matching « optimum »

Cette sous-section compare brièvement les performances d'encodage obtenues avec l'algorithme hybride NE/oMANE (basé LLE) lorsque le vecteur de matching optimum (dans l'algorithme oMANE) est sélectionné selon deux critères différents : (1) minimisation du résidu de prédiction SSE ; (2) minimisation du coût débit-distorsion RDO. A noter qu'ici l'algorithme opère avec un dictionnaire contenant l'ensemble des patches disponibles dans la fenêtre de recherche causale SW. Les tableaux 4.5 et 4.6 montrent que le critère de sélection du vecteur de matching optimum basé RDO permet d'atteindre de meilleurs résultats que celui basé SSE. En effet, comparés à H.264 Intra, les gains BD-rate moyens obtenus lorsque l'on utilise le critère SSE sont de 22.88% en bas débit et de 13.03% en haut débit tandis que ceux obtenus lorsque l'on applique le critère RDO sont de 24.43% en bas débit et de 14.55% en haut débit. Cela s'explique par le fait que la meilleure prédiction ne mène pas nécessairement à la meilleure efficacité de codage. Nous retiendrons donc pour l'algorithme oMANE le critère de sélection basé RDO.

4.3.4.4 Dictionnaire réduit / fenêtre de recherche causale

Cette sous-section a pour but d'analyser l'impact d'utilisation d'un dictionnaire de taille réduite dans l'algorithme hybride NE/oMANE (basé LLE). Notons que le critère de sélection du vecteur de matching optimum appliqué dans la méthode oMANE est basé ici sur un critère RDO.

Tout d'abord, nous avons comparé les performances RD entre un algorithme hybride NE/oMANE dans lequel la méthode oMANE considère un dictionnaire contenant l'en-

	NE + oMANE (basée LLE)					
Bas débits	SSE / SW		RDO / SW		RDO / Dico S	
QP :26,31,36,41	dB	% débit	dB	% débit	dB	% débit
Barbara	0.98	-14.09	1.11	-15.91	1.38	-19.30
Wool	1.82	-26.86	1.93	-28.09	2.10	-30.04
Snook	2.32	-31.25	2.56	-33.82	2.86	-37.15
Pan0_qcif	2.56	-38.09	2.76	-40.09	3.03	-43.56
Building	3.61	-40.78	3.80	-42.20	4.09	-44.75
Lena	0.05	-1.24	0.07	-1.61	0.11	-2.55
Spincalendar	1.26	-22.07	1.40	-24.15	1.47	-25.08
Foreman	0.45	-8.67	0.50	-9,58	0.60	-11.47
Moyenne	**1.63**	**-22.88**	**1.77**	**-24.43**	**1.96**	**-26.74**

Table 4.5 – *Comparaison entre différentes méthodes de prédiction oMANE dans lesquelles le vecteur de matching est sélectionné selon deux critères différents : SSE et RDO, et dans lesquelles deux types de dictionnaire sont considérés : un dictionnaire contenant tous les patches disponibles dans la fenêtre de recherche causale SW et un dictionnaire réduit pré-défini S à* ***bas débits****.*

	NE + oMANE (basée LLE)					
Haut débits	SSE / SW		RDO / SW		RDO / Dico S	
QP :16,21,26,31	dB	% débit	dB	% débit	dB	% débit
Barbara	0.85	-10.08	1.04	-12.15	1.27	-14.72
Wool	1.30	-14.38	1.46	-15.87	1.62	-17.60
Snook	2.01	-17.42	2.26	-19.38	2.59	-21.90
Pan0_qcif	2.05	-19.17	2.30	-21.19	2.58	-23.40
Building	3.42	-27.59	3.68	-29.07	3.96	-31.07
Lena	0.03	-0.52	0.07	-1.12	0.11	-1.77
Spincalendar	0.93	-11.85	1.07	-13.43	1.15	-14.25
Foreman	0.22	-3.23	0.28	-4.21	0.36	-5.35
Moyenne	**1.35**	**-13.03**	**1.52**	**-14.55**	**1.71**	**-16.26**

Table 4.6 – *Comparaison de plusieurs méthodes de prédiction oMANE dans lesquelles le vecteur de matching est sélectionné selon deux critères différents : SSE et RDO, et dans lesquelles deux types de dictionnaire sont considérés : un dictionnaire contenant tous les patches disponibles dans la fenêtre de recherche causale SW et un dictionnaire réduit pré-défini S à* ***hauts débits****.*

semble des patches disponibles dans la fenêtre causale SW et un algorithme hybride NE/oMANE dans lequel la méthode oMANE utilise un dictionnaire pré-défini S composé uniquement d'un sous ensemble de patches issus de SW. Les tableaux 4.5 et 4.6 montrent que l'utilisation d'un dictionnaire pré-défini S permet d'améliorer davantage les performances d'encodage de l'algorithme hybride NE/oMANE puisque le coût de codage lié aux vecteurs de matching (dans la méthode oMANE) a été réduit. En effet, comparés à H.264 Intra, les gains BD-rate moyens obtenus lorsque l'on utilise la fenêtre de recherche complète SW sont de 24.43% à bas débits et de 14.55% à hauts débits tandis que ceux obtenus lorsque l'on considère un dictionnaire de taille réduit S sont de 26.74% à bas débits et de 16.26% à hauts débits. Par ailleurs, les résultats expérimentaux tendent à montrer que l'approche hybride utilisant un dictionnaire réduit permet de réduire en moyenne le temps d'exécution de 80% à l'encodeur par rapport à l'approche hybride considérant la fenêtre de recherche complète.

La figure 4.11 indique les gains en PSNR (à bas et hauts débits) obtenus pour l'image Barbara lorsque l'on fait varier le nombre de patches définis dans le dictionnaire S ($L \in [32, 1024]$ pour les blocs 8x8 et $L \in [16, 512]$ pour les blocs 4x4) et en fixant la taille de la fenêtre de recherche SW à 64x128 pixels pour les blocs 8x8 et à 32x64 pixels pour les blocs 4x4. Dans la figure 4.11, nous pouvons observer qu'une valeur optimum de la taille du dictionnaire S est déterminée. En effet, des gains jusqu'à 1.46 dB à bas débits et 1.35 dB à hauts débits ont été atteints lorsque l'on utilise un dictionnaire S de 128 patches pour des blocs 8x8 et de 64 patches pour des blocs 4x4. Cependant, nous pouvons remarquer que les gains en PSNR diminuent lorsque la taille du dictionnaire S est trop petite. L'explication réside dans le fait que lorsque peu de patches sont considérés, le dictionnaire S ne contient pas assez de patches significatifs en termes de texture afin que la méthode oMANE puisse bien approximer le bloc courant. Dans ce cas, l'algorithme oMANE tend à avoir le même comportement que l'algorithme NE. Par ailleurs, on peut souligner que les gains en PSNR diminuent lorsque la taille du dictionnaire S devient trop importante. Ceci s'explique par le fait que plus la taille du dictionnaire augmente, plus le coût de codage lié aux vecteurs de matching est élevé. Ainsi, les performances d'encodage de l'algorithme hybride NE/oMANE tendent progressivement à approcher les résultats obtenus lorsque l'on considère la fenêtre

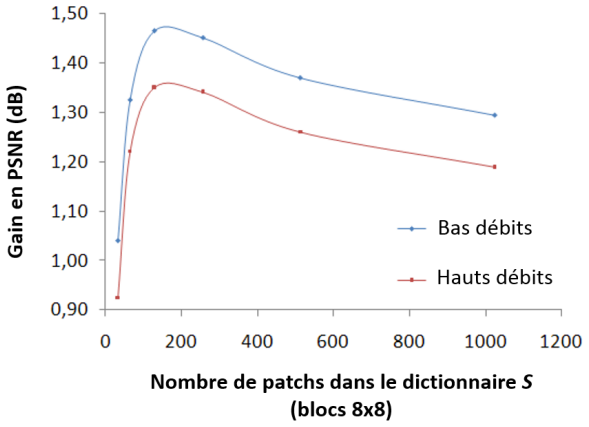

FIGURE 4.11 – *Gains en PSNR par rapport à H.264 Intra versus le nombre de patches défini dans le dictionnaire S avec une taille de fenêtre de recherche fixe, pour l'image Barbara, lorsque la méthode de prédiction hybride NE/oMANE est utilisée (avec la LLE).*

de recherche complète SW.

Nous analysons ensuite le comportement de l'algorithme hybride NE/oMANE lorsque l'on fait varier la taille de la fenêtre de recherche SW et que l'on fixe la taille du dictionnaire pré-défini S. La figure 4.12 montre que les gains en PSNR obtenus respectivement avec l'algorithme NE et l'algorithme hybride NE/oMANE pour l'image Spincalendar et avec différentes tailles de fenêtre (64x128, 128x256, 256x512 et 512x1024 pixels pour les blocs 8x8), la taille du dictionnaire S étant fixée à 512 patches pour les blocs 8x8. A noter qu'ici, la fenêtre de recherche SW et le dictionnaire pré-défini S utilisés pour les blocs 4x4 sont deux fois plus petits que ceux utilisés pour les blocs 8x8. Comparé à H.264 Intra, des gains jusqu'à 1.06 db à bas débit et 1 db à hauts débits sont réalisés avec la méthode de compression basée NE alors que ceux obtenus avec l'algorithme hybride NE/oMANE sont jusqu'à 2.13 db à bas débits et 1.69 db à hauts débits. En augmentant la taille de la fenêtre de recherche, on peut s'apercevoir que les performances de codage de l'algorithme hybride NE/oMANE ont été significativement améliorées par rapport à l'algorithme NE. Pour la méthode oMANE, cela augmente les chances de retenir des voisins qui sont plus similaires au patch d'entrée (contenant le bloc courant et son template), conduisant ainsi à une bonne prédiction du bloc courant tout en gardant un coût de codage constant pour le vecteur de matching.

4.3.4.5 Analyse sur la complexité de calcul

La prédiction spatiale d'un bloc inconnu dans un encodeur H.264 est basée sur de simples propagations (i.e. extrapolation) et combinaisons linéaires de pixels limitrophes le long de directions privilégiées. Ainsi, la charge de calcul pour la prédiction est très faible. Cependant, ces outils d'extrapolation peuvent atteindre des bons résultats uniquement lorsque l'on traite des textures assez simples mais cette approche a tendance à échouer lorsqu'il s'agit de propager des textures bi-dimensionnelles plus complexes.

Supposons une matrice de dictionnaire donnée $\mathbf{A} \in \mathbb{R}^{N \times K}$, $\mathbf{A}(k)$ de taille $N_1 \times K$, et $\mathbf{A}(u)$ de taille $N_2 \times K$ où $N_1 + N_2 = N$. De même, les pixels du template sont stockés dans le vecteur colonne $X(k)$ de taille N_1 et les pixels inconnus du bloc à prédire sont

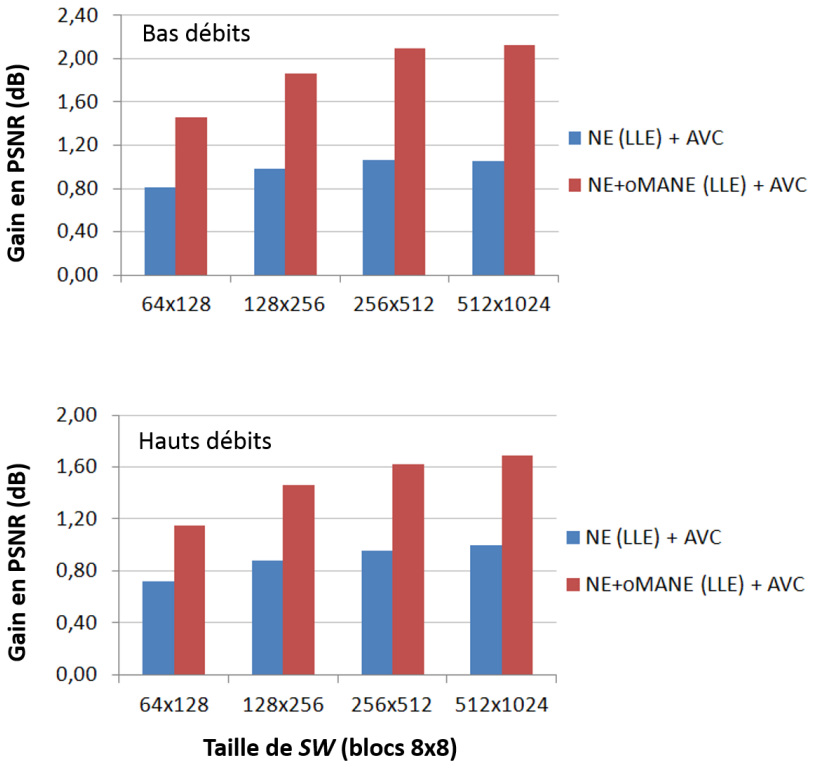

FIGURE 4.12 – *Gains en PSNR par rapport à H.264 Intra versus la taille de la fenêtre de recherche SW avec un nombre fixe de patches dans le dictionnaire prédéfini S, pour l'image Spincalendar, lorsque la méthode de prédiction hybride NE/oMANE est utilisée (méthode NE : la LLE).*

sauvegardés dans le vecteur colonne $X(u)$ de taille N_2.

Dans la méthode NE basée LLE, 20 patches (K) sont sélectionnés pour approximer le template, ces K patches étant les plus similaires au template au sens de la SAD (Somme des Différences Absolues). Pour sélectionner K patches parmi l'ensemble des patches possibles situés dans la fenêtre de recherche SW (soit M patches), nous avons besoin grossièrement de $MN_1+M(N_1-1)$ additions et MN_1 valeurs absolues, suivies de M comparaisons. Ensuite, le calcul de la matrice de covariance locale D requiert KN_1 additions et K^2N_2 opérations multiplication-addition. Afin de résoudre le système d'équations linéaires, nous avons besoin de $O(K^3)$ opérations arithmétiques (avec la méthode d'élimination gaussienne). Pour finir,

Méthode de prédiction	**Encodeur**	**Décodeur**
NE (LLE)	11.686 sec	2.629 sec
NE (NMF)	19.476 sec	7.844 sec
NE/oMANE (LLE, SW)	248.826 sec	2.869 sec
NE/oMANE (LLE, Dico S)	52.579 sec	3.126 sec
NE/oMANE (NMF, Dico S)	867.006 sec	8.685 sec
H.264	0.152 sec	0.054 sec

TABLE 4.7 – *Comparaison du temps d'exécution de différentes méthodes de prédiction à la fois à l'encodeur et au décodeur (valeurs obtenues pour une image de taille* 176x144 *pixels).*

la prédiction des pixels inconnus nécessite KN_2 opérations multiplication-addition.

Dans la méthode NE basée NMF, 20 patches (K) sont sélectionnés au début pour approximer le template. La mise à jour du vecteur V requiert $t(K^2(N_1+1)+KN_1$ corrélations et $t(2K)$ multiplications par éléments. Enfin, la prédiction des pixels inconnus nécessite KN_2 opérations multiplication-addition.

Comparée aux méthodes NE (LLE et NMF), la méthode oMANE utilisant un dictionnaire réduit S requiert au départ la construction d'un dictionnaire S. Ce dictionnaire contient $M' = 512$ patches sélectionnés à partir de la fenêtre de recherche SW, où $M' << M$. Les M' patches sont sélectionnés en minimisant la somme des erreurs absolues sur le template du bloc à prédire. Ainsi, pour sélectionner M' patches, comme ci-dessus, nous avons besoin de réaliser $MN_1+M(N_1-1)$ additions et MN_1 valeurs absolues, suivies de MN_1 comparaisons. Puis, l'algorithme oMANE recherche les L plus proches voisins du patch entier X dans le dictionnaire S. Chaque voisin est pointé par un vecteur de matching candidat (voir Fig. 4.4). Cette recherche requiert $M'N+M'(N-1)$ additions et $M'N$ valeurs absolues, suivies de M' comparaisons. Ensuite, pour chaque vecteur de matching candidat, l'algorithme identifie, dans le dictionnaire S, les $K-1$ plus proches voisins du patch pointé par le vecteur candidat (voir Fig. 4.4). Cette recherche requiert $M'N_1+M'(N_1-1)$ additions et $M'N_1$ valeurs absolues, suivies de M' comparaisons. Puis, La sélection du vecteur de matching "optimum" (parmi les L vecteurs candidats) qui minimise un critère donné nécessite de calculer L fois l'approximation LLE/NMF. Pour finir, la prédiction des pixels inconnus nécessite KN_2 opérations multiplication-addition.

Pour la méthode oMANE sans dictionnaire réduit (i.e. travaillant sur l'intégralité de la fenêtre de recherche SW), la recherche des L plus proches voisins du patch entier dans la fenêtre de recherche nécessite $MN+M(N-1)$ additions et MN valeurs absolues, suivies de M comparaisons. Ensuite, la recherche des $K-1$ plus proches voisins du patch pointé par un vecteur candidat requiert $MN_1+M(N_1-1)$ additions et MN_1 valeurs absolues, suivies de M comparaisons. Puis, comme ci-dessus, la sélection du vecteur "optimum" nécessite de calculer L fois l'approximation LLE/NMF et la prédiction des pixels inconnus requiert KN_2 opérations multiplication-addition.

Afin que l'analyse soit complète, le tableau 4.7 fournit les temps d'exécution des algorithmes de prédiction à la fois à l'encodeur et au décodeur, mesurés avec notre code C/C++ (sur un processeur de 3-GHz). Ces valeurs ont été données pour une image de taille 176x144 pixels. Tout d'abord, à l'encodeur, on peut observer que l'algorithme oMANE basé LLE utilisant un dictionnaire réduit nécessite beaucoup moins de CPU que celui opérant sur l'intégralité de la fenêtre de recherche SW étant donné que oMANE est directement appliqué sur le dictionnaire réduit. Par ailleurs, côté décodeur, nous pouvons souligner que la méthode oMANE avec dictionnaire réduit requiert un peu plus de CPU que celui sans dictionnaire réduit puisque le décodeur doit également construire le dictionnaire. En effet, les temps d'exécution de ces deux algorithmes sont respectivement de 52.579 sec et 248.826 sec à l'encodeur puis de 3.126 sec et 2.869 sec au décodeur. De plus, on peut remarquer qu'à l'encodeur et au décodeur la méthode NE/oMANE basée NMF nécessite beaucoup plus de CPU que celle utilisant la LLE puisque la NMF utilise une méthode de mise à jour multiplicative itérative. Pour finir, au décodeur, nous pouvons noter que le temps d'exécution de la méthode NE (LLE ou NMF) est légèrement différent de celui de la méthode hybride NE/oMANE associée. En effet, au décodeur, la prédiction basée oMANE n'a pas besoin d'effectuer le même traitement qu'à l'encodeur étant donné que le vecteur de matching, permettant de récupérer les mêmes voisins lorsque l'on calcule la solution LLE ou NMF, est transmis au décodeur.

Bas débits	NE + oMANE (LLE, RDO, Dico S)		JPEG2000 (Jasper Soft)	
QP :26,31,36,41	dB	% débit	dB	% débit
Barbara	1.38	-19.30	-1.4	25.36
Wool	2.10	-30.04	-1.83	34.10
Snook	2.86	-37.15	-1.32	19.84
Pan0_qcif	3.03	-43.56	-2.03	37.38
Building	4.09	-44.75	-1.28	21.75
Lena	0.11	-2.55	-1.23	33.93
Spincalendar	1.47	-25.08	-1.16	24.96
Foreman	0.60	-11.47	-2.6	64.50
Moyenne	**1.96**	**-26.74**	**-1.64**	**32.72**

Table 4.8 – *Comparaison de la méthode hybride NE/oMANE (utilisant la LLE comme méthode NE, le critère RDO et le dictionnaire réduit S) et JPEG2000 (Jasper Soft.) par rapport aux modes de prédiction intra H.264 à* ***bas débits****.*

Hauts débits	NE + oMANE (LLE, RDO, Dico S)		JPEG2000 (Jasper Soft)	
QP :16,21,26,31	dB	% débit	dB	% débit
Barbara	1.27	-14.72	-1.32	16.61
Wool	1.62	-17.60	-1.35	15.60
Snook	2.59	-21.90	-0.95	8.79
Pan0_qcif	2.58	-23.40	-2.08	16.92
Building	3.96	-31.07	-1.17	10.61
Lena	0.11	-1.77	-1.34	27.77
Spincalendar	1.15	-14.25	-1.33	18.79
Foreman	0.36	-5.35	-2.27	38.23
Moyenne	**1.71**	**-16.26**	**-1.50**	**19.16**

Table 4.9 – *Comparaison de la méthode hybride NE/oMANE (utilisant la LLE comme méthode NE, le critère RDO et le dictionnaire réduit S) et JPEG2000 (Jasper Soft.) par rapport aux modes de prédiction intra H.264 à* ***hauts débits****.*

4.3.5 Comparaison de la méthode NE/oMANE avec JPEG2000

Les tableaux 4.8 et 4.9 reportent les performances de codage (avec les mesures de Bjontegaard à bas et hauts débits) obtenues avec la méthode NE/oMANE hybride proposée (LLE) et JPEG2000 [jpe05, jas05] par rapport aux modes de prédiction Intra H.264. Pour calculer les performances de codage obtenues avec JPEG2000, chaque image testée est encodée ici en utilisant six ratios de compression différents. Pour chaque image testée, les ratios de compression sont déterminés de telle sorte que les débits cibles obtenus avec JPEG2000 soient à peu près égaux à ceux réalisés avec la méthode hybride NE/oMANE lorsque l'on utilise six pas de quantification différents comme indiqué dans le tableau 4.10.

