



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 Télécommunications

Ecole doctorale MATISSE

présentée par

Angélique Drémeau

préparée à l'INRIA Rennes - Bretagne Atlantique
Institut National de Recherche en Informatique et en Automatique

Décompositions

parcimonieuses :

approches Bayésiennes

et application à la

compression d'image

**Thèse soutenue à Rennes
le 19 novembre 2010**

devant le jury composé de :

Laurent DAUDET

Professeur, Université Paris Diderot / Président,
Examinateur

Pierre VANDERGHEYNST

Professeur, EPFL / Rapporteur

Jérôme IDIER

Directeur de recherche, CNRS / Rapporteur

Béatrice PESQUET-POPESCU

Professeur, Télécom ParisTech / Examinatrice

Jean-Jacques FUCHS

Professeur, Université de Rennes 1 / Directeur de
thèse

Christine GUILLEMOT

Directrice de recherche, INRIA Rennes / Co-
directrice de thèse

Remerciements

Je voudrais d'abord remercier tous les membres du jury de l'intérêt qu'ils ont porté à cette thèse. Merci à Laurent Daudet pour avoir présidé le jury, Pierre Vandergheynst et Jérôme Idier pour avoir accepté d'être rapporteurs de mes travaux, Béatrice Pesquet-Popescu pour son rôle d'examineur. Merci à mes directeurs de thèse, Jean-Jacques Fuchs et Christine Guillemot.

Je tiens à remercier également Cédric Herzet, sans qui cette thèse n'aurait pu voir le jour. A la fois coach dans mes moments de découragement, grand sage face à mes questions, collaborateur lors de nos discussions scientifiques, philosophe devant ma perplexité,... Merci pour tout cela.

Merci encore à Jean-Jacques Fuchs pour son soutien dans les moments difficiles.

Merci aux membres présents et passés du projet TEMICS pour l'ambiance de travail décalée.

Merci à ma famille pour leur confiance sans mesure, aux amis pour leur empathie.

Table des matières

Table des matières	iii
Notations	ix
Introduction	1
1 Compression d'image	5
1.1 Éléments de théorie de l'information	5
1.1.1 Mesures de performance	6
1.1.2 Fonction débit-distorsion	8
1.2 Compression d'image	10
1.2.1 Principe général	10
1.2.2 Codage entropique	12
1.2.3 Quantification	17
1.3 Codage par transformation	21
1.3.1 Principe général	22
1.3.2 État de l'art	24
1.3.3 Formats de compression : JPEG et JPEG2000	25
1.4 Codage par prédiction	27
1.4.1 Principe général	27
1.4.2 Méthode de prédiction intra utilisée dans H.264	29
1.4.3 État de l'art	30
1.5 Optimisation débit-distorsion	31
2 Décompositions parcimonieuses	35
2.1 Problèmes d'optimisation	35
2.1.1 Mesures relâchées de parcimonie	36
2.1.2 Equivalence et unicité des solutions de (\mathcal{P}_p)	38
2.1.3 Approximations parcimonieuses	39

2.2	Recherche d'approximations parcimonieuses	40
2.2.1	Cas particulier : base orthonormée	40
2.2.2	Algorithmes de recherche	41
2.3	Dictionnaires	51
2.3.1	Introduction à l'apprentissage de dictionnaires	51
2.3.2	Algorithmes d'apprentissage	52
2.4	Annexe : définition et propriétés des normes	55
3	Transformations adaptatives	57
3.1	Codage par transformation et parcimonie	57
3.1.1	Considérations débit-distorsion	57
3.1.2	Redondance et coût de codage	61
3.2	Structuration du dictionnaire	63
3.2.1	Motivations	63
3.2.2	Ensemble de bases	64
3.2.3	Concaténations de bases locales	65
3.3	Optimisation débit-distorsion	67
3.4	Bases locales prédéfinies : les DCT directionnelles	69
3.4.1	Principe	69
3.4.2	Implémentation	71
3.4.3	Evaluation des performances	73
3.5	Bases locales apprises : étude d'un algorithme de la littérature	75
3.5.1	Algorithme de Sezer	75
3.5.2	Améliorations	77
3.5.3	Implémentation	81
3.5.4	Evaluation des performances	83
3.6	Vers une approche probabiliste	85
3.6.1	Définition d'un cadre probabiliste	86
3.6.2	L'algorithme de Sezer revisité	87
3.7	Bases locales apprises : un nouvel algorithme Bayésien	88
3.7.1	Une approche alternative	88
3.7.2	Un nouvel algorithme	89
3.7.3	Estimation de la variance de bruit	91
3.7.4	Evaluation des performances en reconstruction	92
3.7.5	Evaluation des performances en compression	94
3.8	Annexes	101
3.8.1	Calcul du pas de quantification optimal au sens débit-distorsion (3.15)	101

3.8.2	Calcul de l'équation de mise à jour de la variance de bruit (3.61)	102
3.8.3	Images utilisées	102
4	Prédiction et parcimonie	105
4.1	Introduction	106
4.2	Prédiction basée sur un mélange de décompositions parcimonieuses	108
4.2.1	Définition d'un cadre probabiliste	108
4.2.2	Développement d'un nouvel algorithme	110
4.3	Evaluation des performances	113
4.3.1	Paramètres du modèle	114
4.3.2	Schéma de prédiction et d'encodage	115
4.3.3	Analyse des performances	116
4.3.4	Perspectives	124
4.4	Annexe : calcul de l'intégrale $\int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}, c) d\mathbf{y}$	126
5	Algorithmes gloutons Bayésiens	129
5.1	Introduction	130
5.2	Définition d'un cadre probabiliste	131
5.3	Formulation du problème	132
5.4	Développement de nouveaux algorithmes	134
5.4.1	Bayesian Matching Pursuit (BMP)	134
5.4.2	Bayesian Orthogonal Matching Pursuit (BOMP)	137
5.4.3	Bayesian Stagewise Orthogonal Matching Pursuit (BS-tOMP)	138
5.4.4	Bayesian Subspace Pursuit (BSP)	140
5.5	Evaluation des performances	142
5.5.1	Modèle Bernoulli-Gaussien, dictionnaire aléatoire	143
5.5.2	Modèle 0 – 1, dictionnaire aléatoire	146
5.5.3	Modèle Bernoulli-Gaussien, dictionnaire en cosinus	148
5.6	Annexes	149
5.6.1	Preuve du théorème (10)	149
5.6.2	Calcul de l'expression du seuil (5.16) et de l'équation de mise à jour des coefficients (5.17) dans BMP	150
5.6.3	Calcul de l'équation de mise à jour des coefficients (5.23) dans BOMP	152
	Conclusion	153

Bibliographie	157
Publications	166

Notations

Relations d'ordre

$x \propto y$	x est proportionnel à y
$x \triangleq y$	x est défini par y
$x \ll y$	x est très inférieur à y
$f = \mathcal{O}(g)$	f est de l'ordre de g : il existe K tel que $f \leq Kg$
$f \sim g$	f est équivalent à g : $f = \mathcal{O}(g)$ et $g = \mathcal{O}(f)$

Vecteurs et matrices

\mathbf{x}	Vecteur
\mathbf{x}^T	Transposée du vecteur \mathbf{x}
$\hat{\mathbf{x}}$	Un estimateur du vecteur \mathbf{x}
\mathbf{X}	Matrice
\mathbf{X}^{-1}	Inverse de la matrice \mathbf{X}
\mathbf{X}^T	Transposée de la matrice \mathbf{X}
\mathbf{X}^+	Pseudo-inverse de Moore-Penrose de la matrice \mathbf{X} : si $\mathbf{X} \in \mathbb{R}^{N \times M}$ est de rang plein (cf. [85]), $\mathbf{X}^+ = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ si $N > M$ $\mathbf{X}^+ = \mathbf{X}^T (\mathbf{X} \mathbf{X}^T)^{-1}$ si $N < M$
\mathbf{I}_N	Matrice identité de dimension $N \times N$
$\ \mathbf{x}\ _p$	Norme ℓ_p de \mathbf{x} : $\ \mathbf{x}\ _p = (\sum_{i=1}^M x_i ^p)^{\frac{1}{p}}$, si $0 < p < \infty$
$\langle \mathbf{x}, \mathbf{y} \rangle$	Produit scalaire entre les vecteurs \mathbf{x} et \mathbf{y}

Probabilités

X	Variable aléatoire
\mathbb{X}	Vecteur aléatoire
$H(X)$	Entropie de la variable aléatoire X
$I(X; Y)$	Information mutuelle entre les variables aléatoires X et Y : $I(X; Y) = \int_x \int_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy$

$E_X[f(x)]$	Espérance de $f(x)$ par rapport à la distribution de X : $E_X[f(x)] = \int_x p(x)f(x)dx$
$E_{X Y}[f(x)]$	Espérance conditionnelle de $f(x)$ par rapport à la distribution de X sachant Y : $E_{X Y}[f(x)] = \int_x p(x y)f(x)dx$
$\mathcal{N}(m, \sigma)$	Distribution normale de moyenne m et de variance σ
$Ber(p)$	Distribution de Bernoulli de paramètre p
$\delta_a(x)$	Distribution de Dirac décentrée : $\delta_a(x) = \delta(x - a)$
<i>i.i.d.</i>	<i>indépendantes et identiquement distribuées</i> Se dit de variables aléatoires qui ont toutes la même loi de probabilité et sont mutuellement indépendantes.

Ensembles

\mathcal{X}	Ensemble d'éléments
\mathbb{N}	Entiers positifs
\mathbb{R}	Nombres réels

Introduction

Contexte d'étude

La compression d'image est un domaine de recherche déjà riche d'une longue histoire. Trouvant ses racines théoriques dans la théorie de l'information initiée par Shannon en 1948 [99], elle a depuis fait appel à de nombreux outils mathématiques, de plus en plus sophistiqués. La problématique est cependant toujours d'actualité, sous-jacente des applications de stockage des données, ou de transmission à travers des canaux à bande passante limitée. Malgré les développements technologiques considérables ces dernières années, l'être humain, insatiable, déclare indispensable ce qu'hier il considérait superflu, et nous sommes sans cesse confrontés à de nouveaux besoins exigeant l'accès à toujours plus d'information.

La compression d'image met en jeu différentes techniques. On peut grossièrement les diviser en quelques grandes familles. Certaines ont pour but de structurer la redondance présente dans l'image que l'on souhaite compresser. C'est le cas par exemple de la transformation, qui cherche à concentrer l'information dans un petit nombre d'éléments en changeant de domaine, et de la prédiction, qui calcule un estimateur de l'information courante à partir d'une information passée. D'autres réduisent l'information à traiter en la quantifiant par exemple. Enfin, un grand pan de recherche de la compression d'image est occupé par le codage entropique, qui cherche à traduire l'information en une séquence de symboles la plus courte possible.

Parallèlement à ce domaine de recherche, la thématique des décompositions parcimonieuses s'est développée. Les décompositions parcimonieuses sont attachées à la description d'un signal (exactement ou de façon approchée) comme combinaison d'un petit nombre d'atomes choisis dans un très grand ensemble appelé dictionnaire. A notre connaissance, le nom de "décompositions parcimonieuses" est né à la fin des années 1990

sous la plume de Chen et Donoho [15]. Mais la notion mathématique, elle, intéressait les scientifiques depuis quelques temps déjà. On la trouvait en régression dans le domaine des statistiques [13, 107], ou en résolution de problèmes inverses bien définis [39] en traitement du signal.

On peut comprendre intuitivement l'intérêt des scientifiques de la communauté de la compression d'image pour les décompositions parcimonieuses : décrire une image avec un petit nombre d'éléments permettrait de réduire la quantité d'information à transmettre ou stocker. Mais la mise en pratique n'est pas si immédiate et a révélé quelques écueils. Jusqu'à présent, malgré de nombreuses contributions prometteuses [53, 86], elle n'a pas réussi à s'imposer comme alternative sérieuse aux schémas de compression plus "classiques" qui, quoique très souvent liés à l'idée de parcimonie, ne la formalisent pas explicitement.

Dans cette thèse, nous nous sommes intéressés à l'utilisation des décompositions parcimonieuses dans des schémas de compression d'image, via des méthodes Bayésiennes. Les problèmes reposant sur des décompositions parcimonieuses peuvent être en effet interprétés de façon probabiliste relativement intuitivement et la définition d'un cadre Bayésien permet l'utilisation d'outils probabilistes performants pour leur résolution. Quoique envisagée très tôt (dès 1997 par Olshausen et Field [81] par exemple), cette approche est restée peu exploitée jusqu'à récemment, où plusieurs algorithmes de recherche de décompositions parcimonieuses "Bayésiens" ont vu le jour [119, 103]. C'est dans la continuité de ces contributions (et parfois même parallèlement !) que nos travaux se sont inscrits.

Organisation du manuscrit

Ce manuscrit est composé de cinq chapitres. Les deux premiers présentent le contexte théorique des travaux réalisés pendant cette thèse, l'un s'attardant sur les aspects de compression d'image, l'autre sur les décompositions parcimonieuses. Ils sont parfaitement indépendants l'un de l'autre. Les trois derniers chapitres exposent les travaux proprement dits.

Ainsi, le **chapitre 2** introduit les notions de parcimonie, décompositions parcimonieuses et dictionnaires favorisant la parcimonie des décompositions. Différents algorithmes de la littérature cherchant à résoudre des problèmes de recherche de décompositions parcimonieuses et d'apprentissage de dictionnaires sont exposés. Dans le **chapitre 1**, nous présentons quelques éléments

de la théorie de l'information utiles à la compréhension du document. Nous étudions ensuite plus attentivement deux techniques de codage d'image en particulier : le codage par transformation et le codage par prédiction. Ces deux techniques de compression sont respectivement l'objet d'étude des chapitres 3 et 4. Le **chapitre 3** s'intéresse plus spécifiquement aux transformations adaptatives. Une première partie est consacrée à l'étude d'un schéma de compression optimisant le choix de la base de transformation par une segmentation en bintree de l'image et l'utilisation d'ensembles de bases locales. Ces bases locales sont d'abord considérées comme prédéfinies, puis elles sont apprises par un algorithme de la littérature. Cet algorithme nous permet alors d'introduire un nouvel algorithme d'apprentissage Bayésien, favorisant la parcimonie de la décomposition. Le développement de ce nouvel algorithme constitue la deuxième grande partie du chapitre 3. L'aspect codage par prédiction est abordé dans le **chapitre 4**. Inspiré de contributions récentes s'appuyant sur des décompositions parcimonieuses, un nouvel algorithme de prédiction Bayésien reposant sur un mélange de décompositions parcimonieuses est proposé. Enfin, approfondissant l'idée de structuration de la parcimonie des décompositions, mise en évidence dans le chapitre 3, le **chapitre 5** propose l'étude d'un modèle Bernoulli-Gaussien et s'intéresse à son utilisation dans la dérivation d'algorithmes de décompositions parcimonieuses.

Chacun des trois chapitres de contributions a donné lieu à un ou plusieurs papiers de conférences. Ces papiers sont répertoriés en fin de manuscrit.

Compression d'image

1

Ce chapitre rappelle brièvement les fondamentaux de la compression d'image. Nous présentons d'abord quelques éléments de la théorie de l'information, puis, étudions plus attentivement deux techniques de compression d'image en particulier : le codage par transformation (section 1.3) et le codage par prédiction (section 1.4). Ces deux approches feront spécifiquement l'objet de contributions, présentées dans les chapitres 3 et 4 de ce manuscrit.

1.1 Éléments de théorie de l'information

La compression d'image repose sur des concepts fondamentaux de la théorie de l'information. Initiée par Shannon en 1948 (cf. [99]), cette théorie introduit les notions de quantité d'information, d'entropie d'une source et définit les bornes théoriques sur le nombre de bits nécessaires à la représentation d'une information. S'appuyant sur cette théorie, la compression d'image cherche le *meilleur* moyen de réduire la taille d'une information - l'image - tout en conservant sa qualité visuelle, *i.e.*, à approcher *au mieux* les bornes de performance posées par la théorie de l'information.

Dans la suite de cette section, nous utiliserons les notations suivantes. Le signal source est modélisé par un vecteur aléatoire $\mathbb{Y} = [Y_1, \dots, Y_N]^T$ dans l'alphabet \mathcal{Y}^N et de densité de probabilité $p(\mathbf{y}) = \Pr\{\mathbb{Y} = \mathbf{y}\}$, $\mathbf{y} \in \mathcal{Y}^N$. On note $\mathbf{y} = [y_1, \dots, y_N]^T$ une réalisation de \mathbb{Y} . Le résultat du codage de la source, appelé signal reconstruit, et les variables associées sont différenciés par un $\hat{\cdot}$. On appelle schéma de codage une opération

$$s : \begin{cases} \mathcal{Y}^N \rightarrow \hat{\mathcal{Y}}^N, \\ \mathbf{y} \mapsto \hat{\mathbf{y}} = s(\mathbf{y}), \end{cases} \quad (1.1)$$

de l'ensemble \mathcal{Y}^N vers l'ensemble $\hat{\mathcal{Y}}^N$.

1.1.1 Mesures de performance

Quantité d'information et qualité de reconstruction sont des notions fondamentales de la théorie de l'information. Elles sont quantifiées par des mesures de débit et de distorsion.

Débit binaire

Le débit binaire mesure la quantité d'information transmise ou stockée en bits par unité de mesure.

S'il s'agit d'images fixes, l'unité de mesure considérée est le pixel. Le débit binaire - assimilé au débit par la suite - s'exprime alors en bits par pixel (abrégé en bpp).

Par souci de simplicité, on formalise ici le débit binaire obtenu par un schéma de codage *par symbole*, noté s (cf. sous-section 1.2.2). \mathcal{Y} est vu ici comme un ensemble fini.

Dans le cas unidimensionnel ($N = 1$), le *débit binaire* est défini par

$$R_{y,s} = l_y, \quad (1.2)$$

où l_y est la longueur de la séquence de bits correspondant au codage de y .

On définit également le *débit binaire moyen*, noté $\bar{R}_{Y,s}$, par

$$\bar{R}_{Y,s} = \sum_{y \in \mathcal{Y}} p(y) R_{y,s} = \sum_{y \in \mathcal{Y}} p(y) l_y. \quad (1.3)$$

$p(y)$ est alors appelée *probabilité d'apparition* du symbole $y \in \mathcal{Y}$.

Dans le cas multidimensionnel avec $\mathbf{y} \in \mathcal{Y}^N$ réalisation de \mathbb{Y} , les deux notions sont respectivement définies par

$$R_{\mathbf{y},s} = \frac{1}{N} \sum_{i=1}^N R_{y_i,s} = \frac{1}{N} \sum_{i=1}^N l_{y_i}, \quad (1.4)$$

et

$$\bar{R}_{\mathbb{Y},s} = \sum_{\mathbf{y} \in \mathcal{Y}^N} p(\mathbf{y}) R_{\mathbf{y},s} = \frac{1}{N} \sum_{i=1}^N \sum_{y_i \in \mathcal{Y}} p(y_i) l_{y_i}, \quad (1.5)$$

où $p(y_i)$ est la probabilité d'apparition du symbole $y_i \in \mathcal{Y}$.

Distorsion

La distorsion mesure l'altération de la forme originale d'un signal, à mesure que ce signal subit différentes opérations (compression, transmission, etc.). Dans le cas unidimensionnel ($N = 1$), une mesure de distorsion est une opération

$$d : \mathcal{Y} \times \hat{\mathcal{Y}} \rightarrow \mathbb{R}^+ \quad (1.6)$$

du produit cartésien des ensembles \mathcal{Y} et $\hat{\mathcal{Y}}$, vers l'ensemble des réels positifs. Elle est dite *bornée* si son maximum est fini, *i.e.*,

$$\max_{(y, \hat{y}) \in (\mathcal{Y}, \hat{\mathcal{Y}})} d(y, \hat{y}) < \infty. \quad (1.7)$$

Dans la plupart des cas, l'ensemble $\hat{\mathcal{Y}}$ est le même que l'ensemble \mathcal{Y} . Des mesures de distorsion communes sont alors par exemple :

- ◆ la distorsion de Hamming telle que

$$d(y, \hat{y}) = \begin{cases} 0 & \text{si } y = \hat{y}, \\ 1 & \text{si } y \neq \hat{y}, \end{cases} \quad (1.8)$$

- ◆ l'erreur quadratique telle que

$$d(y, \hat{y}) = (y - \hat{y})^2. \quad (1.9)$$

Cette dernière mesure de distorsion est très utilisée pour des alphabets continus. Elle est intéressante de par sa simplicité et son lien avec l'estimation au sens des moindres carrés. Cependant, dans les applications de type compression d'image qui nous intéressent ici, cette mesure ne rend pas bien compte de la distorsion perçue par l'oeil humain. Des alternatives ont été proposées mais en général, elles sont hautement non linéaires et difficiles à manipuler ([68]). On conserve donc pour le moment l'erreur quadratique comme mesure de distorsion, en remarquant que diminuer l'erreur quadratique signifie souvent améliorer la qualité visuelle de reconstruction. Cette mesure peut ensuite être corrigée via l'introduction de coefficients de pondération permettant de mieux s'approcher de notre sensibilité perceptuelle sans compliquer l'optimisation mathématique.

Pour un signal source multidimensionnel $\mathbf{y} \in \mathcal{Y}^N$ et son pendant reconstruit $\hat{\mathbf{y}} \in \hat{\mathcal{Y}}^N$, on définit une mesure de distorsion par

$$D_{\mathbf{y},s} = \frac{1}{N} \sum_{i=1}^N d(y_i, \hat{y}_i). \quad (1.10)$$

$D_{y,s}$ dépend de la réalisation y et du schéma de codage utilisé s via la réalisation reconstruite \hat{y} .

La distorsion moyenne, notée $\bar{D}_{\mathbb{Y},s}$, est alors

$$\bar{D}_{\mathbb{Y},s} = E_{\mathbb{Y},\hat{\mathbb{Y}}}[D_{y,s}], \quad (1.11)$$

où $E_{\mathbb{Y},\hat{\mathbb{Y}}}[D_{y,s}]$ est l'espérance mathématique de $D_{y,s}$ par rapport à la distribution de $(\mathbb{Y}, \hat{\mathbb{Y}})$. $\bar{D}_{\mathbb{Y},s}$ dépend de la source \mathbb{Y} et du schéma de codage utilisé s via la source reconstruite $\hat{\mathbb{Y}}$.

1.1.2 Fonction débit-distorsion

On distingue deux grands types de codage : les codages avec et sans perte.

Codage sans perte

Lors d'un codage sans perte, l'information reconstruite est l'exacte copie de l'information originale ; il n'y a pas de distorsion entre l'information originale et l'information reconstruite. On cherche alors à déterminer le schéma de codage s^* tel que

$$s^* = \operatorname{argmin}_{s \in \mathcal{S}} \bar{R}_{\mathbb{Y},s}, \quad (1.12)$$

où \mathcal{S} est l'ensemble des schémas de codage possibles.

Dans ce cas, la théorie de l'information affirme que le débit atteignable par un codage sans perte est borné inférieurement par l'entropie de la source définie par les définitions 1, 2 et le théorème 1 ci-dessous.

Définition 1 (Entropie d'une source discrète) L'entropie $H(Y)$ d'une variable aléatoire Y est définie par

$$H(Y) = - \sum_{y \in \mathcal{Y}} p(y) \log_2 p(y). \quad (1.13)$$

$H(Y) \geq 0$ avec égalité si et seulement si une réalisation $y \in \mathcal{Y}$ de Y se produit avec une probabilité 1.

Définition 2 (Entropie conditionnelle d'une source discrète) Pour deux variables aléatoires Y_1 et Y_2 à réalisations dans \mathcal{Y}_1 et \mathcal{Y}_2 respectivement, l'entropie conditionnelle $H(Y_2|Y_1)$ est définie par

$$H(Y_2|Y_1) = - \sum_{y_2 \in \mathcal{Y}_2} \sum_{y_1 \in \mathcal{Y}_1} p(y_2, y_1) \log_2 p(y_2|y_1). \quad (1.14)$$

Théorème 1 (Règle de chaîne) Pour un vecteur aléatoire $\mathbb{Y} = [Y_1, \dots, Y_N]$,

$$H(\mathbb{Y}) = H(Y_1) + \sum_{i=2}^N H(Y_i | Y_{i-1}, \dots, Y_1). \quad (1.15)$$

Cette relation se réduit à

$$H(\mathbb{Y}) = \sum_{i=1}^N H(Y_i), \quad (1.16)$$

si et seulement si les $\{Y_i\}_{i=1}^N$ sont indépendants.

Le codage sans perte représente un enjeu important lorsqu'une reconstruction parfaite de la source est nécessaire. Cependant, cette exigence restreint considérablement les capacités de compression. Lorsque l'application considérée supporte une certaine distorsion de la source reconstruite, le codage avec perte s'avère donc plus intéressant.

Codage avec perte

Lors d'un codage avec perte, on ne retient qu'une partie de l'information originale, jugée pertinente. Il y a donc une distorsion non nulle entre l'information originale et l'information reconstruite. On cherche alors à déterminer le schéma de codage s^* qui minimise le débit moyen sous contrainte d'une distorsion moyenne donnée, *i.e.*,

$$s^* = \operatorname{argmin}_{s \in \mathcal{S}} \bar{R}_{\mathbb{Y},s} \quad \text{soumis à} \quad \bar{D}_{\mathbb{Y},s} \leq \bar{D}_c \quad (1.17)$$

où \bar{D}_c est une distorsion cible fixée.

Cette optimisation est généralement très compliquée à résoudre. Une approche sous-optimale consiste à restreindre l'ensemble *complet* des schémas de compression possibles \mathcal{S} à un sous-ensemble \mathcal{S}_r de schémas de compression particuliers, par exemple constitués des mêmes types d'opérations, comme on le verra à la section 1.5.

La limite de performance débit-distorsion atteignable par un schéma de compression s^* solution de (1.17) est donnée par une fonction débit-distorsion $R(D)$ (appelée OPTA pour Optimum Performance Theoretically Attainable). Cette fonction établit la quantité d'information minimale nécessaire à la représentation d'une source pour une distorsion donnée, ou inversement, la distorsion minimale atteignable pour un débit donné. Elle est formalisée par le théorème 2 et utilise la notion d'information mutuelle, définie en fonction de l'entropie comme suit.

Définition 3 (Information mutuelle) L'information mutuelle entre deux vecteurs aléatoires \mathbb{Y} et \mathbb{X} est donnée par

$$I(\mathbb{Y}; \mathbb{X}) = H(\mathbb{Y}) - H(\mathbb{Y}|\mathbb{X}). \quad (1.18)$$

Théorème 2 (Théorème Débit-Distorsion) La fonction débit-distorsion d'une source \mathbb{Y} i.i.d. de distribution de probabilité $p(\mathbf{y})$ associée à la mesure de distorsion bornée $d(\mathbf{y}, \hat{\mathbf{y}})$ est telle que

$$\bar{R}(\bar{D}_c) = \min_{\hat{\mathbb{Y}} \in \hat{\mathcal{Y}}_{\bar{D}_c}^N} I(\mathbb{Y}; \hat{\mathbb{Y}}), \quad (1.19)$$

où $\hat{\mathcal{Y}}_{\bar{D}_c}^N = \{\hat{\mathbb{Y}} | \bar{D}_{\mathbb{Y},s} \leq \bar{D}_c\}$ et $I(\mathbb{Y}; \hat{\mathbb{Y}})$ est l'information mutuelle entre \mathbb{Y} et $\hat{\mathbb{Y}}$.

En pratique, la fonction OPTA n'est connue dans sa forme analytique que pour des sources simples, comme des sources gaussiennes. Au delà, le calcul s'avère rapidement trop complexe. En outre, la définition même de la fonction exige de caractériser la source de façon probabiliste, ce qui peut être problématique dans le cas de sources complexes, comme les images ou les vidéos. Enfin, les bornes définies nécessitent parfois des schémas de codage lourds pour être atteintes.

1.2 Compression d'image

1.2.1 Principe général

Atteindre la limite de performance débit-distorsion nécessite donc de connaître la distribution statistique de la source. Or, il n'existe pas de distribution décrivant de façon fine les signaux images. Comme nous l'avons évoqué dans la section précédente, l'optimisation (1.17) est donc réalisée sur un ensemble restreint de schémas de compression. Cet ensemble est défini par un certain nombre d'hypothèses probabilistes sur le signal (explicites ou plus souvent implicites) validées expérimentalement *a posteriori*.

Une hypothèse naturelle, très couramment exploitée, suppose des dépendances statistiques entre les pixels. Ainsi beaucoup de schémas de compression tendent à *structurer la redondance* présente dans l'image de façon à permettre une manipulation plus aisée de l'information et in fine, un codage moins coûteux. On cherchera par exemple à concentrer l'information dans un petit nombre d'éléments : une image contenant des pixels de même couleur pourra alors possiblement être parfaitement décrite via un unique coef-

ficient. Deux grands types de techniques permettent de structurer la redondance inter-pixel :

- ◆ le codage par transformation,
- ◆ le codage prédictif.

Elles font l'objet des deux sections suivantes 1.3 et 1.4 et des chapitres de contributions 3 et 4.

Dans la suite de ce manuscrit, on s'intéressera exclusivement aux schémas de compression avec perte. Par définition, et comme nous l'avons évoqué dans la section précédente, ce type de codage repose sur l'*extraction de l'information jugée pertinente*. Cette opération peut être réalisée de plusieurs façons, chacune répondant à des hypothèses particulières sur le signal. La plus courante est appelée quantification et consiste à représenter l'information par des symboles pris dans un ensemble fini appelé alphabet de quantification. Il existe là encore plusieurs types de quantification, nous abordons les plus utilisés dans la section 1.2.3.

Une fois quantifiée, l'information peut être transcrite en une séquence de bits. On parle alors de *codage entropique binaire*. Le choix d'un codage entropique est fortement lié à la nature de l'information à coder. On pourra par exemple préférer un codage par "symbole" ou en "bloc" suivant les cas. La sous-section 1.2.2 expose quelques éléments théoriques du codage entropique.

Les trois étapes "structuration de la redondance"/"extraction de l'information pertinente"/"codage entropique" sont réalisées à l'encodeur et permettent de représenter une image en une séquence de bits (que l'on veut la plus petite possible). Le décodeur, quant à lui, opère à l'inverse, en reconstruisant l'image à partir de la séquence de bits.

Les étapes de codage entropique et de structuration de la redondance sont en général inversibles. Ce sont des opérations "sans perte". Ce n'est pas le cas pour l'étape de quantification, qui ne conserve qu'une partie seulement de l'information originale.

La figure 1.1 présente une illustration schématique de la chaîne standard complète d'encodage/décodage. De façon simplifiée, on représente ici l'étape de structuration de la redondance par un opérateur inversible t . Un opérateur c également inversible correspond à l'étape de codage entropique. Enfin, on note q l'étape d'extraction de l'information réalisée à l'encodage et son pendant - mais pas inverse ! - h , l'étape d'approximation de l'information réalisée au décodage. Pour des raisons de simplicité, le signal source est vu ici comme une variable aléatoire unidimensionnelle Y .

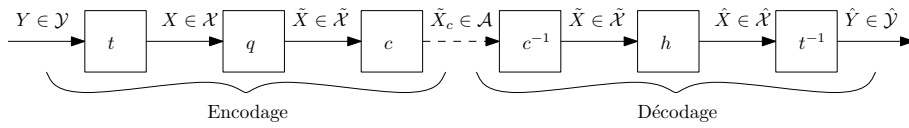


Figure 1.1 Illustration des différentes étapes formant un schéma de compression complet

Dans la suite de cette section, nous nous référerons aux notations de la figure 1.1 pour représenter les signaux qui nous occupent à chaque étape. Nous détaillons dans les sous-sections 1.2.2 et 1.2.3 les étapes de codage entropique et de quantification.

1.2.2 Codage entropique

On se restreint dans un premier temps à l'étude du codage entropique par symbole. Dans ce type de codage, chaque coefficient du vecteur source considéré est codé séparément. Reprenant les notations de la figure 1.1, on considère alors une source \tilde{X} , variable aléatoire à réalisations dans l'ensemble $\tilde{\mathcal{X}}$, appelé alphabet. Cet ensemble est supposé fini, comme l'est, par exemple, un alphabet de quantification. L'opération de codage entropique est une opération inversible, sans perte, définie comme suit.

Définition 4 *Un codage entropique de source est une opération*

$$c : \begin{cases} \tilde{\mathcal{X}} \rightarrow \mathcal{A}, \\ \tilde{x} \mapsto \tilde{x}_c = c(\tilde{x}), \end{cases} \quad (1.20)$$

où \mathcal{A} est un ensemble de séquences de symboles de taille finie.

$c(\tilde{x})$ représente la séquence de symboles - ou mot de code - associée à la réalisation \tilde{x} ; $l_{\tilde{x}}$ est sa longueur (*i.e.*, le nombre de symboles utilisés dans $c(\tilde{x})$). \mathcal{A} est appelé code. Il est dit binaire si les symboles utilisés pour former les mots de code sont 0 et 1. Dans la suite de cette section, on considère un tel codage. On cherche alors à minimiser le nombre de bits moyen utilisés pour représenter les réalisations de \tilde{X} . L'opération de codage étant une opération sans perte, cela revient à approcher le plus possible la borne sur le débit définie par l'entropie de la source \tilde{X} (cf. sous-section 1.1.2).

Codage à longueur fixe (FLC pour fixed length code en anglais)

Tous les éléments de l'ensemble $\tilde{\mathcal{X}}$ peuvent être représentés par des mots de codes de même longueur $l_{\tilde{x}} = \lceil \log_2 |\tilde{\mathcal{X}}| \rceil$ bits. Ce codage n'est optimal

(*i.e.*, il n'atteint l'entropie de la source) que pour des sources \tilde{X} uniformes, *i.e.*, pour lesquelles tous les mots de code ont la même probabilité d'être utilisés (appelée probabilité d'apparition). Dans tous les autres cas, il est plus intéressant de prendre en compte les fréquences d'utilisation relative à chaque mot de code en utilisant des codes à longueur variable. Le but est alors d'optimiser le code \mathcal{A} pour minimiser $\bar{R}_{\tilde{X},c}$ (défini par l'expression (1.3)) et approcher ainsi l'entropie de la source.

Code préfixe

Un code binaire est appelé code préfixe si aucun mot de code n'est préfixe d'un autre mot de code. Cette définition est appelée condition de préfixe.

Les deux exemples suivants (empruntés à Mallat dans [68]) illustrent cette définition.

Considérons le code qui à $\{\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4\}$ associe les mots binaires

$$\{c(\tilde{x}_1) = 0, c(\tilde{x}_2) = 10, c(\tilde{x}_3) = 110, c(\tilde{x}_4) = 101\}.$$

Ce code ne vérifie pas la condition de préfixe et conduit à des ambiguïtés de décodage. Ainsi, le message 1010 peut correspondre à $c(\tilde{x}_2)c(\tilde{x}_2)$ ou à $c(\tilde{x}_4)c(\tilde{x}_1)$.

Inversement, le code

$$\{c(\tilde{x}_1) = 0, c(\tilde{x}_2) = 10, c(\tilde{x}_3) = 110, c(\tilde{x}_4) = 111\}$$

vérifie la condition de préfixe. Tout message codé par ce code sera décodé de manière unique.

Le théorème de Shannon (théorème 3) définit les bornes sur le débit moyen atteignable par un code préfixe optimal (*i.e.*, de longueur moyenne minimale).