Suite aux simulations réalisées sur l'ensemble des séquences de test, on peut remarquer que les performances de codage obtenues avec JPEG2000 sont très en dessous d'H.264 Intra [sof] (en moyenne : 1.64 db en dessous à bas débits et 1.50 dB à dessous à hauts débits). Ces résultats sont en phase avec la conclusion rapportée dans les publications antérieures

NE+oMANE (LLE, RDO, Dico S)			JPEG2000 (Jasper Soft)		
Pas de quantification (qp)	PSNR (dB)	Bitrate (bpp)	Taux de compression (τ)	PSNR (dB)	Bitrate (bpp)
16	46.924	2.852	0.119	45.031	2.850
21	42.372	1.776	0.074	40.952	1.772
26	38.542	1.042	0.043	36.571	1.025
31	35.167	0.639	0.027	32.838	0.646
36	31.677	0.365	0.015	29.232	0.358
41	28.253	0.206	0.009	26.860	0.216

Table 4.10 – *Tests Bitrate pour l'image Barbara.*

[APRSK+04, HW02] qui utilisent le même codec JPEG2000 [jpe05, jas05]. En effet, les auteurs dans [APRSK+04] ont rapporté que par rapport à JPEG2000, le gain en qualité de H.264 Intra s'élevait à 5 dB pour 4 bpp et à moins de 1 dB à 0.16 bpp. De même, les auteurs dans [HW02] montrent que H.264 Intra surpasse largement JPEG2000 à bas débit.

En revanche, on peut observer que la méthode hybride proposée offre significativement de meilleures efficacités de codage par rapport à H.264 Intra (en moyenne : 1.96 dB au-dessus à bas débits et 1.71 dB au-dessus à hauts débits).

4.4 Conclusion

Dans ce chapitre, nous avons introduit deux nouvelles méthodes de prédiction, appelées MANE et oMANE (basées sur la LLE ou la NMF) pour le codage intra, dans lesquelles la recherche K-NN est assistée par un vecteur de matching. Pour la méthode MANE, le vecteur de matching se réfère au patch le plus proche du bloc courant avec son template. Quant à la méthode oMANE, le vecteur de matching se réfère à celui qui minimise le coût débit-distorsion après reconstruction. Pour représenter des textures structurellement plus complexes, l'algorithme hybride s'avère être une alternative efficace par rapport respectivement à H.264 Intra et à la méthode NE conventionnelle au dépens d'une augmentation de complexité. De plus, il a été montré que la méthode hybride NE/oMANE basée LLE requiert considérablement moins de calcul que celle basée NMF pour une efficacité de codage similaire. Concernant l'algorithme oMANE, un dictionnaire de taille réduite (basé uniquement sur les pixels du template) a également été utilisé pour accélérer le processus de recherche des K-NN côté encodeur. Les résultats d'expérimentations ont montré que le temps d'exécution de l'algorithme hybride NE/oMANE a été considérablement réduit par rapport à l'algorithme hybride opérant sur la fenêtre de recherche complète. La considération d'un dictionnaire de taille réduite a également permis d'augmenter l'efficacité de codage de l'algorithme étant donné que le coût de codage relatif à un vecteur de matching a été réduit.

Ces outils de prédiction permettent de réduire efficacement les redondances entre le bloc courant et la région de l'image précédemment décodée/reconstruite. Cependant, certains blocs de l'image peuvent également présenter de meilleures similarités avec un patch situé sur la zone non reconstruite de l'image. Afin de mieux décorréler le signal image de façon globale, le chapitre suivant se propose d'étudier de nouveaux schémas de compression

basés sur le concept d'image résumé de type épitome, l'épitome étant considéré comme un dictionnaire de patches de texture les plus représentatifs de l'image.

CHAPITRE 5

Schéma de compression intra image basé sur le concept d'épitome

Les épitomes sont des outils efficaces pour une multitude d'applications telles que la segmentation, le dénoising, la reconnaissance visuelle, l'indexation et la synthèse de texture. Nous nous sommes cependant orientés vers une autre utilisation de cet outil : la prédiction d'images. Des premières méthodes de prédiction intra basées sur une approche épitome ont été abordées dans la sous-section 2.2.1 dans laquelle l'épitome est construit sur la base d'un algorithme EM (Expectation Maximinization). Ce type d'épitome préserve la texture globale et les caractéristiques de forme de l'image d'origine mais introduit également des artefacts visuels non désirés. Dans notre étude, nous avons plutôt retenu une autre approche s'appuyant sur la factorisation d'image [WWOH08] dans laquelle l'épitome est composé uniquement de patches provenant de l'image d'origine, permettant ainsi de ne pas introduire de tels artefacts.

Ce chapitre vise à introduire de nouveaux schémas de compression d'images fixes basés sur le concept d'épitome menant à un codage plus efficace. Dans un premier temps, un algorithme d'extraction d'épitome basé sur un modèle de translation pure a été mis en place. Ensuite, un premier schéma de codage a été proposé dans lequel une image de prédiction issue de l'épitome codé/décodé via la carte d'assignation associée est construite. Cette image de prédiction va servir d'image de référence pour encoder l'image courante. Un second schéma a été proposé encodant dans un premier temps l'épitome et reconstruisant le reste de l'image avec un esprit « inpainting » en utilisant un scanning adaptatif défini selon un critère inspiré des travaux de Criminisi [CPT04].

5.1 Construction de l'épitome et reconstruction de l'image

Nous décrivons dans cette section un nouvel outil de factorisation permettant d'extraire une version compactée d'une image source. L'objectif de cette représentation est de décorréler le signal source de manière à ne conserver que les patterns (patches de texture) les plus représentatifs de l'image. Une image est décrite ici par son image épitome E et une carte d'assignation Φ. L'épitome E résume le contenu de l'image d'origine en termes de texture et contient un ensemble d'épitomes charts issus de l'image d'origine. Cette carte d'assignation Φ correspond à un ensemble de vecteurs d'assignation. La carte d'assignation indique pour chaque bloc de l'image d'origine quel patch de texture provenant de l'épitome

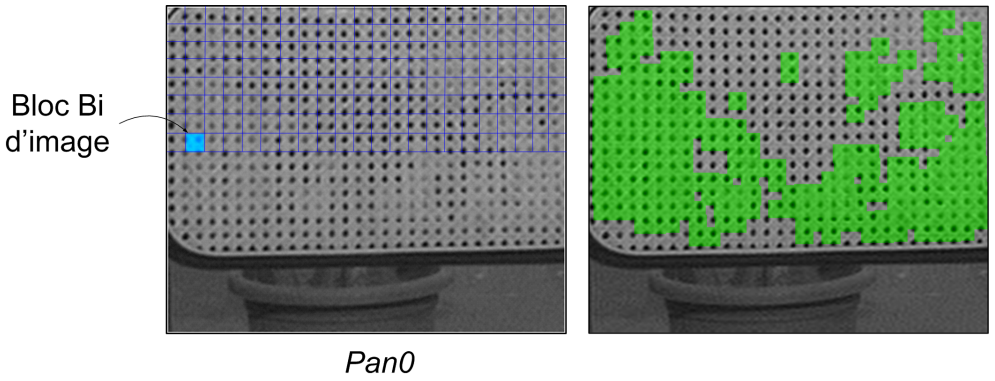

FIGURE 5.1 – *Un bloc d'image donné B_i (représenté en bleu) et ses appariements (soulignés en vert) pour une tolérance d'erreur donnée ϵ.*

est utilisé pour sa reconstruction.

Notons Y l'image d'origine de taille $N \times M$ pixels. L'image Y est divisée en une grille régulière de blocs, et chaque bloc B_i est approximé à partir d'un patch épitome via un vecteur d'assignation φ_i. La procédure de construction de l'épitome est essentiellement composée de trois étapes :

1. Détermination des similitudes
2. Création des épitomes charts
3. Amélioration de la qualité de reconstruction en effectuant par la suite une recherche du meilleur appariement et en mettant à jour la carte d'assignation en conséquence

5.1.1 Recherche des similarités

La méthode de construction de l'épitome procède dans un premier temps à l'analyse de l'image d'origine dans le but de détecter les zones de l'image présentant des similarités en termes de texture. Pour cela, nous venons rechercher pour chaque bloc de l'image d'origine, l'ensemble des patches dans l'image présentant un contenu similaire. En d'autres termes, pour chaque bloc $B_i \in Y$ ($B_i \in$ grille bloc de l'image), on détermine à partir de l'image d'origine une liste $\mathrm{L}_{match}(B_i) = \{M_{i,0}, M_{i,1}, ...\}$ d'appariements $M_{i,l}$ approximant B_i avec une tolérance d'erreur maximum ϵ pré-définie. Dans notre implémentation, la recherche des appariements a été effectuée à l'aide d'un algorithme de bloc matching utilisant une distance euclidienne moyenne. Notons qu'une recherche exhaustive est effectuée sur toute l'image. Une fois que toutes les listes d'appariements ont été créées pour l'ensemble des blocs de l'image, une nouvelle liste $\mathrm{L}'_{match}(M_{j,l})$, indiquant l'ensemble des blocs de l'image susceptibles d'être représentés par un même appariement $M_{j,l}$, est créé. Afin d'exploiter au mieux la redondance dans l'image, les appariements trouvés lors de cette phase de recherche peuvent ne pas être alignés avec la grille bloc de l'image et donc appartenir à la grille "pixel". La figure 5.1 illustre le processus de recherche des similarités en indiquant l'ensemble des matchs trouvés pour un bloc donné B_i dans l'image test " *Pan0_ qcif* ".

5.1.2 Création des épitomes charts

La seconde partie de l'approche consiste à sélectionner des patches de texture issus de l'image d'origine afin de construire les épitomes charts, l'ensemble des épitomes charts

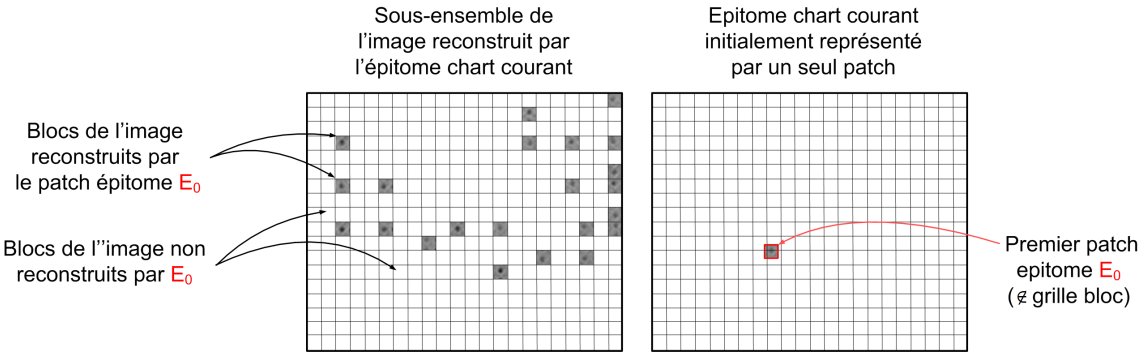

FIGURE 5.2 – *Etape d'initialisation d'un épitome chart : le sous-ensemble de l'image reconstruit par l'épitome chart courant (à gauche), l'épitome chart courant initialement composé d'un seul patch (à droite).*

constituant l'épitome final E. On peut noter que chaque épitome chart représente des zones bien spécifiques de l'image d'origine. En premier lieu, afin d'améliorer la qualité de construction d'un épitome par rapport à l'approche décrite dans la littérature [WWOH08], de nouveaux critères de sélection ont été proposés permettant ainsi d'optimiser l'initialisation et l'extension d'un épitome chart par l'ajout d'un nouvel élément de texture. Par ailleurs, une optimisation supplémentaire a été apportée dans le processus d'extension d'un épitome chart, reposant sur la considération de blocs dits "induits".

5.1.2.1 Initialisation

Soit n l'index de l'épitome chart courant EC_n. Initialement, n est mis à zéro. Soit $Y \in R^{N\times M}$ et $Y' \in R^{N\times M}$ désignant respectivement l'image d'entrée et l'image reconstruite par un patch de texture donné. L'épitome chart EC_n est initialisé par le patch de texture le plus représentatif des blocs de l'image non reconstruits jusqu'à présent par l'épitome courant (E0 sur la figure 5.2). Afin d'initialiser un épitome chart, nous introduisons le critère de sélection suivant basé sur la minimisation d'un critère MSE (Mean Square Error) :

$$\min\left(\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\left(Y_{i,j}-Y'_{i,j}\right)}{N\times M}\right) \tag{5.1}$$

Le critère de sélection (5.1) prend en considération les erreurs de prédiction sur l'ensemble de l'image. Autrement dit, ce critère est appliqué non seulement pour les blocs de l'image qui sont approximés par un patch de texture donné mais également pour les blocs de l'image qui ne sont pas approximés par ce patch. Dans notre implémentation, la valeur zéro est assignée aux pixels de l'image qui ne sont pas reconstruits par ce patch lors du calcul de l'erreur de reconstruction de l'image. Par conséquent, ce critère permet à un épitome chart d'être étendu par un motif permettant la reconstruction du plus grand nombre de blocs dans l'image ainsi que la minimisation de l'erreur de reconstruction. La figure 5.2 illustre l'étape d'initialisation en indiquant les blocs de l'image reconstruite une fois que le premier patch épitome est sélectionné.

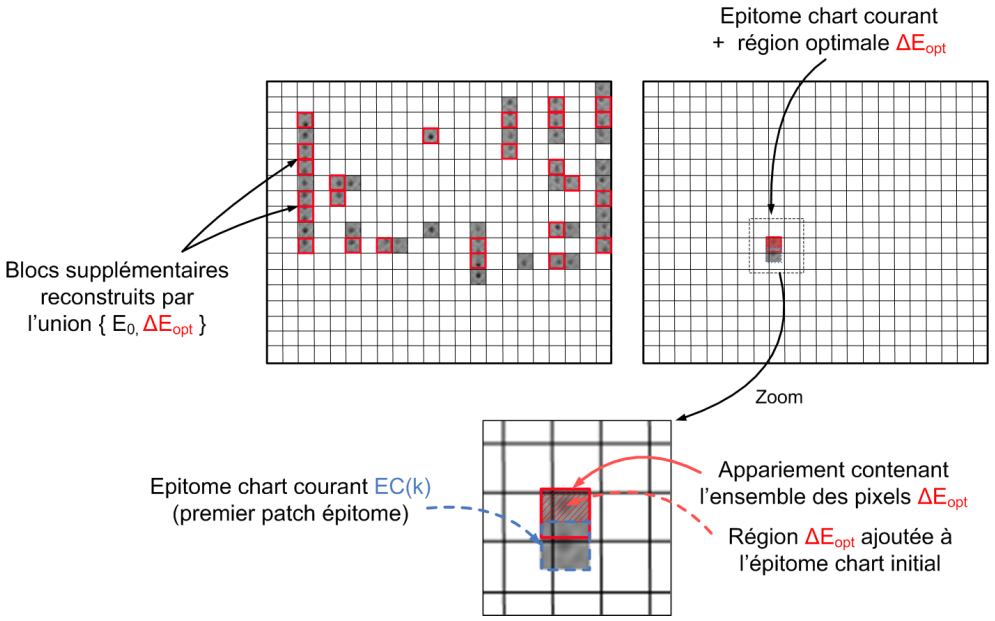

FIGURE 5.3 – *Etape d'extension d'un épitome chart : le sous-sensemble de l'image reconstruit par l'épitome chart courant et la région optimale by the current chart epitome ΔE_{opt} (à gauche), l'épitome chart courant étendu par la région optimale ΔE_{opt} (à droite).*

5.1.2.2 Extension

L'épitome chart courant EC_n est ensuite progressivement étendu par un élément de patch optimal ΔE_{opt} issu de l'image d'origine, puis au fur et à mesure que l'épitome chart courant est élargi, nous gardons la trace du nombre de blocs supplémentaires ayant été reconstruits dans l'image, comme illustré dans la figure 5.3.

Soit k le nombre de fois où l'épitome chart courant est étendu. L'épitome chart initial $EC_n\,(k=0)$ correspond au patch de texture retenu à l'étape d'initialisation. Lors du processus d'accroissement d'un épitome chart, on procède tout d'abord à la recherche de l'ensemble des appariements $M_{j,l}$ connexes à la zone de l'image relative à l'épitome chart courant $EC_n\,(k)$ et susceptibles de représenter d'autres blocs de l'image. Par conséquent, plusieurs candidats ΔE sont en mesure d'être utilisés comme une extension de l'épitome chart courant. Désignons par m le nombre de candidats déterminés après k étapes d'extension de l'épitome chart courant. Pour chaque candidat d'accroissement ΔE, l'ensemble des blocs de l'image supplémentaires pouvant être reconstruits, est spécifié dans la liste $\mathrm{L}'_{match}(M_{j,l})$ associée exclusivement à l'appariement contenant l'ensemble des pixels ΔE. Ensuite, nous venons sélectionner le candidat optimal parmi l'ensemble des candidats trouvés en adoptant un critère de type débit/distorsion, donné par les équations (5.2) et (5.3). Soit $Y \in R^{N\times M}$ et $Y' \in R^{N\times M}$ désignant respectivement l'image d'entrée et l'image reconstruite à partir de l'information de texture contenue dans l'épitome courant E_{curr} et la région d'accroissement candidate ΔE. Il est important de souligner que l'épitome courant est composé des épitomes charts précédemment construits ainsi que de l'épitome chart courant.

$$\min\left(D_{E_{curr}+\Delta E} + \lambda \times R_{E_{curr}+\Delta E}\right) \tag{5.2}$$

$$\text{avec } E_{curr} = \textstyle\sum_{i=0}^{n} EC_i$$

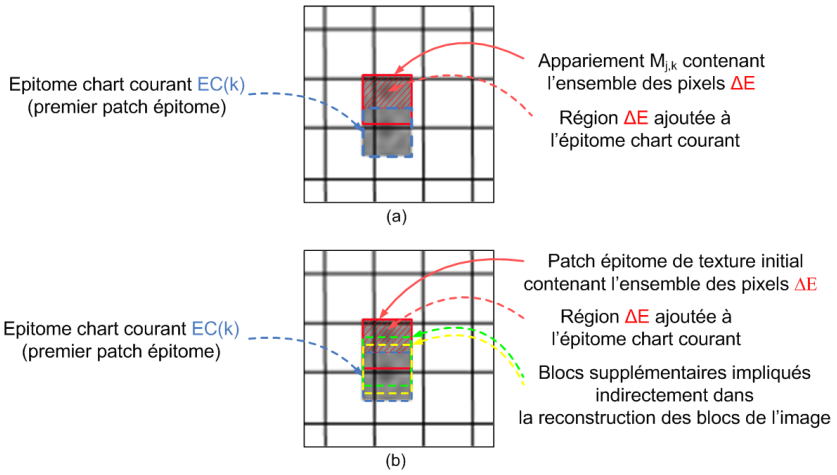

FIGURE 5.4 – *Extension d'un épitome chart : la méthode initiale (a), l'algorithme proposé (b)*

$$\Delta E_{opt}^{k} = \underset{m}{\operatorname{argmin}} \left(\frac{\sum_{i=0}^{N} \sum_{j=0}^{M} \left(Y_{i,j} - Y'_{i,j} \right)}{N \times M} + \lambda \left(\frac{E_{curr} + \Delta E_m}{N \times M} \right) \right) \tag{5.3}$$

Le premier terme du critère (5.3) correspond à l'erreur de reconstruction moyenne par pixel obtenue lorsque l'image d'entrée est reconstruite à l'aide de l'information de texture contenue à la fois dans l'épitome courant $E_{curr} = \sum_{i}^{n} EC_i$ et un incrément candidat ΔE. Comme dans l'étape d'initialisation, lorsque les pixels de l'image ne sont pas impactés par l'ensemble (épitome courant, incrément ΔE), une valeur zéro est assignée. Le second terme du critère (5.3) se réfère à un coût moyen par pixel relatif à la construction de l'épitome, qui est grossièrement estimé comme le nombre de pixels dans l'épitome courant E_{curr} et son incrément ΔE divisé par le nombre total de pixels de l'image.
Une fois l'incrément optimal ΔE_{opt}^{k} sélectionné localement, l'épitome chart courant devient :

$$EC_n\,(k+1) = EC_n\,(k) + \Delta E_{opt}^{k} \tag{5.4}$$

Ensuite, nous continuons d'agrandir l'épitome chart courant jusqu'à ce qu'il n'y ait plus d'appariement $M_{j,l}$ chevauchant la région associée à l'épitome chart courant et permettant de représenter d'éventuels blocs. Par conséquent, lorsque l'épitome chart courant E_{curr} ne peut plus être étendu et que l'image n'est pas intégralement reconstruite par l'épitome courant, l'index n est incrémenté de 1 et un nouvel épitome chart est initialisé à un nouvel emplacement dans l'image. Le processus d'extraction des épitomes charts prend fin dès que l'image entière est représentée par l'épitome. Les deux premières étapes de la méthode de construction d'un épitome sont résumées dans l'algorithme 5.1.