Théorème 3 (Shannon) *Il existe un code préfixe c^* tel que*

$$H(\tilde{X}) \leq \bar{R}_{\tilde{X},c^*} \leq H(\tilde{X}) + 1. \quad (1.21)$$

Codage de Huffman

Un code préfixe optimal peut être construit par un algorithme simple proposé par Huffman dans [51]. Cet algorithme construit un arbre binaire minimisant le nombre de bits moyen $\bar{R}_{\tilde{X},c}$. Les correspondances entre symboles et mots de code forment alors une table que le décodeur devra connaître pour transcrire les séquences binaires en symboles de $\tilde{\mathcal{X}}$ (le cas échéant, il faudra la lui transmettre).

La figure 1.2 illustre par un exemple pratique le déroulement de l'algorithme de Huffman. Une description théorique détaillée peut être trouvée dans [20].

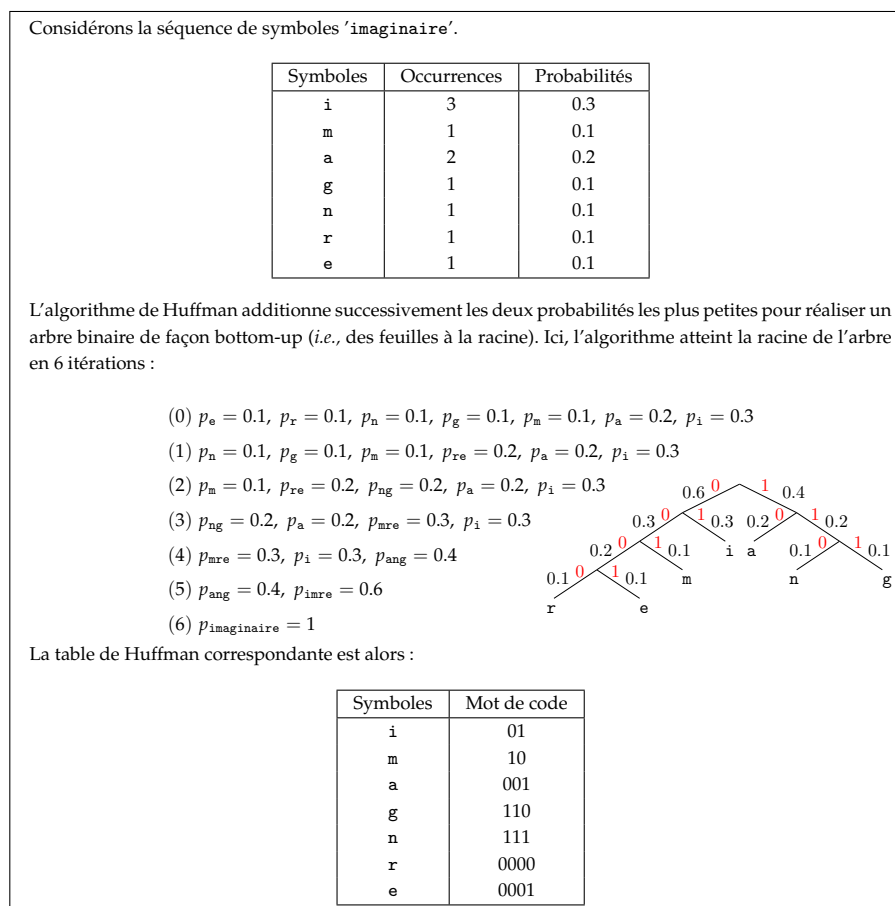


Figure 1.2 Exemple d'utilisation de l'algorithme de Huffman

On peut montrer (cf. [20]) qu'il n'existe pas d'autres codes préfixes de longueur moyenne plus petite que celle atteinte par un code construit par l'algorithme de Huffman. Cet algorithme est optimal et satisfait donc le théorème de Shannon.

Toutes les considérations d'optimalité considérées ci-dessus sont prises au sens du codage par symbole. Ce type de codage peut s'avérer assez peu intéressant lorsque l'entropie de la source est faible (par exemple pour $|\tilde{\mathcal{X}}|$ petit). Dans ce cas en effet, le surcoût potentiel de 1 bit posé par le théorème de Shannon devient important.

Une solution à ce problème est de travailler sur des “blocs” (*i.e.*, des suites) de plusieurs symboles. On peut alors montrer la proposition 1.

Proposition 1 *Le code de Huffman c_h pour un bloc de taille n requiert en moyenne un nombre de bits par symbole qui vérifie*

$$H(\tilde{X}) \leq \bar{R}_{\tilde{X}, c_h} \leq H(\tilde{X}) + \frac{1}{n}. \quad (1.22)$$

Une preuve de cette proposition est donnée dans [68]. On note que lorsque $n \rightarrow +\infty$, *i.e.*, lorsque l'on traite des séquences de bits très longues, les performances en débit atteignables par un codeur de Huffman tendent vers l'entropie de la source \tilde{X} .

Cependant, l'algorithme de Huffman n'est pas idéal pour le codage par bloc. Pour une longueur de bloc n , il nécessite le calcul des probabilités de toutes les combinaisons de n symboles et une modification de la longueur des blocs entraîne un recalcul complet. Le codage arithmétique permet d'étendre facilement la longueur des blocs sans réinitialiser les calculs.

Codage arithmétique

Introduit par Rissanen en 1976 dans [92], le codage arithmétique enregistre les symboles de façon plus structurée qu'un codage de Huffman. Le code est construit progressivement au fur et à mesure de la prise en compte des symboles.

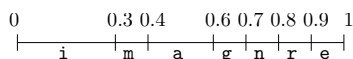
Le codage arithmétique procède par partitionnement successif de l'intervalle $[0, 1]$, résultant en un découpage de $[0, 1]$ en $|\tilde{\mathcal{X}}|^n$ intervalles, où $|\tilde{\mathcal{X}}|$ est le nombre de symboles de $\tilde{\mathcal{X}}$ et n est la longueur du bloc considéré. Chaque intervalle caractérise alors une séquence de n symboles possible, et ce sans ambiguïté.

Cette affirmation est illustrée par la figure 1.3 qui présente le codage arithmétique de la séquence 'imaginaire' introduite précédemment dans la figure 1.2. On trouve une formalisation du codage arithmétique dans [68].

Lorsque la taille des blocs augmente, la longueur des intervalles de codage diminue et leur nombre augmente. La transcription binaire des éléments choisis pour représenter chaque intervalle de codage est calculée progressivement en ajoutant des bits au fur et à mesure de l'augmentation de la séquence de symboles codée. Witten *et al.* en proposent dans [114] des implémentations efficaces. Notant $[a, a + b]$ un intervalle de codage, b correspondra à la probabilité d'apparition du bloc considéré et la séquence binaire associée aura une

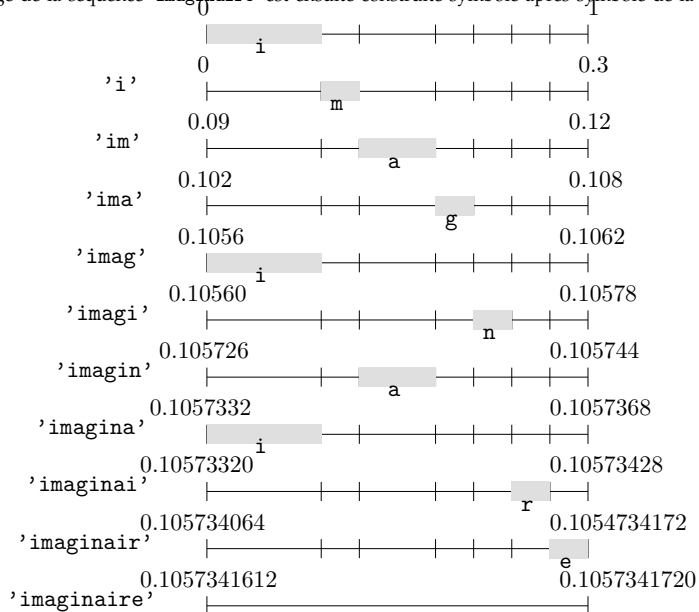
Reprenons la séquence de symboles 'imaginaire'. Dans cet exemple précis, $|\mathcal{X}| = 7$ et $n = 10$, 'imaginaire' est une séquence possible parmi 7^{10} possibilités.

Dans un premier temps, on associe à chaque symbole un intervalle dans l'espace des probabilités dont la longueur dépend de sa probabilité d'apparition (donnée dans la figure 1.2) :



Cette table d'association devra être connue du décodeur.

L'encodage de la séquence 'imaginaire' est ensuite construite symbole après symbole de la façon suivante :



Pour une taille de bloc égale à 10, l'intervalle $[0.1057341612, 0.1057341720[$ caractérise la séquence 'imaginaire' sans ambiguïté. De plus, comme les intervalles ne se recoupent pas, n'importe quel élément de $[0.1057341612, 0.1057341720[$ suffit à déterminer l'intervalle de codage.

Figure 1.3 Exemple d'utilisation du codage arithmétique

longueur l_b telle que

$$-\lceil \log_2 b \rceil \leq l_b \leq -\lfloor \log_2 b \rfloor + 2. \quad (1.23)$$

On peut alors vérifier qu'un code arithmétique c_a permet d'atteindre un nombre de bits moyen par symbole tel que

$$H(\tilde{X}) \leq \bar{R}_{\tilde{X}, c_a} \leq H(\tilde{X}) + \frac{2}{n}. \quad (1.24)$$

Ses performances sont donc légèrement moins bonnes que celles obtenues par un codage de Huffman (notons que lorsque $n \rightarrow +\infty$, l'écart de performance avec le codage de Huffman est tout à fait négligeable), mais sa relative

simplicité et sa grande adaptativité en fait un codage très souvent utilisé.

Codage par plages de zéros

Le codage par plages de zéros - Run-Length Encoding (RLE) en anglais - est utilisé pour des blocs constitués de symboles binaires uniquement. Ici, ce ne sont pas les probabilités d'apparition des symboles qui servent de base à l'encodage mais les probabilités d'apparition des *suites* de symboles identiques (par exemple 000, 11, etc...).

Une fois ces probabilités d'apparition établies, l'algorithme procède comme un codeur entropique "classique", par exemple, comme l'algorithme de Huffman en construisant un arbre binaire de façon bottom-up. Il suffit alors d'indiquer au décodeur le premier symbole du bloc (par exemple 0 ou 1) - ou de fixer une règle du premier symbole, par exemple 1, et ajouter 0 comme "longueur" possible de suite lors de l'apprentissage de la table - avant de transmettre le code binaire proprement dit.

Ce type de codage est très populaire. On le trouve dans le standard de compression JPEG par exemple, utilisé pour coder les indices des coefficients de transformée non nuls (on a alors des blocs de 0 et 1 qui se prêtent très bien au codage par plages). Le format de compression JPEG est décrit de façon plus approfondie dans la sous-section 1.3.3.

1.2.3 Quantification

La quantification précède le codage entropique. Son objectif est double : extraire l'information jugée pertinente et faciliter le codage entropique. L'information quantifiée doit alors pouvoir être décrite par un nombre de symboles suffisamment petit pour atteindre de bonnes performances en débit mais aussi suffisamment grand pour ne pas trop détériorer l'information originale.

Dans cette sous-section, nous présentons quelques méthodes de quantification et dérivons la fonction débit-distorsion OPTA pour l'une d'entre elles.

Reprenant les notations de la figure 1.1 étendues au cas multidimensionnel, on considère une source $\mathbb{X} = [X_1, \dots, X_N]^T$, vecteur aléatoire à réalisations dans \mathcal{X}^N . Cet ensemble peut être continu et non borné, comme - c'est souvent le cas - l'ensemble des réels. L'opération de quantification est une opération non inversible, définie comme suit.

Définition 5 Une quantification est une opération

$$q : \begin{cases} \mathcal{X}^N \rightarrow \tilde{\mathcal{X}}^N, \\ \mathbf{x} \mapsto \tilde{\mathbf{x}} = q(\mathbf{x}), \end{cases} \quad (1.25)$$

de l'ensemble \mathcal{X}^N vers l'ensemble fini $\tilde{\mathcal{X}}^N$.

Une quantification scalaire approxime chaque coefficient de \mathbf{x} indépendamment les uns des autres de sorte que q est constitué de N opérations séparées. Lorsque les coefficients de \mathbf{x} sont très interdépendants, un quantificateur vectoriel qui quantifie ensemble les N coefficients de \mathbf{x} peut nettement améliorer la performance d'un quantificateur scalaire. Cependant, un quantificateur vectoriel est d'une plus grande complexité qu'un quantificateur scalaire. Si une étape de structuration de la redondance est utilisée en outre de la quantification, les coefficients qui en résulteront seront par définition moins dépendants les uns des autres, et le gain apporté par le quantificateur vectoriel ne vaudra pas le coût de calcul. Les quantificateurs scalaires sont donc en pratique plus souvent utilisés.

Dans les schémas de codage que nous proposerons par la suite, nous utiliserons également une quantification scalaire. Nous détaillons donc ici plus spécifiquement ce type de quantification, mais renvoyons le lecteur curieux au livre de Gersho et Gray [41] pour une présentation approfondie de la quantification vectorielle.

Quantification scalaire

Puisqu'une quantification scalaire de \mathcal{X}^N vers $\tilde{\mathcal{X}}^N$ peut être vue comme N opérations séparées, étudier un quantificateur scalaire de dimension N revient à étudier N quantificateurs unidimensionnels. Dans la suite de cette sous-section, on considère donc un quantificateur scalaire q avec $N = 1$ et la variable aléatoire X à réalisations dans \mathcal{X} .

Si $\mathcal{X} = [a, b]$ (avec éventuellement $a = -\infty$, $b = +\infty$), l'opération de quantification revient à partitionner l'intervalle $[a, b]$ en K intervalles $\{] \alpha_{k-1}, \alpha_k]\}_{1 \leq k \leq K}$, puis à associer à chaque valeur x un point de quantification \tilde{x}_k tel que

$$\forall x \in] \alpha_{k-1}, \alpha_k], \quad q(x) = \tilde{x}_k. \quad (1.26)$$

La longueur de chaque intervalle de quantification $] \alpha_{k-1}, \alpha_k]$ est appelée pas de quantification et est noté $\Delta_k \triangleq \alpha_k - \alpha_{k-1}$.

Quantificateur de haute résolution

Un quantificateur est dit de haute résolution si la distribution de probabilités de la variable X , $p(x)$, peut être vue comme constante sur chaque intervalle de quantification $]\alpha_{k-1}, \alpha_k]$.

C'est le cas si les pas de quantification Δ_k sont suffisamment petits par rapport aux variations de $p(x)$ pour que l'on puisse négliger ces variations sur chaque intervalle de quantification.

La proposition suivante donne une expression de l'erreur quadratique moyenne $\bar{D}_{X,q}$ d'un quantificateur de haute résolution en fonction des pas de quantification Δ_k . Une preuve peut être trouvée dans [68].

Proposition 2 *Pour un quantificateur de haute résolution, l'erreur quadratique moyenne $\bar{D}_{X,q}$ est minimisée lorsque $x_k = \frac{\alpha_k + \alpha_{k-1}}{2}$, soit*

$$\bar{D}_{X,q} = \frac{1}{12} \sum_{k=1}^K \Pr\{X \in]\alpha_{k-1}, \alpha_k]\} \Delta_k^2, \quad (1.27)$$

où $\Pr\{X \in]\alpha_{k-1}, \alpha_k]\} = \int_{\alpha_{k-1}}^{\alpha_k} p(x) dx$.

Quantificateur scalaire uniforme

Un quantificateur scalaire uniforme est un quantificateur scalaire pour lequel tous les pas de quantification sont identiques :

$$\Delta_k = \Delta \quad \forall k \in \{1, \dots, K\}. \quad (1.28)$$

Si on considère un quantificateur scalaire uniforme de haute résolution, l'expression (1.27) devient

$$\bar{D}_{X,q} = \frac{\Delta^2}{12}. \quad (1.29)$$

La distorsion est alors indépendante de la source.

Dans la sous-section 1.2.2, nous avons vu que le codage entropique, opération sans perte, cherche à s'approcher au mieux de l'entropie de \tilde{X} . Précédant le codage entropique, une quantification optimale est donc une quantification qui minimise l'entropie $H(\tilde{X})$ sous contrainte d'une distorsion donnée.

Dans [42], Gish et Pierce ont établi une relation entre l'entropie $H(\tilde{X})$ et l'entropie différentielle de X formalisée dans la définition 6. C'est le résultat du théorème 4.

Définition 6 L'entropie différentielle de la variable aléatoire X est définie par

$$H_d(X) = - \int_{-\infty}^{+\infty} p(x) \log_2 p(x) dx. \quad (1.30)$$

Théorème 4 (Gish et Pierce) Si q est un quantificateur de haute résolution par rapport à $p(x)$, alors

$$H(\tilde{X}) \geq H_d(X) - \frac{1}{2} \log_2(12\bar{D}_{X,q}). \quad (1.31)$$

Il y a égalité si et seulement si q est un quantificateur uniforme.

Ce théorème montre que sous hypothèse de haute résolution, le quantificateur scalaire optimal est un quantificateur scalaire uniforme. Supposant que le codage entropique est optimal et atteint l'entropie $H(\tilde{X})$, on obtient $\bar{R}_{X,q} = H(\tilde{X})$ le débit binaire moyen atteint par le quantificateur et, en remplaçant dans (1.31),

$$\bar{D}_{X,q} = \frac{1}{12} 2^{2H_d(X)} 2^{-2\bar{R}_{X,q}}. \quad (1.32)$$

La quantification scalaire uniforme de haute résolution est l'une des rares opérations avec perte réalisées sur des signaux images dont on sache calculer et exprimer analytiquement la fonction débit-distorsion OPTA.

Lorsque la condition de haute résolution n'est pas satisfaite, le quantificateur scalaire uniforme n'est pas optimal. On cherche alors à construire un quantificateur propre aux données que l'on cherche à quantifier. Dans le cas particulier où l'on utilise un codeur entropique à longueur fixe, le débit est directement proportionnel au nombre de points de quantification. Le quantificateur optimal sera alors celui qui minimise la distorsion de quantification sous contrainte d'un nombre fixé de points de quantification. Lloyd en 1957 [64] et Max en 1960 [76] ont déterminé deux conditions nécessaires pour la construction de tels quantificateurs : la règle dite du *plus proche voisin* et la condition du *centroïde*. Ces deux conditions sont à la base des algorithmes d'apprentissage d'alphabet de quantification, dont le plus utilisé est l'algorithme de Lloyd-Max (cf. [41]).

Quantificateur uniforme à zone morte

Un quantificateur scalaire à zone morte - dead-zone en anglais - est un quantificateur scalaire où l'intervalle autour de zéro est plus large. La zone morte - ou dead-zone -, notée T , qualifie cet intervalle, qui permet à l'ensemble des valeurs considérées comme petites, d'être quantifiées à une seule et même valeur zéro.

Stricto sensus, ce type de quantificateur est non-uniforme ($T \neq \Delta$). Toutefois, si tous les autres pas de quantification sont égaux, on qualifie le quantificateur de uniforme à zone morte, et si T reste de grandeur comparable aux autres pas de quantification, on considère que l'expression (1.29) reste valide.

Ce type de quantificateur est très répandu en compression d'image. C'est celui que nous utiliserons dans les schémas de compression que nous proposerons dans la suite de ce manuscrit, en précisant la longueur de la zone morte.

La figure 1.4 présente une illustration des quantificateur scalaire uniforme et quantificateur scalaire uniforme à zone morte. Sur la figure 1.4(a) est représenté graphiquement un quantificateur scalaire uniforme de pas de quantification Δ tandis que la figure 1.4(b) montre un quantificateur scalaire uniforme à zone morte $T = 2\Delta$.

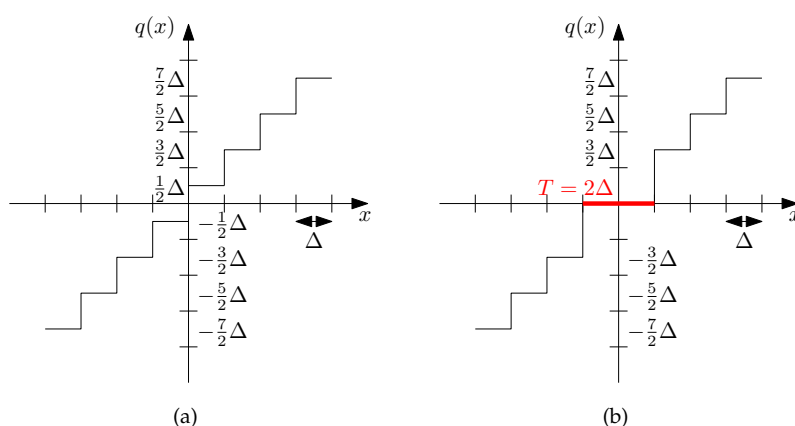


Figure 1.4 Illustration des quantificateur scalaire uniforme (a) et quantificateur scalaire uniforme à zone morte (b, ici $T = 2\Delta$)

1.3 Codage par transformation

La transformation est l'une des deux principales méthodes utilisées pour structurer la redondance présente dans l'image.

Dans cette section, nous présentons le principe général du codage par transformation, et les bases théoriques sur lequel il repose. Puis nous établissons un bref état de l'art des transformations utilisées dans la littérature,

avant de décrire deux formats de compression basés sur le codage par transformation, les standard JPEG et JPEG2000.

1.3.1 Principe général

L'idée de codage par transformation a été introduite par Kramer et Mathews en 1956 (cf. [57]). Reprenant les notations de la figure 1.1 étendues au cas multidimensionnel, on considère une source $\mathbb{Y} = [Y_1, \dots, Y_N]^T$, vecteur aléatoire à réalisations dans \mathcal{Y}^N . L'opération de transformation est une opération sans perte, inversible, définie comme suit.

Définition 7 Une transformation est une opération

$$t : \begin{cases} \mathcal{Y}^N \rightarrow \mathcal{X}^N, \\ \mathbf{y} \mapsto \mathbf{x} = t(\mathbf{y}), \end{cases} \quad (1.33)$$

de l'ensemble \mathcal{Y}^N vers l'ensemble \mathcal{X}^N .

Le plus souvent, on parlera de transformations linéaires, de sorte que $t(\mathbf{y}) = \mathbf{T}\mathbf{y}$, où \mathbf{T} est appelée matrice ou base de transformation.

Le but de la transformation est de représenter le vecteur signal \mathbf{y} , réalisation du vecteur aléatoire \mathbb{Y} , avec un petit nombre de coefficients décorrélés, réalisations x_i de variables aléatoires X_i . Ayant ainsi concentré l'information, on espère quantifier et coder ces coefficients plus efficacement.

Pourtant, il n'existe pas de résultat théorique établissant l'intérêt de la décorrélation : même si les variables X_i sont indépendantes (par exemple, décorrélées et gaussiennes), la théorie de l'information montre que la quantification vectorielle sera toujours meilleure en terme de performance débit-distorsion que la quantification scalaire. En pratique cependant, on a pu constater que plus les variables X_i sont indépendantes ou du moins décorrélées, plus la quantification scalaire est efficace et moins il y a à gagner à utiliser une quantification vectorielle, complexe à mettre en oeuvre.

Deux autres arguments, l'un intuitif, l'autre subjectif, peuvent également renforcer cette observation. L'argument intuitif tient dans l'idée que le codage par transformation opère comme un simple quantificateur vectoriel : il prend avantage de la redondance présente dans le vecteur d'entrée pour mieux coder le vecteur entier, il le fait alors d'une façon simple et ad hoc qui permet ensuite de réaliser la quantification par des quantificateurs scalaires. Enfin, on a pu montrer que certains systèmes biologiques comme l'oreille ou l'oeil qui nous intéresse ici semblent opérer dans le domaine transformé : l'oeil, par

exemple, est peu sensible aux variations rapides (correspondant aux hautes fréquences dans le domaine fréquentiel).

On peut donner une intuition mathématique de l'intérêt de la transformation en considérant un codage sans perte constitué des seules deux opérations de transformation et codage entropique. Dans la suite de cette section, nous assimilerons donc le vecteur aléatoire \mathbb{X} résultant de la transformation du vecteur \mathbb{Y} au vecteur aléatoire $\tilde{\mathbb{X}}$ "entrant" dans le codeur entropique. On comprend alors que la transformation aura pour objectif de minimiser le débit atteignable par le codeur entropique et ainsi de s'approcher de l'entropie jointe $H(\mathbb{Y})$.

La meilleure performance en débit que peut atteindre un schéma de compression est l'entropie jointe de la source $H(\mathbb{Y})$. Or, puisque la transformation est une opération inversible, on a

$$H(\mathbb{Y}) = H(\mathbb{X}), \quad (1.34)$$

et la règle de chaîne établit que

$$H(\mathbb{X}) = H(X_1) + \sum_{i=2}^N H(X_i | X_{i-1}, \dots, X_1) \leq \sum_{i=1}^N H(X_i). \quad (1.35)$$

Si le codage utilisé est un codage entropique par symbole de type Huffman ou arithmétique, nous avons vu dans la sous-section 1.2.3 que l'optimalité est obtenue pour un débit égal à la somme des entropies $\sum_i H(X_i)$. Une façon d'atteindre l'entropie jointe de la source $H(\mathbb{Y})$ est alors de rendre les coefficients X_i indépendants. C'est l'objet de la transformation. Plus précisément, la transformation cherche à décorréliser les coefficients X_i afin de s'approcher de leur indépendance. On a alors

$$H(\mathbb{Y}) = H(\mathbb{X}) \simeq \sum_{i=1}^N H(X_i). \quad (1.36)$$

Pour un signal quelconque, il est très difficile de calculer la base de transformation "optimale", *i.e.*, résultant en des coefficients de transformation indépendants. Mais si le signal \mathbb{Y} est un vecteur aléatoire gaussien, les coefficients de transformation X_i sont des variables gaussiennes dans n'importe quelle base de transformation, et dans ce cas, on montre que la base de transformation optimale est la base de Karhunen-Loève (KLT pour Karhunen-Loève Transform en anglais). Cette base diagonalise la matrice de covariance du vecteur \mathbb{X} , résultant en une décorrélation complète des coefficients X_i . Les coefficients étant gaussiens, leur décorrélation entraîne leur

indépendance. Dans le cas général (non-gaussien), la base de Karhunen-Loève ne sera pas optimale car les coefficients de transformation obtenus ne seront pas indépendants, mais du moins seront-ils décorrélés, améliorant ainsi les performances d'un codage entropique par symbole.

Si elle constitue ainsi la solution la plus adaptée pour la décorrélation des coefficients X_i , la KLT présente l'inconvénient d'être complexe en calcul (elle repose sur une analyse en composantes principales) et dépendante des données. On la remplace en pratique par la transformée en cosinus discret (DCT pour Discrete Cosine Transform en anglais), introduite en 1974 dans [2]. Shanmugam montre en effet dans [98] que cette transformée et la KLT sont proches (elles sont ainsi asymptotiquement équivalentes si la matrice de covariance du signal \mathbb{Y} est une matrice de Toeplitz). La DCT est très populaire et fréquemment utilisée dans les schémas de compression par transformation. Le format de compression JPEG, décrit dans la sous-section 1.3.3, en est un exemple.

D'autres codages, comme le codage par plages de zéros, seront d'autant plus efficaces que la transformation favorisera une "parcimonie structurée" des coefficients, *i.e.*, avec beaucoup de zéros successifs et très peu de coefficients non nuls. Le codage EZW introduit par Shapiro dans [100], par exemple, utilise les dépendances interéchelles entre les coefficients issus d'une transformation en ondelettes. Dans ce cas, ce sont les entropies conditionnelles $H(X_i|X_{i-1}, \dots, X_1)$ que l'on cherche à approcher au mieux afin, in fine, d'atteindre l'entropie jointe $H(\mathbb{Y})$.

1.3.2 Etat de l'art

Les recherches menées dernièrement en compression par transformation tendent à montrer que l'adaptation de la transformée aux caractéristiques locales de l'image permet une amélioration notable des performances. En pratique, l'optimisation de la transformée peut être réalisée à deux niveaux :

- ◆ dans le domaine spatial, en adaptant le support de la transformée,
- ◆ dans le domaine transformé, en adaptant les atomes de la base de projection aux caractéristiques du signal que l'on cherche à décrire.

Plusieurs contributions considérant ces approches peuvent être trouvées dans la littérature.

L'adaptation du support de la transformée exploite l'idée d'appliquer la transformation sur des blocs de l'image plutôt que sur l'image en entier. Cette

approche peut en effet non seulement avoir un intérêt en terme de coût de calcul, mais également en terme de performance débit-distorsion. Plusieurs techniques peuvent être considérées. La plus simple est le découpage de l'image en blocs de taille fixe, par exemple de taille 8×8 pixels comme dans le format de compression JPEG. Mais on trouve, par exemple dans [13], l'utilisation d'une DCT sur des blocs de tailles différentes via un quadtree [45, 104, 102]. Version anisotropique du quadtree, le bintree [97] permet une segmentation en blocs rectangulaires (nous y reviendrons dans la sous-section 3.2.3). Plus récemment, des arbres plus complexes ont été proposés, comme un bintree adaptatif [52] ou des arbres segmentant l'image en polygones [116]. Enfin dans [78], ce sont des méthodes de chevauchement de blocs qui sont considérées et comparées.

Parallèlement, de nouvelles transformées ont émergées, tendant vers une meilleure prise en compte des caractéristiques de l'image. Dans [50], les auteurs remplacent la DCT utilisée dans un schéma de codage de type JPEG par d'autres transformées, mieux adaptées aux statistiques locales des blocs. Dans la même idée, Sezer *et al.* ([96]) optimisent un ensemble de bases sur un ensemble d'entraînement pour maximiser la parcimonie des vecteurs transformés ; nous y ferons plus largement référence dans le chapitre 3. On peut également citer les DCT directionnelles ([122]), ondelettes ([67]), ondelettes orientées ([12]), curvelettes ([10]), contourlettes ([26]) et bandelettes ([61]), très efficaces pour décrire les contours d'une image.

On trouve enfin des contributions considérant des approches hybrides, adaptant la transformation dans les domaines spatial et transformé. Ainsi, dans [61], les auteurs optimisent la taille des blocs et la direction de bases de bandelettes simultanément.

1.3.3 Formats de compression : JPEG et JPEG2000

Le formats de compression JPEG et JPEG2000 ont été proposés respectivement en 1992 et 2000 par l'organisme de standardisation Joint Photographic Experts Group. Ils constituent les deux formats de compression avec perte d'image fixe les plus populaires et sont basés tous deux sur un schéma de codage par transformation.

JPEG

Comme nous l'avons évoqué dans la sous-section 1.3.2, le format JPEG

procède par découpage de l'image en blocs de taille 8×8 pixels. Les blocs sont alors traités successivement, d'abord transformés indépendamment via l'utilisation d'une DCT, puis quantifiés selon une matrice de quantification de taille 8×8 .

Le premier coefficient quantifié, appelé DC, représente la valeur moyenne du bloc considéré et est prédit à partir du bloc encodé précédemment. Ainsi, seule la différence entre la valeur du coefficient DC courant et celle du coefficient DC précédant est encodé, typiquement beaucoup plus petite en valeur absolue. Ce type de codage différentiel est appelé DPCM, pour Differential Pulse Code Modulation en anglais. Les 63 autres coefficients, appelés AC, sont encodés en utilisant directement les valeurs des coefficients du bloc courant.

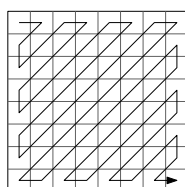


Figure 1.5 Ordre de scanning des coefficients de transformation utilisé dans le format de compression JPEG

JPEG utilise un codeur entropique bidimensionnel. Les coefficients AC sont traités selon un scanning en zigzag qui ordonnent les coefficients approximativement de la fréquence la plus basse vers la fréquence la plus haute (cf. illustration de la figure 1.5). Les coefficients successifs quantifiés à 0 forment des plages de zéros interrompues par des coefficients quantifiés non nuls. Le codage considère alors des couples (longueur,valeur), où "longueur" représente le nombre de coefficients AC consécutifs nuls entre le coefficient non nul courant et le coefficient non nul le précédant, et "valeur" correspond à la valeur (non nulle) du coefficient courant. Un symbole spécial "fin de bloc" (EOB pour End of Block en anglais) est utilisé pour signaler la fin des coefficients non nuls dans le bloc considéré. La séquence de couples (longueur,valeur) est alors compressée en utilisant des codes de Huffman ou arithmétique.

JPEG2000

Nous présentons ici les principes fondamentaux du format de compression JPEG2000. Une description plus détaillée peut être trouvée dans [72].

Le format JPEG2000 est né de l'idée de résoudre certains problèmes liés au

format JPEG.

Le plus connu est l'effet de mosaïque, du au découpage en blocs de l'image, qui apparaît sur les images lorsque le taux de compression devient important. Pour résoudre ce problème, le codage JPEG2000 procède sur l'image en entier (cependant un découpage en "tuiles" est également possible).

L'image est d'abord décomposée en sous-bandes par une transformation en ondelettes ([67]) sur plusieurs niveaux. Les coefficients issus de la transformée sont ensuite quantifiés scalairement par sous-bande, puis encodés par un codeur arithmétique complexe appelé EBCOT (pour "embedded block coding with optimized truncation" en anglais). Nous ne décrivons pas ici son fonctionnement mais renvoyons le lecteur à l'article de Taubman [106]. Cet encodage génère un train (*i.e.*, séquence) binaire imbriqué et organisé de façon progressive. Il est dit scalable, *i.e.*, le fichier compressé de l'image peut être tronqué à n'importe quel endroit et permettre cependant un décodage pertinent de l'image, résultant en une compression flexible et adaptative.

Enfin, JPEG2000 donne la possibilité de définir des régions d'intérêt *i.e.*, de spécifier à l'encodeur des zones de l'image que l'on souhaite de meilleure qualité et transmises en priorité. Cette possibilité constitue un avantage supplémentaire non négligeable par rapport au format de compression JPEG.

1.4 Codage par prédiction

La prédiction constitue la deuxième méthode permettant de structurer la redondance présente dans une image. Nous en présentons ici le principe général. Nous exposons ensuite l'exemple pratique de la prédiction dite intra utilisée dans le format de compression vidéo H.264 puis terminons cette section par un bref état de l'art des méthodes de prédiction développées dans la littérature.

1.4.1 Principe général

Jusqu'à présent, nous n'avons considéré que des codages sans mémoire, où l'information est codée indépendamment des actions passées à l'encodeur et au décodeur. Cependant, la prédictabilité d'un signal peut constituer un intérêt non négligeable : mieux on peut prédire un signal à partir du passé, moins on aura besoin d'envoyer de nouvelles informations.

La prédictibilité d'un signal est intimement liée à l'idée de redondance. Dans un codage vidéo, il pourra s'agir de redondance temporelle, entre deux ou plusieurs images successives, et, si les images sont codées via un découpage en blocs, de redondance entre les blocs constituant une image. Chacune de ces redondances donne lieu à un type de prédiction particulier : prédiction "inter", exploitant la redondance temporelle, et prédiction "intra", basée sur la redondance entre blocs d'une même image.

La prédiction intra peut être similairement utilisée dans le codage d'image fixe basé sur un découpage en blocs. C'est donc elle qui nous intéresse particulièrement ici. La prédiction peut être considérée seule (suivie par exemple d'une quantification vectorielle) ou associée à une transformation. Il y a en effet complémentarité entre les deux méthodes : la transformation structure la redondance entre les pixels d'un bloc, tandis que la prédiction exploite la redondance entre blocs.

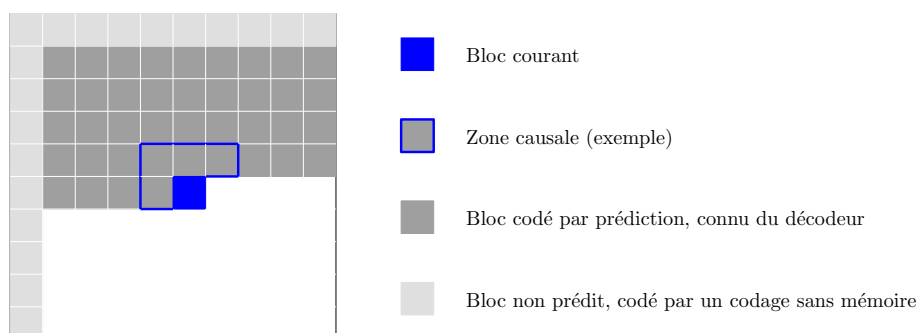


Figure 1.6 Schéma de codage prédictif standard

Un schéma de codage prédictif standard est illustré sur la figure 1.6. Les blocs situés le long du bord gauche et en haut de l'image constituent la "base causale" du codage prédictif. Ils ne sont pas prédits mais encodés indépendamment via par exemple un codage par transformation. Les autres blocs sont ensuite codés par prédiction successivement de la gauche vers la droite et de haut en bas sur la base des blocs précédemment encodés et reconstruits - formant la *zone causale* de la prédiction. La zone causale peut varier selon les schémas proposés (en terme de taille et de position). Par exemple, dans la prédiction utilisée dans le format de compression vidéo H.264 détaillée dans la sous-section 1.4.2, la zone causale est constituée des pixels adjacents au bloc que l'on souhaite prédire. Le bloc prédit est ensuite soustrait au bloc

courant connu de l'encodeur, résultant en une information résiduelle qui devra être transmise au décodeur. Cette information résiduelle peut alors être codée par transformation par exemple.

1.4.2 Méthode de prédiction intra utilisée dans H.264

La prédiction intra utilisée dans le format de compression H.264 repose sur une division de l'image en macroblocs de taille 16×16 pixels et en blocs de taille 4×4 pixels et 8×8 pixels. Nous focalisons ici sur la prédiction intra réalisée sur des blocs de taille 4×4 pixels mais référons au livre de Richardson ([91]) pour une présentation complète du format de compression H.264.

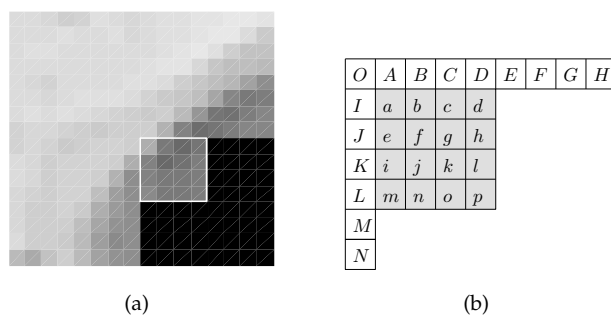


Figure 1.7 Illustration de la prédiction intra de type H.264 (a) et notations des pixels (b)

Il y a 9 modes de prédiction pour les blocs de taille 4×4 pixels. L'encodeur sélectionne le mode de prédiction qui minimise la différence entre le bloc prédit P et le bloc original que l'on cherche à encoder. La figure 1.7 montre un bloc de taille 4×4 pixels que l'on souhaite prédire et sa zone causale. Les pixels au dessus et sur la gauche (indités de A, \dots, O) ont été précédemment encodés et reconstruits et sont donc disponibles à l'encodeur et au décodeur. Les pixels a, b, \dots, p du bloc prédit sont calculés sur la base des pixels A, \dots, O comme illustré sur la figure 1.8.

Les flèches de la figure 1.8 indiquent la direction de prédiction de chaque mode. Pour les modes 3–8, les pixels prédits sont formés à partir d'une moyenne pondérée des pixels A, \dots, O . Par exemple, si le mode 4 est sélectionné, le pixel en haut à droite du bloc considéré (indité par d dans la figure 1.7(b)) est prédit par $\llbracket \frac{B}{4} + \frac{C}{2} + \frac{D}{4} \rrbracket$, où $\llbracket \cdot \rrbracket$ signifie ici "arrondi" à l'entier le plus proche.

Notons que le mode 8 ne peut être utilisé que si le bloc situé en diagonal

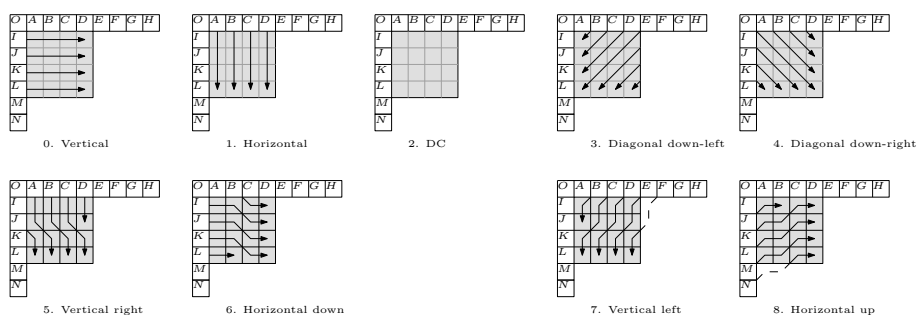


Figure 1.8 Illustration des différents modes de prédiction définis dans la prédiction intra H.264

à gauche du bloc traité est connu du décodeur. Dans le cas contraire, le mode est dit non-causal et n'est pas considéré pendant la prédiction.

Le bloc résiduel issu de la différence entre le bloc original et sa prédiction est ensuite codé par une transformation DCT suivie d'une quantification scalaire uniforme, puis transmis au décodeur.

1.4.3 État de l'art

La plupart des contributions de la littérature concernant la prédiction spatiale partent de la méthode utilisée dans la compression vidéo H.264 et en proposent des améliorations ou extensions.

Un premier axe de recherche est constitué par le partitionnement des blocs considérés lors de la prédiction. Dans [112], l'auteur emprunte les blocs de taille rectangulaire à la prédiction inter de H.264 [91] (16×8 , 8×16 , 8×4 et 4×8 pixels) et les ajoute aux blocs carrés classiquement utilisés dans la prédiction intra de H.264. On trouve également dans des articles plus récents (cf. par exemple [21]) des partitionnements plus complexes, reposant par exemple sur un modèle paramétrique linéaire.

D'autres travaux se sont plus particulièrement intéressés aux prédicteurs en eux-mêmes. Certains proposent de simplement enrichir la méthode de prédiction intra de H.264 en ajoutant des lignes de pixels supplémentaires aux pixels de référence [75], ou des modes bidirectionnels, issus de la combinaison de deux modes de prédiction utilisés dans H.264, aux 9 modes déjà existants [117]. D'autres envisagent de tout autres prédicteurs. Conservant l'idée de baser la prédiction sur les pixels causaux entourant le bloc à prédire, la méthode dite de "Template Matching" [105] propose de rechercher dans un voisinage causal des pixels de même configuration spatiale et de caractéristiques simi-

lares. Le bloc à prédire est alors approximé par le bloc entouré des pixels les plus “ressemblants”. Pour une meilleure adéquation, le template matching est réalisé sur des sous-blocs de taille 2×2 pixels issus de la division en 4 du bloc à prédire. Enfin, citons encore les travaux [23] proposant des manipulations géométriques de blocs de référence, pris dans un voisinage causal, et codant ensuite les caractéristiques de la transformation résultant en la meilleure approximation du bloc considéré.

Un dernier axe de recherche exploré dans la littérature est constitué par le codage des résidus de prédiction. Dans [117], parallèlement à leur proposition de modes bidirectionnels, les auteurs élaborent des transformées directionnelles séparables dérivées de la KLT (qui, elle, est généralement non-séparable). On peut également citer la méthode de transformation introduite dans [93] qui s’appuie sur une permutation des coefficients du bloc résiduel et permet ainsi une meilleure prise en compte, par la DCT, des directionnalités persistantes.

1.5 Optimisation débit-distorsion

Lorsqu’on ne connaît pas précisément la distribution de la source, la recherche de la limite de performance débit-distorsion est particularisée à un type de source et un type de codage. La fonction débit-distorsion limite qui en découle n’est donc plus théorique et générale, comme celle définie par le théorème débit-distorsion (Théorème 2), mais opérationnelle et propre à la source et au schéma de codage étudiés. Les grandeurs optimisées ne sont plus les débit et distorsion moyens mais les débit et distorsion réels, calculés selon les expressions (1.4) et (1.10).

L’approche “optimisation débit-distorsion” restreint la recherche du schéma de codage optimal à un sous-ensemble \mathcal{S}_r de \mathcal{S} . De façon analogue à l’optimisation (1.17), on cherche alors à déterminer le schéma de codage $s_r^* \in \mathcal{S}_r$ qui minimise le débit réel sous contrainte d’une distorsion donnée *i.e.*,

$$s_r^* = \underset{s \in \mathcal{S}_r}{\operatorname{argmin}} R_{y,s} \quad \text{soumis à } D_{y,s} \leq D_c, \quad (1.37)$$

où \mathcal{S}_r est l’ensemble des schémas de codage fixés par les hypothèses et D_c est une distorsion cible fixée. Ou bien, inversement, le schéma de codage $s_r^* \in \mathcal{S}_r$

qui minimise la distorsion réelle sous contrainte d'un débit donné *i.e.*,

$$s_r^* = \underset{s \in \mathcal{S}_r}{\operatorname{argmin}} D_{\mathbf{y},s} \text{ soumis à } R_{\mathbf{y},s} \leq R_c, \quad (1.38)$$

où R_c est un débit cible fixé.

Ce dernier cas est le plus courant : les applications multimédia imposent plus souvent une contrainte sur le débit que sur la distorsion. C'est donc l'optimisation (1.38) que nous considérons dans la suite de cette section, mais l'analogie peut être faite immédiatement pour l'optimisation du débit (1.37).

Dans la plupart des cas pratiques étudiés, \mathcal{S}_r est fini, *i.e.*, s_r^* est choisi parmi un nombre fini de schémas de codage. On peut alors tracer les points de fonctionnement débit-distorsion de tous les schémas de codage inclus dans \mathcal{S}_r . Ainsi que le montre la figure 1.9, la frontière entre les performances atteignables et non-atteignables par les schémas de codage de \mathcal{S}_r est définie par l'enveloppe convexe de l'ensemble des points de fonctionnement.

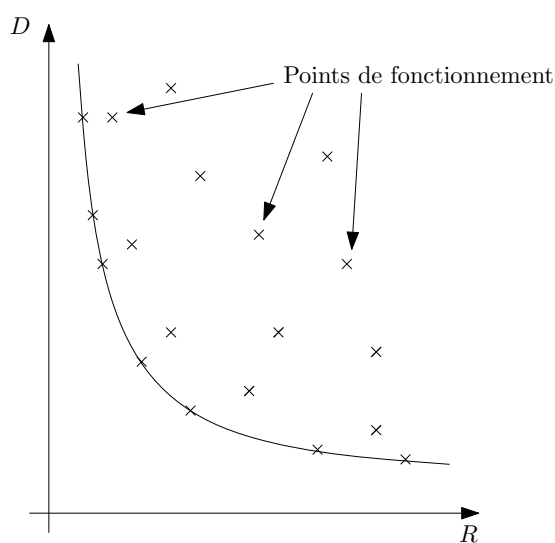


Figure 1.9 Ensemble des points de fonctionnement (R, D) définis par un ensemble de schémas de compression donnés et enveloppe convexe correspondante

La résolution du problème contraint (1.38) discret est basée sur la version discrète de l'optimisation Lagrangienne introduite d'abord par Everett dans [36], puis reprise dans le contexte de codage de source par Shoham et Gersho dans [101]. Par la suite, cette approche a été utilisée très largement dans la littérature.

Théorème 5 Pour $\lambda \geq 0$, la solution $s_r^*(\lambda)$ du problème non contraint

$$\operatorname{argmin}_{s \in \mathcal{S}_r} D_{y,s} + \lambda R_{y,s} \quad (1.39)$$

est également solution du problème contraint (1.38) de contrainte $R_c = R_{y,s_r^*(\lambda)}$, i.e., $R_{y,s} \leq R_{y,s_r^*(\lambda)}$.

Ortega et Ramchandran spécifient dans [82], que les solutions du problème non-contraint (1.39) sont les points de fonctionnement situés sur l'enveloppe convexe définie comme sur la figure 1.9.

Le multiplicateur Lagrangien λ permet de sélectionner des points de compromis débit-distorsion spécifiques. Graphiquement, il représente la pente de la tangente à l'enveloppe convexe au point débit-distorsion considéré. Minimiser la fonction de coût Lagrangienne (1.39) lorsque $\lambda = 0$ est équivalent à minimiser la distorsion, i.e., à sélectionner le point de fonctionnement de l'enveloppe convexe le plus proche de l'axe horizontal. Inversement, minimiser la fonction de coût Lagrangienne (1.39) lorsque λ est très grand, est équivalent à minimiser le débit, et ainsi à trouver le point de fonctionnement de l'enveloppe convexe le plus proche de l'axe vertical. Les valeurs de λ intermédiaires définissent les points de fonctionnement intermédiaires de l'enveloppe convexe.

Le théorème 5 définit une condition suffisante mais pas nécessaire de l'égalité entre les problèmes contraint et non-contraint (1.38) et (1.39). La figure 1.10 en donne l'illustration.

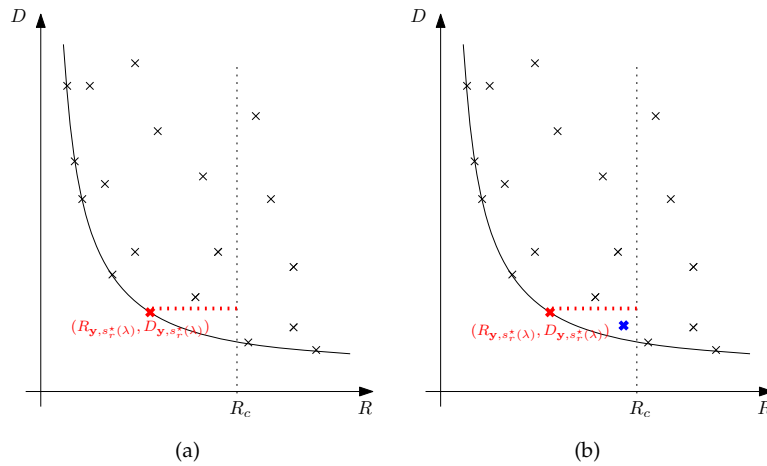


Figure 1.10 Illustration de la non-nécessité de la condition définie par le théorème 5

Sur le graphe 1.10(a), la solution Lagrangienne $(R_{\mathbf{y},s_r^*(\lambda)}, D_{\mathbf{y},s_r^*(\lambda)})$ (en rouge) donne les meilleures performances débit-distorsion sous la contrainte de débit R_c (il n'y a pas d'autres points de fonctionnement dans l'espace délimité par les trois contraintes $D = D_{\mathbf{y},s_r^*(\lambda)}$, $R = R_c$ et l'enveloppe convexe) sans valider la condition $R_c = R_{\mathbf{y},s_r^*(\lambda)}$. Mais ce n'est pas toujours le cas, comme le montre le graphe 1.10(b) où $R_c \neq R_{\mathbf{y},s_r^*(\lambda)}$, et la solution Lagrangienne $(R_{\mathbf{y},s_r^*(\lambda)}, D_{\mathbf{y},s_r^*(\lambda)})$ (en rouge) n'est pas solution de l'optimisation sous contrainte (1.38) (en bleu).

Ceci tient au caractère discret de l'optimisation Lagrangienne. Dans les cas où l'enveloppe convexe est décrite avec un nombre important de points, la configuration 1.10(b) est plus rare et, si elle advient, l'écart entre la solution Lagrangienne et la solution optimale sera négligeable. Par ailleurs, l'optimisation Lagrangienne présente une complexité calculatoire plus faible. Cet atout en fait une approximation souvent en utilisée en pratique.

Décompositions parcimonieuses

2

Dans le chapitre précédent, nous avons introduit quelques principes de la compression d'image et évoqué l'intérêt de la description d'un signal par un petit nombre de coefficients non nuls. Cet intérêt est intuitif en codage par transformation notamment, où l'utilisation ultérieure d'un codage entropique par plages de zéros sera d'autant plus valorisée. Il est un peu moins évident en codage par prédiction ; nous verrons dans le chapitre 4 comment cette idée peut être exploitée comme une information a priori sur le signal que l'on cherche à prédire.

Dans ce chapitre, nous nous restreindrons à l'étude de signaux réels par souci de simplicité, et parce que nous traiterons de signaux images réels par la suite.

La description d'un signal par un petit nombre de coefficients non nuls est appelée *décomposition parcimonieuse*. Formulation et résolution de recherche de décompositions parcimonieuses, taille et propriétés du *dictionnaire* de décomposition sont autant d'axes de recherche de ce domaine. Nous en présentons ici les principaux résultats.

2.1 Problèmes d'optimisation

La décomposition parcimonieuse d'un signal \mathbf{y} dans un dictionnaire \mathbf{D} , ensemble de vecteurs, peut être de deux types :

- elle peut être *représentation*, auquel cas le signal s'exprime *exactement* sous la forme d'une combinaison d'un petit nombre de vecteurs du dictionnaire, $\mathbf{y} = \mathbf{D}\mathbf{x}$,
- ou *approximation*, et dans ce cas, \mathbf{y} est *approché* par la combinaison d'un petit nombre de vecteurs du dictionnaire, $\mathbf{y} \approx \mathbf{D}\mathbf{x}$.

Le vecteur \mathbf{x} est le vecteur de décomposition de \mathbf{y} dans \mathbf{D} . Il contient les coefficients de pondération de la combinaison de vecteurs représentant ou approximant \mathbf{y} .

La *parcimonie* du vecteur $\mathbf{x} = [x_1, \dots, x_M]^T$ est le nombre de coefficients *nuls* dans \mathbf{x} . Moins utilisée, la notion de *diversité* correspond, elle, au nombre de coefficients *non nuls*. Si \mathbf{x} est de dimension M , on a donc

$$\text{diversité} = M - \text{parcimonie}. \quad (2.1)$$

On parle plus souvent de “parcimonie” que de “diversité” mais c’est bien une mesure de diversité qui est prise en compte pour caractériser la parcimonie d’un vecteur \mathbf{x} . Cette mesure “idéale” (par opposition aux mesures “relâchées” qui feront l’objet de la sous-section 2.1.1), formalisée par la pseudo-norme ℓ_0 , compte le nombre de coefficients non nuls. Elle est notée $\|\cdot\|_0$:

$$\|\mathbf{x}\|_0 = \sum_{i=1}^M |x_i|^0 = |\mathcal{I}|, \quad (2.2)$$

où $\mathcal{I} = \{i | x_i \neq 0\}$. Notons que cette mesure de parcimonie ne satisfait pas l’une des trois propriétés des normes, la condition d’homogénéité (2.47) (cf. annexe de ce chapitre), d’où son appellation de pseudo-norme. Les propriétés des normes sont rappelées en annexe de ce chapitre.

Le problème de représentations parcimonieuses “standard” consiste à rechercher le vecteur de coefficients \mathbf{x} le plus parcimonieux qui mène à la reconstruction exacte du vecteur \mathbf{y} dans le dictionnaire \mathbf{D} . On le formalise de la façon suivante :

$$\mathcal{P}_0 : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{soumis à} \quad \mathbf{D}\mathbf{x} = \mathbf{y}. \quad (2.3)$$

Le dictionnaire \mathbf{D} est *en général redondant* : l’ensemble de sélection des vecteurs est exprimé sous la forme d’une matrice \mathbf{D} de M colonnes $\mathbf{d}_k \in \mathbb{R}^N$ avec $M \geq N$, où chaque colonne représente un vecteur - ou *atome*.

Dans la suite de cette section, nous présentons d’abord les mesures relâchées de la parcimonie et leur intérêt dans la résolution des problèmes inverses parcimonieux, puis étudions leur équivalence dans un deuxième temps. La dernière sous-section est consacrée aux approximations parcimonieuses.

2.1.1 Mesures relâchées de parcimonie

Lorsque le dictionnaire \mathbf{D} est redondant, le problème (\mathcal{P}_0) est NP-complet, *i.e.*, il ne peut être résolu qu’en considérant toutes les combinaisons possibles

d'atomes, ce qui n'est pas envisageable en grande dimension. Pour remédier à cette difficulté, des mesures "relâchées" de la parcimonie ont été introduites, correspondant à des (quasi-)normes ℓ_p de \mathbf{x} :

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^M |x_i|^p \right)^{\frac{1}{p}}, \quad 0 < p < \infty. \quad (2.4)$$

Pour $0 < p < 1$, la condition d'inégalité triangulaire (2.48) (cf. annexe de ce chapitre) des normes n'est pas validée, elle est remplacée par la condition d'inégalité quasi-triangulaire

$$\forall (\mathbf{x}_1, \mathbf{x}_2) \in (\mathbb{R}^M)^2, \quad \|\mathbf{x}_1 + \mathbf{x}_2\|_p^p \leq \|\mathbf{x}_1\|_p^p + \|\mathbf{x}_2\|_p^p.$$

La mesure de parcimonie (2.4) constitue alors une quasi-norme. Pour $p \geq 1$ par contre, les trois propriétés des normes sont vérifiées, la mesure de parcimonie (2.4) définit bien une norme. Dans le reste du manuscrit, nous utiliserons abusivement du même terme, norme, pour recouvrir les trois notions de pseudo-norme, quasi-norme et norme.

Le recours à des mesures relâchées de la parcimonie tient à la propriété de convexité de certaines d'entre elles : pour $p < 1$, une norme ℓ_p présentera des courbes de niveaux non-convexes (c'est également le cas pour la mesure "idéale" en norme ℓ_0 de la parcimonie) mais pour $p \geq 1$, elles seront convexes. Or la convexité est souhaitable : elle permet la mise en oeuvre d'algorithmes efficaces pour résoudre le problème (\mathcal{P}_p) , obtenu en substituant la norme ℓ_p à la norme ℓ_0 dans (\mathcal{P}_0) ,

$$\mathcal{P}_p : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_p \quad \text{soumis à} \quad \mathbf{D}\mathbf{x} = \mathbf{y}. \quad (2.5)$$

Cependant, toutes les normes ℓ_p n'apportent pas les mêmes résultats ni les mêmes performances en terme de parcimonie. Ainsi, considérons un signal y unidimensionnel et un dictionnaire \mathbf{D} formé de deux atomes unidimensionnels ($N = 1, M = 2$). La représentation de y dans \mathbf{D} sera le vecteur $\mathbf{x} = [x_1, x_2]$ bidimensionnel. Une interprétation graphique du problème (\mathcal{P}_p) est donnée sur la figure 2.1 pour trois valeurs différentes de p : $p < 1, p = 1, p > 1$. A chacune de ces catégories correspond un graphe (respectivement figures 2.1(a), (b) et (c)). Les courbes de niveaux des normes y sont représentées, ainsi que la droite définie par $y = \mathbf{D}\mathbf{x}$ (en rouge). La solution au problème (2.5) est le point $\mathbf{x}^* = [x_1^*, x_2^*]$, situé à l'intersection de la droite définie par $y = \mathbf{D}\mathbf{x}$ et de la courbe de niveau de plus petite valeur. Elle est indiquée en bleu.

Pour $p \leq 1$, la solution sera effectivement parcimonieuse ($x_1 = 0, x_2 \neq 1$) mais pour $p > 1$, elle sera de dimension supérieure au signal que l'on cherche à représenter ($x_i \neq 0 \quad \forall i \in \{1, 2\}$).

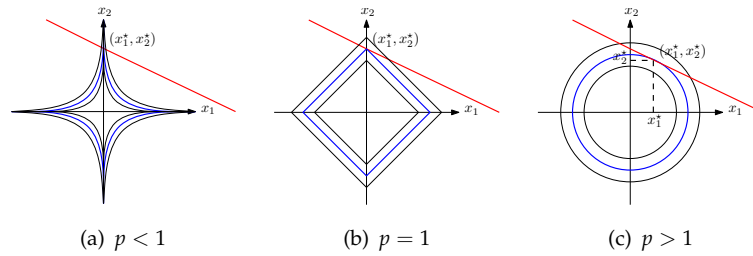


Figure 2.1 Illustration de la (non-)convexité des normes ℓ_p , pour ($p < 1$, figure (a)) $p \geq 1$, figures (b) et (c)

En pratique, beaucoup de contributions proposent de remplacer la norme ℓ_0 par la norme ℓ_1 . Avantaguse de par son caractère convexe, elle présente en outre l'intérêt d'être équivalente à la norme ℓ_0 sous certaines conditions (cf. théorème 6, section suivante). Ce n'est pas le cas pour les normes de paramètre $p > 1$, qui sont convexes mais n'encouragent pas la parcimonie. Dans la suite de cette section, nous restreignons notre étude aux valeurs de p comprises entre 0 et 1.

2.1.2 Equivalence et unicité des solutions de (\mathcal{P}_p)

Plusieurs études menées ces dernières années ont permis de caractériser les solutions admises par le problème (\mathcal{P}_p) selon la valeur du paramètre p . En particulier, les contributions de Donoho et Elad [28] parallèles à celles de Gribonval et Nielsen [43] se sont penchées sur les conditions garantissant l'unicité et l'équivalence des solutions aux problèmes (\mathcal{P}_p) , pour respectivement $p \in \{0, 1\}$ et plus généralement $p \in [0, 1]$. Le théorème 6 résume leurs résultats.

Théorème 6 Soit \mathbf{D} un dictionnaire arbitraire dans un espace de Hilbert de dimension finie ou infinie et $\mu(\mathbf{D}) \triangleq \max_{k \neq k'} |\langle \mathbf{d}_k, \mathbf{d}_{k'} \rangle|$. Si $\mathbf{y} = \sum_i x_i \mathbf{d}_i$ avec

$$\|\mathbf{x}\|_0 < \frac{1}{2} \left(1 + \frac{1}{\mu(\mathbf{D})} \right), \quad (2.6)$$

alors $\mathbf{x}^* = \mathbf{x}$ est la solution unique des problèmes de minimisation (\mathcal{P}_p) , pour $0 \leq p \leq 1$.

La grandeur $\mu(\mathbf{D})$ est appelée *cohérence* du dictionnaire \mathbf{D} . Elle mesure la proximité maximale entre les atomes d'un dictionnaire, sa valeur est comprise entre 0 et 1 si les atomes de \mathbf{D} sont normés à 1. Ainsi, si \mathbf{D} est une base orthonormale, $\mu(\mathbf{D}) = 0$ et le théorème 6 affirme que tout vecteur \mathbf{x} peut être retrouvé sans ambiguïté quelle que soit sa parcimonie; en revanche il suffit que

deux atomes du dictionnaire soient colinéaires (identiques s'ils sont normés à 1), pour que $\mu(\mathbf{D}) = 1$ et qu'aucun vecteur \mathbf{x} ne puisse être retrouvé sans ambiguïté.

Les conséquences du théorème 6 sont très importantes : il autorise et justifie le recours à des mesures relâchées de la parcimonie. En particulier, la norme ℓ_1 permet l'utilisation de techniques de programmation linéaire pour la résolution de (\mathcal{P}_1) . C'est l'approche adoptée par Chen *et al.* dans [16] pour l'algorithme de Basis Pursuit (BP).

2.1.3 Approximations parcimonieuses

L'approximation parcimonieuse autorise un écart par rapport à la décomposition parcimonieuse. Elle repose sur les deux notions de qualité d'approximation et de parcimonie du vecteur de décomposition. La qualité d'approximation est en général mesurée par une erreur quadratique entre le signal réel \mathbf{y} et l'approximation parcimonieuse $\mathbf{D}\mathbf{x}$: $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2$. La parcimonie est prise en compte selon les mesures présentées dans la sous-section 2.1.1.

Trois grands types de problèmes d'optimisation peuvent alors être considérés.

Parcimonisation

Le premier est le pendant du problème (2.5). On souhaite trouver le vecteur \mathbf{x} le plus parcimonieux, *i.e.*, contenant le moins de coefficients non nuls, sous la contrainte que l'erreur d'approximation est inférieure à un seuil fixé $\epsilon \geq 0$. Ce problème est formalisé de la façon suivante

$$\mathcal{P}_p^P : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_p \quad \text{soumis à} \quad \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 \leq \epsilon. \quad (2.7)$$

Approximation

Inversement, on peut également rechercher le vecteur parcimonieux \mathbf{x} qui conduit à l'approximation parcimonieuse la plus proche de \mathbf{y} au sens de l'erreur quadratique, sous la contrainte que \mathbf{x} a une parcimonie supérieure à un seuil fixé, *i.e.*, la mesure de la parcimonie de \mathbf{x} est inférieure à un certain seuil L . Ce problème s'écrit comme suit

$$\mathcal{P}_p^A : \quad \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 \quad \text{soumis à} \quad \|\mathbf{x}\|_p \leq L. \quad (2.8)$$

Si la parcimonie de \mathbf{x} est mesurée par la norme ℓ_0 , le paramètre L correspondra au nombre maximal de coefficients non nuls dans \mathbf{x} .

Régularisation

Enfin, le vecteur \mathbf{x} peut être cherché comme la solution d'un compromis entre qualité d'approximation et parcimonie. Le point de compromis est alors fixé par un paramètre $\lambda \geq 0$ et le problème peut être formulé de la façon suivante

$$\mathcal{P}_p^R : \quad \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_p. \quad (2.9)$$

Un facteur $\frac{1}{2}$ est parfois ajouté en pondération de l'erreur d'approximation $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2$ pour faciliter la dérivation d'algorithmes. On note que cette formalisation peut être vue comme une version Lagrangienne des problèmes de parcimonisation et d'approximation (2.7)-(2.8).

Dans la suite de ce chapitre, nous nous focaliserons sur la recherche d'approximations parcimonieuses en remarquant que le problème (2.5) peut être vu comme un cas particulier du problème (2.7) pour $\epsilon = 0$.

2.2 Recherche d'approximations parcimonieuses

Chacune des formalisations (2.7) à (2.9) conduit à la conception d'un ou plusieurs algorithmes différents. Dans cette section, nous passons en revue les principaux algorithmes existants, en commençant par le cas particulier où le dictionnaire \mathbf{D} est une base orthonormée.

2.2.1 Cas particulier : base orthonormée

L'utilisation d'une base orthonormée (*i.e.*, $\|\mathbf{d}_k\|_2 = 1 \ \forall k \in \{1, \dots, M\}$, $\langle \mathbf{d}_k, \mathbf{d}_{k'} \rangle = 0$ si $k \neq k'$, et $N = M$) comme dictionnaire d'approximation est un cas simple permettant une résolution rapide des problèmes d'approximation (\mathcal{P}_0^P), (\mathcal{P}_0^A) et (\mathcal{P}_0^R) sous la contrainte "idéale" de parcimonie, $p = 0$.

La solution est immédiate pour les problèmes de parcimonisation et d'approximation en remarquant que

$$\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 = \|\mathbf{x} - \mathbf{D}^T \mathbf{y}\|_2^2 = \sum_{i=1}^M (x_i - \mathbf{d}_i^T \mathbf{y})^2. \quad (2.10)$$

On peut montrer que la solution \mathbf{x}^* à ces problèmes est obtenue en seuillant les coefficients $\mathbf{d}_i^T \mathbf{y}$.

Dans le cas de la parcimonisation (\mathcal{P}_0^P), la solution \mathbf{x}^* sera obtenue en mettant à zéro petit à petit, dans l'ordre croissant de leur valeur, les coefficients du produit $\mathbf{D}^T \mathbf{y}$ jusqu'à ce que la contrainte $\|\mathbf{x}^* - \mathbf{D}^T \mathbf{y}\|_2^2 \leq \epsilon$ soit juste encore

satisfaite. Dans le cas de l'approximation (\mathcal{P}_0^A), la solution \mathbf{x}^* sera obtenue en ne retenant que les L plus grands coefficients du produit $\mathbf{D}^T \mathbf{y}$.

La résolution du problème de régularisation (\mathcal{P}_0^R) est un peu moins évidente mais se réduit là encore à une simple opération de seuillage ([96]) :

$$x_i^* = \mathcal{T}_\lambda^0(\mathbf{d}_i^T \mathbf{y}) \triangleq \begin{cases} \mathbf{d}_i^T \mathbf{y} & \text{si } |\mathbf{d}_i^T \mathbf{y}| > \sqrt{\lambda}, \\ 0 & \text{sinon.} \end{cases} \quad (2.11)$$

Un résultat similaire a été obtenu par Donoho dans [27] pour la résolution du problème de régularisation (\mathcal{P}_1^R). Dans le cas d'une base orthonormée, il a montré que ce problème peut être résolu par un seuillage doux :

$$x_i^* = \mathcal{T}_\lambda^1(\mathbf{d}_i^T \mathbf{y}) \triangleq \begin{cases} \mathbf{d}_i^T \mathbf{y} - \lambda/2 & \text{si } \mathbf{d}_i^T \mathbf{y} > \lambda/2, \\ 0 & \text{si } |\mathbf{d}_i^T \mathbf{y}| \leq \lambda/2, \\ \mathbf{d}_i^T \mathbf{y} + \lambda/2 & \text{si } \mathbf{d}_i^T \mathbf{y} < -\lambda/2. \end{cases} \quad (2.12)$$

Le choix d'un seuillage dur ou doux (norme ℓ_0 ou norme ℓ_1) est motivé par le contexte de calcul de l'approximation parcimonieuse. Dans le cas de débruitage, par exemple, où l'on suppose le modèle

$$\mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{n} \quad (2.13)$$

avec \mathbf{n} bruit, le paramètre λ est choisi de façon à seuiller, avec une grande probabilité, juste au dessus de l'amplitude des coefficients de bruit n_i . Un seuillage doux permet alors de reconstruire le signal en évitant les transitions brutales dues au bruit [68]. La figure 2.2 illustre les deux seuillages dur et doux définis respectivement par (2.11) et (2.12).

2.2.2 Algorithmes de recherche

Le cas plus général où \mathbf{D} est un dictionnaire redondant ($M > N$) peut avoir un intérêt non négligeable dans la description de signaux complexes comme les signaux audio ou les images. Considérons un signal \mathbf{y} de dimension N et son approximation parcimonieuse résultant de la combinaison de $L < N$ atomes. Pour une parcimonie donnée (pour un L fixé), plus le dictionnaire de représentation sera redondant, *i.e.*, plus l'ensemble de sélection des vecteurs contiendra de vecteurs différents, plus l'approximation parcimonieuse aura de chance d'être proche du signal \mathbf{y} . Réciproquement, pour une erreur d'approximation donnée, un dictionnaire très redondant diminuera le nombre d'atomes nécessaires pour approximer le signal \mathbf{y} .

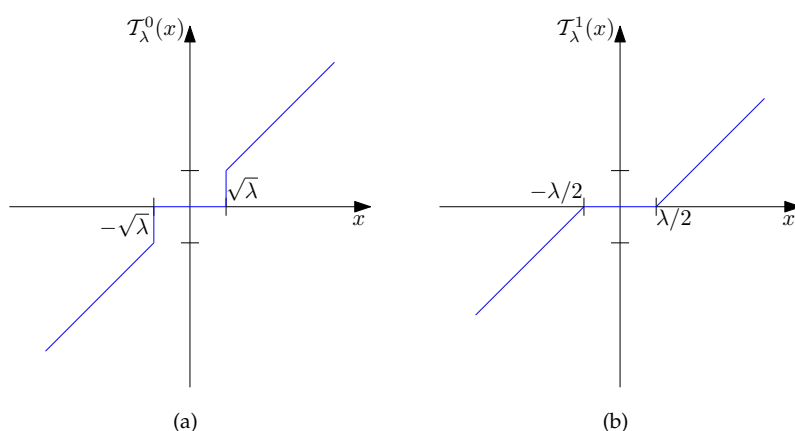


Figure 2.2 Illustration des seuillages dur (a) et doux (b), définis respectivement par (2.11) et (2.12)

Cependant, lorsque le dictionnaire \mathbf{D} est redondant, les problèmes (\mathcal{P}_0^p) , (\mathcal{P}_0^A) et (\mathcal{P}_0^R) , qui utilisent la mesure “idéale” de parcimonie sont NP-complets. On ne trouve alors dans la littérature que des algorithmes sous-optimaux, cherchant à approcher au mieux la solution optimale.