5.1.2.3 Blocs inférents / blocs induits

Une autre amélioration apportée à l'outil d'extraction d'epitome chart, concerne la prise en compte de blocs dits " induits ". En effet, jusqu'à présent, lors de l'extension d'un épitome chart par une région plus grande, nous considérions uniquement les blocs reconstruits à partir de la texture issue du patch épitome initial contenant exclusivement l'ensemble des pixels de l'incrément ΔE (Figure 5.4.a). Cependant, la fusion de l'épitome

Algorithme 5.1 : Pseudocode de l'algorithme de construction d'épitome

Inputs : L'image source Y, la tolérance d'erreur ϵ
1. Trouver les auto-similarités dans l'image :
foreach *image* $B_i \in Y$ *(*$B_i \in$ *grille bloc)* **do**
  Déterminer l'ensemble des appariements dans l'image :
  $\mathrm{L}_{match}(B_i) = \{M_{i,0}, M_{i,1}, ...\}$
end
foreach *appariement* $M_{j,l}$ *trouvé (*$M_{j,l} \in$ *grille pixel)* **do**
  Déterminer l'ensemble des blocs de l'image : $\mathrm{L'}_{match}(M_{j,l})$ qui peuvent être approximés par $M_{j,l}$
end
2. Créer les épitomes charts :
init $n = 0$
while *tous les blocs de l'image ne sont pas représentés par l'épitome courant* **do**
  Initialiser un épitome chart EC_n par l'appariement le plus représentatif $M_{j,l}$
  Etendre EC_n :
  Init $k = 0$
  while *il y a des appariements* $M_{j,l}$ *qui chevauchent* $EC_n(k)$ **do**
    Déterminer l'incrément optimal ΔE^k_{opt}
    Etendre l'épitome chart courant $EC_n(k+1) = EC_n(k) + \Delta E^k_{opt}$
    k=k+1
  end
  n=n+1
end
3. Raffiner la carte :
foreach $B_i \in Y$ *(*$B_i \in$ *grille bloc)* **do**
  Rechercher le meilleur appariement dans l'épitome final
end

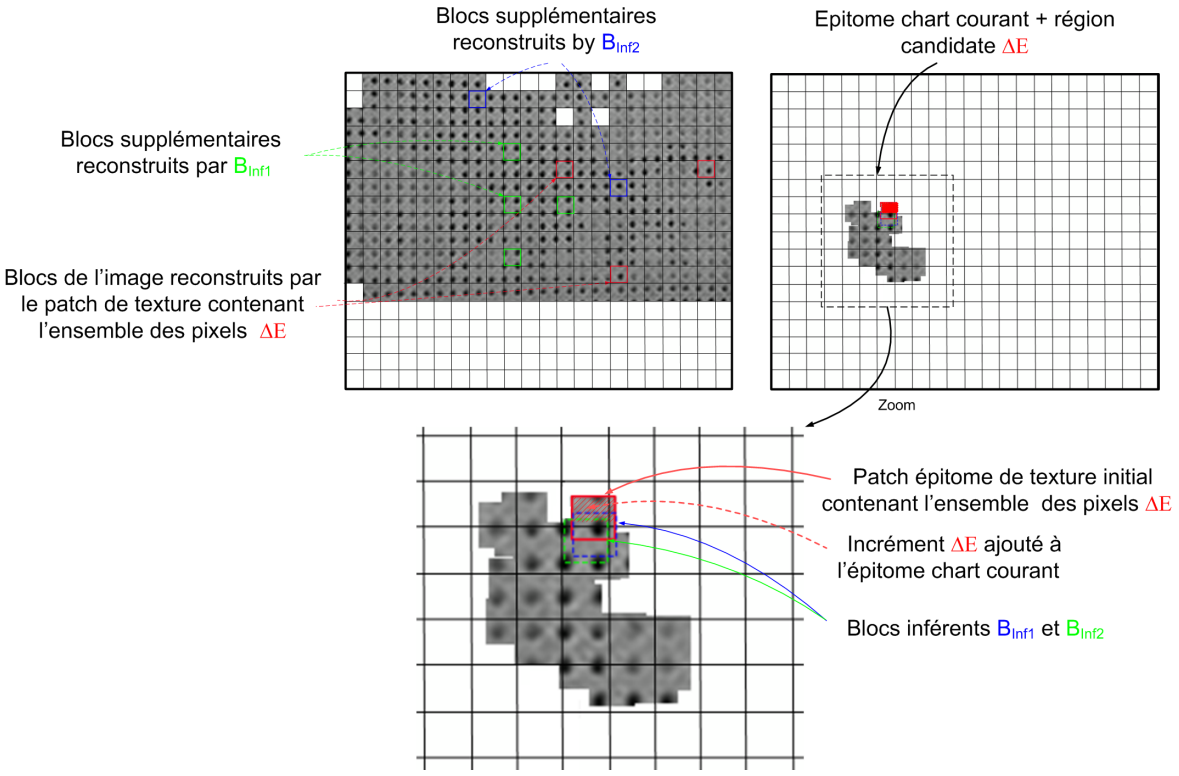

FIGURE 5.5 – *Exemple qui illustre la considération des blocs inférents lors de l'extension d'un épitome chart : le sous-ensemble de l'image reconstruit par l'épitome chart courant (en haut à gauche), l'épitome chart courant agrandi par un incrément (en haut à droite).*

chart courant avec un incrément candidat ΔE peut engendrer d'autres patches qui pourraient représenter des blocs de l'image supplémentaires. On peut souligner que les blocs " induits " ne requièrent pas l'ajout de texture autre que l'incrément ΔE (Figure 5.4.b). Par ailleurs, nous pouvons remarquer que le nombre de blocs " induits " dépend également de la répartition spatiale locale des pixels disponibles dans l'épitome courant. La Figure 5.5 illustre plus précisément le principe des blocs " induits " après plusieurs étapes d'extension d'un épitome chart.

5.1.2.4 Tests préliminaires : évaluation des outils développés

Après avoir introduit de nouveaux outils pour sélectionner des patches de texture issus de l'image d'origine visant à améliorer le procédé d'extraction d'une image " résumé ", il est judicieux d'effectuer des tests préliminaires permettant de consolider l'efficacité de l'algorithme présenté jusqu'ici par rapport à un algorithme intégrant un critère issu des travaux de H. Hoppe . Ce critère, introduit dans [WWOH08], est donné par l'équation suivante :

$$\max_{\Delta E} |I^{E_{curr}+\Delta E} \backslash I^{E_{curr}}| - |\Delta E| \tag{5.5}$$

où $I^{E_{curr}}$ et $I^{E_{curr}+\Delta E}$ désignent respectivement le sous ensemble de l'image reconstruit par l'épitome courant E_{curr} et le sous ensemble reconstitué par la réunion d'un incrément candidat ΔE à l'épitome courant E_{curr}. L'objectif étant de vérifier que l'usage d'un critère

Bas débits	Algorithme proposé	
ϵ=10,12,14,16	dB	% débit
Foreman_qcif	0.68	-13.1
Pan0_qcif	0.38	-11.4
BarbaraCrop1	1.37	-20.2
BarbaraCrop2	0.59	-7.6
CityCrop	0.17	-3.2
Moyenne	**0.64**	**-11.09**

Hauts débits	Algorithme proposé	
ϵ=6,8,10,12	dB	% débit
Foreman_qcif	0.71	-6.2
Pan0_qcif	0.47	-14.9
BarbaraCrop1	0.54	-6.9
BarbaraCrop2	0.5	-3.4
CityCrop	0.3	-3.8
Moyenne	**0.51**	**-7.83**

Table 5.1 – *Comparaison des performances de l'outil d'extraction d'épitome proposé par rapport à l'algorithme basé sur un critère issu de [WWOH08], au sens des mesures de Bjontegaard.*

de sélection de type SSE lors du processus de l'initialisation d'un épitome chart ainsi que l'utilisation conjointe d'un critère de type débit/distorsion et de blocs " induits " lors du processus d'extension d'un épitome chart sont mieux adaptés pour choisir des éléments de texture qu'un critère mesurant uniquement l'impact de l'ajout d'un élément sur la surface reconstruite dans l'image (Eq. (5.5)). Pour cela, nous avons mis en oeuvre un outil d'évaluation inspiré de la métrique de Bjontegaard [Bjo01], outil couramment utilisé en compression pour évaluer les performances d'un encodeur et permettant de fournir une évaluation du gain en qualité à débit constant, ainsi que le gain en débit à qualité identique. Rappelons toutefois qu'en codage, les mesures de Bjontegaard sont généralement générées à partir de quatre valeurs de couple {qualité, débit} obtenues à l'aide de quatre pas de quantification différents afin de couvrir une certaine plage de débit, chaque pas de quantification étant associé à un débit donné.

Une approche similaire peut également être adoptée pour jauger les performances d'un algorithme d'extraction d'épitome. En effet, le seuil d'erreur de reconstruction défini dans l'algorithme peut alors s'apparenter au pas de quantification utilisé en codage d'image dans la mesure où ce seuil agit directement sur la taille de l'épitome généré ainsi que sur la qualité de l'image reconstruite à partir de cette représentation compacte, ce seuil étant relatif à la dégradation de la qualité maximale que l'on accepte d'introduire dans la reconstruction de l'image. Ainsi, plus le seuil d'erreur de reconstruction sera élevé, plus la taille de l'épitome sera réduite. Il est donc possible d'évaluer le rendu sur la qualité de l'image reconstruite ainsi que le rendement vis-à-vis de la taille de l'épitome au sens des mesures de Bjontegaard en se basant sur la connaissance de quatre valeurs de couple {qualité image reconstruite, taille épitome} obtenues à partir de quatre seuils d'erreurs de valeurs différentes.

Le tableau 5.1 fournit les gains en termes de qualité de reconstruction de l'image (évalué en dB) ainsi que les pourcentages de réduction de la taille de l'épitome (évalué en pixels) obtenus sur des images test avec l'algorithme d'extraction d'épitome intégrant les outils développés par rapport à celui utilisant un critère issu de [WWOH08]. Nous constatons que le procédé d'extraction d'épitome développé intégrant conjointement les critères proposés et les blocs " induits " est plus efficace que l'algorithme basé uniquement sur un critère de type " Hoppe " pour l'ensemble des images test. En effet, nous remarquons que l'outil développé jusqu'à maintenant offre une meilleure qualité de reconstruction de l'image pour une taille d'épitome donnée.

Les figures 5.6 et 5.7 comparent visuellement quelques résultats de factorisation obtenus pour deux images croppées de Barbara avec l'algorithme d'extraction d'épitome proposé et

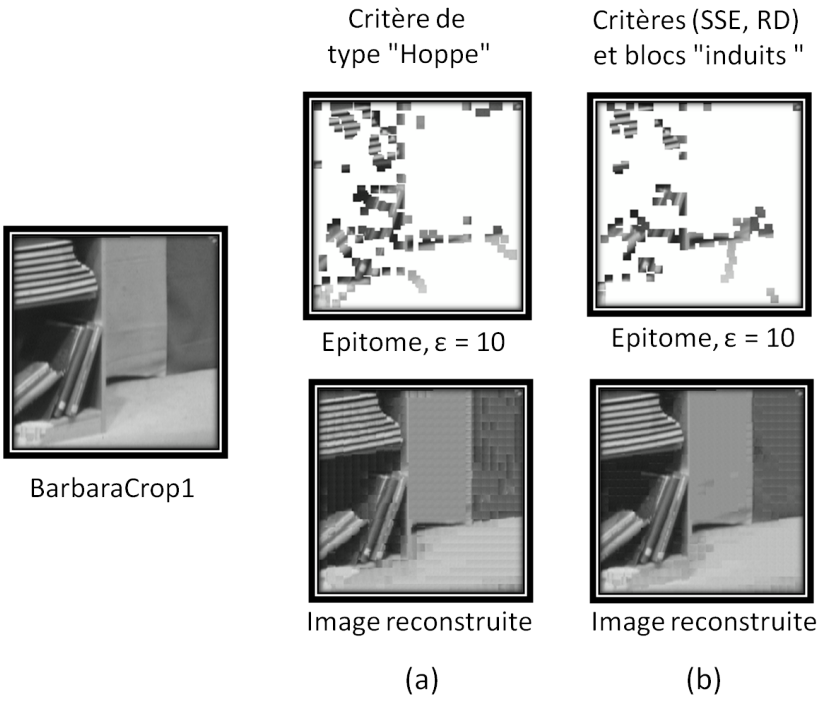

Figure 5.6 – *Comparaison de l'épitome et de l'image reconstruite obtenus sur une première image croppée de Barbara : (a) critère issu de [WWOH08], (b), critères SSE, RD et blocs "induits"*

celui basé sur un critère issu de [WWOH08]. On peut s'apercevoir que l'approche proposée tend à atténuer les artefacts induits au signal tout en réduisant le contenu de l'épitome. Cet effet est clairement visible dans la figure 5.7.

5.1.3 Amélioration de la qualité de reconstruction (raffinement de la map)

Une fois la construction de l'épitome terminée, un raffinement est effectué afin d'obtenir une meilleure qualité de reconstruction. En effet, lorsqu'un patch de texture est ajouté à l'épitome, cet ajout peut fournir une meilleure reconstruction pour d'autres blocs de l'image, pour lesquels un autre patch de l'épitome leur a déjà été attribué lors des étapes précédentes. C'est pourquoi, après la génération de l'épitome, une recherche du meilleur appariement est opérée entre chaque bloc de l'image et l'épitome mettant ainsi à jour la carte d'assignation (étape 3 de l'algorithme 5.1).

5.1.4 Matching avec une précision sous-pixellique

Afin d'affiner la qualité de l'image reconstruite obtenue à partir de l'outil de factorisation, il est possible de représenter plus efficacement l'image d'origine en utilisant une précision sous-pixellique. Pour ce faire, il existe différentes manières de procéder pour fournir une reconstruction de plus grande précision.

La manière la plus intuitive pour aller chercher une reconstruction à la précision sous-pixellique, est de sur-échantillonner l'image épitome obtenue à la résolution spatiale de l'image d'origine, comme décrit en figure 5.8. Les positions sous-pixels de l'image épitome

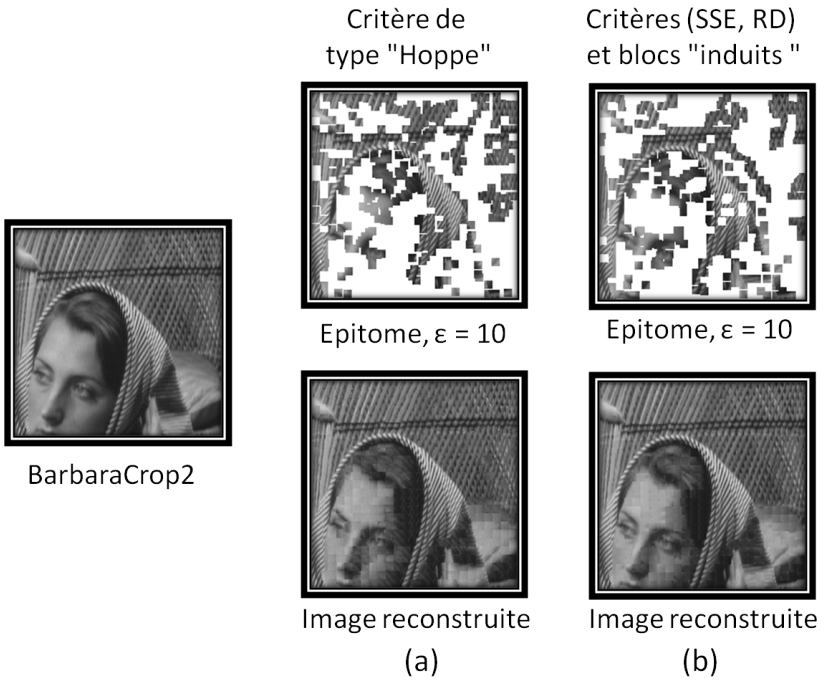

FIGURE 5.7 – *Comparaison de l'épitome et de l'image reconstruite obtenus sur une seconde image croppée de Barbara : (a) critère issu de [WWOH08], (b), critères SSE, RD et blocs "induits"*

sont donc générées à l'aide de filtres interpolateurs. Cette approche permet ainsi la reconstruction d'une image de meilleure qualité au détriment d'un coût de vecteurs plus élevé dû à l'utilisation de vecteurs d'assignation de plus grande précision.

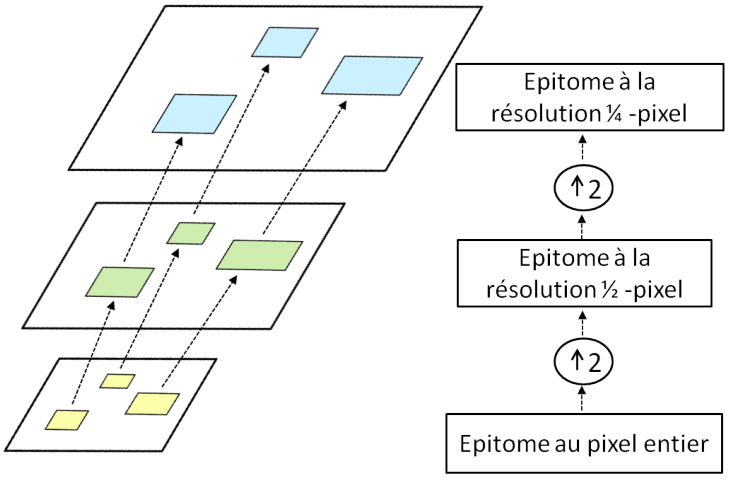

FIGURE 5.8 – *Image épitome au quart de pel issue de l'image épitome de la couche de base interpolée*

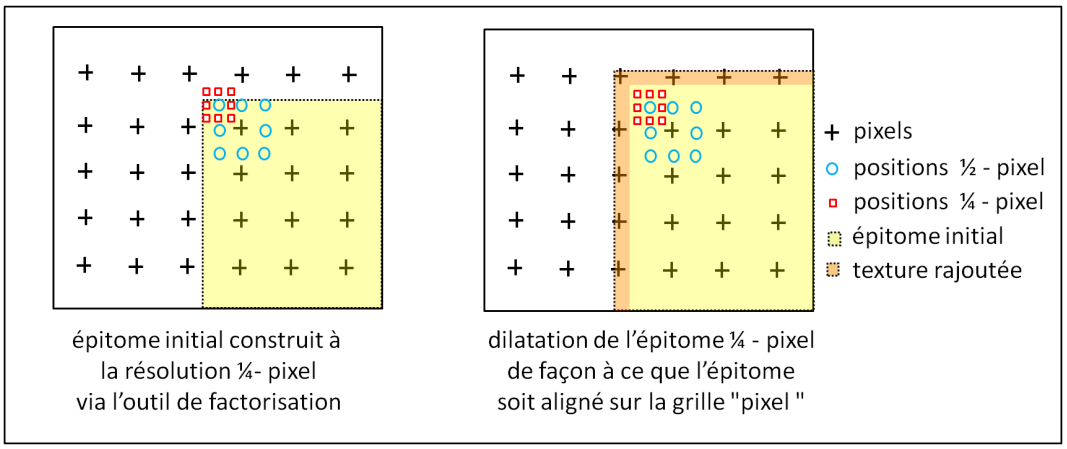

FIGURE 5.9 – *Alignement de l'image épitome construite au quart de pixel sur la grille pixel de l'image source*

Cependant, cette approche paraît sous-optimale dans la mesure où l'épitome a été initialement construit sur la base d'un modèle de translation effectuant une recherche au pixel entier et non une recherche sous-pixellique. Or, on peut penser qu'il serait plus judicieux de considérer la précision sous-pixellique au moment de la construction de l'épitome. En effet, cela permettrait d'améliorer a priori la recherche des similarités et ainsi de réduire encore la taille de l'épitome. Dans ce manuscrit, nous nous sommes donc intéressés à l'étude de cette seconde approche.

En considérant une précision allant jusqu'au quart de pixel, l'algorithme proposé opère de la manière suivante :

1. Dans un premier temps, une interpolation au quart de pixel est appliquée sur l'image source I_0 menant à $I_{0(\uparrow 4)}$. Notons que le quart de pixel est effectué ici à l'aide de filtres séparables FIR [WSBL03].
2. Une recherche au quart de pixel est ensuite réalisée pour déterminer les similarités (matching) contenues dans l'image $I_{0(\uparrow 4)}$. Un épitome au quart de pixel $E_{0(\uparrow 4)}$ est par la suite extrait de l'image interpolée $I_{0(\uparrow 4)}$ à l'aide de l'algorithme de factorisation d'image décrit précédemment.
3. Une fois la construction de l'épitome $E_{0(\uparrow 4)}$ terminée, $E_{0(\uparrow 4)}$ est étendu avec de l'information issue de l'image interpolée $I_{0(\uparrow 4)}$ de façon à ce que l'épitome $E_{0(\uparrow 4)}$ soit aligné sur la grille pixel de l'image source, comme illustré dans la figure 5.9. Ce traitement est essentiel afin de pouvoir extraire l'épitome E_0 à la résolution spatiale de l'image d'origine directement à partir des pixels issus de $E_{0(\uparrow 4)}$ sur une grille de sous-échantillonnage. Ceci est cohérent dans la mesure où le processus de sur-échantillonnage effectué à l'étape 1 ne vient pas altérer le signal image sur ces positions pixel. Ainsi, les valeurs des pixels aux positions pixel de l'épitome $E_{0(\uparrow 4)}$ sont identiques aux valeurs des pixels correspondant à l'image d'origine. De ce fait, deux cas de figure peuvent donc être envisagés pour obtenir l'épitome E_0 à la résolution pixel, chacun conduisant au même résultat :
   - Soit E_0 est obtenu en récupérant directement l'information de texture contenue aux positions pixel de $E_{0(\uparrow 4)}$.
   - Soit on vient déterminer dans un premier temps la structure de E_0 à l'aide de masques permettant de localiser la région épitome à la résolution quart de pixel.

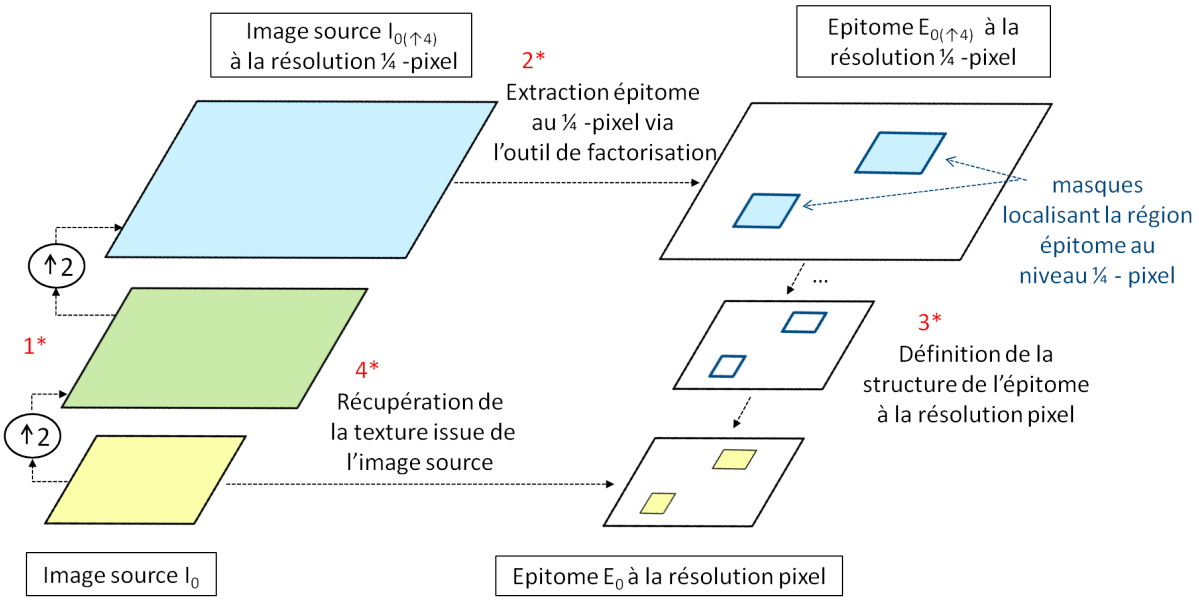

FIGURE 5.10 – *Image épitome au quart de pixel construite à l'aide de l'outil de factorisation*

Puis, on vient remplir le contenu de E_0 directement à partir de l'information de texture issue de l'image d'origine I_0, comme illustré dans la figure 5.10.

4. Pour finir, une phase de raffinement est réalisée via l'algorithme de recherche du meilleur appariement afin d'exploiter éventuellement les pixels rajoutés lors de l'extension de l'épitome $E_{0(\uparrow 4)}$.

Soulignons néanmoins qu'une démarche différente aurait également pu être adoptée pour extraire l'épitome E_0 à partir de sa version sur-résolue $E_{0(\uparrow 4)}$. Cette approche consiste à sous échantillonner l'image épitome $E_{0(\uparrow 4)}$ obtenue à la résolution quart de pixel par le biais de filtres FIR [WSBL03]. Pour ce faire, il est donc nécessaire d'effectuer du padding supplémentaire autour de l'épitome $E_{0(\uparrow 4)}$ avec de l'information de texture issue de $I_{0(\uparrow 4)}$ afin d'utiliser proprement les filtres FIR aux bords de l'épitome $E_{0(\uparrow 4)}$. On peut noter cependant que cette approche présente un inconvénient dans la mesure où les motifs de texture introduits dans l'épitome final E_0 risquent d'être légèrement différents de ceux contenus dans l'image source. En effet, le fait de sous-échantillonner le signal épitome au quart de pixel $E_{0(\uparrow 4)}$ pour obtenir E_0 engendre une légère perte d'information par rapport aux données contenues dans l'image source I_0. Par conséquent, la démarche la plus adaptée pour récupérer l'épitome à la résolution spatiale de l'image d'origine est celle utilisée dans la description de l'algorithme puisqu'elle permet non seulement de s'affranchir des modifications de l'épitome dues aux contraintes des filtres FIR mais également évite d'avoir dans l'épitome final E_0 des motifs légèrement modifiés par rapport à ceux de l'image source.

La figure 5.11 confirme l'intérêt de construire un épitome en utilisant une précision sous-pixellique. On peut constater que lorsque l'on intègre une précision au quart de pixel dans la procédure de construction de l'épitome, cela permet de réduire de manière significative l'information de texture contenue dans l'épitome. En effet, le contenu de l'épitome est réduit pratiquement de moitié par rapport à l'approche utilisant une précision pixellique, pour un même ϵ.

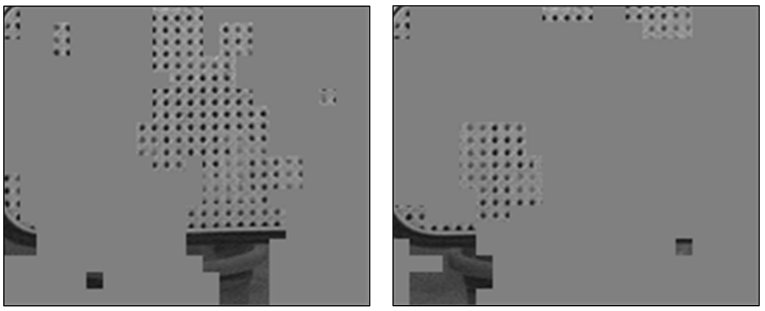

Figure 5.11 – *Exemple de construction d'un épitome : précision au pixel entier (a), précision au quart de pixel (b).*

5.1.5 Reconstruction de l'image

Une fois que l'image épitome est générée, nous pouvons reconstruire une approximation de l'image d'origine à partir de la texture de l'épitome et de la carte d'assignation associée. Quelques résultats de factorisation obtenus pour les images test : Building$_{DS}$, Wool et Pan0_qcif sont présentés dans la figure 5.12. On peut constater que des différences importantes subsistent entre l'image d'origine et l'image reconstruite dues à l'utilisation d'un seuil d'erreur de reconstruction ϵ. Ces différences sont notamment perceptibles sur les zones homogènes, comme on peut le voir lors de la reconstruction de l'image Pan0_qcif. Pour des applications dédiées à la compression vidéo, les exigences en qualité sont telles qu'il est nécessaire de proposer un nouveau schéma de codage permettant d'encoder ces différences, comme décrit dans la section suivante.

5.2 1ère approche étudiée : Algorithme de compression basé épitome/map

Nous avons vu dans la section précédente que les dégradations induites au signal source à l'issue de la factorisation demeurent relativement importantes dues à l'utilisation d'un seuil d'erreur de reconstruction. Afin d'améliorer la qualité de reconstruction de l'image, nous proposons de décrire un nouveau schéma de codage dans lequel l'image source est encodée en prédiction inter image par rapport à une image de référence construite à partir de l'épitome et de la carte d'assignation. Ce schéma de codage s'apparente à de la scalabilité SNR. Par ailleurs, nous pouvons noter qu'en compression d'images, les patterns issus de l'image résumé peuvent être mieux adaptés pour prédire la texture 2D puisque des patches 2D sont considérés à la place des fonctions d'extrapolation orientées mono-directionnelles.

5.2.1 Description de l'algorithme

5.2.1.1 Schéma global

Le schéma de compression basé épitome est décrit dans la figure 5.13. L'objectif de cette méthode est d'utiliser la reconstruction de l'image obtenue à partir de l'épitome décodé comme une image de référence pour encoder l'image source. Ce schéma de codage nécessite donc d'encoder non seulement l'erreur de prédiction mais également l'épitome {texture, carte d'assignation}

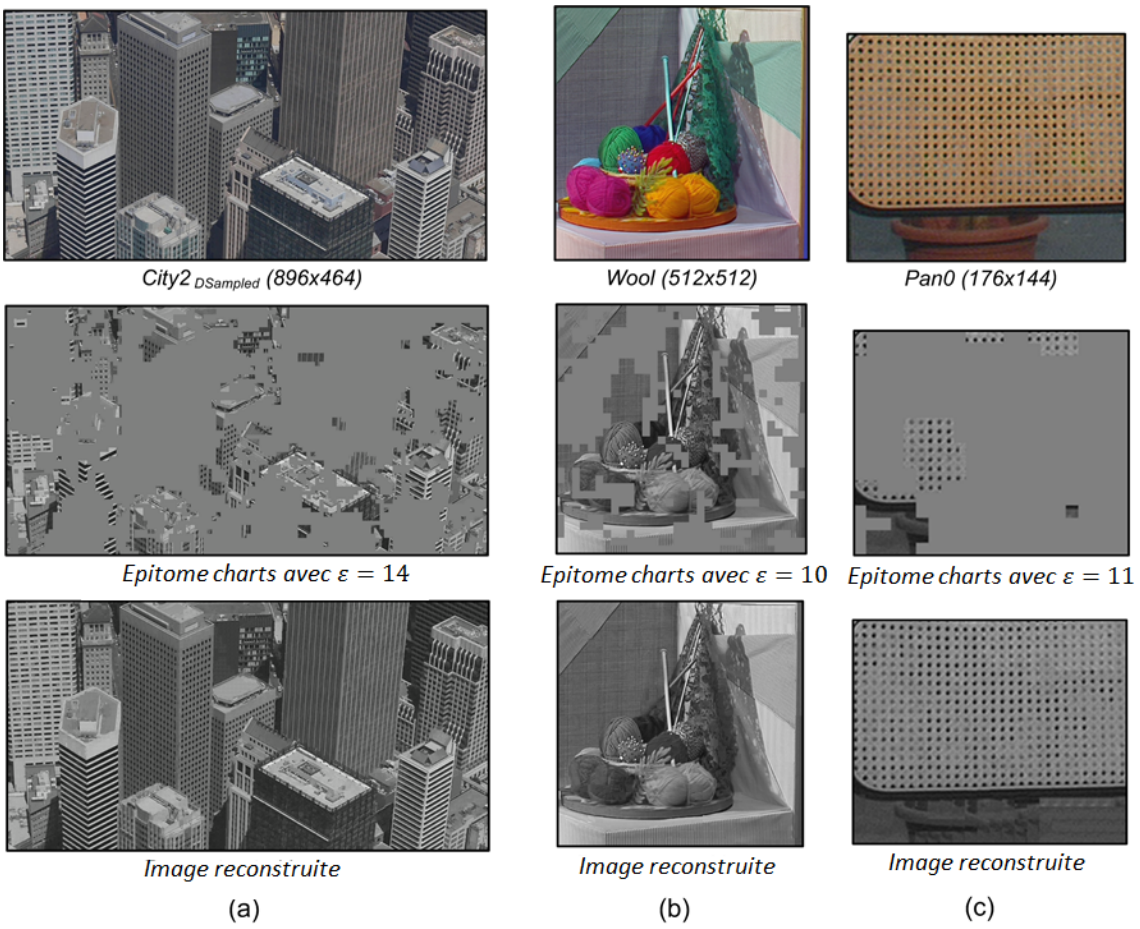

Figure 5.12 – *Résultats de factorisation de trois images : $Building_{DS}$ (version sous-échantillonnée de Building) (a), Wool (b) et Pan0 (c)*

5.2.1.2 Codage de texture de l'épitome

La texture de l'épitome est encodée en utilisant un encodeur de type H.264 Intra (software de référence KTA [sof]). Afin d'améliorer le codage de l'épitome, une optimisation supplémentaire a été apportée sur l'épitome avant de l'encoder. En effet, étant donné que l'épitome issu de la factorisation est construit au départ sur la base d'une grille "pixel", la structure de l'épitome contient des ruptures artificielles entre les zones "épitome"/"non épitome". Par conséquent, ces contours à l'intérieur des blocs 8x8 vont engendrer un coût de codage de texture nettement plus important. Afin d'éviter cela, on propose d'adapter la structure de l'épitome en considérant les contraintes du codeur. Pour cela, l'épitome initial est paddé de façon à ce qu'il soit aligné sur la structure bloc du codeur (i.e. bloc 8x8). Nous avons choisi d'effectuer du padding sur l'épitome avec de la texture réelle et non par propagation de pixel. En effet, le fait de padder l'épitome avec de l'information image permet d'inclure dans le processus de la reconstruction de l'image dite " reconstruite " une autre phase de raffinement en exploitant éventuellement les pixels ajoutés à l'épitome initial par le biais d'un algorithme de recherche du meilleur appariement, comme illustré dans la figure 5.14. Notons que si l'épitome avait été paddé par le biais d'un processus de propagation de pixels (comme dans MPEG4 part 2 et plus particulièrement dans la compression des vidéos objects planes : VOP), cette information paddée ne nous servirait à rien lors de la

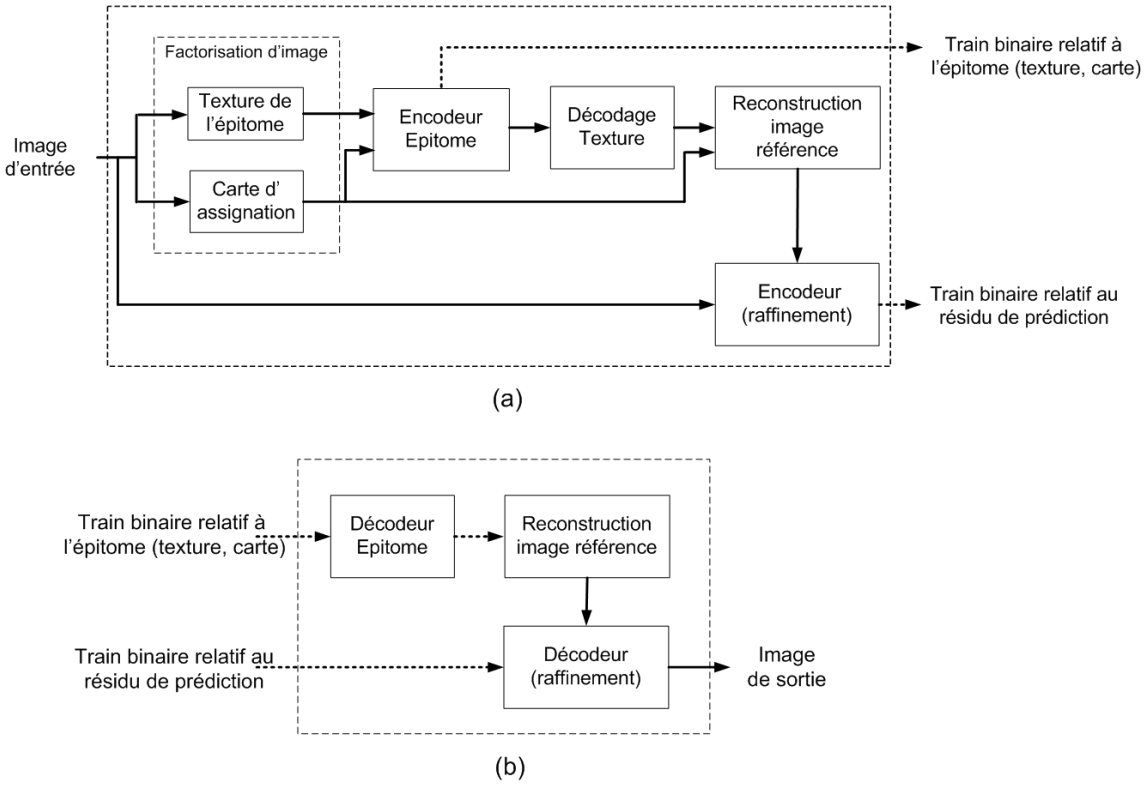

Figure 5.13 – *Synopsis de l'encodeur basé épitome (a), Synopsis du décodeur basé épitome (b)*

reconstruction de l'image, puisqu'il y a très peu de chance que cette information paddée (et codée) se retrouve dans l'image initiale.

Après avoir paddé l'épitome issu de la factorisation et raffiné la carte d'assignation, nous avons décidé d'encoder l'épitome dans sa version "non compactée" (cf. figure 5.15.a). En effet, au lieu de réagencer les structures de l'épitome afin que ce dernier puisse tenir dans une image de taille plus petite, comme cela a été fait dans [WWOH08], nous avons choisi de travailler avec un épitome à la même taille que l'image d'origine puisque les blocs de l'image à reconstruire qui auront un co-localisé dans l'épitome n'auront pas besoin de vecteurs. Ceci permet ainsi d'effectuer du mode "skip" lors de l'encodage de l'image courante. La figure 5.15 illustre la différence entre un épitome dit "compacté" et un épitome dit "non compacté".

Concernant l'encodage de la texture de l'épitome, on peut souligner que la répartition des blocs texture ne se prête pas vraiment bien au standard utilisé. En effet, moins l'épitome contient d'éléments redondants, plus il devient difficile de prédire efficacement les zones de texture présentes dans l'épitome. Ce phénomène se situe principalement sur les zones de frontières de l'épitome.

5.2.1.3 Codage des vecteurs d'assignation (carte d'assignation)

Rappelons que la carte d'assignation résulte de l'outil de factorisation et précise pour chaque bloc de l'image courante, quel patch de texture de l'épitome lui est le plus similaire. L'image de prédiction reconstruite à partir de l'épitome encodé/décodé et de la carte d'assignation est ensuite utilisée comme image de référence pour encoder l'image courante. La transmission de ces données est donc nécessaire pour assurer le bon fonctionnement de l'al-

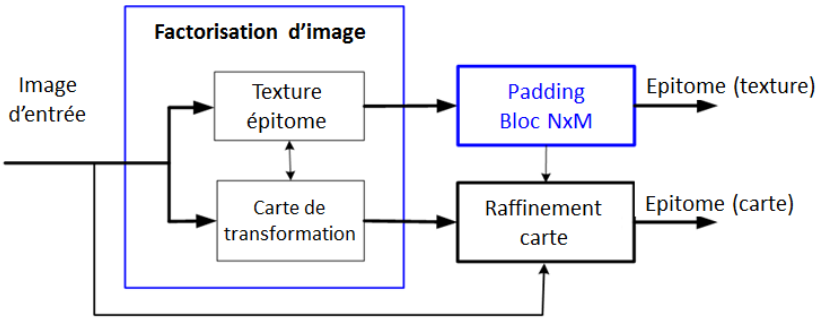

Figure 5.14 – *Padding de l'épitome et raffinement de la carte*

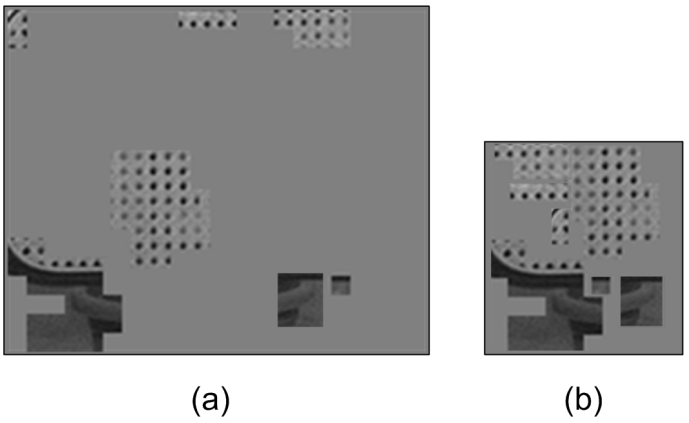

Figure 5.15 – *Illustration d'un épitome dans sa version "non compactée" (a) et dans sa version "compactée" (b).*

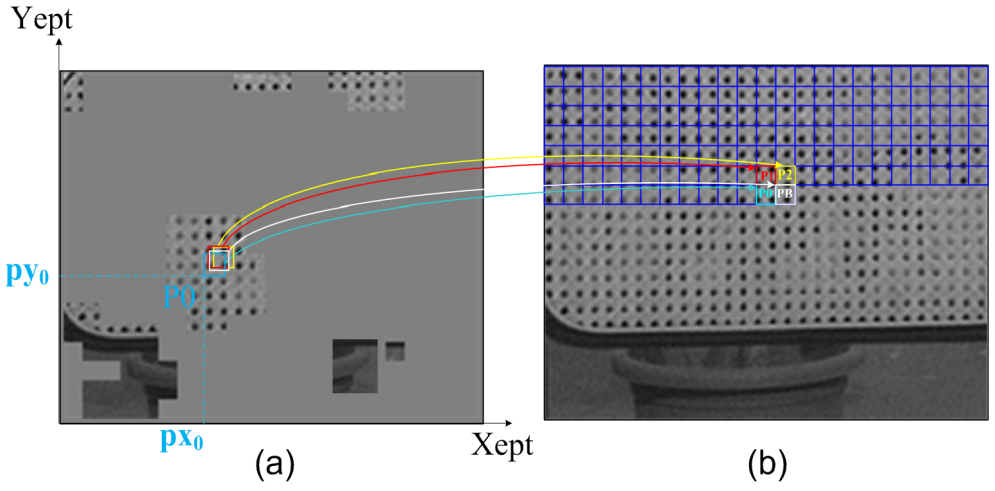

Figure 5.16 – *Principe de prédiction du vecteur d'assignation du bloc courant à partir des vecteurs d'assignation des blocs voisins et de l'image épitome : (a) image épitome, (b) image courante*

gorithme au décodeur. Par ailleurs, il faut souligner que le codage de la carte d'assignation peut être effectué indépendamment du codage de l'image courante.

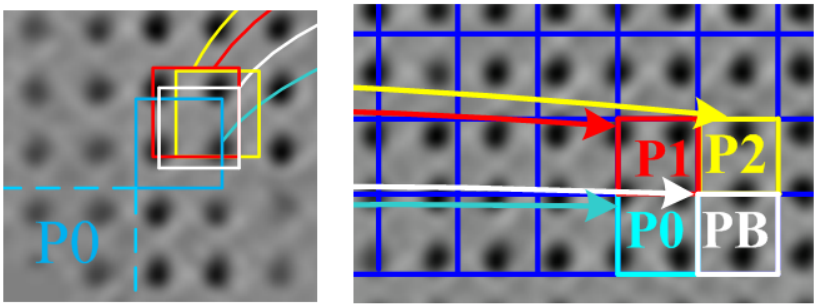

FIGURE 5.17 – *Détails de la prédiction du vecteur d'assignation du bloc courant.*

Le coût de codage d'information associé à la carte d'assignation, dépend de quatre paramètres :

- Le nombre de blocs dans l'image n'appartenant pas à l'épitome (plus le contenu de l'épitome est faible, plus le coût des vecteurs est élevé)
- La taille des blocs utilisés (plus les blocs utilisés sont de petite taille, plus le nombre de vecteurs à coder est élevé)
- La résolution sous-pixellique utilisée pour la prédiction basée épitome (le coût associé à un vecteur à la précision quart de pixel est multiplié par quatre)
- L'entropie de l'information

Dans l'objectif de réduire l'entropie de l'information de la carte d'assignation, nous utilisons un codage prédictif. Ainsi, nous considérons non pas l'entropie de la source mais celle des résidus de prédiction de cette source. L'efficacité de la méthode dépend de la pertinence du prédicteur utilisé. Si on se réfère au codage du vecteur de mouvement en prédiction inter image de MPEG4, afin d'exploiter les redondances spatiales des champs de vecteurs de mouvement, la valeur du prédicteur est fonction de la valeur des vecteurs voisins déjà encodés/décodés. Dans le cadre de MPEG4, ce prédicteur correspond à un médian des vecteurs voisins, en réalité tout est basé sur le fait que localement le champ de mouvement est considéré comme homogène, ce phénomène étant en effet souvent observé.

Pour le codage du vecteur d'assignation à partir d'un épitome de texture, on part d'un principe similaire en ce sens que plus une texture est compactée dans l'épitome plus on espère qu'elle soit redistribuée dans l'image à reconstruire. Autrement dit, vu du bloc courant à reconstruire on espère que les composantes du vecteur d'assignation du bloc courant soient proches de celles des vecteurs d'assignation des blocs voisins. A l'extrême supposons que l'on ait une image composée de textures régulières et d'un épitome constitué d'un seul pattern. Vu de l'image reconstruite, chaque bloc va pointer sur le même pattern de l'épitome alors tous les blocs de l'image reconstruite vont avoir des vecteurs différents vis à vis de l'image épitome, d'où la nécessité de travailler avec la position du bloc pointé dans l'épitome. Ainsi, lorsque l'on va utiliser un prédicteur pour encoder un vecteur d'assignation d'un bloc donné de l'image, ce prédicteur correspondra à la position dans l'épitome du pattern utilisé.