Algorithmes de seuillage itératifs

Résolvant le problème de régularisation (\mathcal{P}_p^R) avec $p = 0$ ou $p = 1$, les algorithmes de seuillage itératifs sont une extension des algorithmes de seuillage dur et doux présentés dans la sous-section précédente 2.2.1 au cas général où \mathbf{D} est un dictionnaire redondant quelconque.

On trouve dans la littérature différentes versions d’algorithmes de seuillage itératifs. Les premiers travaux significatifs ont été réalisés par Kingsbury et Reeves. Dans [55], ils dérivent une méthode de seuillage itérative permettant d’approcher la solution du problème (\mathcal{P}_0^R) . Cependant, leur contribution ne fait réellement aucune connexion avec la fonction objectif (\mathcal{P}_0^R) . On trouve une version plus explicite et légèrement différente de leur résultat dans [8]. Blumensath et Davies montrent ainsi que le problème (\mathcal{P}_0^R) peut être résolu par l’équation de mise à jour

$$x_i^{(n+1)} = T_\lambda^0(x_i^{(n)} + \mathbf{d}_i^T(\mathbf{y} - \mathbf{D}\mathbf{x}^{(n)})), \quad (2.14)$$

où T_λ^0 est définie par l’équation (2.11).

Un résultat similaire peut être prouvé pour le problème (\mathcal{P}_1^R) . Dans [24], Daubechies *et al.* mettent ainsi en évidence qu’une solution peut être obtenue

en itérant l'expression

$$x_i^{(n+1)} = \mathcal{T}_\lambda^1(x_i^{(n)} + \mathbf{d}_i^T(\mathbf{y} - \mathbf{D}\mathbf{x}^{(n)})), \quad (2.15)$$

où \mathcal{T}_λ^1 est définie par l'équation (2.12). Dans la même idée, on peut également citer les travaux de Combettes et Pesquet [18].

Algorithmes de poursuite

Les algorithmes de poursuite, ou algorithmes gloutons, cherchent à résoudre les problèmes (\mathcal{P}_0^P) ou (\mathcal{P}_0^A) . Leur procédure est itérative : à chaque itération, un ou plusieurs (dans le cas des algorithmes *stagewise*, abordés en dernière partie de ce paragraphe) atomes sont ajoutés à la décomposition selon des considérations locales. La recherche se poursuit jusqu'à atteindre le critère d'arrêt, qui peut être une erreur d'approximation maximale ϵ , ou un nombre maximal d'atomes dans la décomposition L .

Les décisions étant prises localement à chaque itération, il n'y a aucune garantie d'obtenir un optimum global pour un dictionnaire \mathbf{D} quelconque. Cependant, ces algorithmes sont en général simples et rapides, ils sont donc très souvent utilisés.

Il existe de nombreux algorithmes de poursuite. Nous présentons les plus populaires :

- ◆ Matching Pursuit (MP), introduit dans la communauté du traitement du signal en 1993 par Mallat et Zhang [71],
- ◆ Orthogonal Matching Pursuit (OMP), évolution de MP proposée par Pati *et al.* dans [84].

Par la suite, des variantes et extensions de ces deux algorithmes ont été dérivées. Citons pour exemple les algorithmes Optimized Orthogonal Matching Pursuit (OOMP) [90], Complementary Matching Pursuit (CMP) [88] et Complementary Orthogonal Matching Pursuit (COMP) [89].

La plupart des algorithmes gloutons estiment successivement le support de la décomposition parcimonieuse et les valeurs des coefficients du vecteur parcimonieux. Le support est défini comme un vecteur $\mathbf{s} = [s_1, \dots, s_M]^T$ tel que $\forall i \in \{1, \dots, M\}$,

$$s_i = \begin{cases} 1 & \text{si } x_i \neq 0, \\ 0 & \text{sinon,} \end{cases} \quad (2.18)$$

où \mathbf{x} est le vecteur de décomposition parcimonieuse.

L'algorithme MP repose sur une sélection des atomes les plus corrélés avec le signal. Son processus général est donné par l'Algorithme 1.

Algorithme 1: Matching Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Mise à jour du support de la décomposition parcimonieuse (sélection de l'atome le plus corrélé avec le résidu)

$$\hat{s}_j^{(n)} = \begin{cases} 1 & \text{si } j = \operatorname{argmax}_i \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle^2, \\ \hat{s}_j^{(n-1)} & \text{sinon.} \end{cases} \quad (2.16)$$

2. Calcul du coefficient du vecteur \mathbf{x} correspondant

$$\hat{x}_j^{(n)} = \begin{cases} \hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle & \text{si } j = \operatorname{argmax}_i \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle^2, \\ \hat{x}_j^{(n-1)} & \text{sinon.} \end{cases} \quad (2.17)$$

3. Mise à jour du résidu : $\mathbf{r}^{(n)} = \mathbf{r}^{(n-1)} - \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle \mathbf{d}_j$.

Algorithme 2: Orthogonal Matching Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Mise à jour du support de la décomposition parcimonieuse (sélection de l'atome le plus corrélé avec le résidu)

$$\hat{s}_j^{(n)} = \begin{cases} 1 & \text{si } j = \operatorname{argmax}_i \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle^2, \\ \hat{s}_j^{(n-1)} & \text{sinon.} \end{cases} \quad (2.19)$$

2. Calcul des coefficients du vecteur \mathbf{x} correspondants

$$\hat{\mathbf{x}}_{\hat{s}^{(n)}} = \mathbf{D}_{\hat{s}^{(n)}}^+ \mathbf{y}, \quad (2.20)$$

où $\mathbf{D}_{\hat{s}^{(n)}}^+$ est la pseudo-inverse de $\mathbf{D}_{\hat{s}^{(n)}}$, matrice formée des colonnes \mathbf{d}_i telles que $\hat{s}_i^{(n)} \neq 0$.

3. Mise à jour du résidu : $\mathbf{r}^{(n)} = \mathbf{y} - \mathbf{D}_{\hat{s}^{(n)}} \hat{\mathbf{x}}_{\hat{s}^{(n)}}$.

Remarquons que rien n'empêche un atome d'être sélectionné plusieurs fois, de sorte qu'il faut parfois un nombre important d'itérations pour atteindre le critère d'arrêt. Cependant, l'algorithme est assuré de converger si le signal \mathbf{y} est contenu dans l'espace engendré par les atomes du dictionnaire \mathbf{D} (cf. [54]).

L'algorithme OMP réalise la même mise à jour du support que MP, mais calcule les valeurs des coefficient non nuls du vecteur parcimonieux d'une autre manière. Ainsi, au lieu de ne mettre à jour qu'un coefficient par itération (à partir de la projection du résidu sur l'atome considéré), OMP réestime tous les coefficients non nuls en projetant le signal \mathbf{y} sur l'espace engendré par tous les atomes sélectionnés. L'opération est réalisée par une orthogonalisation de Gram-Schmidt. Ainsi, un atome déjà choisi ne peut l'être à nouveau et à la N -ième itération, les N atomes sélectionnés et orthogonalisés forment une base

orthogonale de \mathbb{R}^N capable de représenter sans erreur le signal \mathbf{y} . OMP est donc assuré de converger en un nombre fini d'itérations, au plus égal à la taille du signal N . La description de l'algorithme OMP est donnée par l'Algorithme 2.

Plusieurs travaux se sont intéressés à la capacité de reconstruction des algorithmes MP et OMP. Tropp [111], ainsi que Gribonval et Vandergheynst [44] donnent des conditions pour que les algorithmes retrouvent la décomposition exacte d'un signal \mathbf{y} :

Théorème 7 *Supposons que le signal \mathbf{y} admet une décomposition $\mathbf{y} = \sum_{i=1}^L x_i \mathbf{d}_i$ dans un dictionnaire \mathbf{D} arbitraire. Pour $\mathcal{I} = \{i | x_i \neq 0\}$, on note $\Phi_{\mathcal{I}}$ l'opérateur tel que $\Phi_{\mathcal{I}} \mathbf{z} = \sum_{i \in \mathcal{I}} z_i \mathbf{d}_i$. Si*

$$\sup_{i \notin \mathcal{I}} \|\Phi_{\mathcal{I}}^+ \mathbf{d}_i\|_1 < 1, \quad (2.21)$$

où $\Phi_{\mathcal{I}}^+$ est la pseudo-inverse de $\Phi_{\mathcal{I}}$, alors MP et OMP retrouvent la décomposition, i.e., à chaque itération n , un atome "correct" \mathbf{d}_{k_n} est choisi ($k_n \in \mathcal{I}$).

Comme nous l'avons vu, les algorithmes MP et OMP ne permettent de sélectionner qu'un atome à chaque itération. Les algorithmes de poursuite *par étape* (stagewise en anglais) remédient à cette limitation : à chaque itération, plusieurs atomes peuvent être choisis, accélérant ainsi le processus global d'optimisation. Parmi les plus connus, on trouve l'algorithme Stagewise OMP (StOMP) [30], l'algorithme Subspace Pursuit (SP) [22], similaire à l'algorithme Compressive Sampling Matching Pursuit (CoSaMP) [80] ou encore l'algorithme d'analyse en composantes morphologiques (MCA pour Morphological Component Analysis en anglais) [9]. Nous détaillons ici les algorithmes StOMP et CoSaMP/SP.

L'algorithme StOMP peut être vu comme une variante de l'algorithme OMP décrit dans le paragraphe précédent. Le calcul des coefficients du vecteur parcimonieux est identique, mais le choix des atomes ajoutés au support de la décomposition parcimonieuse à chaque itération n est réalisé par un seuillage de paramètre $T^{(n)}$ sur le produit au carré $\langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle^2$. Donoho *et al.* proposent dans [30] deux approches différentes pour fixer la valeur du paramètre $T^{(n)}$ à chaque itération. L'Algorithme 3 présente les principales opérations de StOMP.

L'algorithme CoSaMP/SP résoud exclusivement le problème (\mathcal{P}_0^A) , mais offre un degré de liberté supplémentaire pour la résolution : la désélection d'atomes. Pour cela, CoSaMP/SP s'appuie sur la connaissance du nombre de coefficients non nuls autorisés, L . Son principe est décrit dans l'Algorithme 4.

Algorithme 3: Stagewise Orthogonal Matching Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Mise à jour du support de la décomposition parcimonieuse

$$\hat{s}_j^{(n)} = \begin{cases} 1 & \text{si } \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle^2 > T^{(n)}, \\ \hat{s}_j^{(n-1)} & \text{sinon.} \end{cases} \quad (2.22)$$

2. Calcul des coefficients du vecteur \mathbf{x} correspondants

$$\hat{\mathbf{x}}_{\hat{s}^{(n)}} = \mathbf{D}_{\hat{s}^{(n)}}^+ \mathbf{y}, \quad (2.23)$$

où $\mathbf{D}_{\hat{s}^{(n)}}^+$ est la pseudo-inverse de $\mathbf{D}_{\hat{s}^{(n)}}$, matrice formée des colonnes \mathbf{d}_i telles que $\hat{s}_i^{(n)} \neq 0$.

3. Mise à jour du résidu : $\mathbf{r}^{(n)} = \mathbf{y} - \mathbf{D}_{\hat{s}^{(n)}} \hat{\mathbf{x}}_{\hat{s}^{(n)}}$.

Les deux algorithmes CoSaMP et SP se distinguent par le choix du paramètre P . Dans SP, il est fixé à L , le nombre de coefficients non nuls autorisé. Dans CoSaMP, il est égal à $2L$.

Algorithmes d'optimisation convexe

Les algorithmes d'optimisation convexe s'intéressent au problème d'optimisation (\mathcal{P}_1^R) . On trouve parmi eux les algorithmes basés sur une programmation quadratique comme les algorithmes Basis Pursuit Denoising (BPD) [16] et Global Matched Filter (GMF) [40]. Nous ne les détaillons pas ici, mais quelques résultats importants méritent d'être mentionnés au regard des algorithmes précédents.

De nombreuses simulations numériques tendent à montrer ([16]) que si le signal \mathbf{y} a une décomposition très parcimonieuse dans un dictionnaire \mathbf{D} bien structuré, la décomposition parcimonieuse est parfaitement retrouvée par les algorithmes BPD et GMF. Cette observation a donné lieu à une série de résultats théoriques sur des dictionnaires différents (cf. par exemple [43], [111]). Un des résultats les plus généraux est celui obtenu par Fuchs dans [39].

Théorème 8 *Supposons que le signal \mathbf{y} admet une décomposition $\mathbf{y} = \sum_{i=1}^L x_i \mathbf{d}_i$ dans un dictionnaire \mathbf{D} arbitraire. Pour $\mathcal{I} = \{i | x_i \neq 0\}$, on note $\Phi_{\mathcal{I}}$ l'opérateur tel que $\Phi_{\mathcal{I}} \mathbf{z} = \sum_{i \in \mathcal{I}} z_i \mathbf{d}_i$. Si λ est suffisamment petit et*

$$|\langle (\Phi_{\mathcal{I}}^+)^* \text{sign}(\mathbf{x}), \mathbf{d}_i \rangle| < 1, \quad \forall i \notin \mathcal{I}, \quad (2.28)$$

où $\Phi_{\mathcal{I}}^+$ est la pseudo-inverse de $\Phi_{\mathcal{I}}$ et $(\Phi_{\mathcal{I}}^+)^*$ est l'adjoint de $\Phi_{\mathcal{I}}^+$, alors la résolution du problème (\mathcal{P}_1^R) conduit à la "bonne" décomposition : chaque coefficient non nul de \mathbf{x}^* , solution du problème (\mathcal{P}_1^R) , correspond à un indice $i \in \mathcal{I}$.

Les algorithmes d'optimisation convexe présentent en général de bonnes

Algorithme 4: Subspace Pursuit/Compressive Sampling Matching Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$ et $\hat{\mathbf{s}}^{(0)} = \mathbf{y}$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Sélection des P atomes les plus corrélés avec le résidu

Soit $\mathcal{I} = \{i \in \{1, \dots, M\} | \hat{s}_i^{(n-1)} = 1\}$.

$$\hat{\mathbf{s}}^{(n)} = \underset{\mathbf{s}}{\operatorname{argmax}} \sum_i s_i \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle^2 \text{ soumis à } \|\mathbf{s}\|_0 = P \text{ et } \forall i \in \mathcal{I}, s_i = 1, \quad (2.24)$$

2. Calcul des coefficients du vecteur \mathbf{x} correspondants

$$\tilde{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}} = \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^+ \mathbf{y}, \quad (2.25)$$

où $\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^+$ est la pseudo-inverse de $\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}$, matrice formée des colonnes \mathbf{d}_i telles que $\hat{s}_i^{(n)} \neq 0$.

3. Sélection des L atomes correspondant aux L coefficients de $\tilde{\mathbf{x}}$ les plus grands

$$\hat{\mathbf{s}}^{(n)} = \underset{\mathbf{s}}{\operatorname{argmax}} \sum_i s_i |\tilde{x}_i^{(n)}| \text{ soumis à } \|\mathbf{s}\|_0 = L, \quad (2.26)$$

4. Mise à jour du vecteur \mathbf{x}

$$\hat{\mathbf{x}}^{(n)} = \tilde{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}, \quad (2.27)$$

5. Mise à jour du résidu : $\mathbf{r}^{(n)} = \mathbf{y} - \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} \hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}$.

performances en terme de qualité d'approximation *vs* parcimonie de la décomposition, au regard des algorithmes gloutons. Mais c'est au prix d'une complexité plus élevée. Ainsi, tandis que MP et OMP admettent respectivement une complexité $\mathcal{O}(M)$ et $\mathcal{O}(L^3 + M)$ par itération, GMF nécessite près de N^3 opérations.

Algorithmes Bayésiens

Plus récemment, de nouvelles approches se plaçant dans un cadre Bayésien ont été proposées. Ces approches supposent en général le modèle suivant :

$$\mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{n}, \quad (2.29)$$

où \mathbf{n} est un bruit blanc gaussien de variance σ_n^2 .

On trouve dans la littérature différentes distributions de probabilité a priori sur le vecteur aléatoire \mathbb{X} à réalisations $\mathbf{x} \in \mathbb{R}^M$ (dans la suite, nous assimilerons variable aléatoire et réalisation lorsque le contexte est univoque). La plus intuitive est la distribution Laplacienne, qui permet une interprétation probabiliste de la mesure en norme ℓ_1 de la parcimonie ([6]). Pour chaque composante x_i du vecteur \mathbf{x} ,

$$p(x_i) \propto \exp(-\lambda|x_i|), \quad (2.30)$$

où \propto signifie "proportionnel à".

Etant donné le signal \mathbf{y} et le dictionnaire \mathbf{D} , on montre que le problème (\mathcal{P}_1^R) correspond à un problème d'estimation au Maximum A Posteriori (MAP) sur \mathbf{x} utilisant la distribution a priori (2.30)

$$\begin{aligned} \mathbf{x}_{MAP}^* &= \operatorname{argmax}_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}, \mathbf{D}), \\ &= \operatorname{argmax}_{\mathbf{x}} \log p(\mathbf{y}, \mathbf{x}, \mathbf{D}), \\ &= \operatorname{argmax}_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}, \mathbf{D})p(\mathbf{x}), \\ &= \operatorname{argmin}_{\mathbf{x}} \frac{1}{2\sigma_n^2} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1. \end{aligned} \quad (2.31)$$

Cette interprétation Bayésienne du problème (\mathcal{P}_1^R) comme un problème d'estimation MAP ouvre des perspectives d'analyse intéressantes. Ainsi, comme nous l'avons mentionné ci-dessus, d'autres distributions a priori sur \mathbf{x} peuvent être envisagées, possiblement plus complexes. On peut également considérer d'autres approches d'estimation, en particulier, la minimisation de l'erreur quadratique moyenne (MMSE pour Minimum Mean Square Error en anglais) :

$$\mathbf{x}_{MMSE}^* = \int_{\mathbf{x}} \mathbf{x} p(\mathbf{x}|\mathbf{y}, \mathbf{D}) d\mathbf{x}. \quad (2.32)$$

Cette estimation nécessite en général l'évaluation de la probabilité a posteriori $p(\mathbf{x}|\mathbf{y}, \mathbf{D})$. Ce n'est pas le cas de l'estimation MAP qui peut s'évaluer à partir de la distribution de la probabilité jointe $p(\mathbf{y}, \mathbf{x}, \mathbf{D})$ (cf. équation (2.31)), en général plus simple à calculer. Enfin, le cadre Bayésien permet le recours à des méthodes probabilistes pour résoudre les problèmes d'estimation MAP (2.31) et MMSE (2.32).

Nous présentons ici quelques méthodes Bayésiennes proposées dans la littérature.

Dans [121], Zayyani *et al.* considèrent une approche MAP basée sur un modèle Bernoulli-Gaussien. Les auteurs introduisent une factorisation des x_i telle que $x_i = q_i s_i$, $\forall i \in \{1, \dots, M\}$, et supposent que les variables q_i suivent des lois Gaussiennes de variance σ_q^2 et de moyenne nulle et les composantes s_i sont des variables de Bernoulli. La distribution de probabilités de \mathbf{y} , vecteur

d'observation, dépend alors du modèle a priori tel que $\forall i \in \{1, \dots, M\}$

$$\begin{aligned}
p(x_i) &= \int_{q_i} \sum_{s_i} p(x_i, q_i, s_i) dq_i, & (2.33) \\
&= \int_{q_i} \sum_{s_i} p(x_i | q_i, s_i) p(q_i) p(s_i) dq_i, \\
&= \int_{q_i} \delta_0(x_i) p(q_i) p(s_i = 0) dq_i + \int_{q_i} \delta_{q_i}(x_i) p(q_i) p(s_i = 1) dq_i, \\
&= p(s_i = 0) \delta_0(x_i) + p(s_i = 1) p_{Q_i}(x_i), \\
&= p(s_i = 0) \delta_0(x_i) + p(s_i = 1) \mathcal{N}(0, \sigma_q^2).
\end{aligned}$$

La notation $p_{Q_i}(x_i)$ est utilisée ici pour différencier la variable aléatoire Q_i de sa réalisation x_i . Un algorithme itératif est ensuite proposé estimant successivement les variables $\mathbf{q} = [q_1, \dots, q_M]^T$ et $\mathbf{s} = [s_1, \dots, s_M]^T$ au sens du maximum a posteriori. L'estimation de \mathbf{q} pour \mathbf{s} fixé est relativement aisée du fait de la Gaussianité de \mathbf{q} . En revanche, l'estimation de \mathbf{s} pour \mathbf{q} fixé est plus subtile. Les auteurs ont recours à une méthode du gradient. Pour cela, ils "convertissent" chaque variable aléatoire discrète s_i en une variable continue via un mélange de Gaussiennes centrées en 0 et 1 de faibles variances σ_0^2 et σ_1^2 :

$$p(s_i) = p \mathcal{N}(0, \sigma_0^2) + (1 - p) \mathcal{N}(0, \sigma_1^2), \quad (2.34)$$

où p est un paramètre compris entre 0 et 1 pondérant le mélange. En terme de complexité, leur approche est beaucoup plus coûteuse que les algorithmes de poursuite ou d'optimisation convexe, cependant elle atteint des performances relativement bonnes en termes de PSNR *vs* parcimonie. Cette même distribution a priori (2.33) est utilisée dans d'autres articles signés des mêmes auteurs ([119, 120]). On la retrouve également dans un rapport technique de Soussen *et al.* [103], s'appuyant sur l'algorithme Single Most Likely Replacement (SMLR). Cet algorithme fut introduit en 1982 par Kormylo et Mendel [56, 77] pour la déconvolution d'un signal issu d'un processus Bernoulli-Gaussien. Son approche repose sur la maximisation de fonctions de vraisemblance de la forme $S(\mathbf{s}|\mathbf{y})$ (estimation MAP marginalisée) ou $S(\mathbf{q}, \mathbf{s}|\mathbf{y})$ (estimation MAP jointe). Nous reviendrons sur ce modèle Bernoulli-Gaussien en particulier dans le chapitre 5 pour le développement de nouveaux algorithmes de recherche.

Un modèle Bernoulli-Gaussien différent est envisagé dans une estimation MMSE par Baron *et al.* ([4]). Chaque composante x_i est décrite selon une distribution Gaussienne dépendant d'une variable de Bernoulli s_i , résultant en un mélange de Gaussiennes :

$$p(x_i) = p(s_i = 0) \mathcal{N}(0, \sigma^2(s_i = 0)) + p(s_i = 1) \mathcal{N}(0, \sigma^2(s_i = 1)), \quad (2.35)$$

Les auteurs utilisent un algorithme de propagation de croyance pour estimer les distributions marginales a posteriori $p(x_i|\mathbf{y}, \mathbf{D})$. Cette technique repose sur des graphes factoriels qui permettent un calcul rapide des marginales en exploitant la factorisation de la distribution de probabilités jointe (ici, $p(\mathbf{x}|\mathbf{y}, \mathbf{D})$). Dans [4], les auteurs montrent que, particularisé au modèle (2.35), l'algorithme qui en résulte a une complexité $\mathcal{O}(M \log^2(M))$, ce qui reste compétitif au regard des algorithmes de poursuite ou d'optimisation convexe.

He et Carin considèrent dans [49] le cas limite $\sigma^2(s_i = 0) \rightarrow 0 \quad \forall i \in \{1, \dots, M\}$. Le modèle sur \mathbf{x} (2.35) évolue alors en un mélange de deux distributions différentes :

$$p(x_i) = p(s_i = 0)\delta_0(x_i) + p(s_i = 1)\mathcal{N}(0, \sigma^2(s_i = 1)), \quad (2.36)$$

l'une de Dirac (δ_0), l'autre Gaussienne ($\mathcal{N}(0, \sigma^2(s_i = 1))$) de moyenne nulle et de variance $\sigma^2(s_i = 1)$ (remarquons que nous retombons alors sur une distribution de probabilités similaire à (2.33)). He et Carin supposent en outre que les variables $\sigma^2(s_i = 1)$ et p_i sont aléatoires et suivent respectivement une loi Inverse-Gamma et une loi Beta. Une méthode MCMC (pour Monte Carlo Markov Chain en anglais, cf. [11]) est ensuite utilisée pour estimer la distribution a posteriori $p(\mathbf{x}|\mathbf{y}, \mathbf{D})$.

Enfin, on trouve dans la littérature le modèle suivant : pour chaque composante x_i ,

$$p(x_i|\sigma_i^2) = \mathcal{N}(0, \sigma_i^2), \quad (2.37)$$

où σ_i^2 est inconnu, $\forall i \in \{1, \dots, M\}$. C'est le modèle considéré dans l'algorithme Sparse Bayesian Learning (SBL) introduit par Tipping dans [108]. Supposant le vecteur de variances $\boldsymbol{\sigma}^2 = [\sigma_1^2, \dots, \sigma_M^2]^T$ connu, la distribution a posteriori de \mathbf{x} peut être exprimée analytiquement comme une distribution Gaussienne de moyenne $\boldsymbol{\mu}$ et de variance $\boldsymbol{\Sigma}$:

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}, \boldsymbol{\sigma}^2) &= \frac{p(\mathbf{x}, \mathbf{y}|\boldsymbol{\sigma}^2)}{\int p(\mathbf{x}, \mathbf{y}|\boldsymbol{\sigma}^2) d\mathbf{x}}, \\ &= \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \end{aligned} \quad (2.38)$$

avec

$$\boldsymbol{\mu} = \sigma_n^{-2} \boldsymbol{\Sigma} \mathbf{D}^T \mathbf{y}, \quad (2.39)$$

$$\boldsymbol{\Sigma} = (\sigma_n^{-2} \mathbf{D}^T \mathbf{D} + \mathbf{A})^{-1}, \quad (2.40)$$

où $\mathbf{A} = \text{diag}(\sigma_1^{-2}, \dots, \sigma_M^{-2})$. Il s'agit alors d'estimer les variances σ_i^2 . Pour ce faire, SBL réalise une estimation au Maximum de Vraisemblance (ML pour

Maximum of Likelihood) de type II (ou maximisation d'évidence), *i.e.*, une maximisation de la *vraisemblance marginale* $p(\mathbf{y}|\sigma^2)$ ([5][66]). En raison de l'inversion matricielle (2.40), la complexité de l'algorithme est assez élevée, $\mathcal{O}(N^3)$ (cf. lemme d'inversion matricielle [115]). Ce défaut a motivé la conception d'un SBL rapide ([109]), de complexité inférieure, $\mathcal{O}(MN^2)$. Notons enfin que SBL a été dérivé à l'origine pour résoudre des problèmes de régression. Lorsque \mathbf{D} est carré et constitué de fonctions noyaux définies positives, on obtient le Relevance Vector Machine (RVM), concurrent Bayésien du Support Vector Machine (SVM) présentant de nombreux avantages [38]. Nous renvoyons le lecteur à la thèse de Wipf [113] pour une analyse détaillée de SBL et des algorithmes Bayésiens basés sur des estimations MAP.

2.3 Dictionnaires

Comme nous l'avons vu, la "qualité" de la décomposition parcimonieuse, en terme de parcimonie et d'approximation, dépend de l'algorithme de décomposition parcimonieuse utilisé. Elle est également liée au dictionnaire dans lequel est réalisée la décomposition. Plus le dictionnaire est adapté aux caractéristiques du signal et favorise la parcimonie de la décomposition, "meilleure" est la décomposition parcimonieuse. La définition de dictionnaires constitue donc un enjeu important et fait l'objet d'un grand nombre de contributions.

2.3.1 Introduction à l'apprentissage de dictionnaires

Deux types de dictionnaires peuvent être différenciés :

- ◆ les dictionnaires constitués d'un ensemble prédéfini de fonctions,
- ◆ les dictionnaires appris sur un ensemble de signaux.

Les dictionnaires prédéfinis présentent l'avantage d'être simples d'utilisation. Mais leur "succès" dépend de leur bonne adaptation à une description parcimonieuse des signaux considérés. Ainsi, un dictionnaire bien adapté à des signaux images texturés ne favorisera vraisemblablement pas une bonne décomposition parcimonieuse de signaux images homogènes ou contenant des contours.

L'apprentissage permet de définir un dictionnaire propre à un type de signaux donnés (constituant l'ensemble d'entraînement du dictionnaire) sous

des critères que l'on peut contrôler. Particularisé aux décompositions parcimonieuses, le problème peut se formaliser de la façon suivante. Etant donné un ensemble d'entraînement $\{\mathbf{y}_j\}_{j=1}^K$, on recherche le dictionnaire \mathbf{D}^* qui conduit au meilleur compromis distorsion-parcimonie :

$$\mathbf{D}^* = \arg \min_{\mathbf{D}} \left\{ \sum_j \min_{\mathbf{x}_j} \|\mathbf{y}_j - \mathbf{D}\mathbf{x}_j\|_2^2 + \lambda \|\mathbf{x}_j\|_0 \right\},$$

où \mathbf{x}_j est le vecteur de décomposition parcimonieuse du signal \mathbf{y}_j dans le dictionnaire \mathbf{D} .

Les approches d'apprentissage de dictionnaires adaptés aux décompositions parcimonieuses proposées jusqu'ici présentent toutes un processus itératif en deux étapes :

- ◆ pour un dictionnaire donné, recherche de la décomposition parcimonieuse pour chaque signal de l'ensemble d'entraînement,
- ◆ pour des décompositions parcimonieuses données, recherche du dictionnaire.

Leurs différences reposent sur les méthodes utilisées pour estimer successivement les vecteurs parcimonieux et le dictionnaire.

2.3.2 Algorithmes d'apprentissage

On trouve dans la littérature de nombreuses méthodes d'apprentissage de dictionnaires adaptés aux représentations parcimonieuses. Nous en présentons ici quelques unes, parmi les plus populaires.

Approches Bayésiennes

Les approches Bayésiennes traitent le problème d'optimisation de dictionnaire dans un cadre probabiliste. Chaque signal d'entraînement \mathbf{y}_j est vu comme une combinaison bruitée d'atomes choisis dans un dictionnaires \mathbf{D} ,

$$\mathbf{y}_j = \mathbf{D}\mathbf{x}_j + \mathbf{n}, \quad (2.41)$$

où \mathbf{n} est un bruit blanc gaussien.

Deux approches peuvent alors être envisagées.

La première considère le problème d'estimation au Maximum de Vraisemblance (ML pour Maximum of Likelihood en anglais) suivant :

$$\mathbf{D}^* = \arg \max_{\mathbf{D}} \sum_{j=1}^K \log p(\mathbf{y}_j | \mathbf{D}), \quad (2.42)$$

où

$$p(\mathbf{y}_j|\mathbf{D}) = \int_{\mathbb{R}^M} p(\mathbf{y}_j, \mathbf{x}_j|\mathbf{D})d\mathbf{x}_j = \int_{\mathbb{R}^M} p(\mathbf{y}_j|\mathbf{x}_j, \mathbf{D})p(\mathbf{x}_j)d\mathbf{x}_j. \quad (2.43)$$

Plusieurs distributions de probabilité différentes sont proposées dans la littérature. On trouve ainsi des distributions de Cauchy et de Laplace ([81] et [63]), censées favoriser la parcimonie (la distribution de Laplace correspond à la mesure en norme ℓ_1).

La marginalisation (2.43) est très complexe à résoudre. Olshausen et Field proposent dans [81] de la remplacer par une maximisation :

$$\mathbf{D}^* = \operatorname{argmax}_{\mathbf{D}} \sum_{j=1}^K \max_{\mathbf{x}_j} \log p(\mathbf{y}_j, \mathbf{x}_j|\mathbf{D}). \quad (2.44)$$

Une méthode du gradient est ensuite utilisée pour estimer les vecteurs parcimonieux \mathbf{x}_j d'une part et le dictionnaire \mathbf{D} d'autre part. Cette solution tendant à augmenter les valeurs des atomes du dictionnaire, les auteurs proposent de contraindre la norme ℓ_2 des atomes.

Confrontés à la même marginale (2.43), Lewicki et Sejnowski ([63]) choisissent d'approximer la distribution de probabilité $p(\mathbf{y}_j|\mathbf{D})$ plutôt que de maximiser sur les variables \mathbf{x}_j . Ils ont recours pour cela à une approximation de Laplace qui approche une distribution de probabilité complexe par une Gaussienne. Cette technique permet de résoudre analytiquement l'intégration (2.43) et ainsi de prendre en compte les incertitudes sur la distribution de probabilité des \mathbf{x}_j . Elle présente de plus l'avantage d'éviter la définition de contraintes sur les normes des atomes du dictionnaire. Une simple méthode du gradient peut alors être utilisée pour l'estimation du dictionnaire, sans autre considération.

La deuxième approche proposée dans la littérature considère le problème d'estimation au Maximum A Posteriori (MAP) suivant :

$$(\mathbf{D}^*, \{\mathbf{x}_j^*\}) = \operatorname{argmax}_{\mathbf{D}} \sum_{j=1}^K \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}). \quad (2.45)$$

C'est l'approche adoptée par Murray et Kreutz-Delgado dans [79] et Kreutz-Delgado et Rao dans [58]. Une méthode du gradient est utilisée pour estimer le dictionnaire. Confrontés au même problème d'augmentation des valeurs des atomes que Olshausen et Field, les auteurs choisissent une distribution a priori qui contraint le dictionnaire à avoir une norme de Frobenius unitaire. Mais l'apport fondamental de leur contribution est l'utilisation de l'algorithme de recherche FOCUSS pour réaliser l'étape d'estimation des

vecteurs parcimonieux. Ce choix améliore les performances de l'algorithme d'apprentissage au regard des autres algorithmes Bayésiens précédents.

Méthode des directions optimales (MOD)

La méthode des directions optimales (MOD pour Method of Optimal Directions en anglais), introduite par Egan *et al.* dans [35], s'inspire explicitement de l'algorithme de Lloyd-Max utilisé pour l'apprentissage de dictionnaires de quantification ([41]). L'étape d'estimation des vecteurs parcimonieux est réalisée, comme dans l'algorithme proposé par Kreutz-Delgado et Rao, par un algorithme de recherche (cette fois-ci cependant, OMP est préféré à une norme ℓ_1). L'étape d'estimation du dictionnaire constitue la principale contribution de la méthode MOD et réside dans la minimisation de l'erreur d'approximation totale $\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2$, où $\mathbf{Y} = [\mathbf{y}_1^T, \dots, \mathbf{y}_K^T]^T$ et $\mathbf{X} = [\mathbf{x}_1^T, \dots, \mathbf{x}_K^T]^T$. Cette approche conduit à de bonnes performances au regard des algorithmes proposés jusqu'ici. Notons que dans cette méthode comme dans celle de Olshausen et Field, une normalisation des atomes du dictionnaire est nécessaire.

Algorithme K-SVD

Proposée par Aharon *et al.* dans [1], cette méthode s'appuie sur une décomposition en valeurs singulières (SVD pour Singular Value Decomposition en anglais) pour estimer le dictionnaire \mathbf{D} . Après une estimation de vecteurs parcimonieux par un algorithme de recherche de type OMP, les atomes du dictionnaire sont mis à jour successivement. L'algorithme procède de la façon suivante. La contribution de l'atome considéré, noté \mathbf{d}_k , dans la description des signaux est évaluée par une matrice d'erreur de représentation correspondant à la différence entre les signaux \mathbf{y}_j "utilisant" l'atome \mathbf{d}_k et leurs approximations parcimonieuses "tronquées" (*i.e.*, dans lesquelles on a retiré l'atome \mathbf{d}_k). Les vecteurs obtenus forment une matrice dont on calcule ensuite la SVD. L'atome \mathbf{d}_k peut alors être estimé par le premier vecteur propre ainsi calculé. L'algorithme K-SVD présente de très bonnes performances en considération des autres algorithmes de la littérature.