Pour le codage des vecteurs d'assignation, le résidu ou l'erreur de prédiction du vecteur $P_{err} = P_B - P_{pred}$ va être transmis à la place du vecteur P_B. Le prédicteur correspond à un médian des vecteurs voisins. Ainsi, la prédiction $P_{pred} = \binom{px_{pred}}{py_{pred}}$ d'un vecteur P_B d'un bloc B de l'image est donnée par l'expression :

$$P_{pred} = median(P0, P1, P2) \tag{5.6}$$

soit

$$\begin{cases} px_{pred} = median(px_0, px_1, px_2) \\ py_{pred} = median(py_0, py_1, py_2), \end{cases} \tag{5.7}$$

où $P0 = \binom{px_0}{py_0}$, $P1 = \binom{px_1}{py_1}$ et $P2 = \binom{px_2}{py_2}$ correspondent aux composantes des vecteurs d'assignation des blocs voisins utilisés. Le principe de prédiction d'un vecteur d'assignation est également illustré dans les figures 5.16 et 5.17.

Afin de réduire le coût de codage des résidus des vecteurs d'assignation, on utilise les propriétés statistiques des données à encoder, à savoir le codage entropique. Le but du codage entropique est d'adapter le code binaire associé à un symbole à sa probabilité d'apparition. La limite théorique étant donnée par la définition de l'entropie $H = -\sum pk.log_2(pk)$ avec pk la probabilité d'occurrence de la valeur k. Deux principales techniques de codage entropique sont largement utilisées dans les schémas de codage actuels : le codage d'Huffman et le codage arithmétique. Dans notre cas, nous prenons parti d'utiliser le codage d'Huffman pour encoder l'erreur de prédiction en raison de sa simplicité.

5.2.1.4 Codage du résidu (étape de raffinement)

A l'encodeur, une fois que l'image d'origine est factorisée en un épitome de texture et une carte d'assignation, une image de prédiction est construite à partir de la texture de l'épitome décodé et de la carte d'assignation (Figure 5.13.a). Ensuite, cette image de prédiction est utilisée comme une image de référence lorsque nous encodons l'image courante. Notons qu'il est nécessaire de transmettre au décodeur le bistream relatif à l'épitome {texture, carte d'assignation} en plus du bistream lié au résidu de prédiction. Au niveau décodeur, nous décodons dans un premier temps le bistream associé à l'épitome {texture, carte d'assignation} et nous reconstruisons ensuite une image à la résolution spatiale de l'image courante, qui sert d'image de référence pour le décodeur dédié à l'image courante (Figure 5.13.b).

5.2.2 Résultats expérimentaux

Dans un premier temps, l'outil d'extraction d'épitome construit sous la base d'un modèle de translation simple a été comparé à celui développé dans [WWOH08] en mesurant la qualité de l'image prédite sans transmettre un résidu de prédiction. Nous constatons qu'avec l'image test "Building", notre approche permet une réduction de débit de 12.8% lorsque l'on encode l'épitome de texture et la carte d'assignation.

Le schéma de compression basé sur l'approche épitome a été également évalué en se comparant aux modes intra H.264/AVC. La prédiction basée épitome a été intégrée dans le software de référence KTA [sof]. Le tableau 5.2 présente les résultats obtenus avec l'algorithme proposé à précision pixel pour plusieurs images. En utilisant la métrique de Bjontegaard [Bjo01] (QP = 16, 21, 26, 31, 36, 41), nous pouvons constater que le schéma de compression proposé permet de réduire en moyenne le débit d'environ 9.21% en bas débit et de 5.09% en haut débit. Nous pouvons remarquer également que les performances de codage sont améliorées lorsque un épitome est construit à la précision quart de pixel (tableau 5.2). Dans l'algorithme de compression proposé, précisons que l'épitome et le résidu sont encodés ici avec le même pas de quantification.

Image	Taille	ϵ	Patch	Bas débits		Hauts débits	
				PSNR	% débit	PSNR	% débit
PRECISION PIXELLIQUE							
Building	1792x944	14	16x16	+0.69	-12.34	+0.40	-3.80
$Building_{DS}$	896x464	14	8x8	+0.97	-11.56	+0.99	-8.25
Wool	720x576	10	16x16	+0.30	-4.61	+0.25	-2.52
Pan0	176x144	11	8x8	+0.64	-8.33	+0.71	-5.80
PRECISION AU QUART DE PIXEL							
Wool	720x576	10	16x16	+0.50	-7.87	+0.38	-3.91
Pan0	176x144	11	8x8	+1.29	-16.16	+1.21	-10.05

TABLE 5.2 – *Comparaison des performances de l'algorithme de compression proposé à la précision pixellique et celui au quart de pixel par rapport aux modes de prédiction H.264 Intra.*

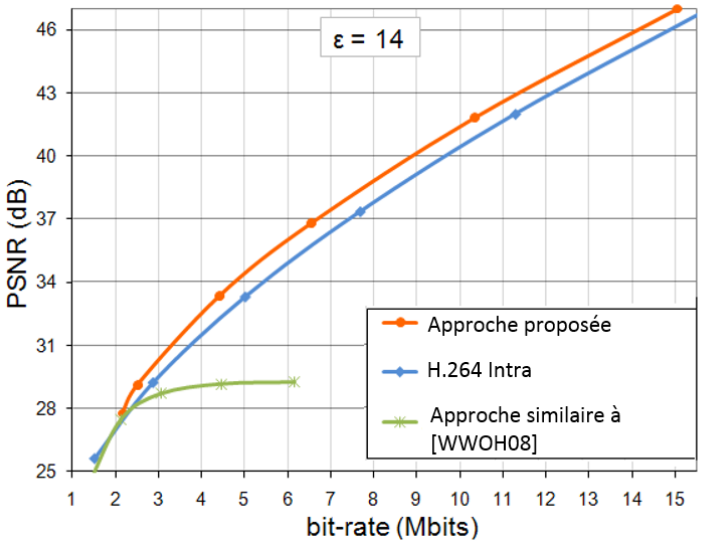

FIGURE 5.18 – *Courbes de performances de codage pour l'image $Building_{DS}$ à la précision pixellique*

La figure 5.18 montre les courbes débit-distorion pour l'algorithme de codage proposé ainsi que pour les modes intra H.264/AVC. Cette figure décrit également une étape intermédiaire représentant la qualité de la prédiction de l'image pour une tolérance d'erreur fixée, en fonction du coût de codage de l'épitome {texture, carte d'assignation}. Cette étape intermédiaire peut être assimilée à l'approche décrite dans [WWOH08]. Sur cette courbe, on constate qu'il est inutile de quantifier plus précisément la texture de l'épitome puisque la qualité de la reconstruction de l'image n'est pas nécessairement meilleure. Ceci est dû au fait que l'on fixe une tolérance d'erreur lors de la construction de l'épitome.

5.2.3 Conclusion

Dans cette section, nous avons introduit une méthode de prédiction basée sur l'approche épitome. Cette nouvelle approche de prédiction offre des résultats intéressants comparée aux modes de prédiction directionnels d'H264. Pour des zones texturées, l'algorithme de compression basé épitome s'avère être une solution alternative pour faire de la prédiction

intra. Cependant, le premier schéma proposé peut être aisément amélioré. En effet, jusqu'à maintenant, la construction de l'épitome est effectuée indépendamment du codage de l'image courante. Par conséquent, certaines zones de l'épitome peuvent ne pas être utilisées lors de l'encodage de l'image source. Par ailleurs, dans l'approche proposée, nous transmettons un vecteur pour chaque bloc de l'image non colocalisé avec l'épitome. Or, lors de l'encodage d'un bloc de l'image courante, la prédiction issue de l'épitome n'est pas toujours retenue. En effet, puisque nous mettons en concurrence ce mode avec ceux des modes directionnels d'H.264, il est possible qu'un mode H.264 soit finalement plus efficace. Les limites soulignées du premier schéma de compression basé épitome nous ont ainsi conduits à nous orienter vers un second schéma de compression décrit dans la section suivante.

5.3 2e approche étudiée : Algorithme de compression orienté « inpainting » via l'épitome et les méthodes de prédiction basées NE/oMANE

5.3.1 Description de l'algorithme

5.3.1.1 Motivations

Dans la section précédente, nous avons présenté un premier schéma de compression basé épitome dans lequel l'image courante est encodée en inter-image par rapport à une image de référence reconstruite à partir de l'image épitome et de la carte d'assignation associée. Malgré des premiers résultats prometteurs, les limites du premier schéma de codage proposé nous ont contraints à revoir la conception du schéma global. En effet, jusqu'à maintenant, un vecteur d'assignation était systématiquement transmis pour chaque bloc de l'image alors que le codeur pouvait retenir un des modes directionnels d'H264 et non la prédiction basée épitome. Afin d'éviter de transmettre inutilement des vecteurs d'assignation, une solution possible serait d'encoder l'image courante directement en référence à l'épitome codé/décodé en autorisant une zone d'excursion sur toute l'image. Dans ce cas, la prédiction résultant de l'image épitome doit être déterminée à l'aide d'un estimateur considérant la position du bloc pointé dans l'épitome et non le vecteur de déplacement. Cependant, le coût des vecteurs d'assignation demeure non négligeable par rapport au débit total. Ce coût est d'autant plus accru que le nombre d'occurrences du mode de prédiction basé épitome est élevé. Une manière possible permettant de restreindre l'utilisation de vecteurs d'assignation consiste à reconstruire des blocs de l'image d'entrée à partir de l'épitome associé en faisant intervenir des méthodes de prédiction s'appuyant uniquement sur le voisinage causal telles que le Template Matching ou d'autres méthodes dérivées (LLE). Dans ce cas, deux types de scanning s'offrent à nous pour encoder l'image courante :

- Soit l'image courante est encodée avec un ordre de scanning classique (i.e. raster scan) dans lequel les modes directionnels d'H264 sont directement mis en concurrence avec différents modes de prédiction issus de l'épitome : un mode de prédiction réalisé à l'aide du template (ie. TM ou LLE) et un mode de prédiction avec vecteur permettant d'effectuer un adressage direct dans l'épitome. Cette approche présente quelques similarités avec celle étudiée dans [WHW09] où une prédiction issue d'un épitome de type [JFK03] est déterminée par template matching.
- Soit nous venons directement encoder les blocs de l'image localisés en dehors de l'épitome en utilisant un ordre de scanning adaptatif. En effet, au lieu de parcourir les blocs de l'image courante en raster scan, il peut être intéressant de venir directement propager l'information de texture issue de l'épitome suivant un scanning dépendant

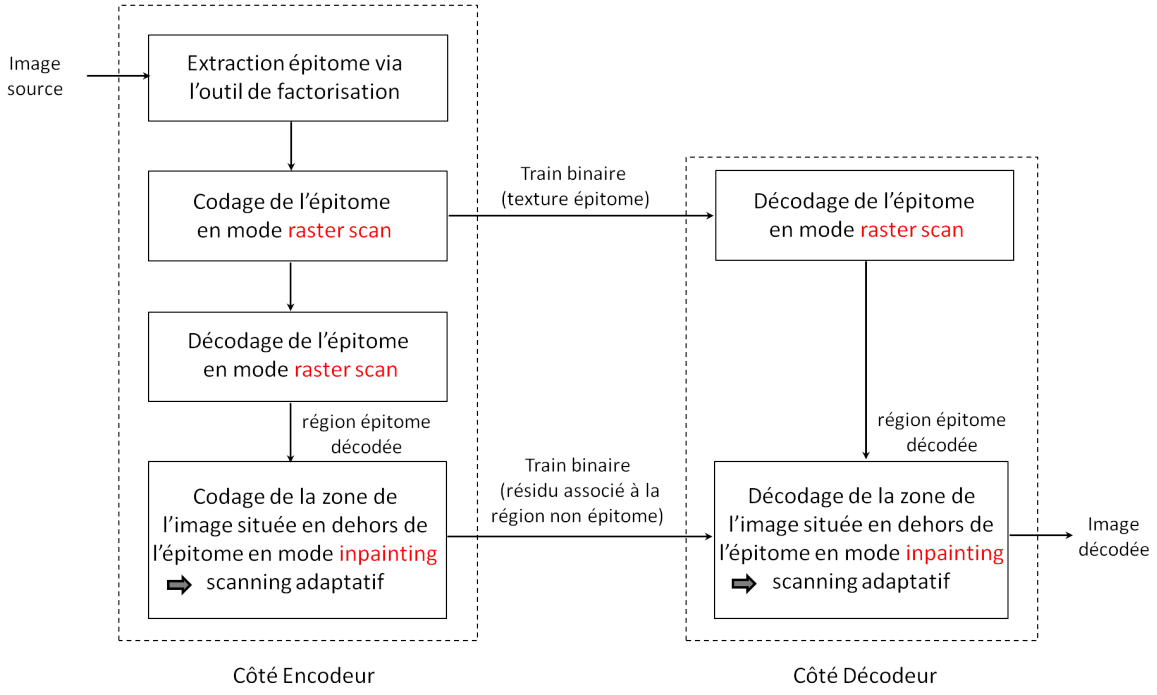

FIGURE 5.19 – *Schéma bloc d'un système d'encodage/décodage basé sur le principe d'inpainting de l'épitome avec codage de l'erreur résiduelle*

du contenu structurel disponible dans l'image jusqu'à reconstruire l'intégralité de l'image. La procédure de reconstruction de l'image s'apparente ainsi à une approche de type inpainting dans laquelle le codage du résidu est également pris en compte dans la boucle de reconstruction afin de permettre la remise à jour du voisinage causal de l'image et une meilleure utilisation du TM/LLE. Dans cette section, nous prenons parti d'explorer cette seconde approche.

5.3.1.2 Schéma global

Le second schéma de compression basé sur le concept d'épitome est décrit par la figure 5.19. L'objectif est de proposer une amélioration du schéma de codage présenté dans la section précédente 5.2. L'idée est de montrer comment après le codage et le décodage de l'épitome, il est possible de prédire par "inpainting" les blocs manquants de l'image.

5.3.1.3 Ordre de scanning adaptatif

L'ordre de scanning, mis en place pour encoder les blocs de l'image localisés en dehors de l'épitome, vise à favoriser la propagation des structures linéaires avant la reconstruction des textures. L'algorithme de remplissage dépend uniquement des priorités attribuées pour chaque bloc de l'image situé à la frontière de la région à remplir. Cet ordre de remplissage est régi par une règle de priorité similaire à celle introduite dans [CPT04] pour la suppression d'objets (par inpainting) dans une image, adapté ici afin qu'il convienne à la structure bloc de l'encodeur. Les blocs se trouvant sur la continuation de forts contours et entourés par des pixels dits de hautes confiances, ont la plus haute priorité et sont d'abord traités.

Notons qu'un certain nombre de templates de différentes formes et de tailles diverses $\Psi_{T_i}, i = 1, ..., n$ peuvent être envisagés pour reconstruire un bloc de l'image. Les différents

templates considérés sont représentés dans la figure 5.20. Le critère de priorité a donc été généralisé ici de façon à ce qu'il puisse considérer l'ensemble des templates disponibles. Ce critère sert donc non seulement à fournir la priorité d'un bloc lors de l'inpainting de l'épitome mais également à déterminer le template qui sera associé au bloc lors de sa prédiction dans le cas où plusieurs templates d'orientations différentes peuvent être disponibles pour le bloc. Notons que ce problème ne se pose pas lorsque l'on traite le premier bloc 8x8 d'un nouveau macrobloc 16x16 mais peut se produire lorsque l'on encode les autres blocs 8x8 du macrobloc courant.

Soit un bloc Ψ_B localisé sur la frontière $\delta\Omega$ de la région à remplir Ω (également appelé "ligne de front"), sa priorité est définie comme suit :

$$P(\Psi_B, \Psi_{T_B}) = \max_{i=1,\dots,n} (\alpha \times \log(C(\Psi_B, \Psi_{T_i})) + (1-\alpha) \times \log(D(\Psi_B, \Psi_{T_i}))) \tag{5.8}$$

Le premier terme $C(\Psi_B, \Psi_{T_i})$ du critère est un terme de confiance de type géométrique et permet de mesurer le degré de fiabilité d'un template donné Ψ_{T_i} pour la prédiction d'un bloc Ψ_B situé sur la frontière $\delta\Omega$. L'objectif est de privilégier la reconstruction de blocs dont le voisinage local considéré présente un grand nombre de pixels connus. L'expression de $C(\Psi_B, \Psi_{T_i})$ est donnée par l'équation (5.9). Ce terme est défini comme le rapport entre le nombre de pixels connus dans le template considéré Ψ_{T_i} par rapport au nombre total de pixels contenus dans le patch entier Ψ_{B+T_i}.

$$C(\Psi_B, \Psi_{T_i}) = \frac{card(\Psi_{T_i})}{card(\Psi_{B+T_i})} \tag{5.9}$$

Le second terme $D(\Psi_B, \Psi_{T_i})$ est une fonction qui spécifie la force des isophotes arrivant sur la frontière $\delta\Omega$ qui sépare le template considéré et le bloc à inpainter. Ce terme vient augmenter la priorité des blocs qui présentent des isophotes orthogonaux à la ligne de front $\delta\Omega$. Il joue ainsi un rôle fondamental dans l'ordre de reconstruction des blocs puisqu'il vient favoriser la reconstruction des structures linéaires. L'expression du terme $D(\Psi_B, \Psi_{T_i})$ est fournie par l'équation suivante :

$$D(\Psi_B, \Psi_{T_i}) = \max(D(p), p \in (\delta\Omega \cap \ \Psi_{B+T_i})) \text{ with } D(p) = \frac{\left|\nabla I_p^{\perp} \cdot n_p\right|}{\sigma} \tag{5.10}$$

où σ est un facteur de normalisation (i.e. $\sigma = 255$ pour une image en niveaux de gris) et n_p est un vecteur unitaire orthogonal à la ligne de front $\delta\Omega$ au point p. Le terme $D(p)$ est calculé ici sur l'ensemble des pixels de la ligne de front (séparant le template considéré et le bloc à inpainter) et est défini entre $[0, 1]$. Par conséquent, lorsque $D(p) \to 1$, cela signifie que Ψ_{T_i} possède un isophote perpendiculaire à la ligne de front au point p.

Le paramètre α de l'équation (5.8) contrôle le poids accordé à chacun des termes $C(\Psi_B, \Psi_{T_i})$ et $D(\Psi_B, \Psi_{T_i})$. Ce paramètre permet ainsi d'ajuster le critère de priorité basé confiance/isophote à des fins de compression. L'idée est de pouvoir agir sur le poids de chacun des termes du critère de priorité en vue d'améliorer les performances de codage en termes débit/distorsion. Rappelons que le critère de priorité initial, introduit par Criminisi dans [CPT04], est défini comme le produit du terme de confiance $C(.)$ par le terme spécifiant la force des isophotes $D(.)$. Pour notre application, il a donc fallu transformer le critère initial défini dans [CPT04], d'où l'utilisation du logarithme dans l'équation (5.8). Notons que lorsque α est égal à 0.5, le critère revient à celui défini dans [CPT04].

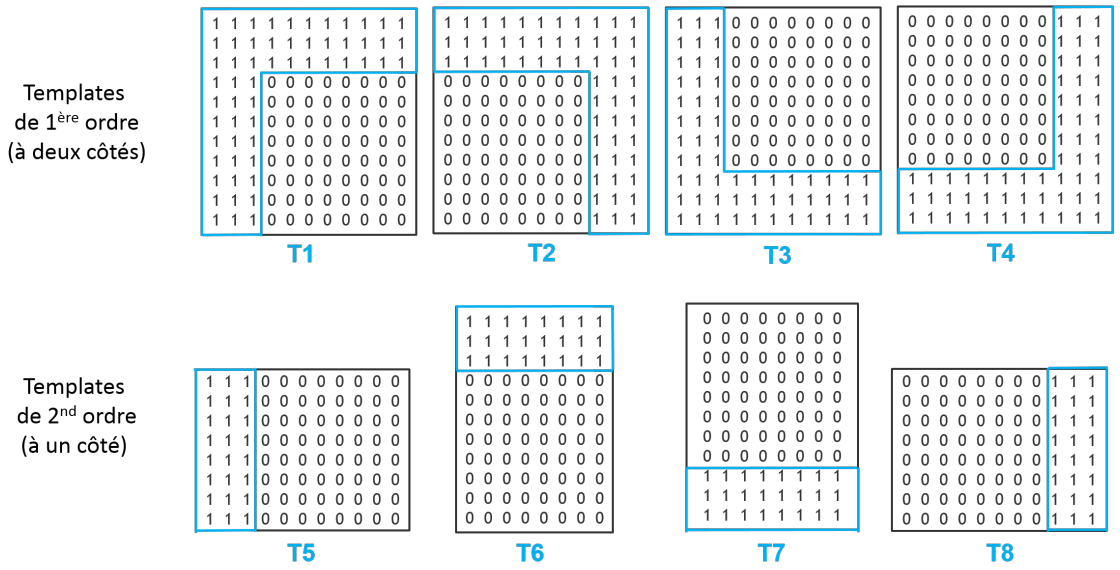

FIGURE 5.20 – *Ensemble des tempates considérés pour calculer la priorité d'un bloc Ψ_B localisé sur la frontière de la région à remplir $\delta\Omega$.*

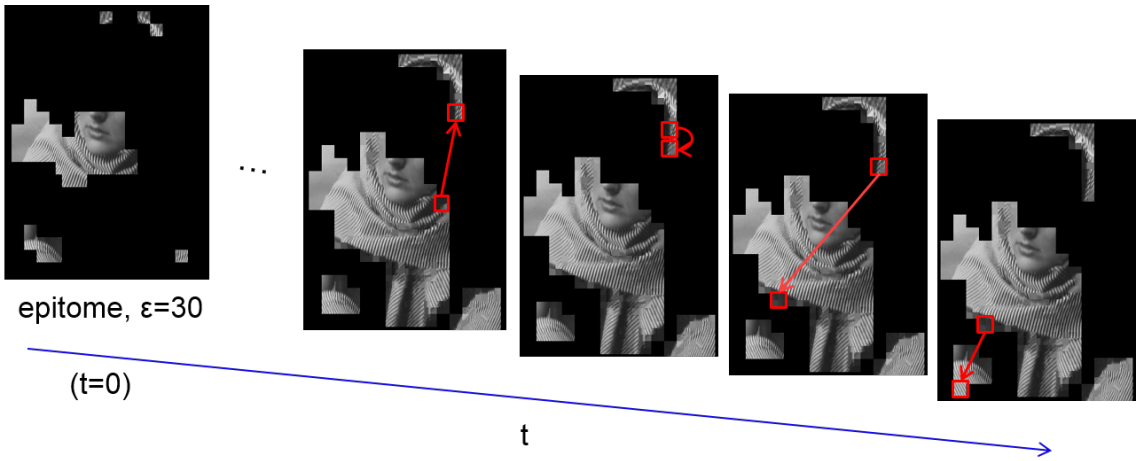

FIGURE 5.21 – *Illustration de l'évolution du scanning adaptatif défini selon une règle de priorité basée confiance/isophote [CPT04] en partant d'un épitome de l'image Barbara.*

La figure 5.21 illustre la progression du scanning des macroblocs lorsqu'un critère de priorité basé confiance/isophote similaire à celui proposé dans [CPT04] est considéré en partant d'un épitome de l'image Barbara. Nous constatons que l'ordre de scanning varie en fonction des structures présentes sur les frontières de l'épitome puis de celles contenues dans l'image en cours de reconstruction. En effet, le critère utilisé vient d'abord favoriser la reconstruction de blocs dont le voisinage dispose de forts contours (i.e. les structures) pour ensuite traiter les blocs dont le voisinage présente de faibles gradients (i.e. les textures). On notera que du fait que l'ordre de reconstruction des blocs est géré par une fonction de priorité $P(\Psi_B, \Psi_T)$ calculée uniquement à partir de la région précédemment encodée, le décodeur peut effectuer le même scanning sans avoir besoin d'aucune information supplémentaire.