Apprentissage de dictionnaires structurés : l'union de bases

Dans le cadre du codage par transformation, l'utilisation de dictionnaires redondants peut avoir des répercussions importantes sur le coût de codage des indices des atomes choisis pour la décomposition parcimonieuse (nous y reviendrons dans le chapitre 3). Une façon de réduire ce coût est d'introduire de la structure dans le dictionnaire, ce peut être simplement

réalisé en considérant un ensemble de dictionnaires, voire de bases. Plusieurs contributions se sont intéressées à l'apprentissage d'unions de bases. Nous en présentons ici deux qui proposent des approches différentes quoique basées sur les mêmes techniques : l'une optimise une union de bases dans son ensemble, tandis que l'autre considère chaque base séparément.

Une première méthode a été introduite par Lesage *et al.* dans [62]. Basé sur une SVD comme l'algorithme de Aharon *et al.*, l'algorithme en propose cependant une autre utilisation. Il procède ainsi en estimant les atomes de chaque base en même temps et non successivement comme dans l'algorithme K-SVD. Les vecteurs parcimonieux sont également estimés de façon différente, via la méthode BCR (pour Block Coordinate Relaxation) décrite dans [94]. Cette méthode permet d'étendre le seuillage doux présenté dans la sous-section 2.2.1 à l'union de bases orthonormées.

Un autre algorithme a été proposé par Sezer *et al.* dans [96]. Au lieu de considérer le dictionnaire dans son ensemble, les auteurs partent ici de l'hypothèse que chaque signal admet une décomposition parcimonieuse dans une unique base. L'algorithme présente donc une étape supplémentaire de classification, où chaque signal est classé en fonction de la base qui minimise l'erreur d'approximation, résultant en plusieurs "sous-ensembles d'entraînement". L'algorithme poursuit ensuite de façon classique en estimant successivement les vecteurs parcimonieux et les bases sur les sous-ensembles correspondants. Les vecteurs parcimonieux sont calculés par un seuillage dur (cf. (2.11)). Les bases sont mises en jour par une méthode basée sur une SVD similaire à celle utilisée par Lesage *et al.* Cet algorithme fera l'objet d'une étude plus approfondie dans la section 3.4 du chapitre 3.

2.4 Annexe : définition et propriétés des normes

On appelle *norme* sur un espace vectoriel E de $\mathbb{K} = \mathbb{R}$ ou \mathbb{C} toute application \mathcal{G} de E vers \mathbb{R}_+ vérifiant les propriétés suivantes :

- ◆ Positivité :

$$\forall \mathbf{x} \in E \setminus \{0\}, \mathcal{G}(\mathbf{x}) > 0, \quad (2.46)$$

- ◆ Homogénéité :

$$\forall \mathbf{x} \in E, \forall \lambda \in \mathbb{K}, \mathcal{G}(\lambda \mathbf{x}) = |\lambda| \mathcal{G}(\mathbf{x}), \quad (2.47)$$

♦ Inégalité triangulaire :

$$\forall (\mathbf{x}_1, \mathbf{x}_2) \in (E)^2, \mathcal{G}(\mathbf{x}_1 + \mathbf{x}_2) \leq \mathcal{G}(\mathbf{x}_1) + \mathcal{G}(\mathbf{x}_2). \quad (2.48)$$

Transformations adaptatives

3

Dans ce chapitre, nous étudions un schéma de compression par transformation reposant sur des dictionnaires structurés.

Les deux premières sections motivent et explicitent l'approche proposée, tandis que dans les suivantes, nous en exposons des implémentations pratiques.

Ainsi, plusieurs contributions de cette thèse sont ici exposées : extension des DCT directionnelles proposées par Zeng et Fu [122] à des supports rectangulaires, étude et amélioration d'un algorithme d'apprentissage favorisant la parcimonie des décompositions, introduit par Sezer *et al.* [96], et enfin élaboration d'un nouvel algorithme d'apprentissage de bases. Chacune de ces contributions a fait l'objet d'un article de conférence [C][F][D].

3.1 Codage par transformation et parcimonie

Cette section formalise et justifie le schéma de compression proposé. Nous étudions ainsi les expressions des débit et distorsion dans la compression par transformation "classique" avant d'envisager l'utilisation d'un dictionnaire redondant en terme de coût de codage.

3.1.1 Considérations débit-distorsion

Soit $\mathbf{y} \in \mathbb{R}^N$ une image vectorisée que l'on cherche à compresser. Par la transformation, \mathbf{y} est décomposé dans une base orthonormée $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_N]$ de la façon suivante :

$$\mathbf{y} = \sum_{k=1}^N x_k \mathbf{d}_k. \quad (3.1)$$

Les coefficients de transformation x_k sont ensuite quantifiés. Considérant un quantificateur scalaire q , le signal \mathbf{y} est alors approximé par $\hat{\mathbf{y}}$ tel que

$$\hat{\mathbf{y}} = \sum_{k=1}^N q(x_k) \mathbf{d}_k, \quad (3.2)$$

où $q(x_k)$ est possiblement nul pour $k \in \{1, \dots, N\}$. Rappelons que pour $\mathbf{x} = [x_1, \dots, x_N]^T$ et q quantificateur scalaire, on utilise les notations $q(\mathbf{x}) \triangleq [q(x_1), \dots, q(x_N)]^T$.

Limite de validité de l'hypothèse de haute résolution

Considérant la décomposition d'une image sur une base d'ondelettes, on peut montrer de façon empirique que les histogrammes des coefficients d'ondelettes sont fortement "piqués" et peuvent être modélisés dans une première approximation par des distributions Laplaciennes. Cette observation est illustrée dans la figure 3.1 par l'histogramme des coefficients de décomposition de l'image "Lena" sur une base d'ondelettes orthogonales, emprunté à Mallat [68].

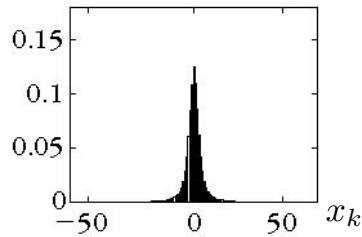


Figure 3.1 Histogramme normalisé des coefficients d'ondelettes pour la décomposition de l'image "Lena"

Cette caractéristique est montrée également pour des bases DCT appliquées sur des blocs d'image [69]. Elle est généralement admise.

Dans ces conditions, la densité de probabilité de la variable aléatoire X correspondant aux réalisations x_k est très mal approximée par une constante dans la zone proche de 0. L'hypothèse de quantificateur de haute résolution n'est pas vérifiée à bas débits (*i.e.*, pour des grands pas de quantification), on ne peut donc pas utiliser la formulation (1.32) (sous-section 1.2.3 chapitre 1) pour exprimer la distorsion en fonction du débit. Le paragraphe suivant étudie et établit le lien entre distorsion et débit dans ce cas.

Etude débit-distorsion

Le débit binaire associé à la compression dépend de la base \mathbf{D} et du

quantificateur q utilisés. On le note $R(\mathbf{D}, q)$. $R(\mathbf{D}, q)$ est composé de deux termes :

- ♦ $R_v(\mathbf{D}, q)$, le nombre de bits nécessaires à la transmission des valeurs quantifiées non nulles,
- ♦ $R_i(\mathbf{D}, q)$, le coût de codage des indices des coefficients quantifiés non nuls.

Mallat montre dans [68] que le nombre de bits $R_v(\mathbf{D}, q)$ obtenu par un codage entropique à longueur variable est proportionnel au nombre de coefficients non nuls de $q(\mathbf{x})$, noté $\|q(\mathbf{x})\|_0 = L$,

$$R_v(\mathbf{D}, q) \propto L. \quad (3.3)$$

Le débit binaire $R_i(\mathbf{D}, q)$ est constitué du coût de codage du nombre de coefficients non nuls et du nombre de bits nécessaires pour spécifier le choix de la combinaison de $\|q(\mathbf{x})\|_0 = L$ vecteurs parmi N . Pour un codage à longueur fixe, on obtient

$$R_i(\mathbf{D}, q) = \log_2 N + \log_2 C_N^L \sim L(1 + \log_2 \frac{N}{L}). \quad (3.4)$$

Le coût de codage total est alors tel que

$$R(\mathbf{D}, q) = R_v(\mathbf{D}, q) + R_i(\mathbf{D}, q) \sim L(2 + \log_2 \frac{N}{L}). \quad (3.5)$$

Si $L \ll N$, le coût de codage des indices des coefficients non nuls domine le débit binaire total, $R(\mathbf{D}, q) \sim L \log_2 \frac{N}{L}$.

Par ailleurs, la distorsion obtenue entre le signal \mathbf{y} et son approximation $\hat{\mathbf{y}}$ est explicitée par

$$\begin{aligned} D(\mathbf{D}, q) &\triangleq \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2, \\ &= \|\mathbf{x} - q(\mathbf{x})\|_2^2, \\ &= \sum_{k=1}^N |x_k - q(x_k)|^2, \\ &= \sum_{k \notin \mathcal{I}} |x_k|^2 + \sum_{k \in \mathcal{I}} |x_k - q(x_k)|^2, \end{aligned} \quad (3.6)$$

où \mathcal{I} est l'ensemble des indices des coefficients non nuls de $q(\mathbf{x})$. Restreignant notre étude aux quantificateurs scalaires uniformes (avec éventuellement une zone morte), l'erreur de quantification sur les coefficients x_k non quantifiés à 0 est comprise entre 0 et $\frac{\Delta}{2}$, où Δ est le pas de quantification. Ainsi,

$$\sum_{k \notin \mathcal{I}} |x_k|^2 \leq D(\mathbf{D}, q) \leq \sum_{k \notin \mathcal{I}} |x_k|^2 + L \frac{\Delta^2}{4}. \quad (3.7)$$

Le terme $\sum_{k \notin \mathcal{I}} |x_k|^2$ constitue l'erreur d'approximation non linéaire, résultant de la parcimonie de $q(\mathbf{x})$.

Pour introduire le théorème 9 explicitant la distorsion en fonction du débit, nous adoptons les notations suivantes. Les coefficients x_k sont classés dans l'ordre décroissant de leur valeur absolue ; on les note

$$|x_{i_k}| \geq |x_{i_{k+1}}| \quad \forall k \in \{1, \dots, N-1\}. \quad (3.8)$$

On écrit alors $|x_{i_k}| \sim C k^{-s}$ s'il existe deux constantes $A, B > 0$ indépendantes de C, k et N telles que

$$A C k^{-s} \leq |x_{i_k}| \leq B C k^{-s}. \quad (3.9)$$

Dans [69], Mallat et Falzon prouvent le résultat ci-dessous.

Théorème 9 *Soit q un quantificateur scalaire uniforme. Il existe un code à longueur variable tel que $\forall s > \frac{1}{2}$ et $c > 0$, si $|x_{i_k}| \sim C k^{-s}$ alors*

$$D(\mathbf{D}, q) \sim C^2 R(\mathbf{D}, q)^{1-2s} \left(1 + \log_2 \frac{N}{R(\mathbf{D}, q)}\right)^{2s-1} \quad \text{pour } R(\mathbf{D}, q) \leq N \quad (3.10)$$

La condition $R(\mathbf{D}, q) \leq N$ impose que l'on se trouve à bas débits, *i.e.*, à moins de 1 bit par pixel (abrégé en bpp). On voit alors que pour de tels débits, la distorsion est proportionnelle à $R(\mathbf{D}, q)^{1-2s}$ au lieu de $2^{-2R(\mathbf{D}, q)}$ à hauts débits (cf. équation (1.32) sous-section 1.2.3 chapitre 1).

Quantificateur optimal

Puisque l'hypothèse de haute résolution n'est plus valide, un quantificateur scalaire uniforme ne minimise pas la fonction débit-distorsion. Reprenons les histogrammes des coefficients de transformation (figure 3.1). Comme nous l'avons mentionné, ces histogrammes peuvent être approximatés par des distributions Laplaciennes. Or, pour celles-ci, on peut montrer [83] que le quantificateur optimal qui minimise la distorsion sous contrainte de débit avec un codage entropique est uniforme de pas de quantification Δ avec un intervalle de quantification autour de 0 défini par $[-\Delta, \Delta]$, *i.e.*, deux fois plus large que les autres intervalles de quantification. Notons que la modification de l'intervalle de quantification autour de 0 n'affecte pas la validité du théorème 9.

Pour une base donnée \mathbf{D} , on peut alors chercher à caractériser le pas de quantification optimal au sens débit-distorsion *i.e.*, à caractériser q_{Δ^*} , quantificateur scalaire uniforme de pas de quantification Δ^* et de zone morte $T = 2\Delta^*$,

tel que

$$q_{\Delta^*} = \underset{q_{\Delta}}{\operatorname{argmin}} D(\mathbf{D}, q_{\Delta}) \text{ soumis à } R(\mathbf{D}, q_{\Delta}) \leq R_c, \quad (3.11)$$

où R_c est un débit cible spécifié comme contrainte sur le débit $R(\mathbf{D}, q_{\Delta})$; $D(\mathbf{D}, q_{\Delta})$ et $R(\mathbf{D}, q_{\Delta})$ sont définis respectivement par (3.6) et (3.5). Comme on le montrera par la suite (cf. sous-sections 3.4.2 et 3.5.3), l'expression $(2 + \log_2 \frac{N}{\|q(\mathbf{x})\|_0})$ dans (3.5) peut être approximée par une constante γ , dépendant de la base \mathbf{D} et du schéma de compression utilisé, de sorte que

$$R(\mathbf{D}, q_{\Delta}) \sim \gamma \|q_{\Delta}(\mathbf{x})\|_0. \quad (3.12)$$

Nous avons vu dans la section 1.5 du chapitre 1 que l'on peut exprimer, sous certaines conditions, ce problème dans une forme Lagrangienne non contrainte. Ainsi,

$$q_{\Delta^*}(\mu) = \underset{q_{\Delta}}{\operatorname{argmin}} D(\mathbf{D}, q_{\Delta}) + \mu R(\mathbf{D}, q_{\Delta}), \quad (3.13)$$

où μ , multiplicateur Lagrangien, est lié implicitement à R_c et définit le point de fonctionnement. Soit, en remplaçant par les expressions de $D(\mathbf{D}, q_{\Delta})$ et $R(\mathbf{D}, q_{\Delta})$,

$$q_{\Delta^*}(\mu) = \underset{q_{\Delta}}{\operatorname{argmin}} \|\mathbf{x} - q_{\Delta}(\mathbf{x})\|_2^2 + \mu \gamma \|q_{\Delta}(\mathbf{x})\|_0. \quad (3.14)$$

On montre alors (cf. annexe 3.8.1 à la fin de ce chapitre) que le pas de quantification optimal, noté $\Delta^*(\mu)$ est lié à μ par la relation

$$\Delta^*(\mu) = \sqrt{\frac{4\gamma\mu}{3}}. \quad (3.15)$$

3.1.2 Redondance et coût de codage

L'approche parcimonieuse se propose de remplacer la base orthonormée utilisée de façon classique dans le codage par transformation par un dictionnaire potentiellement redondant, noté \mathbf{D} , de M colonnes $\mathbf{d}_k \in \mathbb{R}^N$. Des contributions se sont intéressées à cette approche. Citons pour exemples les travaux de Figueras i Ventura *et al.* [53] et Peotta *et al.* [86].

On formalise l'approche de la façon suivante. Si le dictionnaire est de rang plein (ce que nous supposons à chaque fois qu'il s'agira de dictionnaire redondant), tout signal \mathbf{y} est parfaitement représenté mais pas de façon unique. On cherche la représentation la plus parcimonieuse,

$$\mathbf{y} = \sum_{k \in \mathcal{J}} x_k \mathbf{d}_k, \quad (3.16)$$

où \mathcal{J} décrit l'ensemble des indices des vecteurs utilisés pour décrire \mathbf{y} , avec $|\mathcal{J}| \leq N$.

Nous avons évoqué dans le chapitre précédent l'intérêt *intuitif* de l'utilisation d'un dictionnaire redondant plutôt qu'une base dans l'approximation parcimonieuse : plus le dictionnaire de décomposition est redondant, plus le signal dont on cherche une approximation a de chances d'être bien décrit par un petit nombre de vecteurs du dictionnaire. Transposée au codage par transformation, la redondance permettrait donc, pour une parcimonie donnée, de diminuer la distorsion, ou inversement de concentrer davantage l'énergie du signal en un petit nombre de coefficients. Cependant, cet avantage est à mettre en regard du coût de codage de la combinaison des vecteurs, *i.e.*, de la spécification des vecteurs et des coefficients de pondération correspondants utilisés pour décrire le signal.

Après transformation, les coefficients du vecteur parcimonieux sont quantifiés puis codés. On suppose un quantificateur scalaire q ,

$$\mathbf{y} = \sum_{k \in \mathcal{I}} q(x_k) \mathbf{d}_k, \quad (3.17)$$

où \mathcal{I} est l'ensemble des indices des coefficients non nuls de $q(\mathbf{x}) \in \mathbb{R}^M$, avec $|\mathcal{I}| = L$. Notons que la parcimonie de $q(\mathbf{x})$ dans (3.17) dérive du quantificateur qui peut être éventuellement à zone morte, mais également de la décomposition parcimonieuse de \mathbf{y} dans \mathbf{D} , donc des algorithmes et du dictionnaire utilisés (cf. section 2.2 chapitre 2).

Le calcul du coût de codage total $R^R(\mathbf{D}, q)$ est similaire à celui attaché au codage par transformation "classique" (3.5). Il est constitué de la même façon de deux termes, correspondant au nombre de bits nécessaires pour coder les valeurs et les indices des coefficients non nuls de $q(\mathbf{x})$. Cependant, tandis que le premier reste proportionnel à $\|q(\mathbf{x})\|_0 = L$, le deuxième est sensiblement modifié :

$$R_i(\mathbf{D}, q) = \log_2 M + \log_2 C_M^L \sim L(1 + \log_2 \frac{M}{L}). \quad (3.18)$$

Le coût de codage total est donc, dans le cas d'un dictionnaire redondant,

$$\begin{aligned} R^R(\mathbf{D}, q) &= R_v(\mathbf{D}, q) + R_i(\mathbf{D}, q) \\ &\sim L(2 + \log_2 \frac{M}{L}) \geq L(2 + \log_2 \frac{N}{L}). \end{aligned} \quad (3.19)$$

Ainsi, si la redondance peut être intéressante du point de vue de l'approximation du signal \mathbf{y} , son coût - et en particulier le coût de codage du support

de l'approximation de \mathbf{y} (*i.e.*, les indices de coefficients non nuls de $q(\mathbf{x})$) - peut être réductible. Une façon de réduire le coût de codage du support de l'approximation de \mathbf{y} sans se priver d'une bonne description est d'introduire de la structure dans le dictionnaire.

3.2 Structuration du dictionnaire

Dans cette section, nous proposons une structuration de dictionnaire simple basée sur un ensemble de bases locales concaténées en bintree. La première sous-section rappelle quelques contributions de la littérature allant dans ce sens. La deuxième justifie en terme de coût de codage l'utilisation d'un ensemble de bases plutôt qu'un dictionnaire redondant de mêmes dimensions dans le schéma de compression par transformation. Enfin, dans la dernière sous-section, nous complétons cette structuration par une concaténation en arbre.

3.2.1 Motivations

Soit \mathbf{D} un dictionnaire redondant de M colonnes $\mathbf{d}_k \in \mathbb{R}^N$ et \mathbf{y} un signal que l'on cherche à compresser. Pour une parcimonie fixée, *i.e.*, par exemple, pour une approximation de \mathbf{y} avec L atomes de \mathbf{D} , il existe C_M^L combinaisons possibles d'atomes, soit encore C_M^L sous-espaces de génération de $\hat{\mathbf{y}}$ possibles de dimension L .

C'est ce grand nombre qui rend la spécification de la meilleure combinaison coûteuse. On peut dès lors s'interroger sur l'intérêt relatif d'un tel choix : est-il nécessaire de se laisser autant de degrés de liberté ? Ne peut-on pas se satisfaire, en terme de qualité d'approximation, d'un plus petit nombre de sous-espaces ?

A travers l'idée de structuration du dictionnaire, on va donc chercher à forcer le choix de *groupes* d'atomes, *i.e.*, à réduire le nombre de combinaisons d'atomes possibles en les liant les uns aux autres. On trouve dans la littérature un grand nombre de contributions approchant cette idée. Les premières, les plus connues, sont celles utilisant un "découpage" du domaine transformé ou spatial. Par exemple, les paquets d'ondelettes introduits par Coifman, Meyer et Wickerhauser [17] se construisent en divisant l'axe fréquentiel en intervalles de taille variables. Ces transformées sont particulièrement bien adaptées à la décomposition de signaux qui ont des comportements différents selon les intervalles de fréquences. Si au contraire, le signal a des propriétés qui varient

dans l'espace (ou le temps pour des signaux audio par exemple), il sera plus judicieux de décomposer le signal en "arbre" qui segmente l'axe spatial (ou temporel) en intervalles dont les tailles sont adaptées aux structures du signal. Nous y avons fait mention dans l'état de l'art consacré à la compression par transformation (sous-section 1.3.2 chapitre 1).

Dans une approche propre aux décompositions parcimonieuses, Eldar *et al.* ont proposé dans [34] et [33] un dictionnaire redondant sous contrainte de parcimonie structurée en "blocs" ou "sous-espaces". Le vecteur parcimonieux qui en résulte présente de grandes plages de zéros correspondant aux zones du dictionnaire non utilisées. On en trouve une généralisation dans [3] sous forme de "modèles" de parcimonie. Enfin, dernièrement, Yu *et al.* ont proposé dans [118] un algorithme de débruitage basé sur un ensemble de bases apprises sur l'image bruitée par une analyse en composantes principales (PCA pour principal component analysis en anglais). Notons que ces approches "parcimonieuses" peuvent être vues comme autant de façons de structurer le dictionnaire dans le domaine transformé - si tant est qu'une telle distinction "transformé-spatial" ait un sens puisque dans les deux cas, c'est bien le dictionnaire résultant qui transpose le signal dans le domaine transformé.

3.2.2 Ensemble de bases

Sur la base des dernières contributions présentées dans la sous-section précédente, nous proposons de considérer dans un premier temps un ensemble de P bases orthonormées : on suppose que le signal \mathbf{y} que l'on cherche à compresser a une approximation parcimonieuse dans l'une d'entre elles. La parcimonie de l'approximation dans cette base peut être totalement indépendante de la quantification : les coefficients de la transformation sont alors seuillés (selon un paramètre Lagrangien, un critère de distorsion ou un nombre de coefficients non nuls maximum, cf. sous-section 2.2.1 du chapitre 2) puis quantifiés. Elle peut au contraire être obtenue par la seule quantification. C'est l'approche que nous traiterons dans la suite du manuscrit.

On note $\mathcal{D} = \{\mathbf{D}_1, \dots, \mathbf{D}_P\}$ l'ensemble des P bases orthonormées. Si $\mathbf{D}_i = [\mathbf{d}_i^1, \dots, \mathbf{d}_i^N]$ est la base dans laquelle \mathbf{y} a une approximation parcimonieuse, le signal issu de la compression de \mathbf{y} , noté $\hat{\mathbf{y}}$, est tel que

$$\begin{aligned} \hat{\mathbf{y}} &= \sum_{k=1}^N q(x_k) \mathbf{d}_i^k, \\ &= \sum_{k \in \mathcal{I}} q(x_k) \mathbf{d}_i^k, \end{aligned} \quad (3.20)$$

où \mathcal{I} est l'ensemble des indices des coefficients non nuls de $q(\mathbf{x})$, avec $|\mathcal{I}| = \|q(\mathbf{x})\|_0 = L$. On suppose pour simplifier que le quantificateur q utilisé est indépendant de la base \mathbf{D}_i .

On note $R_d(\mathcal{D})$ le coût de spécification de la base choisie pour l'approximation parcimonieuse du signal \mathbf{y} . $R_d(\mathcal{D})$ est égal à $\log_2 |\mathcal{D}| = \log_2 P$ avec un codage à longueur fixe. Le débit binaire total dépend de l'ensemble \mathcal{D} et du quantificateur q , il est noté $R^E(\mathcal{D}, q)$ et s'exprime d'après (3.5) comme

$$\begin{aligned} R^E(\mathcal{D}, q) &= R_v(\mathbf{D}_i, q) + R_i(\mathbf{D}_i, q) + R_d(\mathcal{D}), \\ &\sim L(2 + \log_2 \frac{N}{L}) + \log_2 P. \end{aligned} \quad (3.21)$$

Comparons ce résultat au débit atteint avec le dictionnaire correspondant non structuré, *i.e.*, fait de la concaténation des P dictionnaires. Ce dictionnaire, noté $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_P]$ contient alors $M = PN$ atomes. Le débit binaire total $R^R(\mathbf{D}, q)$ résultant est tel que

$$\begin{aligned} R^R(\mathbf{D}, q) &= R_v(\mathbf{D}, q) + R_i(\mathbf{D}, q), \\ &\sim L(2 + \log_2 \frac{PN}{L}), \\ &\sim L(2 + \log_2 \frac{N}{L} + \log_2 P). \end{aligned} \quad (3.22)$$

On a donc

$$R^R(\mathbf{D}, q) \geq R^E(\mathcal{D}, q), \quad (3.23)$$

avec égalité si et seulement si $L = 1$, *i.e.*, un seul atome est utilisé pour décrire le signal \mathbf{y} .

3.2.3 Concaténations de bases locales

L'ensemble de bases permet de structurer le dictionnaire de décomposition dans le domaine transformé. Dans cette sous-section, nous proposons de combiner cette approche avec une structuration spatiale, via l'utilisation d'un arbre. Notre choix se porte sur une segmentation en bintree, qui exploite, de façon simple, des supports de bases locales anisotropiques (rectangulaires).

On peut considérer la segmentation en bintree de deux manières différentes. D'un côté, elle offre un degré de liberté supplémentaire par rapport à un traitement de l'image en blocs de taille fixée, permettant d'augmenter le nombre de bases d'image possibles (et on peut intuitivement supposer que plus on a de bases à disposition, plus on a de chance de choisir une

base qui décrira au mieux les propriétés de l'image avec peu de coefficients). De l'autre, ce degré de liberté est *limité* puisque le bintree interdit de par sa structure certaines combinaisons de bases locales. Il y a donc bien *structuration* résultant en une réduction du coût de codage de la base d'image utilisée par rapport à un ensemble non structuré de bases d'image.

Avec cette double structuration - concaténation de bases locales en arbre et sélection des bases locales dans un ensemble - on espère obtenir une bonne adaptabilité spatiale de la base de transformation aux géométries de l'image et une bonne description des caractéristiques locales de l'image. Pour éviter les confusions entre bases locales et globales, nous noterons, dans la suite de ce chapitre, \mathbf{B} une base d'image *globale* construite ainsi que nous venons de le voir et \mathbf{D}_i une base *locale* d'indice i choisie dans un ensemble $\mathcal{D} = \{\mathbf{D}_1, \dots, \mathbf{D}_P\}$.

Reprenons l'expression du débit (3.21). Elle est constituée des termes de codages des coefficients de transformation quantifiés non nuls (valeurs et indices) et de la spécification de la base de transformation choisie dans un ensemble (\mathbf{D}_i est alors une base *globale*). Avec le niveau de structuration supplémentaire apporté par la segmentation en bintree, ce dernier coût devient celui de la spécification des bases choisies *localement* et il faut en outre ajouter le coût de codage de la segmentation en bintree, spécifiant les supports des bases locales.

Pour alléger les équations, nous adoptons les notations suivantes :

- ◆ R_x est le débit binaire attaché au codage des coefficients de transformation quantifiés non nuls (valeurs et indices),
- ◆ R_d est le coût de codage des indices des bases locales,
- ◆ R_s est le nombre de bits nécessaires pour coder la segmentation en bintree.

Le débit binaire total qui dépend de la base d'image globale notée \mathbf{B} et du quantificateur global q (nous détaillons ce dernier dans la section suivante) s'exprime donc comme

$$R(\mathbf{B}, q) = R_x + R_d + R_s. \quad (3.24)$$

Nous explicitons chacun de ses termes dans la section suivante.

3.3 Optimisation débit-distorsion

Dans la section précédente, nous avons défini un ensemble de bases d'image. Dans cette section, nous étudions le problème du choix de la meilleure base parmi cet ensemble.

Comme nous l'avons évoqué plusieurs fois, deux paramètres du codage par transformation peuvent affecter les performances débit-distorsion : la base de transformation et le quantificateur des coefficients transformés. On note \mathbf{B}^* la base d'image globale, optimale au sens débit-distorsion, recherchée (parmi un ensemble structuré comme expliqué dans la section précédente), et q^* le quantificateur optimal recherché. Sous sa forme non contrainte, le problème s'exprime de la façon suivante :

$$(\mathbf{B}^*(\mu), q^*(\mu)) = \underset{\mathbf{B}, q}{\operatorname{argmin}} D(\mathbf{B}, q) + \mu R(\mathbf{B}, q), \quad (3.25)$$

avec μ multiplicateur Lagrangien.

Hypothèses de résolution

La solution de (3.25) peut être obtenue de façon efficace par programmation dynamique [87] en s'assurant que :

1. l'ensemble des bases d'image est une concaténation en arbre de bases locales,
2. la distorsion et le débit peuvent être décomposés en termes locaux associés à chaque base locale.

La première condition est satisfaite par notre schéma de compression (cf. sous-section précédente). La segmentation en bintree permet une application *locale* de la quantification. Localement, dans le cas d'une quantification uniforme scalaire avec une zone morte, le pas de quantification dépend des caractéristiques de la base de transformation (cf. équation (3.15)). Nous faisons ici l'hypothèse que ce pas de quantification est le même quelle que soit la base locale choisie. Ainsi, la quantification réalisée sur l'image globale ne dépend que du support de la base de transformation, *i.e.*, de la segmentation en bintree. Le pas de quantification est défini localement par $\Delta^*(\mu) = \{\Delta_j^*(\mu)\}_{j \in \mathcal{F}}$ tel que

$$\forall j \in \mathcal{F} \quad \Delta_j^*(\mu) = \sqrt{\frac{4\gamma_j\mu}{3}}, \quad (3.26)$$

où \mathcal{F} est l'ensemble des feuilles du bintree. Le problème (3.25) se simplifie alors en :

$$\mathbf{B}^*(\mu) = \underset{\mathbf{B}}{\operatorname{argmin}} D(\mathbf{B}, \Delta^*(\mu)) + \mu R(\mathbf{B}, \Delta^*(\mu)). \quad (3.27)$$

En revanche, la deuxième condition doit être vérifiée pour chacun des termes R_x , R_d et R_s que nous avons définis plus haut.

Nous adoptons d'abord une simple implémentation du bintree en assignant "1" aux noeuds internes de l'arbre et "0" aux noeuds feuilles. Ainsi, R_s peut être exprimé comme la somme des débits nécessaires pour coder chaque feuille, notés $\{R_s^j\}_{j \in \mathcal{F}}$, i.e.,

$$R_s = \sum_{j \in \mathcal{F}} R_s^j. \quad (3.28)$$

Nous supposons ensuite que les indices des bases locales sont encodés par un code à longueur fixe. Ainsi, notant l_{FLC} la longueur du code FLC, on a

$$\begin{aligned} R_d &= \sum_{j \in \mathcal{F}} R_d^j, \\ &= |\mathcal{F}| l_{FLC}. \end{aligned} \quad (3.29)$$

Notons que l'utilisation d'un FLC est en général sous-optimale (cf. sous-section 1.2.2 chapitre 1). Ce choix est fait ici pour des raisons de complexité puisqu'il permet de décomposer R_d en une somme de termes locaux. Dans la sous-section suivante, nous décrivons une façon plus efficace de coder les indices des bases locales. L'expression (3.29) constitue alors une borne supérieure sur le débit atteignable par le schéma pratique.

Le débit nécessaire pour coder les coefficients quantifiés de transformation *sur chaque base locale* est proportionnel au nombre de coefficients non nuls (cf. équation (3.12)). D'après l'hypothèse que nous avons posée sur l'indépendance de la quantification par rapport aux indices des bases locales utilisées,

$$\begin{aligned} R_x &= \sum_{j \in \mathcal{F}} R_x^j, \\ &= \sum_{j \in \mathcal{F}} \gamma_j \|q_j(\mathbf{x}_j)\|_0, \end{aligned} \quad (3.30)$$

où $q_j(\mathbf{x}_j)$ représente le vecteur de coefficients de transformation quantifiés sur la j -ième feuille du bintree et γ_j est le facteur de proportionnalité entre le débit local R_x^j et le nombre de coefficients $\|q_j(\mathbf{x}_j)\|_0$. Par hypothèse, la quantification

q_j dépend uniquement du support de la transformée locale, *i.e.*, de la taille du j -ième bloc-feuille.

Résumant ces observations et hypothèses, on obtient finalement que le débit total peut être décomposé comme une somme de termes locaux, *i.e.*,

$$R(\mathbf{B}, \Delta^*(\mu)) = \sum_{j \in \mathcal{F}} R(\mathbf{D}_{i_j}, \Delta_j^*(\mu)), \quad (3.31)$$

avec $R(\mathbf{D}_{i_j}, \Delta_j^*(\mu)) = R_s^j + R_d^j + R_x^j$, où \mathbf{D}_{i_j} est la base locale attachée à la j -ième feuille et choisie dans un ensemble \mathcal{D} , $i_j \in \{1, \dots, P\}$. Le problème (3.25) peut alors être considéré comme $|\mathcal{F}|$ problèmes d'optimisation indépendants, tels que

$$\mathbf{B}^*(\mu) = \sum_{j \in \mathcal{F}} \underset{\mathbf{D}_{i_j}}{\operatorname{argmin}} D(\mathbf{D}_{i_j}, \Delta_j^*(\mu)) + \mu R(\mathbf{D}_{i_j}, \Delta_j^*(\mu)), \quad (3.32)$$

et des méthodes de programmation dynamique standard (cf. [87]) peuvent alors être appliquées.

3.4 Bases locales prédéfinies : les DCT directionnelles

Les DCT directionnelles (DDCT) ont été introduites dans [122] sur des supports carrés. Les auteurs montrent que ces bases donnent de bonnes performances de codage pour des blocs d'image contenant des contours orientés. Dans cette section, nous étendons ces bases à des supports rectangulaires de taille variable afin de les exploiter dans un schéma de compression utilisant une segmentation en bintree de l'image. Des résultats en termes de performance débit-distorsion sont présentés et comparés aux standards de compression JPEG et JPEG2000. Ils ont été présentés lors de la conférence ICASSP 2010 [C].

3.4.1 Principe

L'idée principale des DDCT repose sur la propriété de séparabilité de la transformée DCT bidimensionnelle standard, *i.e.*, une DCT bidimensionnelle peut être réalisée en calculant successivement une DCT unidimensionnelle sur les colonnes du bloc-image considéré puis une DCT unidimensionnelle sur les lignes du bloc issu de la première DCT. La construction d'une DDCT est basée sur la modification de l'ordre de scanning des pixels du bloc pour créer des bases avec des directions privilégiées.

Pour des raisons de clarté, nous exposons la construction d'une transformée DDCT sur un bloc y de taille 8×4 pixels et le mode directionnel "diagonal down-left" représenté sur la figure 3.2. L'extension aux autres modes directionnels est directe.

A chaque mode directionnel, on associe un ordre de scanning défini par un ensemble de vecteurs v_k . Chaque vecteur v_k contient un sous-ensemble de pixels pris selon une direction donnée. Pour le mode "diagonal down-left", les v_k sont constitués des pixels situés sur des flèches allant de haut en bas et de droite à gauche comme illustré sur la figure 3.2. Ils ont ainsi des longueurs différentes.