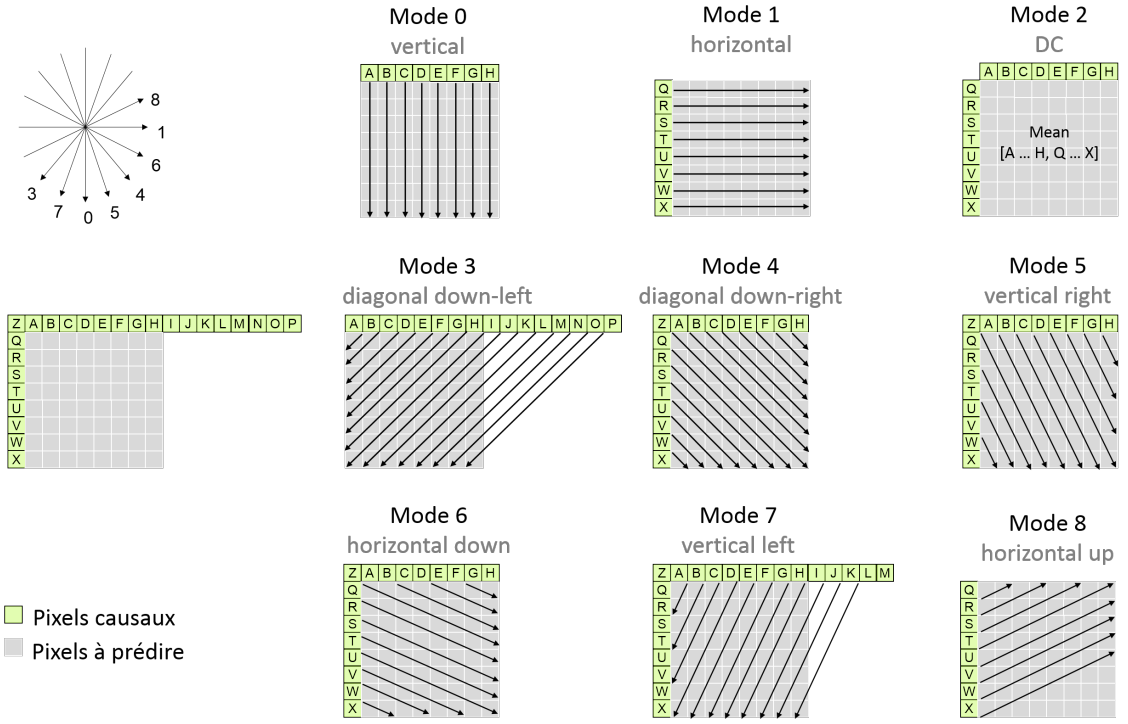

Figure 5.22 – *Les modes de prédiction intra 8x8 du standard H.264 opérant avec un voisinage localisé en haut à gauche.*

5.3.1.4 Adaptation des modes de prédiction intra H.264 en fonction du voisinage considéré

Dans le schéma proposé fondé sur "l'inpainting" de l'épitome avec codage du résidu, nous souhaitons exploiter les modes directionnels définis dans la norme H.264 afin de prédire la région localisée à l'extérieur de la zone épitome lorsque les modes de prédiction basés template ne conviendraient pas.

L'élaboration d'un tel schéma nécessite de modifier les outils de prédiction intra H.264 afin qu'ils puissent être utilisés dans un esprit "inpainting". En effet, pour assurer un bon fonctionnement de l'algorithme, quatre orientations de voisinage doivent être prises en compte : la zone située en haut à gauche, celle en haut à droite, celle en bas à gauche et celle en bas à droite (cf. figure 5.20). Lors du processus de reconstruction de l'image par "inpainting", on peut donc être amené à considérer un voisinage causal dont l'orientation diffère de celle habituellement utilisée en codage intra classique, à savoir le voisinage situé en haut à gauche. Les modes de prédiction intra du standard H.264 opérant uniquement avec le voisinage causal classique, il est donc nécessaire d'adapter ces modes directionnels selon l'orientation du voisinage causal considéré.

Afin d'y parvenir, un ensemble de modes directionnels associés à chaque orientation du voisinage sont générés à partir d'une transformation géométrique appliquée sur les modes directionnels existants de la norme H.264. En effet, des modes analogues au standard sont obtenus pour les trois autres orientations en appliquant respectivement une opération de rotation de 90°, 180° et 270° sur les modes intra classiques. Par ailleurs, afin de ne pas altérer l'efficacité du codeur lors de l'encodage des modes de prédiction directionnels, il est nécessaire que les modes directionnels inhérents à chaque orientation du voisinage soient

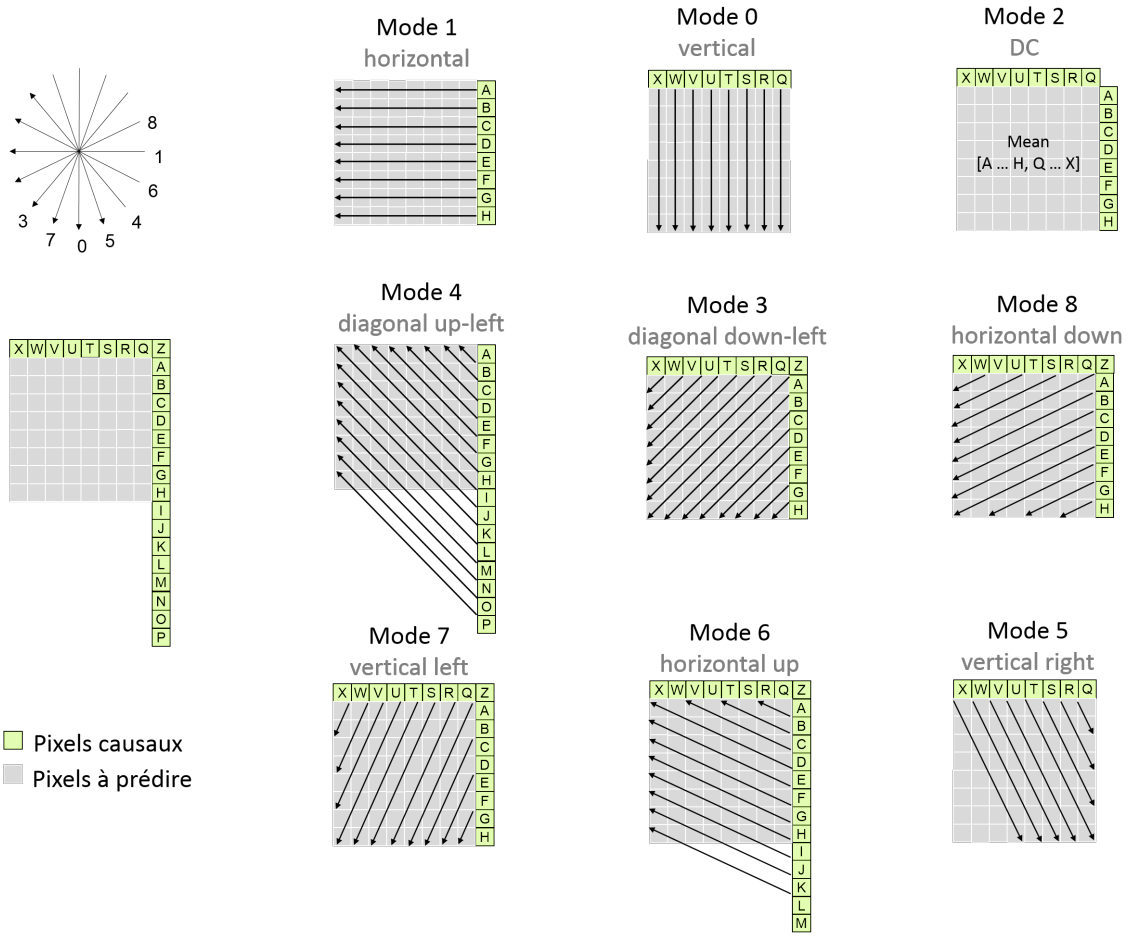

Figure 5.23 – *Les modes directionnels associés à un voisinage situé en haut à droite. Ces modes sont invariants à une opération de rotation de 90° sur les modes directionnels existants de la norme H.264.*

cohérents avec ceux utilisés en raster scan. Il a donc également fallu faire en sorte de rendre les modes de prédiction intra d'H.264 invariants aux rotations de 90°, 180° et 270°. La figure 5.22 rappelle les modes de prédiction intra 8x8 définis dans le standard H.264. La figure 5.23 représente les modes directionnels analogues déterminés pour un voisinage situé en haut à droite.

5.3.1.5 Codage de la texture de l'épitome via les méthodes LLE/oMALLE en raster scan

Rappelons que l'épitome a été construit dans le but de limiter la quantité d'information redondante dans l'image d'origine. Cependant, nous constatons que certaines corrélations subsistent toujours dans l'épitome. Il est ainsi possible de réduire davantage le coût de codage de la texture de l'épitome en exploitant d'autres outils de prédiction plus efficaces pour le codage de la texture 2D que ceux de H.264/AVC. Les résultats obtenus dans le chapitre 4 ont montré qu'en scanning order classique, l'utilisation conjointe des outils de prédiction basés LLE et oMALLE permet de réduire de façon significative le coût de codage par rapport au standard H.264/AVC. Ceci nous a donc incités à nous orienter vers l'utilisation de ces mêmes outils pour encoder non seulement l'image épitome mais

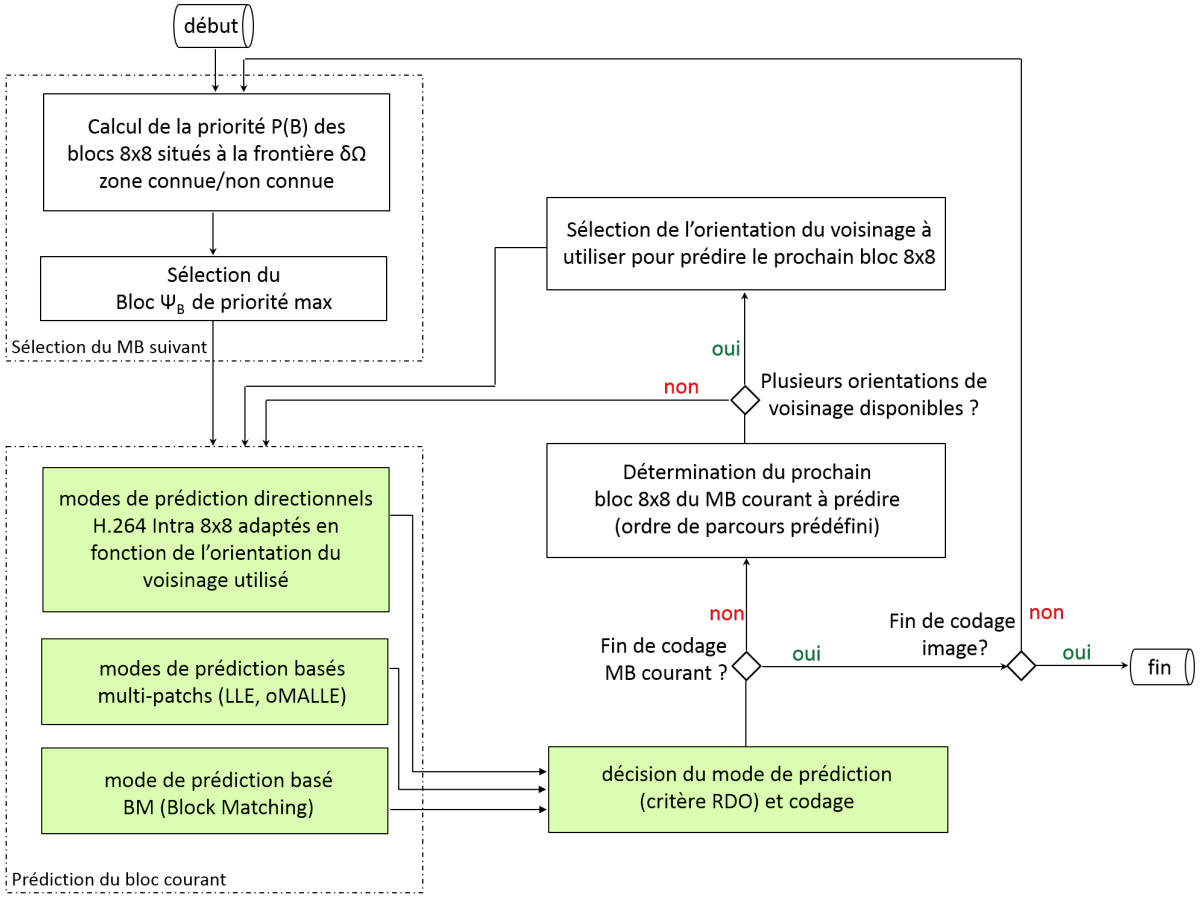

FIGURE 5.24 – *Principe de sélection et de codage des macroblocs dans le cas d'un schéma de compression orienté inpainting avec codage du résidu*

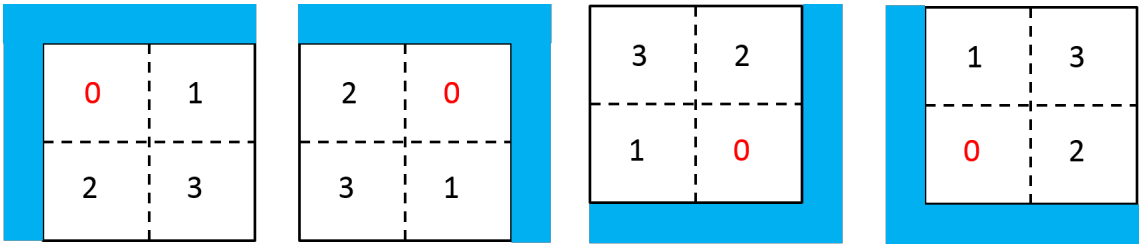

FIGURE 5.25 – *Ordre de parcours des partitions de l'intra 8x8 défini en fonction de la localisation du premier bloc encodé au sein du macrobloc courant. Le signal de référence assimilé au premier bloc 8x8 encodé est représenté en bleu. Les numéros à l'intérieur des blocs correspondent à l'ordre de parcours.*

également, comme nous le verrons plus tard, le reste de l'image.

5.3.1.6 Codage du résidu via les méthodes LLE/oMALLE en mode inpainting

A l'encodeur, après avoir extrait un épitome de l'image source via l'outil de factorisation décrit dans la section 5.1 puis encodé/décodé l'épitome suivant un ordre de parcours "raster scan", le reste de l'image est ensuite encodé en adoptant un ordre de scanning adaptatif

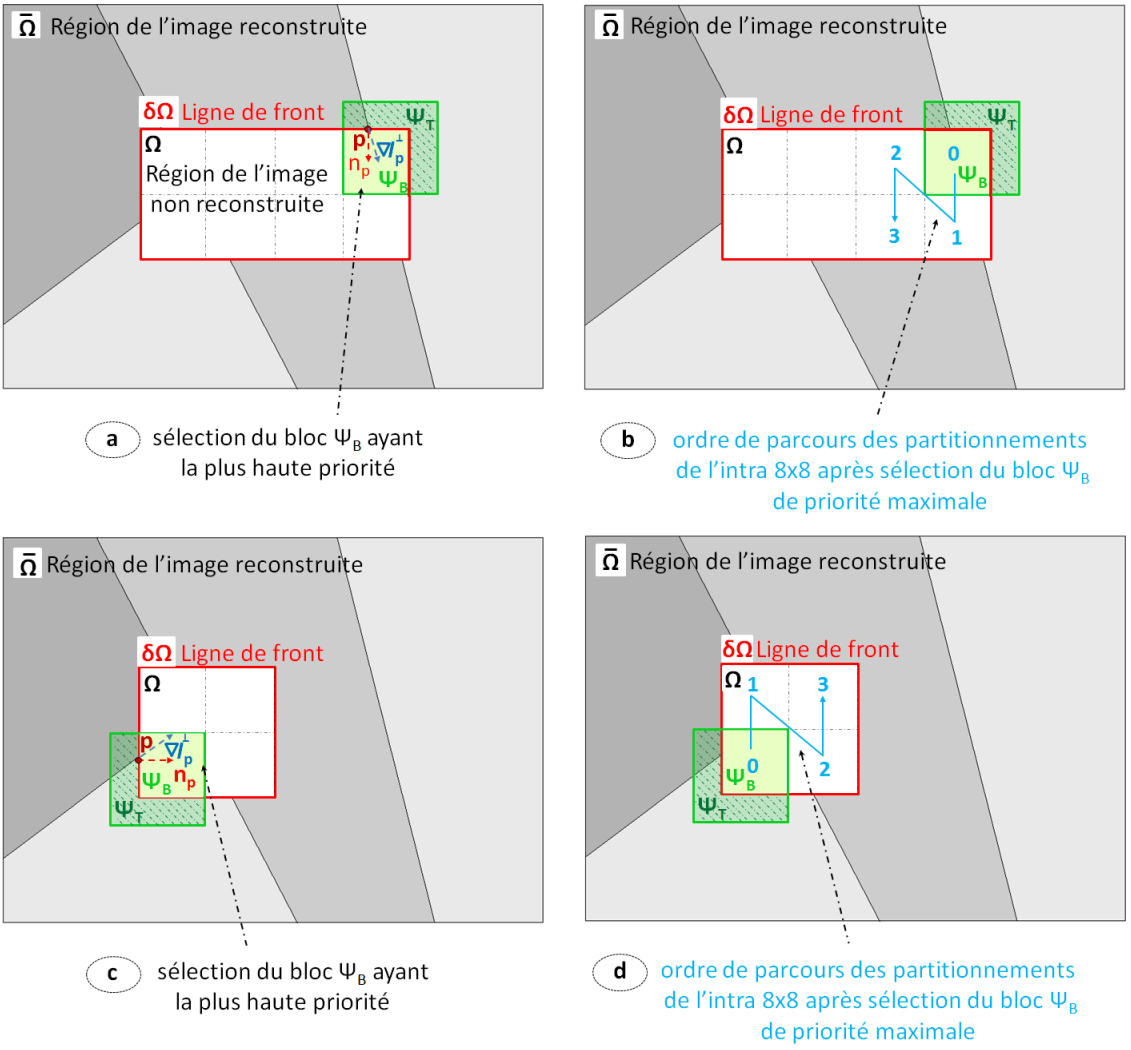

FIGURE 5.26 – *Illustration du principe de reconstruction de macroblocs dans le cas d'un scanning order adaptatif. Deux cas sont présentés : (a,b) et (c,d)*

et selon un esprit "inpainting" (Figure 5.19). Au niveau décodeur, l'épitome est dans un premier temps décodé. Ensuite, le reste de l'image est décodé en suivant le même scanning adaptatif déterminé à l'encodeur (Figure 5.19).

Nous allons décrire maintenant plus précisément le principe de sélection et de codage des macroblocs situés en dehors de la zone épitome (Figure 5.24). Afin de déterminer le prochain macrobloc à encoder, nous venons calculer la priorité sur l'ensemble des blocs 8x8 localisés sur la frontière $\delta\Omega$ délimitant la zone connue/non connue (i.e. décodée/à coder). Ensuite, après avoir sélectionné le bloc de priorité maximale Ψ_B, nous venons prédire le bloc en mettant en concurrence les modes de prédiction directionnels H.264 Intra 8x8 adaptés, les outils de prédiction basés multi-patches (LLE, oMALLE) ainsi que la méthode de prédiction basée Block Matching. Le codeur vient ensuite sélectionner le mode de prédiction qui permet d'obtenir les meilleures performances RDO. Une fois le bloc Ψ_B encodé, nous venons traiter l'intégralité des blocs restants du macrobloc courant. Dans l'implémentation actuelle, nous avons choisi de définir l'ordre de parcours des blocs en fonction de la localisation du premier bloc Ψ_B encodé au sein du macrobloc courant. La figure 5.25

indique les quatre ordres selon lesquels on peut parcourir les différentes sous partitions d'un macrobloc. Comme pour l'adaptation des modes de prédiction directionnels, les trois derniers ordres de parcours sont définis ici en réalisant respectivement une opération de rotation de 90°, 180° et 270° sur l'ordre de parcours adopté dans le standard H.264. Par ailleurs, on peut noter que lorsque plusieurs orientations de voisinage sont disponibles pour encoder le prochain bloc 8x8 du macrobloc courant, nous avons utilisé la même règle de priorité pour déterminer le voisinage à prendre en compte pour prédire le bloc. En effet, le template fournissant la priorité maximale servira de support de prédiction pour le prochain bloc 8x8 à encoder. La figure 5.26 illustre le principe de scanning order adaptatif adopté dans le nouveau schéma pour reconstruire les macroblocs restants de l'image.