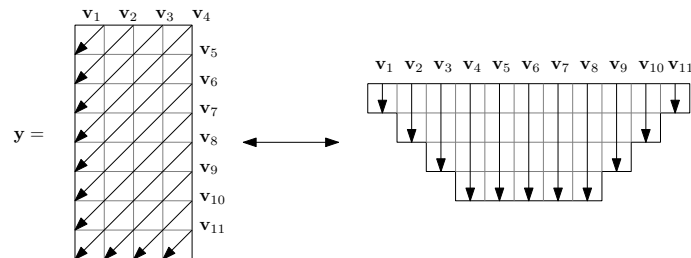


Figure 3.2 Ordonnement des pixels pour le mode "diagonal down-left" sur un bloc rectangulaire

Les pixels peuvent être ensuite réordonnés, résultant en un tableau de pixels "pyramidal", dont les colonnes sont les vecteurs v_k . Considérant ce tableau, le processus de construction de la DDCT est alors similaire à la DCT bidimensionnelle standard. Une première DCT unidimensionnelle est d'abord réalisée sur les colonnes v_k ; une seconde ensuite sur les lignes de la matrice pyramidale, comme illustré sur la figure 3.3.

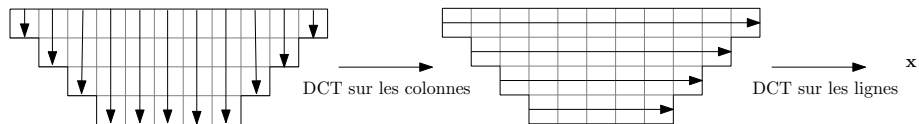


Figure 3.3 Transformée DCT "diagonal down-left" sur un bloc rectangulaire

Le bloc de transformation x est enfin obtenu en réarrangeant les coefficients dans leurs positions initiales dans y .

3.4.2 Implémentation

Dans cette section, nous détaillons l'implémentation du schéma de compression que nous proposons et illustrons ses performances.

Dictionnaire

On considère un ensemble de bases d'image construites comme expliqué dans la section 3.2. Les supports des bases locales ont des tailles qui varient de 4×4 pixels à 32×32 pixels. Pour chaque taille de support, on considère 7 modes directionnels correspondant à 7 modes de prédiction intra du format de compression vidéo H.264 (cf. sous-section 1.4.3 chapitre 1). Les modes "vertical", "horizontal" et "DC" de la prédiction H.264 sont regroupés sous le mode directionnel "1", résultant en la DCT bidimensionnelle "standard". Les modes suivants correspondent à des DCT orientées. La base de transformation est sélectionnée selon une procédure d'optimisation décrite dans la section 3.3.

Ordre de scanning

De la même façon que dans le format de compression JPEG (cf. figure 1.5, sous-section 1.3.3 chapitre 1), les coefficients transformés et quantifiés sont traités selon un ordre de scanning particulier favorisant un ordonnancement de la fréquence la plus basse vers la fréquence la plus haute (cf. [122] pour des blocs carrés). La figure 3.4 montre l'ordre de scanning choisi pour le mode "diagonal down-left" sur un bloc de 8×4 pixels.

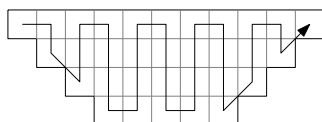


Figure 3.4 Ordre de scanning des coefficients de transformation utilisé pour le mode "diagonal down-left"

Codage des coefficients quantifiés

Après ordonnancement, les coefficients quantifiés de la transformation sont encodés par des codes de Huffman. Les tables de Huffman sont optimisées selon la taille du support des transformées locales. Les indices des coefficients non nuls sont encodés via un codage par plages de zéros (RLE, cf. sous-section 1.2.2 chapitre 1).

Codage des modes directionnels locaux

L'encodage des modes directionnels est réalisé au moyen d'une procédure quadtree comme illustré sur la figure 3.5.

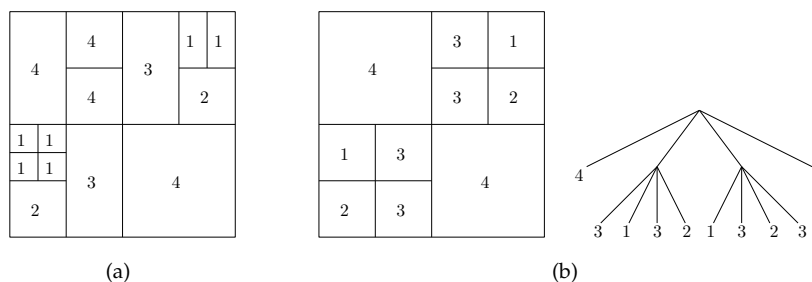


Figure 3.5 (a) Exemple d'une segmentation en bintree et (b) encodage en quadtree des modes directionnels locaux correspondants

La figure 3.5(a) représente les supports des bases locales. Le numéro à l'intérieur de chaque support correspond au mode directionnel choisi localement. La figure 3.5(b) représente l'encodage en quadtree des indices correspondants. On procède de la façon suivante. L'image est segmentée en 4 blocs carrés de dimensions égales. Si toutes les bases locales dans un bloc ont un mode directionnel identique, ce bloc correspond à une feuille du quadtree et est "labellisé" par le mode directionnel commun ; sinon, le bloc est sous-divisé en 4 blocs carrés etc.

Analyse des facteurs de proportionnalité γ_j

Les facteurs γ_j dépendent des bases de transformation et du schéma de codage. Nous faisons ici l'hypothèse qu'ils ne dépendent pas des modes directionnels des bases : seule la taille du support de la transformation est importante. Les facteurs γ_j sont donc déterminés empiriquement par taille de bloc, en considérant un schéma de codage identique à celui défini précédemment mais pour une segmentation en blocs de même taille. Les courbes 3.6(a) et 3.6(b) expriment le rapport $\frac{R_x}{L}$, où L est le nombre de coefficients non nuls sur toute l'image, en fonction de $\log_2 \frac{N}{L}$, où N est la dimension de l'image, pour respectivement une segmentation en blocs de taille 8×8 pixels et 16×16 pixels. On observe que ce rapport évolue très peu, dans un intervalle compris entre 6 et 8.5 pour les blocs de 8×8 pixels et entre 5.5 et 8.5 pour des blocs de 16×16 pixels. Le tableau 3.1 résume les valeurs moyennes choisies pour les 7 tailles de blocs possibles de la segmentation en bintree proposée.

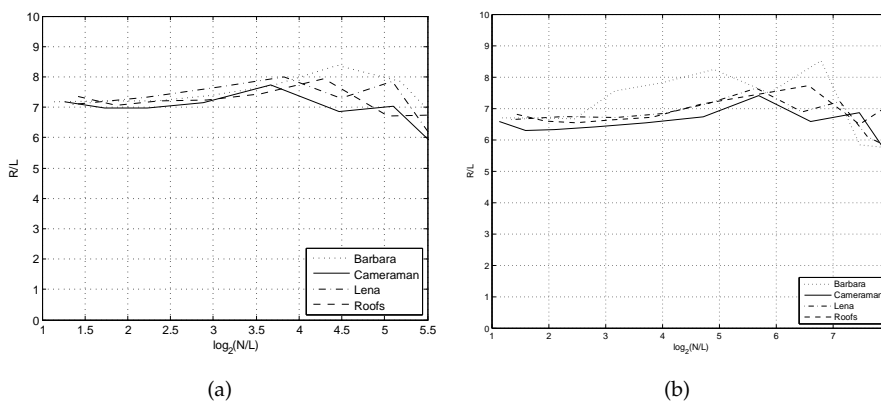


Figure 3.6 Valeurs des γ_j obtenues par compression des images "Barbara", "Cameraman", "Lena" et "Roofs" sur une segmentation en blocs de 8×8 pixels (a) et 16×16 pixels (b).

Taille de bloc	32×32	32×16	16×16	16×8	8×8	8×4	4×4
γ_j	6.80	6.90	6.90	7.0	7.40	7.20	7.10

Table 3.1 Valeurs moyennes de γ_j par taille de blocs pour les DCT directionnelles.

3.4.3 Evaluation des performances

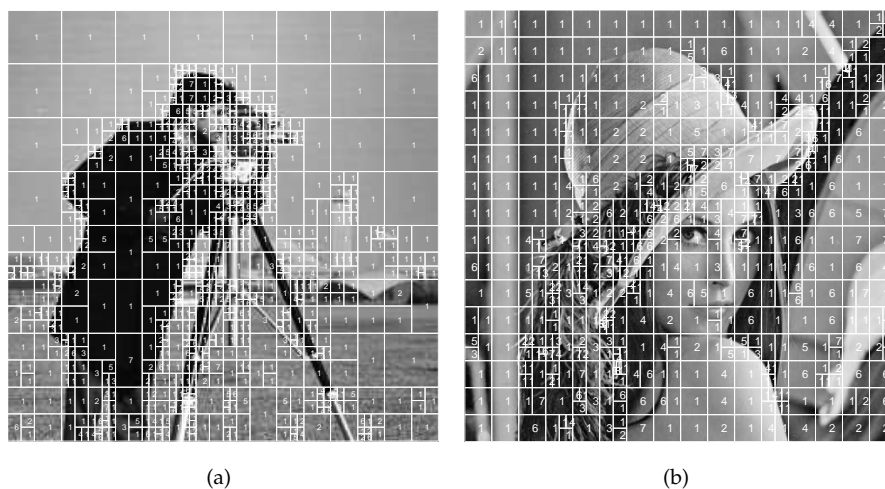


Figure 3.7 (a) Segmentation binaire et modes directionnels associés obtenus pour "Cameraman" (a) à $R= 0.46$ bpp et PSNR= 31.40 dB et "Lena" (b) à $R= 0.08$ bpp et PSNR= 28.30 dB.

La figure 3.7 représente les supports des bases locales composant la base

d'image ainsi que leurs modes directionnels pour les images "Cameraman" et "Lena", à des points débit-distorsion particuliers. On remarque que la DCT conventionnelle (mode "1") est sélectionnée dans les régions homogènes de l'image tandis que les transformées DDCT sont utilisées dans les régions où la directionnalité est plus importante (par exemple, des contours orientés).

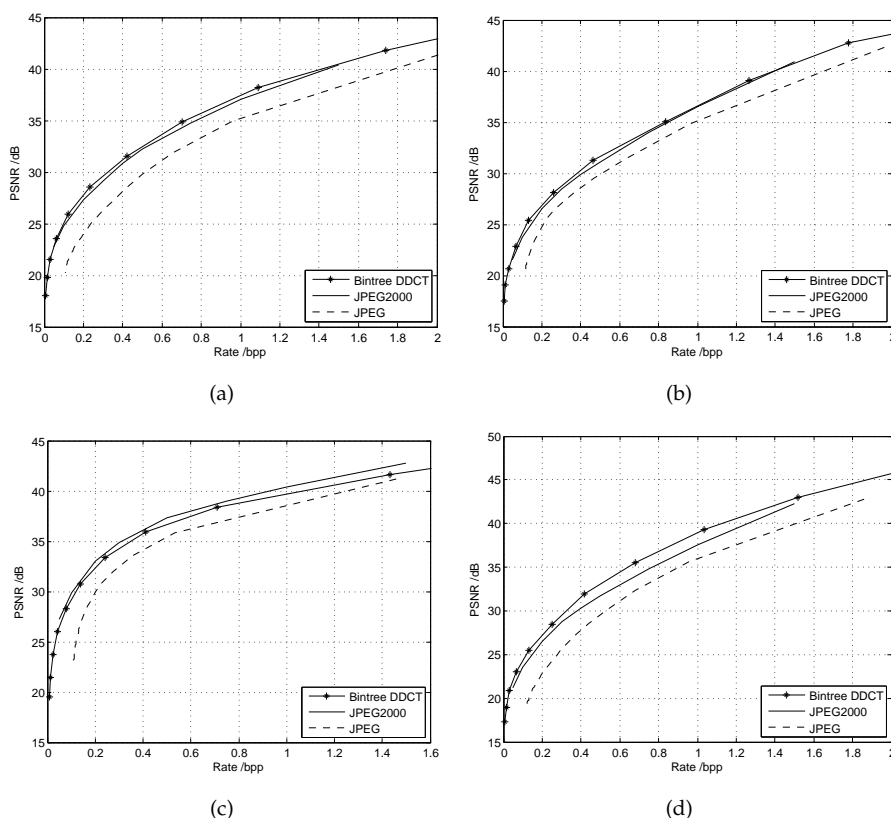


Figure 3.8 Courbes débit-distorsion pour la compression de "Barbara" (a), "Cameraman" (b), "Lena" (c) et "Roofs" (d) avec le schéma de codage proposé utilisant des DCT directionnelles et les standard JPEG2000 et JPEG.

La figure 3.8 compare les performances débit-distorsion obtenues par notre schéma de compression avec celles obtenues par les formats de compression JPEG et JPEG2000 sur quelques images. On remarque que le schéma de compression proposé surpasse JPEG de plus d'1 dB pour l'ensemble des images étudiées et JPEG2000 d'1 dB environ pour la plupart des images à bas et moyens débits. L'image Lena est moins bien encodée avec le schéma de compression proposé qu'avec le standard JPEG2000. Une explication possible

réside dans les caractéristiques des atomes de transformée utilisés : les DCT directionnelles semblent bien décrire les zones et contours orientés, mais pas les textures sans orientation "simple", comme peut l'être la plume du chapeau. Remarque : les images sont rappelées en annexe 3.8.3 de ce chapitre.

3.5 Bases locales apprises : étude d'un algorithme de la littérature

Les bonnes performances obtenues par le schéma de compression exploité dans la section précédente sont encourageantes. Conservant la structuration du dictionnaire détaillée dans la section 3.2, on peut espérer les améliorer encore en utilisant des bases locales apprises selon un critère favorisant la parcimonie des décompositions. C'est l'objet de cette section et des deux suivantes.

Nous nous proposons dans un premier temps d'étudier un algorithme d'apprentissage de bases adaptées aux décompositions parcimonieuses introduit par Sezer *et al.* dans [96]. Cet algorithme, appelé algorithme de Sezer dans la suite, servira d'algorithme de référence pour l'étude et la conception d'un algorithme Bayésien que nous présenterons dans les deux sections 3.6 et 3.7.

Nous adopterons dans la suite de ce chapitre les notations suivantes. Soit $\{\mathbf{y}_j\}_{j=1}^K$ un ensemble d'entraînement pour l'optimisation d'un ensemble de P bases. Nous noterons \mathbf{D} cet ensemble de bases et le définissons de la façon suivante,

$$\mathbf{D} \triangleq [\mathbf{D}_1, \dots, \mathbf{D}_i, \dots, \mathbf{D}_P], \quad \mathbf{D}_i^T \mathbf{D}_i = \mathbf{I}_N, \quad (3.33)$$

où \mathbf{I}_N est la matrice identité de dimension N . Cette notation est abusive : \mathbf{D} n'est jamais considéré comme concaténation mais bien comme un ensemble de P bases. Elle permet cependant une formulation simple des algorithmes. De la même façon, on note \mathbf{x}_{ji} le vecteur de décomposition de \mathbf{y}_j dans la base \mathbf{D}_i et \mathbf{x}_j le vecteur tel que

$$\mathbf{x}_j^T \triangleq [\mathbf{x}_{j1}^T, \dots, \mathbf{x}_{ji}^T, \dots, \mathbf{x}_{jP}^T]^T. \quad (3.34)$$

Les résultats présentés dans cette section ont fait l'objet d'un article dans les proceedings de la conférence SPIE 2010 [F].

3.5.1 Algorithme de Sezer

L'algorithme de Sezer est un algorithme itératif en deux étapes. Dans une première étape, chaque signal d'entraînement est assigné à une famille $\mathcal{S}_i, i \in$

$\{1, \dots, P\}$. Puis, dans une seconde étape, chaque famille \mathcal{S}_i est utilisée pour optimiser une base \mathbf{D}_i sous un critère parcimonie-distorsion.

Plus précisément, à la k -ième itération de l'algorithme, la première étape associe à chaque signal d'entraînement \mathbf{y}_j , $j \in \{1, \dots, K\}$, un indice $c_j^{(k)} \in \{1, \dots, P\}$ correspondant à l'indice de la base qui minimise l'erreur d'approximation du signal sous contrainte de parcimonie. De façon formelle, on définit

$$\forall i \in \{1, \dots, P\}, \quad \mathcal{S}_i^{(k)} = \left\{ j \in \{1, \dots, K\} \mid c_j^{(k)} = i \right\}, \quad (3.35)$$

$$\text{où } c_j^{(k)} = \underset{i \in \{1, \dots, P\}}{\operatorname{argmin}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i^{(k-1)} \mathbf{x}_{ji}^{(k-1)}\|_2^2 + \lambda' \|\mathbf{x}_{ji}^{(k-1)}\|_0 \right\}. \quad (3.36)$$

L'étape de mise à jour des bases, formant la deuxième étape de l'algorithme de Sezer, se formalise de la façon suivante :

$$\begin{aligned} \forall i \in \{1, \dots, P\}, \forall j \in \{1, \dots, K\} \\ \mathbf{D}_i^{(k)} = \underset{\mathbf{D}_i}{\operatorname{argmin}} \left\{ \sum_{j \in \mathcal{S}_i^{(k)}} \min_{\mathbf{x}_{ji}} \{ \|\mathbf{y}_j - \mathbf{D}_i \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_0 \} \right\} \\ \text{soumis à } \mathbf{D}_i^T \mathbf{D}_i = \mathbf{I}_N. \end{aligned} \quad (3.37)$$

Elle est en pratique elle-même réalisée selon une procédure itérative, optimisant successivement pour chaque famille \mathcal{S}_i les représentations parcimonieuses associées aux signaux de la famille considérée et la base \mathbf{D}_i . Les représentations parcimonieuses sont calculées par une opération de seuillage dur (cf. section 2.2.1 chapitre 2), selon la formalisation "standard"

$$\begin{aligned} \forall i \in \{1, \dots, P\}, \forall j \in \{1, \dots, K\}, \\ \mathbf{x}_{ji}^{(k,l)} = \underset{\mathbf{x}_{ji}}{\operatorname{argmin}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i^{(k,l-1)} \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_0 \right\}, \end{aligned} \quad (3.38)$$

où (l) est le numéro de l'itération dans l'étape de mise à jour des bases. Notons que le paramètre λ' est défini par l'utilisateur et permet de fixer le compromis entre parcimonie et distorsion. Les bases \mathbf{D}_i sont ensuite estimées. Le problème se formalise, à la l -ième itération de l'étape, de la façon suivante :

$$\begin{aligned} \forall i \in \{1, \dots, P\}, \\ \mathbf{D}_i^{(k,l)} = \underset{\mathbf{D}_i}{\operatorname{argmin}} \left\{ \sum_{j \in \mathcal{S}_i^{(k)}} \|\mathbf{y}_j - \mathbf{D}_i \mathbf{x}_{ji}^{(k,l)}\|_2^2 \right\} \text{ soumis à } \mathbf{D}_i^T \mathbf{D}_i = \mathbf{I}_N, \end{aligned} \quad (3.39)$$

et revient à calculer

$$\forall i \in \{1, \dots, P\}, \quad \mathbf{D}_i^{(k,l)} = \mathbf{V} \mathbf{U}^T, \quad (3.40)$$

 Algorithme 5: Algorithme de Sezer

0. Initialisation

- $\mathbf{D}^{(0)} = [\mathbf{D}_1^{(0)}, \dots, \mathbf{D}_i^{(0)}, \dots, \mathbf{D}_P^{(0)}]$,
- $\forall i \in \{1, \dots, P\}, \forall j \in \{1, \dots, K\}, \mathbf{x}_{ji}^{(0)} = \underset{\mathbf{x}_{ji}}{\operatorname{argmin}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i^{(0)} \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_0 \right\}$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Classification

$$\forall i \in \{1, \dots, P\}, \mathcal{S}_i^{(k)} = \left\{ j \in \{1, \dots, K\} \mid c_j^{(k)} = i \right\}, \quad (3.35)$$

$$\text{où } c_j^{(k)} = \underset{i \in \{1, \dots, P\}}{\operatorname{argmin}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i^{(k-1)} \mathbf{x}_{ji}^{(k-1)}\|_2^2 + \lambda' \|\mathbf{x}_{ji}^{(k-1)}\|_0 \right\}. \quad (3.36)$$

2. Mise à jour des bases

$\forall i \in \{1, \dots, P\}$,

$$\mathbf{D}_i^{(k)} = \underset{\mathbf{D}_i}{\operatorname{argmin}} \left\{ \sum_{j \in \mathcal{S}_i^{(k)}} \min_{\mathbf{x}_{ji}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_0 \right\} \right\} \text{ soumis à } \mathbf{D}_i^T \mathbf{D}_i = \mathbf{I}_N. \quad (3.37)$$

où $\mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{V}^T$ est la décomposition en valeurs singulières de $\sum_{j \in \mathcal{S}_i} \mathbf{x}_{ji}^{(k,l)} \mathbf{y}_j^T$. La justification de ce calcul peut être trouvée dans [62, 96]. A l'issue de la procédure itérative de l'étape, au bout de l_{fin} itérations,

$$\forall i \in \{1, \dots, P\}, \mathbf{D}_i^{(k)} = \mathbf{D}_i^{(k, l_{fin})}. \quad (3.41)$$

L'algorithme 5, dans l'encadré, résume les deux étapes de l'algorithme de Sezer.

3.5.2 Améliorations

Dans cette section, nous introduisons quelques modifications dans l'algorithme de Sezer afin d'améliorer sa convergence et ses performances en termes de distorsion *vs* débit. Pour étudier la pertinence de ces modifications, nous appliquons les bases apprises sur des blocs de taille 8×8 pixels et adoptons un schéma de codage simple, utilisant des codes de Huffman pour encoder les coefficients quantifiés et un encodeur RLE pour coder les indices des coefficients non nuls. Les performances débit-distorsion atteintes sont alors comparées à celles obtenues par l'algorithme de Sezer original.

Initialisation

La fonction que l'on cherche à optimiser est multimodale : selon l'initialisation, l'algorithme converge vers un minimum local particulier, différentes initialisations conduisent donc potentiellement à différents résultats. A

l'étape d'initialisation, Sezer *et al.* réalisent une classification des blocs d'entraînement en K sous-ensembles au moyen de gradients d'image. Sur chaque sous-ensemble, la base est ensuite initialisée par une transformée de Karhunen Løve (KLT) en utilisant les signaux correspondants. Or, même si l'on peut penser que des signaux de mêmes caractéristiques géométriques auront des décompositions parcimonieuses dans une même base, il n'en existe pas de preuve évidente. Il peut donc être intéressant de décorréliser l'initialisation des bases de l'ensemble des signaux d'entraînement. Nous procédons en initialisant d'abord K bases indépendamment de l'ensemble d'entraînement et en reliant ensuite par (3.35) les signaux d'entraînement à l'une d'entre elles. Dans ce but, nous proposons d'utiliser les DDCT introduites par Zeng *et al.* [122] décrites dans la section précédente. Etant indépendantes des signaux d'entraînement, ces bases permettront une classification des signaux selon le seul critère de parcimonie de leurs décompositions ; et si effectivement, il y a bien lien entre parcimonie des décompositions et similarités géométriques des signaux, ces bases n'y feront pas obstacle, présentant elles-mêmes des caractéristiques géométriques fortes.

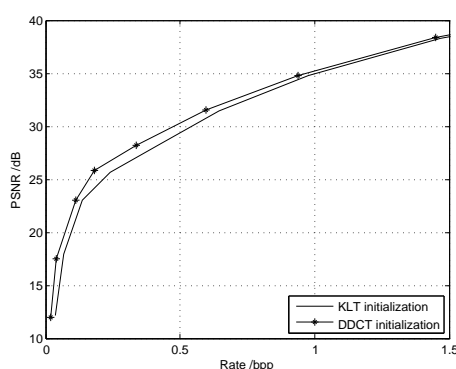


Figure 3.9 Performances débit-distorsion obtenues par deux initialisations différentes : KLT and DDCT.

La figure 3.9 présente les performances débit-distorsion obtenues par l'algorithme de Sezer initialisé par des KLT et par des DDCT sur l'image Barbara. On observe que cette dernière initialisation améliore le PSNR par plus d'1 dB à bas débits.

Critère de parcimonie

Autre enjeu important : le choix de la norme ℓ_p utilisée comme mesure de la parcimonie dans (3.38), dont nous avons déjà eu l'occasion (cf. chapitre 2) de

mentionner le rôle peut conduire à des performances différentes. Le cas $p = 0$ utilisé par Sezer implémente le “vrai” critère de parcimonie. Cependant, d’autres choix de p peuvent potentiellement conduire à des points fixes différents et par ailleurs, se montrer plus judicieux en terme de rapidité d’exécution de l’algorithme. Nous nous sommes intéressés au cas $p = 1$. Dans ce cas, la relation (3.38) est remplacée par

$$\mathbf{x}_{ji}^{(k)} = \underset{\mathbf{x}_{ji}}{\operatorname{argmin}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i^{(k)} \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_1 \right\}, \quad (3.42)$$

et résolue par une opération de seuillage doux (cf. section 2.2.1 chapitre 2).

Pour ce choix de norme, nous avons pu constater de façon expérimentale que l’algorithme résultant était plus rapide.

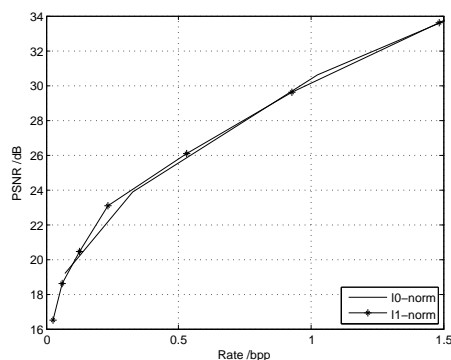


Figure 3.10 Performances débit-distorsion obtenues par deux normes d’optimisation différentes : normes ℓ_0 and ℓ_1 .

Les courbes comparatives débit-distorsion sont données dans la figure 3.10 sur l’image Roofs. Les deux algorithmes sont initialisés avec les DDCT. On peut voir que le choix de la norme ℓ_1 conduit à une légère augmentation du PSNR.

Prise en compte de la quantification

Enfin, puisque les bases apprises sont appliquées dans un contexte de compression d’image, il semble intéressant d’intégrer la connaissance du schéma de codage utilisé après l’étape de transformation. En pratique, cela revient à prendre la quantification en compte et ainsi, à contraindre les \mathbf{x}_j à prendre leurs valeurs dans un ensemble fini de points. En particulier, si on considère une quantification scalaire uniforme à zone morte $T = 2\Delta$, où Δ est le pas de quantification, les coefficients des \mathbf{x}_j peuvent prendre leurs valeurs

dans l'ensemble $\{0\} \cup \{\pm(\frac{3}{2}\Delta + k\Delta)\}_{k \in \mathbb{N}}$. Cette restriction permet de lier implicitement l'ensemble des bases optimisées au quantificateur utilisé dans le schéma de codage considéré.

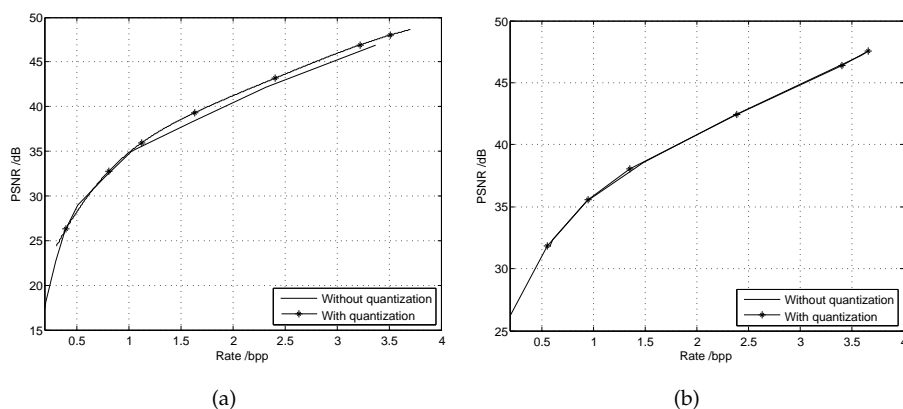


Figure 3.11 Performances débit-distorsion obtenues avec et sans intégration de la quantification dans l'algorithme d'apprentissage : (a) sur des blocs de taille 4×4 pixels, (b) sur des blocs de taille 8×8 pixels.

La figure 3.11 analyse la pertinence de l'intégration de la quantification dans l'algorithme d'apprentissage sur l'image Peppers. Les algorithmes sont initialisés avec des DDCT et utilisent la norme ℓ_1 . Pour des blocs de taille 4×4 pixels, la prise en compte de la quantification améliore les performances débit-distorsion à hauts et moyens débits (cf. figure 3.11(a)). Cependant, ce n'est pas le cas à bas débits ou pour des blocs de taille plus grande comme on peut le voir sur la figure 3.11(b) pour des blocs de taille 8×8 pixels. Plusieurs raisons peuvent expliquer ces résultats décevants. La plus probable réside dans l'étape de classification rendue instable par la quantification. A bas débits, la différence entre les valeurs réelles et quantifiées est grande (en raison de la largeur des intervalles de quantification), de plus elles sont peu nombreuses. La classification repose donc sur un petit nombre de coefficients qui, d'une itération à l'autre, peuvent prendre des valeurs très différentes. Lorsque la taille des blocs augmente, ce comportement est renforcé par la contrainte de parcimonie (le rapport "nombre de coefficients non nuls sur nombre total de coefficients" diminue). De par la structure de l'algorithme, la prise en compte de la quantification ne semble donc pas pertinente. Nous ne considérerons pas cette modification dans la version finale de l'algorithme d'apprentissage utilisé dans le schéma de compression présenté dans la sous-section suivante.

3.5.3 Implémentation

L'implémentation du schéma de compression utilisant les bases apprises par l'algorithme de Sezer est similaire à celle détaillée dans la sous-section 3.4.2.

Apprentissage des bases

La base de transformation est choisie selon une procédure d'optimisation décrite dans la section 3.3 parmi un ensemble fait de la concaténation de bases locales en bintree.

Selon la taille du support (compris entre 4×4 pixels et 32×32 pixels), on utilise entre 50000 et 400000 signaux d'entraînement pour optimiser un ensemble de 6 bases avec la procédure et les améliorations décrites dans les sous-sections précédentes 3.5.1 et 3.5.2. Les bases sont initialisées avec des DDCT dont les modes directionnels correspondent aux modes de prédiction intra du standard de compression vidéo H.264 (les modes "DC", "vertical" et "horizontal" ne sont pas considérés). L'apprentissage est ainsi réalisé pour plusieurs valeurs de λ' ($\lambda' \in \{5, 16, 51, 161\}$). La base DCT est ajoutée aux bases apprises, résultant pour une taille de bloc fixée en 4 ensembles de 7 bases. A chaque point de compression global, spécifié par le paramètre μ , la valeur de λ' correspondant à l'ensemble générant les meilleures performances en débit-distorsion est codée et transmise (une même valeur de λ' est ainsi utilisée sur l'ensemble de la segmentation en bintree, *i.e.*, pour toutes les tailles de blocs).

Ordre de scanning

Contrairement aux DCT directionnelles, les ordres de scanning correspondant aux bases de Sezer ne peuvent être déterminés de façon théorique, ils sont appris sur un ensemble d'images transformées par les bases apprises puis quantifiées. L'apprentissage résulte en 168 ordres de scanning différents correspondant à chaque base, chaque λ' et chaque taille de bloc. Pour la base DCT, l'ordre de scanning "en zigzag" utilisé dans JPEG (cf. figure 1.5, sous-section 1.3.3 chapitre 1) est conservé.

Codage

L'encodage des valeurs quantifiées des coefficients transformés et des indices des coefficients non nuls est réalisé comme décrit dans la sous-section 3.4.2. De même, l'encodage des indices des bases locales se fait au moyen d'un

Taille de bloc	32×32	32×16	16×16	16×8	8×8	8×4	4×4
$\lambda' = 5$	6.90	7.0	7.10	7.30	7.50	7.20	7.30
$\lambda' = 16$	6.80	6.90	7.0	7.30	7.50	7.20	7.20
$\lambda' = 51$	6.90	7.0	7.10	7.20	7.40	7.20	7.20
$\lambda' = 161$	6.80	6.80	6.80	7.10	7.30	7.10	7.30

Table 3.2 Valeurs moyennes de γ_j par taille de blocs et valeur de λ' pour les bases apprises par l'algorithme de Sezer.

quadtree.

Analyse des facteurs de proportionnalité $\{\gamma_j\}$

Nous reprenons l'hypothèse que les facteurs γ_j ne dépendent pas des indices

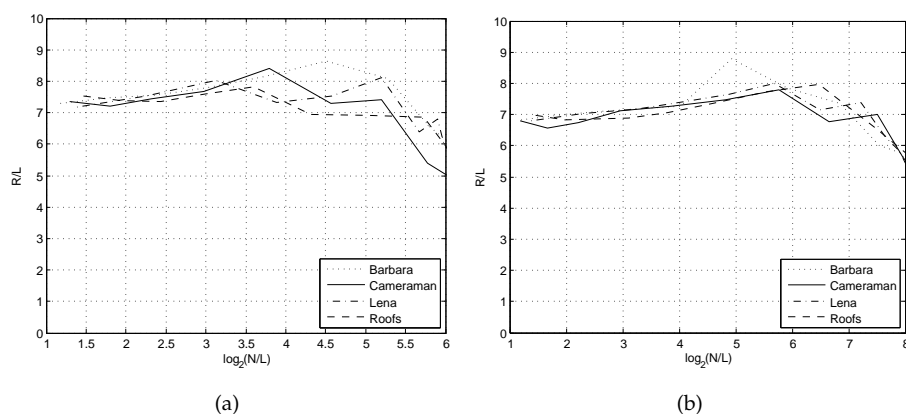


Figure 3.12 Valeurs des γ_j obtenues par compression des images "Barbara", "Cameraman", "Lena" et "Roofs" sur une segmentation en blocs de 8×8 pixels (a) et 16×16 pixels (b) pour $\lambda' = 16$.

des bases mais uniquement de la taille du support de la transformée et, ici, du paramètre λ' . Ils sont donc déterminés empiriquement par taille de blocs et par valeur de λ' , en considérant un schéma de codage identique à celui défini précédemment mais pour une segmentation en blocs de même taille. Les figures 3.12(a) et 3.12(b) illustrent graphiquement les valeurs des γ_j obtenues pour respectivement des blocs de taille 8×8 pixels et 16×16 pixels pour $\lambda' = 16$. Le tableau 3.2 résume les valeurs moyennes choisies pour les 7 tailles de blocs possibles de la segmentation en bintree proposée et les 4 valeurs de λ' choisies.

3.5.4 Evaluation des performances

Plusieurs analyses permettent de caractériser les performances atteintes par le schéma de compression proposé.

Une première comparaison des performances obtenues par les DCT directionnelles d'une part et les bases apprises par l'algorithme de Sezer d'autre part est réalisée en considérant une segmentation de l'image en blocs de même taille.

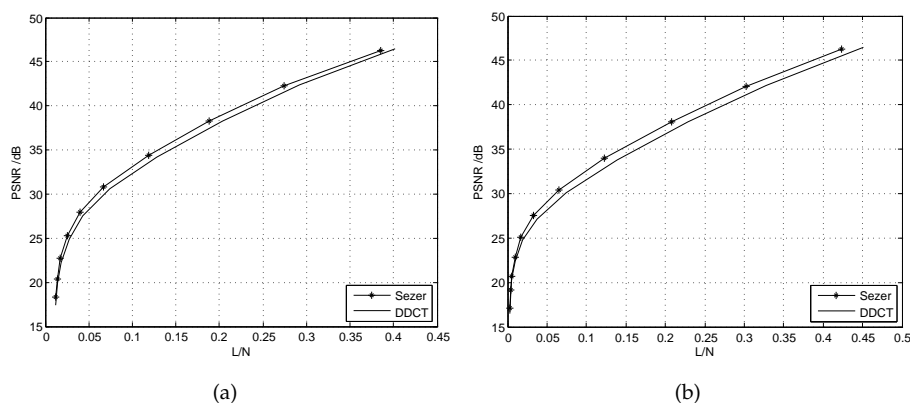


Figure 3.13 Performances parcimonie-distorsion atteintes par les deux ensembles de bases, DDCT et bases apprises par l'algorithme de Sezer, pour une segmentation de l'image "Cameraman" en blocs de taille 8×8 pixels (a) et 16×16 pixels (b).

La figure 3.13 présente les performances parcimonie-distorsion atteintes par les bases DDCT d'une part et les bases apprises par l'algorithme de Sezer d'autre part, pour un découpage de l'image "Cameraman" en blocs de taille 8×8 pixels (3.13(a)) et 16×16 pixels (3.13(b)). On observe ainsi que pour un PSNR donné, les blocs transformés par les bases de Sezer présentent une plus grande parcimonie que ceux transformés par les DCT directionnelles. L'apprentissage a donc bien favorisé la parcimonie des décompositions.

L'écart entre les deux courbes de performances parcimonie-distorsion se retrouve quoique moins prononcé sur la figure 3.14 qui présente les performances en débit-distorsion obtenues par chaque ensemble de bases.

Enfin, la figure 3.15 illustre et compare les performances débit-distorsion atteintes par le schéma de compression basé sur une segmentation en bintree et l'utilisation de bases apprises par l'algorithme de Sezer avec le même schéma utilisant des DCT directionnelles et les standards de compression JPEG et JPEG2000. Le gain apporté par l'apprentissage est dans l'ensemble

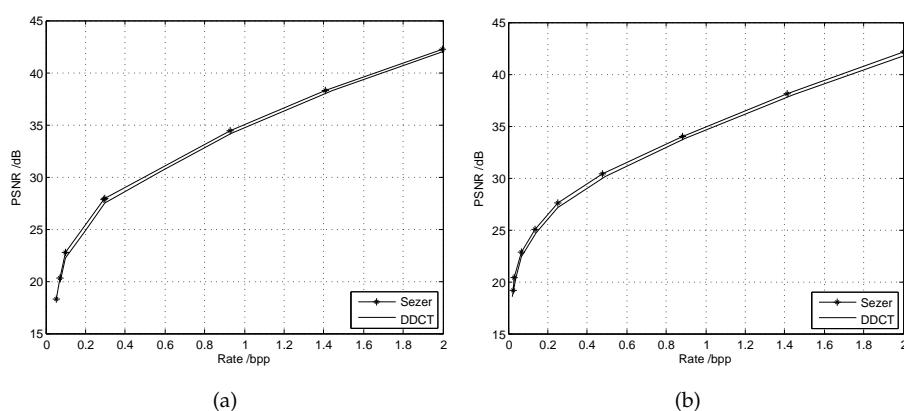


Figure 3.14 Performances débit-distorsion atteintes par les deux ensembles de bases, DDCT et bases apprises par l’algorithme de Sezer, pour une segmentation de l’image “Cameraman” en blocs de taille 8×8 pixels (a) et 16×16 pixels (b).

peu significatif, voire, dans le cas de l’image “Roofs”, inexistant. Cette dernière image présente des zones fortement orientées (cf. annexe 3.8.3), déjà très bien captées par les DCT directionnelles.

L’apprentissage de bases selon l’algorithme introduit par Sezer *et al.* a permis d’augmenter la parcimonie des vecteurs de représentation mais cela n’a pas suffi pour améliorer de façon significative les performances débit-distorsion. Plusieurs raisons peuvent être avancées : une parcimonie encore trop faible, un codage des coefficients de transformation quantifiés coûteux. Pour cette dernière raison, il peut être intéressant de comparer les coûts de codage des valeurs et indices des coefficients de transformation quantifiés pour chaque ensemble de bases, DDCT et bases apprises par l’algorithme de Sezer. La figure 3.16 représente graphiquement ces coûts en fonction de la parcimonie des décompositions pour la compression des images “Barbara”, “Cameraman”, “Lena” et “Roofs”. On peut ainsi observer que si le coût de codage des valeurs est en général plus faible pour le schéma de compression utilisant les bases apprises que celui utilisant les DDCT, c’est un comportement inverse qui régit le coût de codage des indices. Malgré l’apprentissage des ordres de scanning, le codage des indices des coefficients de transformation non nuls, réalisé par plages de zéros, est moins efficace, donc plus coûteux, que celui attaché au codage par DDCT. De par leurs constructions, les DCT directionnelles concentrent l’énergie du signal considéré sur les mêmes atomes en priorité alors qu’aucune contrainte de ce type n’est imposée par l’algorithme de Sezer aux bases optimisées. Les atomes utilisés dans les décompositions par-

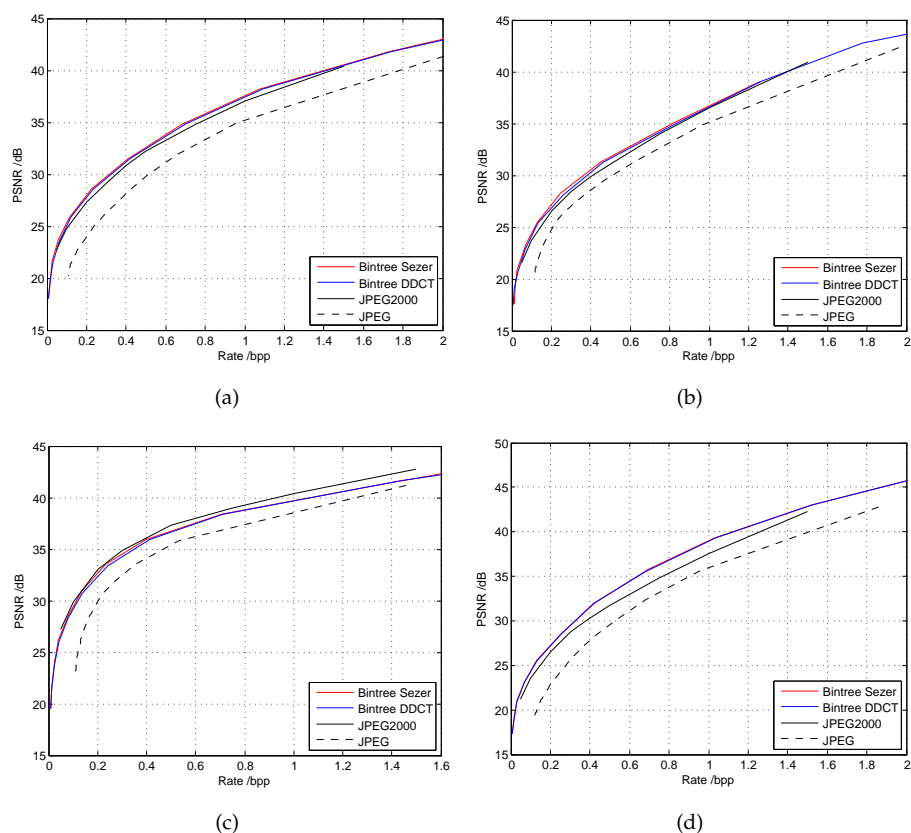


Figure 3.15 Performances débit-distorsion pour la compression de “Barbara” (a), “Cameraman” (b), “Lena” (c) et “Roofs” (d) avec le schéma de codage proposé utilisant des bases apprises par l’algorithme de Sezer, des DCT directionnelles et le standard JPEG2000 et JPEG.

cimonieuses sont alors choisis de façon “uniforme”, sans priorité. Ce handicap peut expliquer en partie les performances relativement décevantes du schéma de compression utilisant les bases apprises par l’algorithme de Sezer.

3.6 Vers une approche probabiliste

Une des raisons avancées pour expliquer les résultats peu satisfaisants de la section précédente invoque la faible amélioration de la parcimonie introduite par l’apprentissage de Sezer. Et en effet, nous avons eu l’occasion de souligner certaines faiblesses de la structure de l’algorithme, en particulier son étape de classification des signaux d’entraînement. Dans les sections 3.6 et 3.7 nous développons une nouvelle approche d’apprentissage susceptible

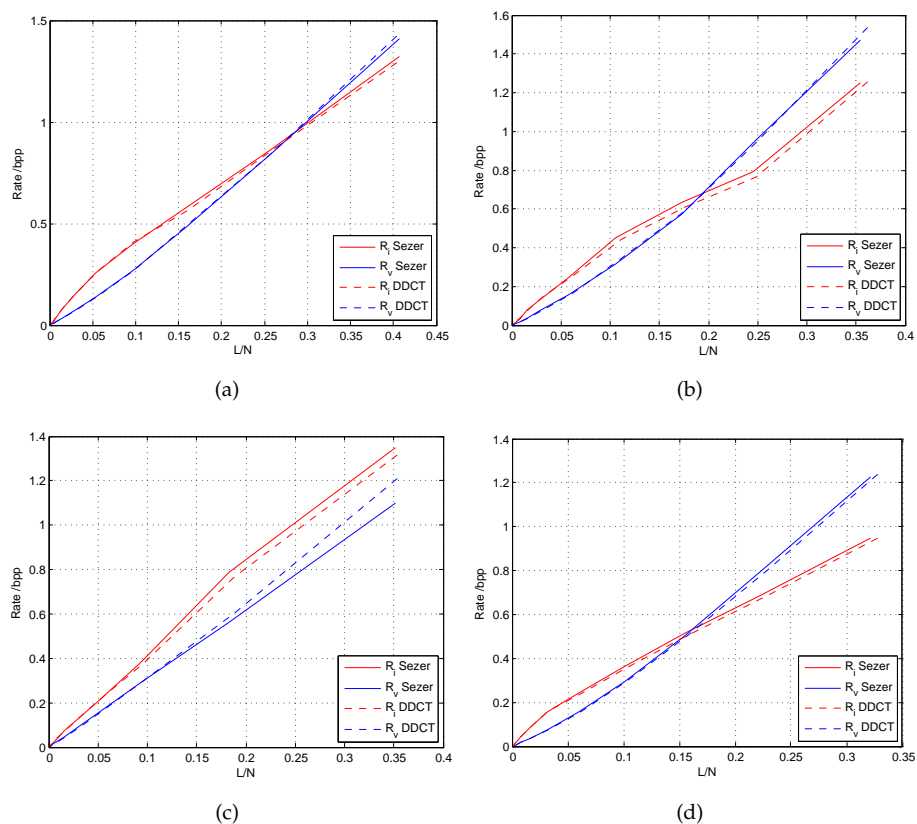


Figure 3.16 Débits R_i et R_v obtenus pour la compression de “Barbara” (a), “Cameraman” (b), “Lena” (c) et “Roofs” (d) en fonction de la parcimonie (nombre de coefficients non nuls rapporté au nombre de pixels, L/N) avec le schéma de codage proposé utilisant des bases apprises par l’algorithme de Sezer et le même utilisant des DCT directionnelles.

de résoudre les problèmes d’instabilités liés à cette opération.

La section 3.6 peut être vue comme une transition entre l’algorithme de Sezer présenté dans la section précédente et le nouvel algorithme. Nous y proposons une interprétation probabiliste de l’algorithme de Sezer et posons les bases de l’approche alternative qui fera l’objet de la section suivante.

3.6.1 Définition d’un cadre probabiliste

Dans cette section et la suivante, nous assimilerons variable aléatoire et réalisation lorsque le contexte est univoque.

On considère le modèle suivant pour chaque signal d’entraînement $\mathbf{y}_j, j \in$

$\{1, \dots, K\}$:

$$p(\mathbf{y}_j | \mathbf{D}) = \int_{\mathbb{R}^M} \sum_{c_j=1}^P p(\mathbf{y}_j | \mathbf{x}_j, \mathbf{D}, c_j) p(\mathbf{x}_j | c_j) p(c_j) d\mathbf{x}_j, \quad (3.43)$$

avec

$$p(\mathbf{y}_j | \mathbf{x}_j, \mathbf{D}, c_j = i) = \mathcal{N}(\mathbf{D}_i \mathbf{x}_{ji}, \sigma^2 \mathbf{I}_N) \quad (3.44)$$

où $\mathcal{N}(m, \Gamma)$ est une distribution Gaussienne de moyenne m et de covariance Γ , et

$$p(\mathbf{x}_j | c_j = i) \propto \exp(-\lambda \|\mathbf{x}_{ji}\|_0), \quad (3.45)$$

où $\lambda > 0$ et \propto signifie "proportionnel à"¹. Cette expression impose la parcimonie de \mathbf{x}_{ji} .

Le modèle (3.43)-(3.45) peut être interprété comme suit : chaque \mathbf{y}_j est considéré comme une combinaison bruitée de vecteurs choisis dans une unique base ; le choix de la base est indicé par c_j . La parcimonie est encouragée via la distribution a priori (3.45) qui pénalise les \mathbf{x}_{ji} avec beaucoup de coefficients non nuls. $p(\mathbf{y}_j | \mathbf{D})$ peut alors être vue comme un mélange de Gaussiennes $\mathcal{N}(\mathbf{D}_i \mathbf{x}_{ji}, \sigma^2 \mathbf{I}_N)$ où chaque élément est pondéré par un facteur dépendant de la parcimonie de \mathbf{x}_{ji} et de la probabilité a priori $p(c_j = i)$.

3.6.2 L'algorithme de Sezer revisité

Dans cette sous-section, nous montrons que l'algorithme de Sezer peut être considéré comme une implémentation particulière d'un problème au maximum a posteriori (MAP) dans le contexte probabiliste exposé dans la sous-section précédente. Posons $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_j, \dots, \mathbf{x}_K]$ une matrice dont les colonnes sont les vecteurs parcimonieux \mathbf{x}_j et $\mathbf{c} = [c_1, \dots, c_K]^T$ un vecteur formé des indices c_j . Si on fait les deux hypothèses suivantes

$$\forall i \in \{1, \dots, P\}, \forall j \in \{1, \dots, K\}, p(c_j = i) = \frac{1}{P} \text{ et } \lambda' = 2\lambda\sigma^2, \quad (3.46)$$

où P est le nombre de bases considérées et λ' le facteur Lagrangien utilisé dans l'algorithme de Sezer, alors les récursions (3.35)-(3.38) peuvent être re-

1. Remarque : (3.45) ne définit pas une distribution de probabilité "propre" puisque le facteur de normalisation est ∞ . Cependant cette "entorse" ne conduit à aucune difficulté dans la suite de la dérivation de l'algorithme.

formulées comme suit :

$$\mathbf{c}^{(k)} = \underset{\mathbf{c}}{\operatorname{argmax}} \sum_{j=1}^K \log p(\mathbf{y}_j, \mathbf{x}_j^{(k-1)}, \mathbf{D}^{(k-1)}, c_j), \quad (3.47)$$

$$(\mathbf{D}^{(k)}, \mathbf{X}^{(k)}) = \underset{(\mathbf{D}, \mathbf{X})}{\operatorname{argmax}} \sum_{j=1}^K \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}, c_j^{(k)}). \quad (3.48)$$

L'équivalence entre (3.47)-(3.48) et (3.35)-(3.38) est évidente en prenant le modèle (3.43)-(3.45) en compte.

De (3.47)-(3.48), on déduit que l'algorithme de Sezer est équivalent à une optimisation séquentielle du problème MAP suivant :

$$(\mathbf{D}^*, \mathbf{X}^*, \mathbf{c}^*) = \underset{(\mathbf{D}, \mathbf{X}, \mathbf{c})}{\operatorname{argmax}} \sum_{j=1}^K \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}, c_j). \quad (3.49)$$

Remarquons que la formulation MAP de l'algorithme de Sezer donne une connexion entre le paramètre utilisateur λ' et les paramètres "physiques" du modèle λ, σ^2 .

3.7 Bases locales apprises : un nouvel algorithme Bayésien

Cette section explore une approche alternative au problème d'estimation MAP (3.49).

La première sous-section explicite cette alternative qui est ensuite développée dans la seconde sous-section. La qualité de l'algorithme proposé est évaluée dans les dernières sous-sections à travers ses performances en reconstruction et en compromis débit-distorsion, comparées à celles obtenues par l'algorithme de Sezer.

Cette contribution a fait l'objet d'un article dans les proceedings de la conférence ICASSP 2010 [D].

3.7.1 Une approche alternative

Dans la dernière sous-section, nous avons mis en évidence le fait que l'algorithme de Sezer peut être interprété comme un algorithme itératif pour résoudre un problème d'estimation MAP joint (sur \mathbf{D} , \mathbf{X} et \mathbf{c}). Cette formulation suggère des approches alternatives pour l'optimisation du dictionnaire.

Ici, nous considérons le problème d'estimation MAP *marginalisé* suivant :

$$(\mathbf{D}^*, \mathbf{X}^*) = \operatorname{argmax}_{(\mathbf{D}, \mathbf{X})} \sum_{j=1}^K \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}), \quad (3.50)$$

où

$$p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}) = \sum_{c_j=1}^P p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}, c_j). \quad (3.51)$$

Le problème (3.50) n'a pas de solution analytique simple. Néanmoins, il peut être résolu de façon efficace au moyen d'un algorithme EM ([25]).

3.7.2 Un nouvel algorithme

L'algorithme EM ([25]) est une méthode itérative permettant de résoudre des problèmes d'estimation au sens du maximum de vraisemblance impliquant des variables cachées. L'algorithme alterne entre deux étapes. Dans l'étape E-step (pour "expectation step" en anglais), une borne inférieure sur la fonction objectif que l'on cherche à maximiser (ici, $\sum_j \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D})$, cf. (3.50)) est calculée en utilisant la valeur courante des paramètres d'intérêt en compte. La valeur des paramètres est ensuite mise à jour en maximisant la borne inférieure (constituant ainsi l'étape M-step pour "maximization step"). Dans notre cas, la variable \mathbf{c} est considérée comme cachée. Appliquées au problème (3.50), les étapes E-step et M-step peuvent être formalisées comme suit : à la k -ième itération de l'algorithme,

E-step :

$$\begin{aligned} \mathcal{Q}(\mathbf{D}, \mathbf{X}, \mathbf{D}^{(k)}, \mathbf{X}^{(k)}) &= E_{\mathbf{c} | \mathbf{Y}, \mathbf{X}^{(k)}, \mathbf{D}^{(k)}} [\log p(\mathbf{X}, \mathbf{Y}, \mathbf{c}, \mathbf{D})], \\ &= \sum_{j=1}^K \sum_{i=1}^P w_{ji}^{(k)} \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}, c_j = i), \end{aligned} \quad (3.52)$$

où $w_{ji}^{(k)} \triangleq p(c_j = i | \mathbf{y}_j, \mathbf{x}_j^{(k-1)}, \mathbf{D}^{(k-1)})$. $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_K]$ est la matrice dont les colonnes sont les signaux d'entraînement \mathbf{y}_j .

M-step :

$$(\mathbf{D}^{(k+1)}, \mathbf{X}^{(k+1)}) = \operatorname{argmax}_{(\mathbf{D}, \mathbf{X})} \mathcal{Q}(\mathbf{D}, \mathbf{X}, \mathbf{D}^{(k)}, \mathbf{X}^{(k)}). \quad (3.53)$$

Les équations du E-step (3.52) et M-step (3.53) sont particularisées au modèle (3.43)-(3.45) dans l'algorithme 6.

Il peut être intéressant de comparer les opérations réalisées par l'algorithme proposé et l'algorithme de Sezer. En particulier, l'étape de E-step (3.58) peut être vue comme une version "douce" de la classification effectuée par l'algorithme de Sezer. Tandis qu'une décision dure $c_j^{(k)}$ est prise sur les valeurs des c_j dans (3.36), l'algorithme EM calcule une probabilité a posteriori de c_j , probabilité que le vecteur \mathbf{y}_j ait une décomposition parcimonieuse dans la i -ème base. Les probabilités $w_{ji}^{(k)} = p(c_j = i | \mathbf{y}_j, \mathbf{x}_j^{(k-1)}, \mathbf{D}^{(k-1)})$ sont calculées de la façon suivante :

$$\begin{aligned} w_{ji}^{(k)} &\propto p(c_j = i, \mathbf{y}_j, \mathbf{x}_j^{(k-1)}, \mathbf{D}^{(k-1)}), \\ &\propto p(\mathbf{y}_j | c_j = i, \mathbf{x}_j^{(k-1)}, \mathbf{D}^{(k-1)}) p(\mathbf{x}_j^{(k-1)} | c_j = i, \mathbf{D}^{(k-1)}) p(c_j = i), \\ &\propto \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y}_j - \mathbf{D}_i^{(k-1)} \mathbf{x}_{ji}^{(k-1)}\|_2^2 - \lambda \|\mathbf{x}_{ji}^{(k-1)}\|_0\right) p(c_j = i). \end{aligned} \quad (3.54)$$

Il est alors facile de voir, d'après (3.36), que, si $p(c_j = i) = \frac{1}{P}$, $\forall i, \forall j$,

$$c_j^{(k)} = \underset{i}{\operatorname{argmax}} p(c_j = i | \mathbf{y}_j, \mathbf{x}_j^{(k-1)}, \mathbf{D}^{(k-1)}). \quad (3.55)$$

L'algorithme de Sezer peut être, dans ce cas, interprété comme une version seuillée de l'approche que nous proposons, basée sur l'algorithme EM.

L'étape M-step est trop complexe à calculer. On la remplace par une procédure itérative qui assure l'augmentation de la fonction $\mathcal{Q}(\mathbf{D}, \mathbf{X}, \mathbf{D}^{(k)}, \mathbf{X}^{(k)})$, résultant en un algorithme EM *généralisé* (GEM pour "generalized expectation-maximization"). En pratique, la procédure est relativement similaire à la mise à jour des bases réalisée dans l'algorithme de Sezer. Elle alterne entre deux étapes : elle calcule les représentations parcimonieuses des signaux dans chaque base par une opération de seuillage dur (cf. section 2.2.1 chapitre 2),

$$\begin{aligned} \forall i \in \{1, \dots, P\}, \forall j \in \{1, \dots, K\} \\ \mathbf{x}_{ji}^{(k,l)} = \underset{\mathbf{x}_{ji}}{\operatorname{argmin}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i^{(k,l-1)} \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_0 \right\}, \end{aligned} \quad (3.56)$$

où (l) est le numéro de l'itération dans l'étape M-step et $\lambda' = 2\lambda\sigma^2$, puis met à jour les bases. C'est dans cette dernière opération que tient la différence principale entre les deux algorithmes. Tandis que dans l'algorithme de Sezer, chaque base \mathbf{D}_i est optimisée en n'utilisant que les vecteurs contenus dans le sous-ensemble \mathcal{S}_i (cf. équation (3.40)), ici l'ensemble d'entraînement entier $\{\mathbf{y}_j\}_{j=1}^K$ est utilisé en pondérant chaque contribution des \mathbf{y}_j par w_{ji} , *i.e.*, la probabilité de choisir la base \mathbf{D}_i sachant son approximation parcimonieuse dans

 Algorithme 6: Algorithme d'apprentissage basé EM

0. Initialisation

$$\mathbf{D}^{(0)} = [\mathbf{D}_1^{(0)}, \dots, \mathbf{D}_i^{(0)}, \dots, \mathbf{D}_P^{(0)}],$$

$$\forall i \in \{1, \dots, P\}, \forall j \in \{1, \dots, K\}, \mathbf{x}_{ji}^{(0)} = \underset{\mathbf{x}_{ji}}{\operatorname{argmin}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i^{(0)} \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_0 \right\}.$$

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. E-step

$$\forall i \in \{1, \dots, P\}, \forall j \in \{1, \dots, K\},$$

$$w_{ji}^{(k)} \propto \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y}_j - \mathbf{D}_i^{(k-1)} \mathbf{x}_{ji}^{(k-1)}\|_2^2 - \lambda \|\mathbf{x}_{ji}^{(k-1)}\|_0\right) p(c_j = i). \quad (3.58)$$

2. M-step

$$\forall i \in \{1, \dots, P\},$$

$$\mathbf{D}_i^{(k)} = \underset{\mathbf{D}_i}{\operatorname{argmin}} \left\{ \sum_{j=1}^K w_{ji}^{(k)} \min_{\mathbf{x}_{ji}} \left\{ \|\mathbf{y}_j - \mathbf{D}_i \mathbf{x}_{ji}\|_2^2 + \lambda' \|\mathbf{x}_{ji}\|_0 \right\} \right\} \text{ soumis à } \mathbf{D}_i^T \mathbf{D}_i = \mathbf{I}_N. \quad (3.59)$$

cette base. Les bases sont donc calculées selon

$$\forall i \in \{1, \dots, P\}, \mathbf{D}_i^{(k,l)} = \mathbf{V}\mathbf{U}^T, \quad (3.57)$$

où $\mathbf{U}\Lambda^{1/2}\mathbf{V}^T$ est la décomposition en valeurs singulières de $\sum_{j=1}^K w_{ji}^{(k)} \mathbf{x}_{ji}^{(k,l)} \mathbf{y}_j^T$ et (l) est le numéro de l'itération dans l'étape de mise à jour des bases. Les complexités de l'approche basée EM et de l'algorithme de Sezer sont donc similaires.

3.7.3 Estimation de la variance de bruit

Le cadre probabiliste que nous avons défini présente l'avantage supplémentaire de permettre l'estimation des paramètres du modèle. Parmi ceux-ci, la variance de bruit σ^2 peut être vue comme une mesure de la différence entre les signaux réels \mathbf{y}_j et leurs approximations parcimonieuses dans les bases correspondantes. Puisque ces dernières sont réestimées à chaque itération de l'algorithme, il peut être pertinent de répercuter la modification sur la variance de bruit, *i.e.*, de la mettre à jour également. L'estimation de ce paramètre est réalisée en incluant σ^2 comme nouvelle variable inconnue dans le problème MAP (3.50), *i.e.*,

$$(\mathbf{D}^*, \mathbf{X}^*, (\sigma^2)^*) = \underset{(\mathbf{D}, \mathbf{X}, \sigma^2)}{\operatorname{argmax}} \sum_{j=1}^K \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}). \quad (3.60)$$

Les équations de l'algorithme EM sont adaptées à ce nouveau problème en ajoutant l'expression de mise à jour suivante dans l'étape M-step :

$$(\sigma^2)^{(k)} = \frac{1}{NK} \sum_{j=1}^K \sum_{i=1}^P w_{ji}^{(k)} \|y_j - \mathbf{D}_i^{(k)} \mathbf{x}_{ji}^{(k)}\|_2^2. \quad (3.61)$$

Les détails de ce calcul sont donnés dans l'annexe 3.8.2 de ce chapitre.

Dans la section suivante, nous verrons que l'estimation de la variance de bruit améliore de façon significative la convergence de l'algorithme d'apprentissage.

3.7.4 Evaluation des performances en reconstruction

Dans cette sous-section, on cherche à évaluer la capacité de l'algorithme proposé à retrouver des bases utilisées lors de la génération des données d'entraînement. Pour cela, on compare les performances de trois algorithmes :

- ◆ "Sezer" : algorithme d'apprentissage proposé dans [96] et défini dans l'algorithme 5,
- ◆ "EM" : algorithme défini dans l'algorithme 6 où l'estimation de la variance de bruit (3.61) est également implémentée,
- ◆ "EM-seuillé" : similaire à l'algorithme "EM" où l'étape E-step est remplacée par l'opération de seuillage (3.55).

Remarquons que "Sezer" et "EM-seuillé" sont similaires mais diffèrent par l'estimation de la variance de bruit implémentée dans le dernier.

Génération des données d'entraînement

On utilise des signaux synthétiques pour tester la capacité des algorithmes à retrouver les bases originales qui génèrent les données. 500 signaux d'entraînement \mathbf{y}_j de dimension $N = 16$ sont générés selon le modèle (3.43)-(3.45). On considère un ensemble de 6 matrices aléatoires orthonormales $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_6]$ générés selon une loi uniforme. Chaque base est sélectionnée avec une probabilité $p(c_j) = 1/6$. Les vecteurs \mathbf{x}_j contiennent $L = 4$ coefficients non nuls à des positions aléatoires. Les amplitudes des coefficients non nuls sont définies selon une distribution Gaussienne de moyenne nulle et de variance $\sigma_x^2 = 16$.

Initialisation des algorithmes

L'ensemble de bases $\mathbf{D}^{(0)}$ est initialisé à partir des bases originales comme

suit :

$$\forall i \in \{1, \dots, P\} \quad \mathbf{D}_i^{(0)} = \mathbf{D}_i \mathbf{M}^T, \quad (3.62)$$

où $\mathbf{M} = GS(\mathbf{I}_{16} + N(a))$, GS représente le processus de l'orthogonalisation de Gram-Schmidt et $N(a)$ représente une matrice de taille 16×16 dont les éléments sont des réalisations *i.i.d.* d'une loi uniforme sur $[-a, a]$. Cette formulation permet de contrôler l'écart de $\mathbf{D}^{(0)}$ par rapport à \mathbf{D} . On initialise les $\mathbf{x}_{ji}^{(0)}$ en résolvant le problème (3.56) avec $\mathbf{D}^{(0)}$. La variance de bruit est initialisée par $(\sigma^2)^{(0)} = (L/N)\sigma_x^2$. Enfin, on pose $\lambda = \log N$ selon le résultat établi par Donoho et Johnstone dans [29].

Evaluation des performances en reconstruction

Les performances des algorithmes sont évaluées via le taux de détections manquées (MDR pour missed-detection rate en anglais) représentant le nombre relatif d'atomes originaux qui ne correspondent à aucun atome estimé. Puisque tous les atomes sont de normes unitaires, deux atomes \mathbf{d}_1 et $\hat{\mathbf{d}}_1$ sont considérés comme "correspondants" si et seulement si $|\mathbf{d}_1^T \hat{\mathbf{d}}_1| \geq \zeta$, où ζ est fixé à 0.99. Le MDR est évalué en fonction du rapport signal-à-bruit (SNR pour signal-to-noise ratio en anglais) défini par $\text{SNR} \triangleq 10 \log((L/N)\sigma_x^2/\sigma^2)$.

Les trois algorithmes sont initialisés de la même manière et appliqués sur le même ensemble de données. Les algorithmes sont itérés 50 fois. L'étape de M-step est implémenté en itérant 10 fois entre (3.59) et (3.56).

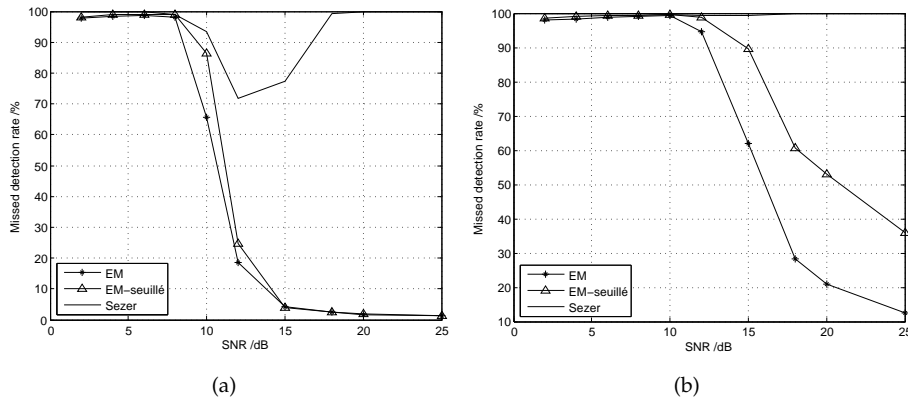


Figure 3.17 Comparaison entre les algorithmes "Sezer", "EM" et "EM-seuillé" pour différentes initialisations : $a=0.3$ (a) et $a=0.4$ (b).

Les figures 3.17(a) et 3.17(b) montrent le MDR atteint par les différents algorithmes pour $a = 0.3$ and $a = 0.4$. On peut remarquer que l'approche proba-

biliste proposée conduit à une amélioration claire des performances. Pour des hauts SNR, l'algorithme de Sezer présente des performances médiocres dues à l'étape de classification : avec une petite covariance de bruit, la contrainte de parcimonie est relâchée et augmente les erreurs de classification potentielles. Plus l'initialisation du dictionnaire est "grossière", plus les approches "EM" et "EM-seuillé" surpassent en performance de reconstruction l'algorithme de Sezer. Ainsi, pour $a = 0.4$, l'algorithme de Sezer n'arrive pas à reconstruire les bases originales, tandis que "EM" et "EM-seuillé" atteignent des MDR de respectivement 20 % et 54 % à SNR=20 dB. Enfin, d'une façon générale, les performances de l'algorithme "EM" sont supérieures à celles de l'algorithme "EM-seuillé", prouvant l'intérêt d'une classification "douce" des signaux d'entraînement.

3.7.5 Evaluation des performances en compression

Dans cette sous-section, on s'intéresse à un schéma de compression utilisant des bases apprises par l'algorithme proposé. On compare les performances en termes de parcimonie *vs* distorsion et débit *vs* distorsion de deux algorithmes :

- ◆ "Sezer" : algorithme d'apprentissage proposé dans [96] et défini dans l'algorithme 5,
- ◆ "EM" : algorithme défini dans l'algorithme 6 où l'estimation de la variance de bruit (3.61) est également implémentée.

Implémentation

Pour notre étude, nous adoptons un schéma de codage simple, basé sur une segmentation de l'image en blocs de taille fixe. Nous considérons deux tailles de blocs différentes : 8×8 pixels et 16×16 pixels. Les blocs sont transformés par une base choisie dans un ensemble appris, puis les coefficients de transformation sont quantifiés par un quantificateur scalaire uniforme avec une zone morte deux fois plus large que le pas de quantification.

Apprentissage des bases - Pour les deux tailles de support considérées, 6 bases sont apprises avec les deux algorithmes "EM" et "Sezer". L'apprentissage s'appuie sur un ensemble de 15000 signaux d'entraînement pour la segmentation en blocs de taille 16×16 pixels et 50000 signaux pour la segmentation en blocs de taille 8×8 pixels (notons que ces ensembles d'entraînement ne contiennent pas les images qui sont utilisées pour évaluer les per-

formances des algorithmes). Dans les deux algorithmes, les bases sont initialisées avec des DDCT dont les modes directionnels correspondent aux modes de prédiction intra du standard de compression vidéo H.264 (les modes “DC”, “vertical” et “horizontal” ne sont pas considérés). La base DCT est ajoutée aux bases apprises.

L'apprentissage est réalisé pour plusieurs valeurs de σ^2 . Comme nous l'avons mentionné précédemment, σ^2 peut être considérée comme une mesure de l'erreur d'approximation autorisée entre les signaux d'entraînement et leurs approximations parcimonieuses. Comme les bases apprises seront utilisées ensuite dans un schéma de compression, on peut interpréter cette erreur comme celle due à la quantification des coefficients de transformation (puisque c'est la quantification qui génère la parcimonie). Dans l'algorithme “EM”, la valeur de $(\sigma^2)^{(n)}$, estimée à chaque itération par (3.61), diminue du fait de la mise à jour des bases et des vecteurs parcimonieux. On peut alors poser σ^2 comme une borne inférieure en deçà de laquelle une amélioration de l'approximation parcimonieuse n'est pas pertinente, *i.e.*, la quantification détermine la qualité de l'approximation parcimonieuse. En pratique, dès que $(\sigma^2)^{(n)}$ atteint la valeur de σ^2 , on la fixe à cette valeur. Dans le cas de l'algorithme de Sezer, cette erreur est fixée dès le début dans le paramètre $\lambda' = 2\lambda\sigma^2$.