5.3.2 Résulats expérimentaux

Dans un premier temps, le schéma de compression fondé sur l'inpainting de l'épitome avec codage du résidu a été évalué en se comparant aux modes intra H.264/AVC. Comme pour le premier schéma basé épitome introduit dans la section précédente 5.2, l'épitome et le résidu ont été encodés avec le même pas de quantification. Le tableau 5.3 présente les résultats obtenus avec l'algorithme proposé pour plusieurs images et pour α=0.5 et α=0.9. En utilisant la métrique de Bjontegaard [Bjo01] (QP = 16, 21, 26, 31, 36, 41), nous pouvons constater que le schéma de compression proposé permet de réduire en moyenne le débit d'environ 32.34% en bas débit et de 34.42% en haut débit lorsque $\alpha = 0.5$. Par ailleurs, nous pouvons remarquer que les performances de codage sont légèrement meilleures pour $\alpha = 0.9$.

En se référant aux résultats obtenus avec le premier schéma de codage basé épitome (tableau 5.2) pour les images "Building (DS)" et "Wool", on peut s'apercevoir que le second schéma proposé permet d'améliorer de manière significative l'efficacité de codage. En effet, en prenant l'exemple de l'image "Wool", on peut constater que le pourcentage de réduction de débit engendré par la première méthode de compression basée épitome est de 4.61% à bas débit et de 2.52% à haut débit tandis que le second outil de compression proposé permet d'atteindre un pourcentage de gain en débit de 33.39% à bas débit et de 19.39% à haut débit.

Lors de l'analyse des résultats de simulation, nous avons également pu constater que le scanning order adaptative est différent selon le pas de quantification que l'on utilise pour encoder le résidu. Ceci est assez cohérent puisque lorsque l'on quantifie grossièrement le résidu, on va introduire des micro-structures parasites dues à des effets de blocs qui n'étaient pas initialement présentes dans l'image. Par ailleurs, certaines structures initialement présentes vont disparaître en raison d'une forte quantification.

Il est également intéressant de comparer les perfomances de codage obtenues avec le schéma opérant en scanning order classique et sans épitome puis celui fondé sur l'inpainting de l'épitome avec un scanning order adaptatif, les deux schémas intégrant les mêmes outils de prédiction (méthode hybride LLE/oMALLE, Block Matching et modes AVC). Le tableau 5.4 présente les résultas obtenus avec les deux outils de codage. On peut remarquer que globalement l'approche avec le scanning order adaptif permet d'obtenir des résultats sensiblement meilleurs que l'algorithme opérant en raster scan. Pour que l'étude soit complète, nous avons également comparé la qualité de la prédiction obtenue sur la région localisée en dehors de la zone épitome avec ces deux types de schémas de codage. La figure 5.27 présente les résultats de prédiction obtenus pour l'image Spincalendar lorsque

	Epitome + scan adaptatif			
Low bit rates	$\alpha = 0.5$		$\alpha = 0.9$	
QP :26,31,36,41	dB	% rate	dB	% rate
Barbara	0.98	-14.43	1.13	-16.16
Building (DS)	4.16	-46.11	4.40	-48.12
Spincalendar	2.87	-37.28	3.15	-39.99
Wool	2.19	-31.55	2.33	-33.39
Average	**2.55**	**-32.34**	**2.75**	**-34.42**

	Epitome + scan adaptatif			
High bit rates	$\alpha = 0.5$		$\alpha = 0.9$	
QP :16,21,26,31	dB	% rate	dB	% rate
Barbara	0.81	-9.37	0.92	-10.58
Building (DS)	4.33	-32.79	4.48	-33.89
Spincalendar	2.74	-27.61	2.97	-29.92
Wool	1.69	-18.21	1.81	-19.39
Average	**2.39**	**-21.99**	**2.54**	**-23.44**

Table 5.3 – *Comparaison des performances de codage obtenues avec le schéma de compression basé sur l'inpainting de l'épitome avec codage du résidu par rapport au standard H.264 Intra* 8*x*8.

	raster		scan. adapt.	
Low bit rates	scan		$\alpha = 0.9$	
QP :26,31,36,41	dB	% rate	dB	% rate
Barbara	1.19	-17.07	1.13	-16.16
Building (DS)	4.23	-46.78	4.40	-48.12
Spincalendar	2.87	-37.59	3.15	-39.99
Wool	2.28	-32.73	2.33	-33.39
Average	**2.64**	**-33.54**	**2.75**	**-34.42**

	raster		scan. adapt.	
High bit rates	scan		$\alpha = 0.9$	
QP :16,21,26,31	dB	% rate	dB	% rate
Barbara	0.92	-10.53	0.92	-10.58
Building (DS)	4.32	-32.80	4.48	-33.89
Spincalendar	2.64	-26.83	2.97	-29.92
Wool	1.76	-18.93	1.81	-19.39
Average	**2.41**	**-22.28**	**2.54**	**-23.44**

Table 5.4 – *Comparaison des perfomances de codage de deux types de schéma, à savoir le schéma opérant en scanning order classique et sans épitome puis celui fondé sur l'inpainting de l'épitome avec un scanning order adaptatif, les deux schémas intégrant les mêmes outils de prédiction (méthode hybride LLE/oMALLE, Block Matching et modes AVC).*

l'erreur résiduelle est encodée avec un $QP = 16$. Nous remarquons que la qualité de prédiction obtenue avec l'approche basée sur l'inpainting de l'épitome est meilleure que celle fournie en raster scan. Ceci démontre l'intérêt d'utiliser un épitome car il permet d'apporter de l'information supplémentaire pour la prédiction par rapport au voisinage causal considéré en raster scan.

5.3.3 Conclusion

Cette section a permis d'introduire un nouveau schéma de compression basé épitome dans lequel on vient reconstruire l'intégralité de l'image en propageant directement l'information de texture issue de l'épitome encodé/décodé suivant un scanning order adaptatif basé Criminisi puis en encodant l'erreur résiduelle générée. Les outils de prédiction étudiés dans ce nouveau schéma incluent la méthode hybride LLE/oMALLE, le BM ainsi que les modes AVC. L'objectif de ce nouveau schéma étant d'améliorer le premier schéma de compression basé épitome proposé dans ce document. Afin de mettre en place ce nouvel algorithme, il a fallu en premier lieu casser le traitement "raster scan" du codeur H.264 afin qu'il puisse opérer selon un scanning order adaptatif. La rupture avec l'ordre raster scan a été un point délicat à mettre en oeuvre. En effet, l'implémentation d'une telle approche a engendré un certain nombre de modifications au sein du codeur H.264 afin qu'il puisse prendre en compte l'ensemble des orientations du voisinage causal susceptibles d'être utilisé, notamment au niveau des modes directionnels d'H264 lors de la prédiction spatiale mais également au niveau des contextes lors du codage entropique (CABAC). La rupture

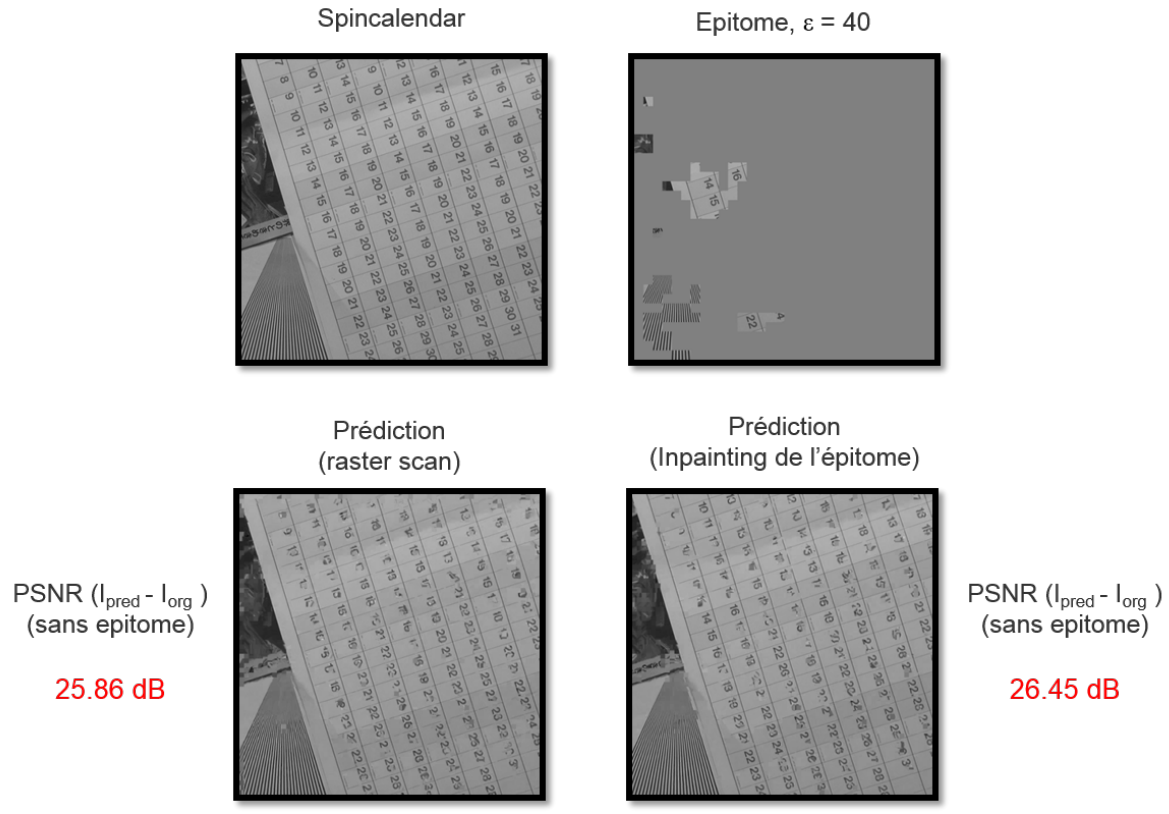

Figure 5.27 – *Résultats de prédiction obtenues avec un scanning order classique et un schéma basé sur l'inpainting de l'épitome pour l'image Spincalendar (QP=16).*

avec le raster scan permet d'apporter une grande avancée par rapport aux précédents standards de compression puisque l'on peut dorénavant adapter le scanning des blocs de l'image en fonction d'un critère spécifique donné.

Les résultats d'expérimentation obtenus avec ce nouveau schéma de codage sont assez encourageants. En effet, ce nouveau schéma permet d'améliorer significativement les performances de compression par rapport non seulement au codeur H.264 mais également au premier schéma de codage basé épitome. Par ailleurs, sur la base des premiers résultats obtenus, nous avons montré que le scanning order adaptatif basé Criminisi permettait d'apporter une légère amélioration par rapport à un schéma de codage intégrant les mêmes outils de prédiction mais dans lequel on fonctionne en raster scan.

Conclusion et perspectives

Rappel des objectifs

L'objet de cette thèse a été la recherche et l'évaluation de techniques pour améliorer les performances en termes de prédiction d'un codeur numérique existant. Ainsi dans le domaine de la compression vidéo, outre la signalétique associée à la structure et aux modes de codage de l'image, la compression en tant que telle concerne principalement l'erreur résiduelle de prédiction, d'une certaine manière on peut considérer cette erreur résiduelle de prédiction comme étant la conséquence des limitations algorithmiques des méthodes de prédiction existantes qu'elles soient de type spatial ou temporel.

L'enjeu a consisté dans une première partie à investiguer, dans un schéma de codage conventionnel des techniques de prédiction qui visent à reproduire et propager des structures et des textures rencontrées dans les images et les vidéos. Notre choix s'est clairement orienté vers les méthodes basées patch et multi-patches de complexités variables, ces outils algorithmiques étant fortement inspirés des techniques utilisées dans les domaines du masquage d'erreur, du remplissage de texture ("inpainting") voir du débruitage. Par ailleurs, certains des modes de prédiction étudiés présentent la particularité intéressante, dans ce contexte de codage vidéo, de ne pas demander d'information additionnelle de signalisation particulière, hormis le mode en question.

L'autre partie de la thèse s'est plus focalisée sur le concept d'épitome dont le rôle est de factoriser une image ou une vidéo en termes de structures, texture et singularités. Ce concept d'épitome a été intégré dans deux schémas de compression, l'un de ces algorithmes s'avère vraiment en rupture avec les techniques traditionnelles dans la mesure où le traitement, par le codeur et le décodeur, des entités (blocs) composant l'image, est réalisé dans un ordre spatial qui dépend du contenu et cela dans un souci de propagation des structures de l'image. Dans ce dernier algorithme de compression les méthodes de prédiction multi-patches y ont été également introduites.

Ces différentes solutions ont été intégrées dans l'un des encodeurs H.264/AVC existant afin d'évaluer leurs performances en tenant compte des contraintes de débit et de distorsion, ainsi que de l'état de l'art relatif à ses différentes techniques.

Rappel des travaux réalisés

Approches conventionnelles basées multi-patches

Correspondance de gabarit et de bloc (Template Matching et Block matching) : la première approche a consisté à intégrer dans un codeur MPEG-4 AVC (KTA) une technique de prédiction hybride en ce sens qu'aux modes de prédiction spatiale mono-directionnelle ont été ajoutées les prédictions basées block matching (BM) et template matching (TM). Ces dernières méthodes permettent de fournir, lorsque cela est possible, des prédictions de contenus plus complexes que celles offertes par les standards de compression actuels. Par ailleurs une des vertus du TM est en effet de ne pas demander de syntaxe spécifique supplémentaire, le décodeur réalisant la même recherche de prédiction que le codeur. Comparativement à MPEG-4 AVC, les gains moyens à bas débits ont été de 17.6%, 20.6% et 24.78% respectivement pour le BM, le TM et le TM+BM sur le panel de séquences testées. L'intérêt principal de cette première contribution est en fait d'allier au TM le BM, permettant ainsi des gains significatifs proches de 25%.

Multi-patches : suite aux premiers résultats relatifs au TM+BM et aux travaux initiés dans [TG12] en prédiction multi-patches de type "neighbor embedding" (NE) appliquée à la prédiction intra, il nous a semblé évident d'investiguer plus en profondeur la prédiction multi-patches. Cet algorithme basé NE permet en effet d'approximer un bloc d'image via une combinaison linéaire estimée sur ses K templates plus proches voisins (K-NN) du template courant, cette combinaison linéaire étant en définitive appliquée aux blocs adjacents aux K templates. Deux méthodes d'approximation NE issues des techniques de réduction de dimensionnalité ont été utilisées, d'une part le Locally Linear Embedding (LLE) et d'autre part le Non Negative Matrix Factorization (NMF). Bien que multi-patches, cette prédiction basée NE est dans le principe très proche de l'algorithme TM en ce sens que la recherche des K-NN est exclusivement liée au template (du bloc courant) avec lequel la recherche des K-NN est effectuée. De fait, cette recherche peut engendrer des inconsistances au sein des patches (template + bloc), et plus particulièrement au niveau des blocs alors que le matching est tout à fait cohérent au niveau des templates. L'approximation étant estimée sur les K-NN, la prédiction est alors entachée d'erreurs inhérentes aux inconsistances contenues dans certains blocs. Pour pallier ce problème d'inconsistances nous avons proposé deux améliorations dont le fil conducteur réside dans le fait que la recherche des K-NN doit être amorcée par le bloc courant et son template (appelé patch courant), la suite revenant à l'association d'un sous-ensemble de K-NN patches, dédiés au calcul de la prédiction, à un vecteur relatif au matching du patch courant et du premier patch du sous-ensemble. Cette première amélioration (MANE) a fourni des résultats supérieurs à la technique TM+BM, la deuxième plus aboutie (oMANE), qui consiste à déterminer le meilleur sous-ensemble (via un critère RDO), a encore accru les performances. Concernant cette dernière méthode oMANE, des optimisations intéressantes ont été réalisées en particulier sur le nombre de patches du K-NN à retenir pour l'approximation du bloc courant, et sur la réduction du coût du vecteur associé au sous-ensemble de patches. Cette réduction de coût s'est principalement opérée par la constitution d'un dictionnaire réduit dédié aux premiers patches des sous-ensembles, de même, une optimisation de la taille de ce dictionnaire réduit a été faite. Il est important de mentionner que ce dictionnaire réduit permet de réduire considérablement le temps de calcul de ce type de prédiction, sachant que le décodeur crée également son dictionnaire puis opère le même traitement, une fois le vecteur décodé.

Approches nouvelles basées épitome

Dans cette deuxième partie, nous nous sommes intéressés à la notion de factorisation d'image avec l'espoir, dans un contexte de compression vidéo, que cette factorisation nous permette de reconstruire l'image initiale. Cette approche nous a amenés à deux schémas de compression de conceptions bien différentes. Tout d'abord la factorisation de l'image s'est effectuée via un épitome, celui-ci a été en quelque sorte élaboré à partir d'un algorithme de correspondance basé translation pure au pel entier ainsi qu'au 1/4 de pel. Un soin tout particulier a été apporté à la construction de l'épitome via les blocs "inférents" ainsi qu'à l'utilisation des interpolateurs au 1/4 de pel. Dans les faits, l'épitome résultant contient un concentré des structures spatialement répétitives ainsi que les singularités de l'image.

Epitome/Carte de vecteurs : le premier schéma de codage s'est inspiré des travaux de [WWOH08], où l'épitome est composé de la texture et d'une carte de vecteurs d'assignation permettant de reconstruire l'image initiale (aux erreurs de correspondance près). Dans cette première méthode, nous avons donc encodé l'épitome (texture + vecteur d'assignation), puis cet épitome codé et décodé est utilisé pour reconstruire l'image, reste alors à encoder l'erreur résiduelle de prédiction entre cette image reconstruite et l'image courante originale. Ces premiers travaux, quant à l'utilisation de l'épitome ont donné des résultats probants comparés à MPEG-4 AVC et [WWOH08], néanmoins bien en deçà de ceux obtenus avec les méthodes multi-patches intégrées dans un codeur conventionnel de la première partie. Ce déficit de performance (vs multi-patches basés NE) vient évidemment de la non utilisation de méthodes de prédiction multi-patches mais aussi du coût de la carte de vecteurs d'assignation utilisée pour construire l'image de prédiction. Cette analyse nous a amenés à la conception d'un autre schéma de codage cette fois ci plus en rupture avec les techniques conventionnelles.

Epitome inpainting : ce deuxième et dernier schéma de codage basé épitome représente à lui seul la synthèse de l'ensemble des algorithmes étudiés dans ces travaux de thèse qui pourrait se résumer en trois mots clés : épitome, prédiction basée NE, inpainting. L'idée sous-jacente est la suivante : à partir d'un épitome codé puis décodé, définir une méthodologie de reconstruction itérative des blocs de l'image situés sur les frontières de l'épitome. Cette méthodologie repose sur un choix judicieux des blocs, à savoir ceux qui présentent la/les structures qui devraient le mieux se prêter à la propagation des structures en question à l'aide des outils de prédictions spatiales. C'est exactement la problématique rencontrée en inpainting, et c'est naturellement la raison pour laquelle le critère de priorité basé isophote/contour utilisé dans [CPT04] a été mis en oeuvre pour estimer la pertinence des structures situées sur les frontières. Ce concept amène à la réalisation d'un ordre de balayage des blocs (ou scanning) non plus fixe mais adaptatif. On tient à mentionner qu'une initiative relative au scanning des sous-partitions dans le macro-bloc avait déjà été proposée dans [STC07] dans le contexte de la prédiction bidirectionnelle spatiale (BIP), où cependant le mode de scanning (se limitant à deux possibilités) devait être transmis au décodeur. Propager efficacement les structures de l'image en fonction du contenu, implique donc que l'ordre de traitement des blocs soit dépendant du contenu. C'est ce qui a été réalisé où un des points délicats a été, entre autres, de traiter convenablement le codage entropique notamment au niveau du CABAC. En effet, l'état du codeur entropique doit être conservé une fois le bloc encodé, pour le traitement futur de l'un des blocs voisins du bloc courant. De plus, traiter les blocs dans un ordre quelconque avec toutes les orientations possibles, quant au voisinage causal, a demandé de rajouter 3 jeux de prédictions spatiales classiques supplémentaires dédiées à ces nouvelles orientations (rotations de 90°, 180° et 270°). De même, de nouveaux templates utilisés pour la prédiction multi-patches ont été

ajoutés.

Pour résumer, ce codeur encode classiquement l'épitome (texture uniquement). Cet épitome codé puis décodé permet d'encoder récursivement les blocs restants de l'image dans un ordre dépendant du contenu, en utilisant les modes de codage classiques, les méthodes de prédictions multi-patches LLE et oMALLE ainsi que le BM. Il est important de noter que le décodeur détermine aussi le même ordre de scanning des blocs, cette information n'étant pas transmise. Sur la base de tous ces outils de prédictions, les performances de ce schéma de compression ont légèrement dépassé celles obtenues avec les schémas conventionnels dotés des mêmes outils de prédiction. En effet, le schéma de codage opérant en raster scan permet d'atteindre en moyenne des gains de 33.54% et 22.28% en bas et hauts débits respectivement par rapport aux modes H.264 Intra. Quant à l'approche inpainting de l'épitome avec scanning order adaptatif, elle permet d'obtenir en moyenne des gains de 34.42% et 23.44% en bas et haut débits respectivement.

Travaux restants et perspectives

Le travail de cette thèse s'est clairement positionné sur la problématique de la prédiction spatiale dédiée au codage intra. Ainsi au moment de clore ce manuscrit, dans la continuité des travaux engagés, des pistes restent encore à affiner mais aussi à explorer.

En tout premier lieu, de façon à se comparer à l'état de l'art actuel en compression, il serait opportun d'implémenter certains des algorithmes étudiés au cours de cette thèse, en particulier oMALLE et/ou le schéma de compression basé épitome, NE et inpainting dans le récent standard MPEG HEVC. Ceci représente une tâche bien délicate, entre autres, au niveau du scanning des CU qu'il faudra alors profondément modifier.

Pour ce qui concerne les améliorations possibles pour l'approche hybride TM/BM, on pourrait envisager des techniques plus efficaces en utilisant un prédicteur de vecteur pour coder les vecteurs pour le BM, une possibilité bien connue étant d'utiliser un médian sur les composantes des vecteurs voisins, lorsque ces composantes sont disponibles.

Concernant les approches conventionnelles pour lesquelles les techniques multi-patches ont été abordées, une des difficultés réside dans la complexité de ces outils de prédiction. Il est indéniable qu'une complexité algorithmique, qui devient rapidement prohibitive, a été sérieusement déportée au décodeur. Ainsi la prédiction oMALLE nécessite la construction du dictionnaire réduit par template matching, la recherche des K-1-NN, l'approximation du template courant via la LLE et enfin le calcul des prédictions. Pour ce qui est du calcul des poids estimés au niveau du template courant, une initiative intéressante a été présentée dans [SYA+13, STC13], où pour la prédiction basée templates le calcul des poids associés à chaque template est effectué à l'aide de l'algo Non Local Mean (NLM). Les coefficients de pondération sont alors issus d'une LUT dont l'accès se fait par la valeur de l'erreur de correspondance de chacun des templates. Cette technique pourrait être avantageusement reprise dans nos travaux.

Du point de vue algorithmique que ce soit pour le TM, le BM, les modes de prédiction NE, et oMALLE (y compris dans le schéma reposant sur l'épitome), pour améliorer les performances de prédiction une solution serait de travailler à la précision sous-pixel, par exemple 1/4 de pel. Cela demande de faire les interpolations à la volée et va à l'encontre de l'objectif du paragraphe précédent en augmentant inévitablement encore la complexité, mais cela devrait aussi améliorer les performances en terme de compression.

Le travail de cette thèse s'est essentiellement focalisé sur la prédiction intra, la suite logique concerne la prédiction inter image basée multi-patches de type NE, ce travail est d'ores et déjà abordé de manière intéressante dans [MCG+13].

Dans les perspectives, il nous semble aussi que le potentiel d'évolution le plus élevé se situe au niveau de l'approche épitome dotée de la prédiction NE dans un esprit inpainting. Ne serait-ce déjà par le fait que les expérimentations n'ont été faites (faute de temps) que sur la base de blocs 8x8, avec de plus un scanning de ces bloc 8x8 imposé dans le MB (ce zigzag étant néanmoins dépendant de l'orientation du voisinage du MB). La première amélioration serait, de fait, de rendre le scanning totalement adaptatif au niveau de toutes les sous partitions 8x8 et 4x4 dans le MB (dans le cas MPEG-4 AVC). Cela permettrait d'être plus fidèle au principe préconisé de propagation des structures du signal. Une autre voie à explorer pourrait également concerner le critère permettant de gérer les priorités d'extension du signal, à savoir le critère basé confiance/isophote. Ce critère, bien que performant en inpainting, est tout de même basé pixel avec une analyse locale dans un environnement très restreint. Il serait alors probablement judicieux d'étendre la profondeur d'analyse dans la zone de recherche des structures ou contours de la zone causale afin de rendre ce nouveau critère plus robuste et en particulier moins sensible au bruit de quantification. Dans ce deuxième schéma de codage basé épitome, celui-ci est construit indépendamment de sa réelle utilisation ultérieure en prédiction et notamment en prédiction multi-patches. Ainsi, il pourrait être judicieux, lors de l'élaboration de l'épitome, de prendre en compte non seulement la reconstruction du patch (comme cela est fait dans ce travail), mais également la reconstruction multi-patches. On laisse malheureusement imaginer au lecteur l'augmentation de la complexité calculatoire au codeur d'une telle approche. Dans le même ordre d'idée, toujours dans ce même schéma de compression, un travail reste à faire pour déterminer, pour une image, la taille optimale de l'épitome sur la base d'outils de prédiction donné et du meilleur compromis débit/distorsion.

Enfin, ce travail s'est beaucoup appuyé sur la mise en application des algorithmes NE (avec ou sans vecteur) dans un codeur classique basé blocs, avec un critère de validation de type rate/distorsion (RDO). Il pourrait être intéressant de rajouter des modes de codage de types synthèses de textures [Ash01] comme dans [RDBT13] utilisant finalement les mêmes outils algorithmiques plutôt basés pixels ; le critère de décision étant alors dédié à la validation de la cohérence des textures reconstruites, une des principales difficultés se situant au codeur dans le choix de la métrique de qualité pour arbitrer la sélection des modes de codage soit classiques soit de synthèse.

Table des figures

Liste des tableaux

Publications personnelles

[1] S. Cherigui, M. Alain, C. Guillemot, D. Thoreau et P. Guillotel, Epitome inpainting with in-loop residue coding for image compression, *IEEE International Conference on Image Processing, ICIP*, 2014 (soumis).

[2] M. Alain, S. Cherigui, C. Guillemot, D. Thoreau et P. Guillotel, Locally Linear Embedding Methods for Inter Image Prediction, *IEEE International Conference on Image Processing, ICIP*, 2013.

[3] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel et P. Perez, Correspondence map-aided neighbor embedding methods for image prediction, *IEEE Transactions on Image Processing*, 2013.

[4] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel et P. Perez, Map-aided locally linear embedding methods for image prediction, *IEEE International Conference on Image Processing, ICIP*, 2012, **(among the 11 nominated for best student paper award)**.

[5] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel et P. Perez, Hybrid template and block matching algorithm for image intra prediction, *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2012, **(best student paper award)**.

[6] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel et P. Perez, Epitome-based image compression using translational sub-pel mapping, *IEEE Multimedia Signal Processing, MMSP*, 2011.

[7] S. Cherigui, C. Guillemot, D. Thoreau et P. Guillotel, *Method of coding a sequence of images and corresponding reconstruction method*, Brevet EP20110305064, WO2011EP58474, 2011.

[8] S. Cherigui, D. Thoreau et E. Francois, *Method for extracting an epitome from an image*, Brevet EP20110305065, WO2011EP72431, 2011.

[9] S. Cherigui, D. Thoreau et C. Guillemot, *Method of coding an image epitome*, Brevet EP20110305063, WO2011EP58495, 2011.

[10] D. Thoreau, S. Cherigui , A. Martin, P. Guillotel et C. Guillemot, *Inter-image prediction method and device and corresponding encoding method and device*, Brevet WO2013EP69903, 2012.

[11] D. Thoreau, S. Cherigui , A. Martin, P. Guillotel et C. Guillemot, *Method for predicting block of pixels of image, involves forming each candidate patch on block*

of pixels, and predicting block of pixels such that one of set of candidate patches is left behind other candidate patches, Brevet FR20120061403, 2012.

[12] S. Cherigui, C. Guillemot, D. Thoreau et P. Guillotel, *Method and device for encoding a block of an image and corresponding reconstructing method and device*, Brevet WO2013EP50157, 2012.

[12] S. Cherigui, C. Guillemot, D. Thoreau et P. Guillotel, *Method and device for encoding a block of an image and corresponding reconstructing method and device*, Brevet EP20120305144, 2012.

[13] D. Thoreau, S. Cherigui, *Method and apparatus for encoding or decoding a video signal using a summary reference picture*, Brevet WO2012EP50182, 2011.

[14] Deux autres brevets non encore publiés

Bibliographie

[APRSK+04] Aravind AL., Bindu P. Rao, Shudhir S. Kudva, Sreenu Babu, Sumam David et Ajit V. Rao : Quality and complexity comparison of h.264 intra mode with jpeg2000 and jpeg. *In IEEE Int. Conf. on Image Processing, ICIP*, 2004. 81

[Ash01] M. Ashikhmin : Synthesizing natural textures. *In Proc. of ACM Symposium on Interactive 3D Graphics*, pages 217–226, 2001. 117

[BD96] S.A. Basith et S.R. Done : Digital video, mpeg and associated artifacts. *In Imperial College London*, 1996. 13

[Bjo01] G. Bjontegaard : Calculation of average psnr differences between rd curves. *In document VCEG-M33, ITU-T VCEG Meeting*, April 2001. 53, 71, 90, 100, 110

[BMBP11] L. Benoît, J. Mairal, F. Back et J. Ponce : Sparse image representation with epitomes. *In IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, 2011. 33

[CFJ05] V. Cheung, B. J. Frey et N. Jojic : Video epitomes. *In IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, pages 42–49, 2005. 27

[CPT04] A. Criminisi, P. Pérez et K. Toyama : Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. on Image Processing*, 13(9):1200–1212, 2004. 83, 103, 104, 105, 115, 122

[CYL+10] X. Chu, S. Yan, L. Li, K. L. Chan et T. S. Huang : Spatialized epitome and its applications. *In IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, pages 311–318, 2010. 27

[DH04] A. Dumitras et B.G. Haskell : An encoder-decoder texture replacement method with application to content-based movie coding. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(6):825–840, 2004. 24

[DHIC03] A. Dumitras, B.G. Haskell, A.C. Inc et C.A. Cupertino : A texture replacement method at the encoder for bit-rate reduction of compressed video. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(2):163–175, 2003. 24

[DLR77] A. P. Dempster, N. M. Laird et D. B. Rubin : Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977. 28

[EA06] M. Elad et M. Aharon : Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. on Image Processing*, pages 3736–3745, 2006. 33

[EA08] M. Elad et M. Aharon : Sparse and redundant modeling of image content using an image signature dictionary. *SIAM Journal on Image Sciences*, pages 228–247, 2008. 33, 38

[FDK02] K. Freivalds, U. Dogrusoz et P. Kikusts : Disconnected graph layout and the polyomino packing approach. *In Lecture Notes in Computer Science*, volume 22-65, pages 378–391, 2002. 39

[GKS00] U. Gargi, R. Kasturi et S.H. Strayer : Performance characterization of video-shot-change detection methods. *IEEE Trans. on Circuits and Systems for Video Technology*, 10(1):1–13, 2000. 42

[HEV10a] Joint call for proposal on video compression technology, 2010. 19

[HEV10b] Vision, Applications and Requirements for High-Performance Video Coding (HVC), 2010. 19

[HEV11] HM : reference Software for HEVC version HM-3.0, 2011. 19

[HW02] T. Halbach et M. Wien : Concepts and performances of next generation video compression standardization. *In Proc. Nordic Signal Processing Sympossium, NORSIG*, oct. 2002. 81

[jas05] Jasper Software Reference Manual (Version 1.900.0). ISO/IEC JTC 1/SC 29/WG 1, ITU-T SG 16, Coding of Still Pictures, 2005. 80, 81

[JF03] N. Jojic et B. J. Frey : Advances in algorithms for inference and learning in complex probability models. *In IEEE Trans. PAMI*, 2003. 29

[JFK03] N. Jojic, J. Frey et A. Kannan : Epitomic analysis of appearance and shape. *In IEEE Int. Conf. on Computer Vision, ICCV*, pages 34–41, 2003. 27, 28, 29, 31, 33, 36, 38, 40, 41, 42, 45, 102, 120

[JGJS98] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola et S. K. Saul : An introduction to variational methods for graphical models. *In Machine Learning*, volume 37, pages 183–233. Kluwer Academic Publishers, 1998. 29

[Jol02] I.T. Jolliffe : Principle component analysis. *2e edition Springer*, 2002. 58

[jpe05] The JPEG-2000 Still Image Compression Standard. ISO/IEC JTC 1/SC 29/WG 1, ITU-T SG 16, Coding of Still Pictures, 2005. 80, 81

[KSE+03] V. Kwatra, A.A. Schödl, I. Essa, G. Turk et A. Bobick : Graphcut textures : image and video texture synthesis using graph cuts. *In Proceedings of ACM SIGGRAPH*, pages 277–286, 2003. 24

[KWR06] A. Kannan, J. Winn et C. Rother : Clustering appearance and shape by learning jigsaws. *In NIPS*, 2006. 27

[LS00] D. D. Lee et H. S. Seung : Algorithms for non-negative matrix factorization. *In Advances in Neural Information Processing Systems, NIPS*, 2000. 58, 59, 61

[MCG+13] A. Martin, S. Cherigui, C. Guillemot, D. Thoreau et P. Guillotel : Locally linear embedding methods for inter image coding. *In IEEE Int. Conf. on Image Processing, ICIP*, 2013. 117

[MZ93] S. Mallat et Z. Zhang : Matching pursuit with time-frequency dictionaries. *IEEE Trans. on Image Processing.*, 41:3397–3415, 1993. 34

[NKCW08] K. Ni, A. Kannan, A. Criminisi et J. Winn : Epitomic location recognition. *In IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, pages 1–8, 2008. 27

[NKCW09] K. Ni, A. Kannan, A. Criminisi et J. Winn : Epitomic location recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 2158–2167, 2009. 27

[NNMB+03] Patrick Ndjiki-Nya, Bela Makai, Gabi Blattermann, Aljoscha Smolic, Heiko Schwarz et Thomas Wiegand : Improved h.264/avc coding using texture analysis and synthesis. *In Int. conf. on Image Processing, ICIP*, pages 849–852, 2003. 24

[NNMS+03] P. Ndjiki-Nya, B. Makai, A. Smolic, H. Schwarz et T. Wiegand : Video coding using texture analysis and synthesis. *In Proceedings of Picture Coding Symposium, St. Malo, France*. Citeseer, 2003. 24

[OF97] B. A. Olshausen et D. J. Field : Sparse coding with an overcomplete basis set : a strategy employed by v1 ? *In Vision Research*, pages 3311–3325, 1997. 33

[OR98] A. Ortega et K. Ramchandran : Rate-distortion methods for image and video compression. 15(6):23–50, 1998. 18

[Pey09] G. Peyre : Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, pages 17–31, 2009. 33

[RDBT13] F. Racape, O. Deforges, M. Babel et D. Thoreau : Spatiotemporal texture synthesis and region-based motion compensation for video compression. *In Signal Processing : Image Communication*, volume 28, pages 993–1005, 2013. 117

[RLF+11a] F. Racape, S. Lefort, E. François, M. Babel et O. Deforges : Adaptive pixel/patch-based synthesis for texture compression. *In Int. conf. on Image Processing, ICIP*, pages 609–612, 2011. 24

[RLF+11b] F. Racape, S. Lefort, E. François, M. Babel et O. Deforges : Characterization and adaptive texture synthesis-based compression scheme. *In European Signal Processing Conference, EUSIPCO*, 2011. 24

[RS00] S. Roweis et L. Saul : Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, Dec. 2000. 59, 60, 61, 120

[SCSI08] D. Simakov, Y. Caspi, E. Schechtman et M. Irani : Summarizing visual data using bidirectional similarity. *In IEEE Conf. Computer Vision and Pattern Recognition, CVPR*, pages 1–8, 2008. 30, 31, 44, 45

[sof] JM reference software version 14.2. *In http ://iphome.hhi.de/suehring/tml/download/KTA/*. 52, 69, 80, 96, 100

[SOH12] G. J. Sullivan, J. R. Ohm et W. J. Han : Overview of the high efficiency video coding HEVC standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 22(12), Dec. 2012. 20

[SSM98] B. Scholkopf, A. Smola et K.R. Muller : Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, Jul. 1998. 59

[STC07] T. Shiodera, A. Tanizawa et T. Chujoh : Bidirectional intra prediction. *In ITU-T SG16/Q.6 VCEG, VCEG-AE14, Marrakech, Morocco*, January 2007. 115

[STC13] T. Shiodera, A. Tanizawa et T. Chujoh : Intra prediction for lossless coding. *In Joint Collaborative Team on Video Coding (JCT VC), Doc. JCTVC-L0161, Geneva, Switzerland*, Jan. 2013. 116

[SW98] G.J. Sullivan et T. Wiegand : Rate-distortion optimization for video compression. 15(6):74–90, 1998. 18

[SYA+13] E. Shiodera, G. Yammine, P. Amon, A. Hutter et A. Kaup : Sample-based weighted prediction with directional template matching for hevc lossless coding. *In PCS*, 2013. 116

[TBS06] T. K. Tan, C. S. Boon et Y. Suzuki : Intra prediction by template matching. *In IEEE Int. Conf. on Image Processing, ICIP*, pages 1693–1696, 2006. 23, 51

[TBS07] T. K. Tan, C. S. Boon et Y. Suzuki : Intra prediction by averaged template matching predictors. *In IEEE Consumer Communications and Networking Conference, CCNC*, pages 405–409, 2007. 24

[TdSL00] J.B. Tenenbaum, V. de Silva et J.C. Langford : A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, Dec. 2000. 59, 120

[TG10] M. Turkan et C. Guillemot : Image prediction : Template matching vs. Sparse approximation. *In IEEE Int. Conf. on Image Processing, ICIP*, pages 789–792, 2010. 51

[TG12] M. Turkan et C. Guillemot : Image prediction based on neighbor embedding methods. *IEEE Trans. on Image Processing*, 21:1885–1898, 2012. 57, 62, 63, 64, 70, 114, 120

[WHW09] Q. Wang, R Hu et Z. Wang : Intra coding in h.264/avc by image epitome. *In PCM*, 2009. 40, 41, 42, 43, 102, 120

[WHW+12] Q. Wang, R. Hu, Z. Wang, C. Zhou et B. Hang : Intra coding and refresh based on compression-oriented video epitomic analysis. *IEEE Trans. on Circuits and Systems for Video Technology*, 22-5:714–726, May 2012. 44, 45, 120

[WHWH10] Q. Wang, R Hu, Z. Wang et B. Hang : Intra coding and refresh based on video epitomic analysis. *In IEEE Int. Conf. on Multimedia Expo, ICME*, 2010. 40, 43, 44

[WHZ+08] L.Y. Wei, J. Han, K. Zhou, H. Bao, B. Guo et H.Y. Shum : Inverse texture synthesis. *In ACM SIGGRAPH*, pages 1–9, 2008. 32

[WL00] Li-Yi Wei et Marc Levoy : Fast texture synthesis using tree-structured vector quantization. *In SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, 2000. 24

[WSBL03] T. Wiegand, G. J. Sullivan, B. Bjontegaard et A. Luthra : Overview of the H.264/AVC video coding standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 13-7:560–576, Jul. 2003. 50, 93, 94

[WWOH08] H. Wang, Y Wexler, E. Ofek et H. Hoppe : Factoring repeated content within and among images. 27(3), 2008. 37, 38, 45, 83, 85, 89, 90, 91, 92, 97, 100, 101, 115, 121, 124

[YC02] S.-L. Yu et C. Chrysafis : New intra prediction using intra macroblock motion compensation. *In Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-C151*, May 2002. 51

[ZYE+08] Y. Zheng, O. D. Yin, X. Escoda, X. Li et C. Gomila : Intra prediction using template matching with adaptive illumination compensation. *In IEEE Int. Conf. on Image Processing, ICIP*, pages 125–128, 2008. 24

Résumé

Au cours de ces dernières années, le domaine de la compression vidéo a connu un essor considérable avec le standard H.264/AVC et l'arrivée de son successeur HEVC. La prédiction spatiale de ces standards repose sur la propagation unidirectionnelle de pixels voisins. Bien que très efficace pour étendre des motifs répondants aux mêmes caractéristiques, cette prédiction présente des performances limitées lorsqu'il s'agit de propager des textures complexes. Cette thèse vise à explorer de nouveaux schémas de prédiction spatiale afin d'améliorer les techniques actuelles de prédiction intra, en étendant ces schémas locaux et monodimensionnels à des schémas globaux, multidimensionnels et multi-patches. Une première méthode de prédiction hybride intégrant correspondance de bloc et correspondance de gabarit (template) a été investiguée. Cette approche hybride a ensuite été étendue en prédiction multi-patches de type "neighbor embedding" (NE). L'autre partie de la thèse est dédiée à l'étude des épitomes dans un contexte de compression d'images. L'idée est d'exploiter la redondance spatiale de l'image d'origine afin d'extraire une image résumé contenant les patches de texture les plus représentatifs de l'image, puis ensuite utiliser cette représentation compacte pour reconstruire l'image de départ. Ce concept d'épitome a été intégré dans deux schémas de compression, l'un de ces algorithmes s'avère vraiment en rupture avec les techniques traditionnelles dans la mesure où les blocs de l'image sont traités, à l'encodeur et au décodeur, dans un ordre spatial qui dépend du contenu et cela dans un souci de propagation des structures de l'image. Dans ce dernier algorithme de compression, des modes de prédiction directionnelle intra H.264 étendus et des méthodes avancées de prédiction multi-patches y ont été également introduits. Ces différentes solutions ont été intégrées dans un encodeur de type H.264/AVC afin d'évaluer leurs performances de codage par rapport aux modes intra H.264 et à l'état de l'art relatif à ces différentes techniques.

Abstract

In recent years, video compression field has increased significantly since the apparition of H.264/AVC standard and of its successor HEVC. Spatial prediction in these standards are based on the unidirectional propagation of neighboring pixels. Although very effective to extend pattern with the same characteristics, this prediction has limited performances to extrapolate complex textures. This thesis aims at exploring new spatial prediction schemes to improve the current intra prediction techniques, by extending these local schemes to global, multidimensional and multi-patches schemes. A hybrid prediction method based on template and block matching is first investigated. This hybrid approach is then extended to multi-patchs-based prediction of type "Neighbor Embedding" (NE). The other part of this thesis is dedicated to the study of epitome image within the scope of image compression. The idea is to exploit spatial redundancies in the original image in order to first extract a summary image containing the texture patches the most representative of the image, and then use this compacted representation to rebuild the original image. The concept of epitome has been incorporated in two compression schemes, one of these algorithms is in rupture with the traditional techniques since the image blocks are processed, both at encoder and decoder sides, in a spatial order that depends on the image content and this in the interest of propagating image structures. In this last compression algorithm, extended H.264 Intra directional prediction modes and advanced multi-patches prediction methods have been also included. These different solutions have been integrated in a H.264/AVC encoder in order to assess their coding performances with respect to H.264 intra modes and the state of the art relative to these different techniques.