Si on suppose l'hypothèse de haute résolution valide (en réalité, elle ne l'est pas, comme nous l'avons vu dans la section 3.1),

$$\sigma^2 = \frac{\Delta^2}{12}, \quad (3.63)$$

où Δ est le pas de quantification choisi. 4 valeurs arbitraires de pas de quantification sont considérées : 4, 7, 13 et 23. Pour chacune de ces valeurs deux ensembles de bases sont entraînés, avec les deux algorithmes “EM” et “Sezer”. Enfin, on pose $\lambda = \log N$ (avec $N = 64$ ou $N = 256$) selon le résultat établi par Donoho et Johnstone dans [29].

Ordre de scanning - Les coefficients transformés et quantifiés sont ordonnancés selon un ordre particulier, favorisant un regroupement des coefficients non nuls. Ces ordres sont appris pour chaque taille de support, chaque σ^2 considéré et chaque base, de la même façon que dans les expérimentations menées dans la sous-section 3.5.3 avec l'algorithme de Sezer “amélioré”. Enfin, pour la DCT, l'ordre de scanning en “zigzag” classique est utilisé.

Codage - Après ordonnancement, les coefficients quantifiés de la transformation sont encodés par des codes de Huffman. Les indices des coefficients non nuls sont traités par un codage par plages de zéros (RLE). Enfin, à

chaque point de compression, la valeur de Δ correspondant à l'ensemble de bases générant les meilleures performances en débit-distorsion est codée et transmise.

Evaluation des performances en termes de parcimonie *vs* distorsion

La figure 3.18 illustre les performances en termes de parcimonie *vs* distorsion atteintes par les deux schémas de compression considérés (l'un utilisant les bases apprises par l'algorithme "EM", l'autre les bases apprises par l'algorithme "Sezer"). Les graphes des figures 3.18 (a)-(c)-(e) sont obtenues pour une segmentation de l'image à compresser en blocs de taille 8×8 pixels ; ceux des figures 3.18 (b)-(d)-(f) pour une segmentation en blocs de taille 16×16 pixels. 3 images sont considérées, "Cameraman", "Peppers" et "Roofs".

Les résultats observés sont assez décevants pour la segmentation en blocs de taille 8×8 pixels. Les bases apprises par l'algorithme "EM" améliorent certes les performances mais cette amélioration reste peu significative quelle que soit l'image compressée. En revanche, sur une segmentation en blocs de taille 16×16 pixels, le gain apporté par les bases apprises par l'algorithme "EM" est très perceptible pour les images "Cameraman" et "Peppers". Pour certaines parcimonies, l'écart entre les deux courbes de performances dépasse 1 dB. Ce n'est pas le cas pour l'image "Roofs", pour laquelle on observe des performances similaires pour les deux schémas de compression.

Plusieurs raisons peuvent être avancées pour expliquer ces résultats. Nous l'avons vu précédemment, l'une des faiblesses de l'algorithme de Sezer réside dans son étape de classification. Pour un rapport L/N donné (*i.e.*, pour une parcimonie donnée), lorsque la dimension des signaux d'entraînement augmente, le nombre de combinaisons d'atomes possibles augmente et constitue une plus grande source d'erreurs pour la classification. Le remplacement de cette étape par un calcul de probabilités a posteriori, pondérant les contributions de chaque signal dans l'optimisation des bases, permet de prendre en compte l'incertitude sur la classification des signaux et justifie l'amélioration des performances observées sur les figures 3.18 (b)-(d). Par ailleurs, les images "Cameraman" et "Peppers" présentent des caractéristiques similaires : contours orientés nets et contrastés, zones homogènes, peu ou pas texturées. L'image "Roofs", au contraire, contient beaucoup de zones fortement texturées et des contours peu contrastés. On peut imaginer améliorer ces résultats en augmentant le nombre de bases apprises, ou en catégorisant les images et adaptant l'ensemble d'entraînement à la catégorie considérée.

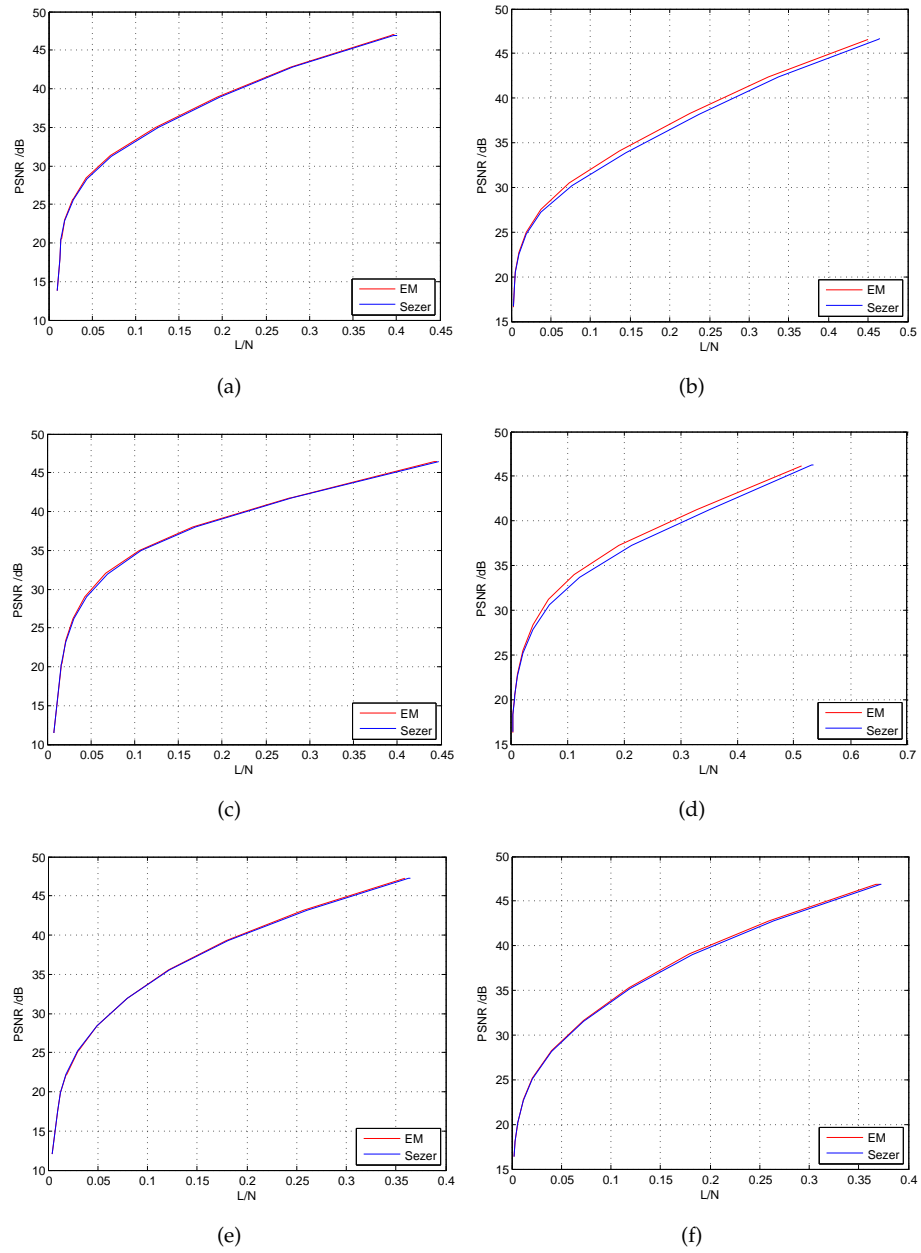


Figure 3.18 Performances parcimonie-distorsion atteintes par les ensembles de bases apprises par les algorithmes “Sezer” et “EM” pour différentes images, “Cameraman” (a)-(b), “Peppers” (c)-(d), “Roofs” (e)-(f), et pour une segmentation en blocs de tailles différentes, 8×8 pixels (a)-(c)-(e) et 16×16 pixels (b)-(d)-(f).

Evaluation des performances en termes de débit *vs* distorsion

La figure 3.19 compare les performances en termes de débit *vs* distorsion atteintes par les deux schémas de compression sur des segmentations en blocs de taille 8×8 pixels (figures 3.19 (a)-(c)-(e)) et 16×16 pixels (figures 3.19 (b)-(d)-(f)) et pour les 3 images “Cameraman”, “Peppers” et “Roofs”.

On observe que quelles que soient la taille des blocs et l’image compressée, les deux schémas de compression présentent des performances relativement similaires. Certes, la courbe rouge de l’algorithme “EM” est légèrement supérieure à la courbe bleue de “Sezer”, mais la différence n’est pas significative.

Pour expliquer ce résultat, une analyse des coûts de codage des indices et des valeurs des coefficients quantifiés peut être intéressante. La figure 3.20 illustre ces coûts en fonction de la segmentation choisie et de l’image compressée.

On constate d’abord que les coûts liés au codage des valeurs quantifiées sont très proches d’un schéma de compression à l’autre. Les algorithmes “EM” et “Sezer” reposent sur une optimisation des bases similaire, les distributions des coefficients issus de la transformation par une base apprise selon l’un ou l’autre algorithme sont donc très ressemblantes. Une plus grande différence est observable entre les coûts de codage des indices des coefficients non nuls. D’une façon générale, ce coût est toujours plus élevé pour les bases apprises par l’algorithme “EM” que pour celles apprises par l’algorithme “Sezer”. Cette augmentation est faible pour la segmentation en blocs de taille 8×8 pixels et pour l’image “Roofs” quelle que soit la segmentation considérée, mais elle est plus élevée pour la compression des images “Cameraman” et “Peppers” en blocs de taille 16×16 pixels. Ces observations expliquent les courbes de la figure 3.19 : même lorsque les performances en termes de parcimonie *vs* distorsion sont améliorées (comme “Cameraman” et “Peppers” pour une segmentation en blocs de tailles 16×16 pixels), elles sont compensées par un plus grand coût de codage des indices.

Nous nous heurtons là encore au problème d’ordonnancement des atomes. Si, dans certains cas, l’algorithme “EM” a permis d’augmenter la parcimonie des vecteurs de transformation pour une distorsion donnée, il n’a pas résolu le problème de sélection uniforme des atomes de la décomposition. Et en effet, aucune contrainte de ce type n’a été prise en compte. Nous verrons dans le chapitre 5 qu’une réponse possible à ce problème peut être apportée par l’utilisation d’un modèle Bernoulli-Gaussien.

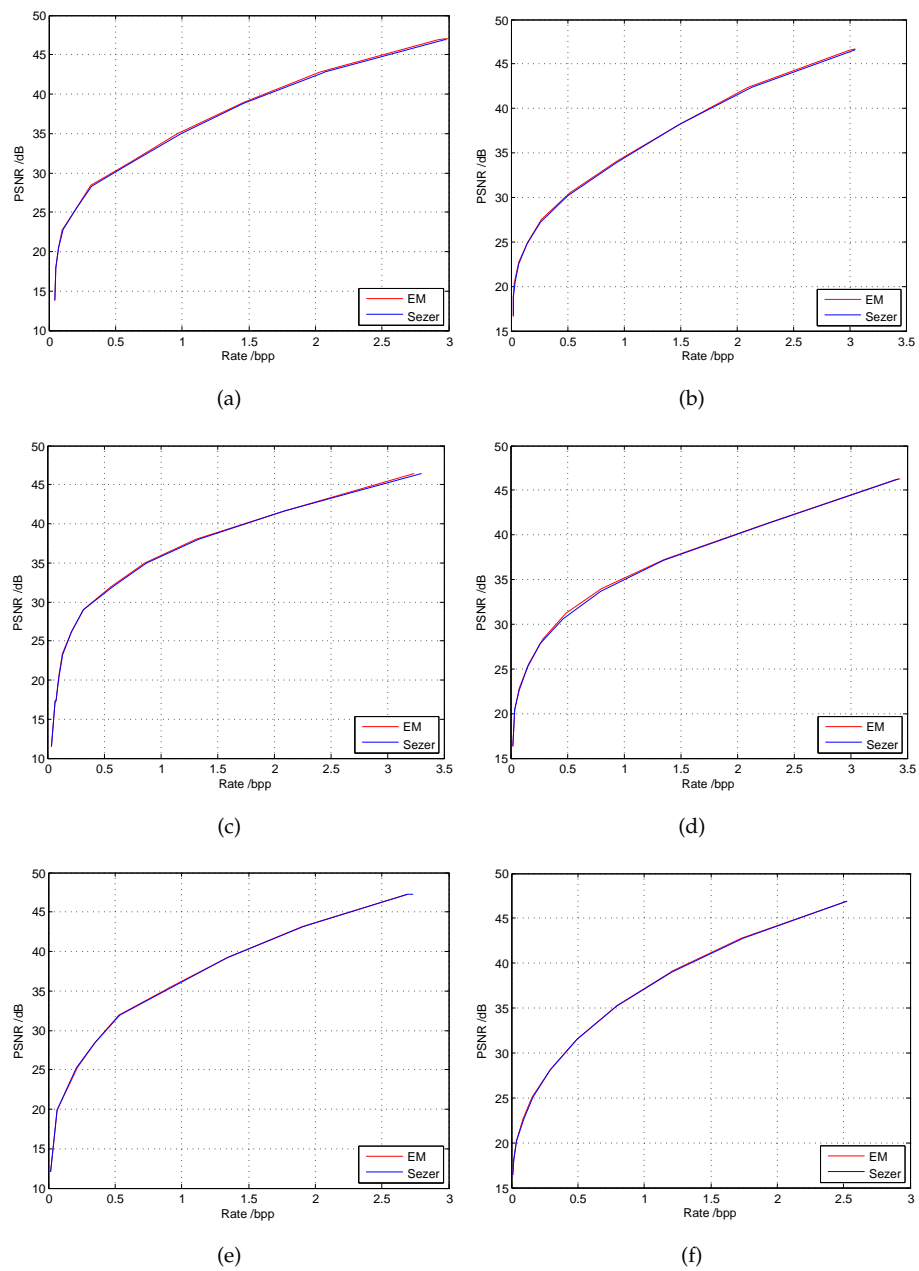


Figure 3.19 Performances débit-distorsion atteintes par les ensembles de bases apprises par les algorithmes “Sezer” et “EM” pour différentes images, “Cameraman” (a)-(b), “Peppers” (c)-(d), “Roofs” (e)-(f), et pour une segmentation en blocs de tailles différentes, 8×8 pixels (a)-(c)-(e) et 16×16 pixels (b)-(d)-(f).

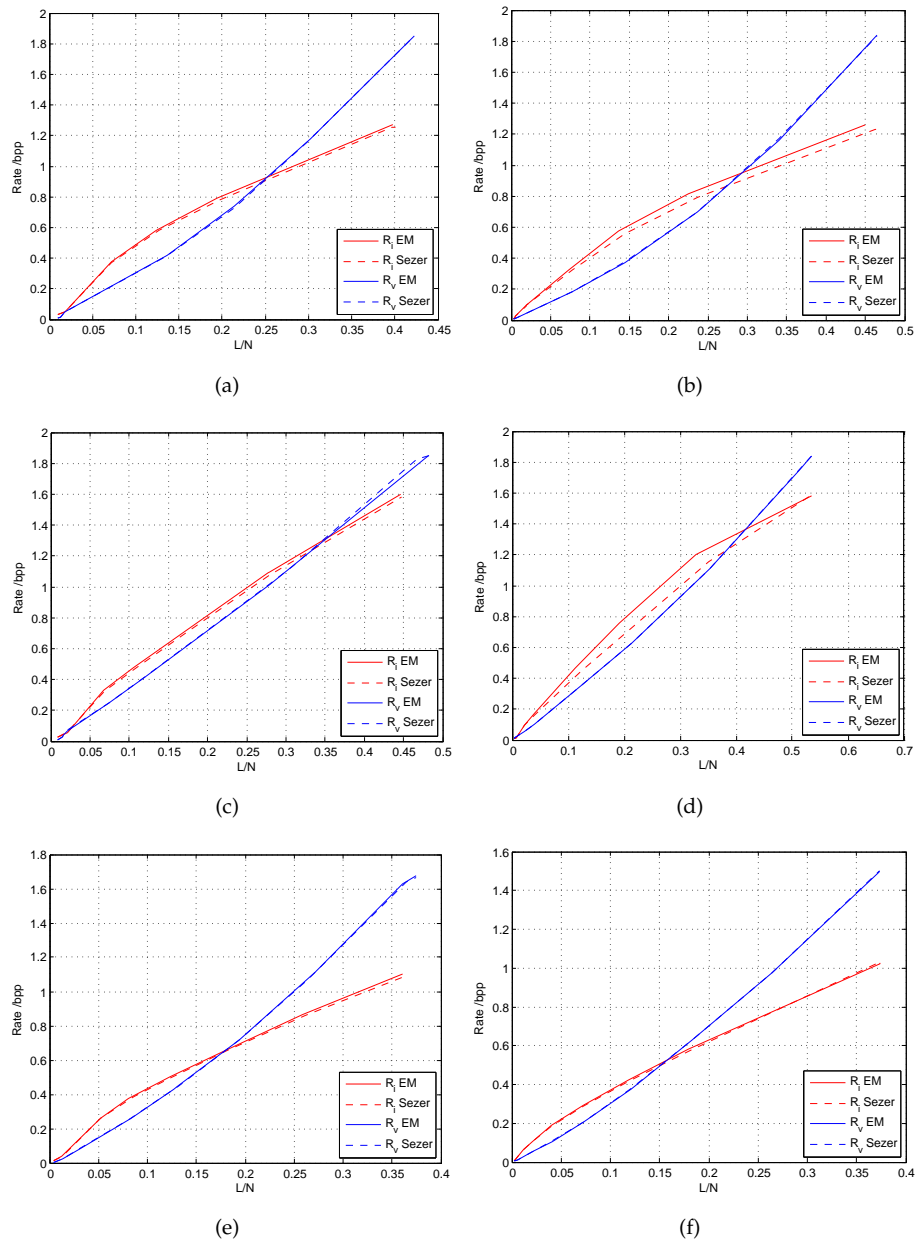


Figure 3.20 Coûts de codage des indices, R_i , et des valeurs des coefficients quantifiés, R_v , obtenus pour la compression des images "Cameraman" (a)-(b), "Peppers" (c)-(d), "Roofs" (e)-(f), avec les ensembles de bases apprises par les algorithmes "Sezer" et "EM" sur une segmentation en blocs de tailles différentes, 8×8 pixels (a)-(c)-(e) et 16×16 pixels (b)-(d)-(f).

3.8 Annexes

3.8.1 Calcul du pas de quantification optimal au sens débit-distorsion (3.15)

Reprenons le problème d'optimisation (3.14) :

$$\begin{aligned} q_{\Delta^*}(\mu) &= \operatorname{argmin}_{q_{\Delta}} \|\mathbf{x} - q_{\Delta}(\mathbf{x})\|_2^2 + \mu\gamma \|q_{\Delta}(\mathbf{x})\|_0, \\ &= \operatorname{argmin}_{q_{\Delta}} \sum_{k=1}^N (|x_k - q_{\Delta}(x_k)|^2 + \mu\gamma \|q_{\Delta}(x_k)\|_0). \end{aligned} \quad (3.64)$$

q_{Δ} est supposé scalaire uniforme avec une zone morte $T = 2\Delta$ où Δ est le pas de quantification. Dans ce cas, optimiser q_{Δ} revient à optimiser Δ pour tout $k \in \{1, \dots, N\}$.

On note $\mathcal{L}_k = |x_k - q_{\Delta}(x_k)|^2 + \mu\gamma \|q_{\Delta}(x_k)\|_0$, $\forall k \in \{1, \dots, N\}$, la fonction objectif de (3.64).

Si $q_{\Delta}(x_k) \neq 0$ (i.e., $q_{\Delta}(x_k) = (\lfloor \frac{x_k - \Delta}{\Delta} \rfloor + \frac{3}{2})\Delta$), l'erreur de quantification est telle que

$$|x_k - q_{\Delta}(x_k)|^2 \leq \frac{\Delta^2}{4}, \quad (3.65)$$

de sorte que, $\forall k \in \{1, \dots, N\}$,

$$\mathcal{L}_k \leq \frac{\Delta^2}{4} + \mu\gamma. \quad (3.66)$$

On note $\mathcal{L}_{\neq 0} \triangleq \frac{\Delta^2}{4} + \mu\gamma$.

En revanche, si $q_{\Delta}(x_k) = 0$, la fonction objectif s'écrit, $\forall k \in \{1, \dots, N\}$,

$$\mathcal{L}_k = x_k^2. \quad (3.67)$$

Puisque Δ doit être le même pour tout $k \in \{1, \dots, N\}$, la décision sur la valeur de $q_{\Delta}(x_k)$ est donc prise par la condition suivante :

$$\text{si } \mathcal{L}_k \leq \mathcal{L}_{\neq 0}, \text{ alors } q_{\Delta}(x_k) = 0, \quad (3.68)$$

et définit la zone morte

$$\frac{T}{2} = \Delta = \sqrt{\mu\gamma + \frac{\Delta^2}{4}} \quad (3.69)$$

$$\Rightarrow \Delta = \sqrt{\frac{4\mu\gamma}{3}}. \quad (3.70)$$

3.8.2 Calcul de l'équation de mise à jour de la variance de bruit (3.61)

A chaque itération k de l'algorithme d'apprentissage basé EM, la fonction que l'on cherche à maximiser dans l'étape M-step s'écrit

$$\begin{aligned} \mathcal{Q}(\mathbf{D}, \mathbf{X}, \mathbf{D}^{(k)}, \mathbf{X}^{(k)}) &= \sum_{j=1}^K \sum_{i=1}^P w_{ji}^{(k)} \log p(\mathbf{y}_j, \mathbf{x}_j, \mathbf{D}, c_j), \\ &\propto -\frac{K}{2} \log(2\pi\sigma^{2N}) \\ &+ \sum_{j=1}^K \sum_{i=1}^P w_{ji}^{(k)} \left(-\frac{1}{2\sigma^2} \|\mathbf{y}_j - \mathbf{D}_i \mathbf{x}_{ji}\|_2^2 - \lambda \|\mathbf{x}_{ji}\|_0 \right) + \log p(c_j = i). \end{aligned} \quad (3.71)$$

Dérivant l'expression par rapport à σ^2 et égalisant le résultat obtenu à zéro pour atteindre l'optimum, on obtient l'équation de mise à jour (3.61), *i.e.*,

$$(\sigma^2)^{(k)} = \frac{1}{NK} \sum_{j=1}^K \sum_{i=1}^P w_{ji}^{(k)} \|\mathbf{y}_j - \mathbf{D}_i^{(k)} \mathbf{x}_{ji}^{(k)}\|_2^2. \quad (3.61)$$

3.8.3 Images utilisées



(a)



(b)



(c)



(d)



(e)

Figure 3.21 Images utilisées pour tester les schémas de compression proposés : “Barbara” (a), “Cameraman” (b), “Lena” (c), “Roofs” (d), “Peppers” (e).

Prédiction et parcimonie

4

Dans le chapitre précédent, nous avons approfondi l'idée de parcimonie dans le codage par transformée et élaboré un schéma de compression exploitant des bases locales favorisant la parcimonie des transformées concaténées en bintree. Ainsi que nous l'avons introduit dans le chapitre 1, le codage par transformée est à mettre en parallèle du codage par prédiction, qui permet une prise en compte différente de la redondance présente dans une image. De nombreux travaux (cf. section 1.4 chapitre 1) se sont intéressés à ce type de compression, et parmi ceux-ci, certains se sont en particulier penchés sur l'intérêt des approches parcimonieuses.

Comme nous l'avons mentionné dans l'introduction du chapitre 2, il n'existe pas de lien évident entre parcimonie et codage par prédiction. A notre connaissance, toutes les méthodes proposées font l'hypothèse d'une décomposition parcimonieuse du signal constitué des données manquantes et observées dans un dictionnaire donné. Ce postulat permet de lier les deux types de données et ainsi de prédire les unes par rapport aux autres. Nous y reviendrons dans la première section de ce chapitre.

Les performances atteintes par ces approches dépassent encore de peu celles obtenues par l'algorithme de prédiction intra utilisé dans le format de compression H.264 [91]. Dans les deuxième et troisième sections de ce chapitre, nous tenterons d'améliorer ces résultats et proposerons une méthode exploitant un dictionnaire lui-même structuré en un ensemble de dictionnaires. Cependant notre étude est prospective : il serait nécessaire d'intégrer l'algorithme proposé dans un schéma de compression vidéo complet de type H.264 pour le valider.

Cette contribution fait l'objet d'un article qui sera publié dans les proceedings de la conférence MMSP en octobre de cette année [A].

4.1 Introduction

Dans cette section, nous introduisons l’approche parcimonieuse dans le problème de prédiction (ou d’inpainting, les deux problèmes se formalisant de façon similaire) puis établissons un bref état de l’art des contributions utilisant cette approche.

L’idée de prédiction basée sur des décompositions parcimonieuses repose sur l’hypothèse que les données que l’on veut prédire et les données observées ont une décomposition parcimonieuse dans un dictionnaire donné. Cette hypothèse constitue une information a priori sur le signal fait de la concaténation des données observées et manquantes. Intuitivement, elle permet de définir un lien entre les deux types de données - la décomposition parcimonieuse des données observées est également celle des données manquantes - et ainsi de prédire les unes en fonction des autres.

L’approche “standard” est formalisée de la façon suivante. Soit $\mathbf{y} = [\mathbf{y}_o^T, \mathbf{y}_m^T]^T$ la concaténation de $\mathbf{y}_o \in \mathbb{R}^{N_o}$, données observées, et $\mathbf{y}_m \in \mathbb{R}^{N_m}$, données manquantes. On suppose que \mathbf{y} a une décomposition parcimonieuse dans le dictionnaire $\mathbf{D} \in \mathbb{R}^{N \times M}$ et que l’on peut l’approcher en ne connaissant que \mathbf{y}_o . Le vecteur de décomposition parcimonieuse \mathbf{x}^* est calculé à partir des données observées comme solution des problèmes (\mathcal{P}_p^P) , (\mathcal{P}_p^A) ou (\mathcal{P}_p^R) (cf. sous-section 2.1.3 chapitre 2). Par exemple, pour le problème de régularisation (\mathcal{P}_p^R) ,

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y}_o - \mathbf{D}^o \mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0, \quad (4.1)$$

avec λ multiplicateur Lagrangien et $\mathbf{D}^o \in \mathbb{R}^{N_o \times M}$ le dictionnaire dont les lignes correspondent aux coefficients de \mathbf{y}_o , comme illustré à la figure 4.1. Le signal \mathbf{y}_m est alors prédit par $\hat{\mathbf{y}}_m^*$ défini comme

$$\hat{\mathbf{y}}_m^* = \mathbf{D}^m \mathbf{x}^*, \quad (4.2)$$

où $\mathbf{D}^m \in \mathbb{R}^{N_m \times M}$ est le dictionnaire dont les lignes correspondent aux coefficients de \mathbf{y}_m (cf. illustration de la figure 4.1).

Avec un choix de dictionnaires pertinents, beaucoup de contributions ([46, 47, 31, 37, 73, 110]) montrent qu’une telle approche offre de bonnes performances en prédiction et inpainting.

Dans [46, 47], Guleryuz considère un ensemble de bases orthonormales et propose un algorithme d’inpainting itératif estimant le signal manquant comme la moyenne des approximations parcimonieuses dans chaque base.

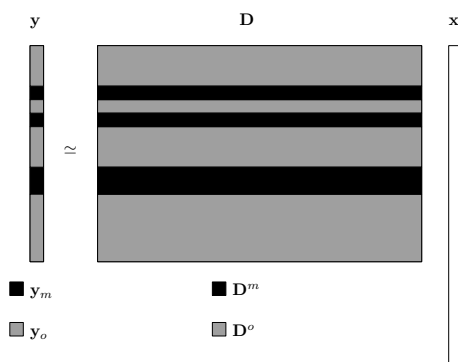


Figure 4.1 “Découpage” du dictionnaire et notations utilisées dans les méthodes de prédiction basées sur les décompositions parcimonieuses.

Nous verrons que la méthode que nous développons dans ce chapitre repose sur une idée similaire. Une autre approche est présentée par Elad *et al.* dans [31]. L’implémentation proposée utilise un dictionnaire redondant fait d’atomes bien adaptés à des zones “cartoon” et texturées. Elad *et al.* ajoutent également un terme de pénalité de variation totale (TV) au problème de décomposition parcimonieuse standard. Enfin, dans [37], Fadili *et al.* introduisent une implémentation de (4.2)-(4.1) basée sur l’algorithme EM.

Plusieurs contributions s’intéressent plus spécifiquement au problème de prédiction basée sur des décompositions parcimonieuses dans le contexte de compression d’images fixes et vidéo (cf. [73, 110]). Ces contributions se distinguent principalement par le choix du dictionnaire utilisé pour approcher de façon parcimonieuse le signal, et le choix des données utilisées comme base de la prédiction. Dans [73], Martin *et al.* considèrent un dictionnaire redondant fait de fonctions discrètes de Fourier réelles et de cosinus, tandis que Türkan *et al.* [110] construisent un dictionnaire à partir de blocs d’image pris dans une zone causale (*i.e.*, déjà décodée, connue au décodeur) large et considèrent 7 voisinages causaux possibles.

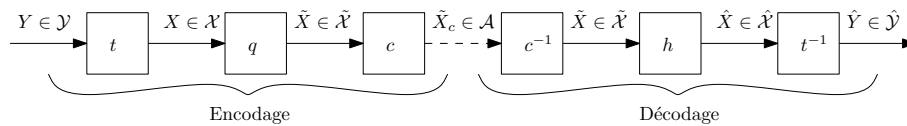
C’est dans ce contexte particulier de compression d’image, que nous proposons un nouvel algorithme de prédiction spatiale. Le but est d’améliorer les performances obtenues par le schéma (4.2)-(4.1) (que nous qualifierons de “standard” pour le différencier de notre approche) au sens débit-distorsion, mesurées après prédiction et après codage. Pour ce faire, notre approche sera donc comparée à la méthode “standard”, mais également au schéma de prédiction intra utilisé dans le format de compression vidéo H.264 [91], dont Martin *et al.* ont montré les bonnes performances dans [73].

4.2 Prédiction basée sur un mélange de décompositions parcimonieuses

Mise à part l'approche de Guleryuz dans [46, 47], la caractéristique commune aux méthodes de prédiction mentionnées dans la section précédente est l'utilisation d'un unique dictionnaire¹ dans le problème de décomposition parcimonieuse (4.1). Ici, nous considérons un ensemble de dictionnaires : on suppose que le vecteur recherché a une décomposition parcimonieuse dans un dictionnaire choisi parmi P . Nous exposons d'abord un cadre probabiliste adapté à la modélisation de telles situations puis proposons un algorithme pratique de prédiction exploitant ce contexte. Enfin, nous montrons qu'une prédiction basée sur ce modèle conduit à un mélange *pondéré* d'approximations parcimonieuses calculées dans chaque dictionnaire (se distinguant ainsi de l'approche de Guleryuz qui propose un *moyennage* des approximations parcimonieuses).

4.2.1 Définition d'un cadre probabiliste

Rappelons le schéma de la figure 1.1 :



Dans ce chapitre comme dans le précédent, les étapes du schéma de compression qui nous intéressent sont celles occupées par les opérateurs t et t^{-1} , *i.e.*, celles de structuration de la redondance présente dans l'image considérée.

Le problème "standard" de prédiction basée sur des décompositions parcimonieuses tel que formalisé par (4.2)-(4.1) fait l'amalgame entre les données observées au décodeur et les "vraies" données, constituées de blocs de l'image originale. Pour être précis, on cherche à prédire une partie de l'image originale - \mathbf{y}_m selon les notations de la section précédente - en fonction de données disponibles au décodeur, donc après transmission et décodage. Ces dernières données sont "bruitées" par rapport aux originales par les erreurs de prédiction, appelées "résidus" de prédiction, et par les erreurs de quantifi-

1. Ce dictionnaire peut être par ailleurs choisi dans un ensemble de dictionnaires ou fait d'atomes de natures différentes (par exemple, DCT, ondelettes, etc.).

cation de ces résidus produites lors de leur encodage. Ce sont ces deux sources d'incertitudes que nous tentons de prendre en compte à travers la définition d'un cadre probabiliste.

Nous adoptons les notations suivantes. Soit $\mathbf{y} = [\mathbf{y}_o^T, \mathbf{y}_m^T]^T$ le vecteur constitué des données *originales* à l'encodeur, \mathbf{y}_o disponibles et \mathbf{y}_m que l'on cherche à prédire. On pose $\mathbf{z} = [\mathbf{z}_o^T, \mathbf{z}_m^T]^T$ le pendant de \mathbf{y} au décodeur, *i.e.*, après codage et transmission, constitué de \mathbf{z}_o données observées et \mathbf{z}_m données manquantes.

On considère un ensemble de P dictionnaires $\mathcal{D} = \{\mathbf{D}_1, \dots, \mathbf{D}_P\}$ avec $\mathbf{D}_i \in \mathbb{R}^{N \times M_i}$, $\forall i$. Notons que nous nous plaçons ainsi dans un cadre plus général que celui utilisé dans les sections 3.6 et 3.7 où l'on supposait des bases orthonormées. On pose \mathcal{X} l'ensemble des vecteurs de décompositions de \mathbf{y} dans chaque dictionnaire \mathbf{D}_i , *i.e.*,

$$\mathcal{X} = \{\mathbf{x}_i\}_1^P. \quad (4.3)$$

De même que dans les sections 3.6 et 3.7, nous assimilerons variable aléatoire et réalisation lorsque le contexte est univoque.

Sur la base de ces définitions, on considère le modèle hiérarchique suivant :

$$p(\mathbf{z}|\mathbf{y}) = \mathcal{N}(\mathbf{H}\mathbf{y}, \Gamma), \quad (4.4)$$

$$p(\mathbf{y}|\mathcal{X}, c = i) = p(\mathbf{y}|\mathbf{x}_i) = \mathcal{N}(\mathbf{D}_i\mathbf{x}_i, \sigma^2 I_N), \quad (4.5)$$

$$p(\mathcal{X}|c = i) = p(\mathbf{x}_i) \propto \exp(-\lambda_i \|\mathbf{x}_i\|_0), \quad (4.6)$$

où² $\mathbf{H} \in \mathbb{R}^{N \times N}$, $\lambda_i > 0$ et $\mathcal{N}(\mu, \Sigma)$ est une distribution Gaussienne de moyenne μ et de covariance Σ . Γ est supposée diagonale avec

$$\Gamma_{jj} = \begin{cases} \sigma_o^2 & \text{si le } j\text{-ème coefficient de } \mathbf{z} \text{ est dans } \mathbf{z}_o, \\ \sigma_m^2 & \text{si le } j\text{-ème coefficient de } \mathbf{z} \text{ est dans } \mathbf{z}_m. \end{cases} \quad (4.7)$$

Le modèle (4.4)-(4.6) peut être interprété de la façon suivante. La distribution (4.4) définit le modèle d'observation, tandis que les définitions (4.5)-(4.6) formalisent les informations a priori sur \mathbf{y} . Ainsi, \mathbf{z} est observé comme une transformation linéaire bruitée de \mathbf{y} , lui-même combinaison bruitée d'atomes d'un unique dictionnaire choisi parmi P ; ce dictionnaire est indexé par c . La parcimonie de la décomposition de \mathbf{y} est encouragée par la distribution a priori (4.6) pénalisant les décompositions \mathbf{x}_i qui ont beaucoup de coefficients non nuls. La figure 4.2 représente un graphe factoriel [59] schématisant les dépendances entre les variables considérées.

2. Remarque : ici encore, (4.6) ne définit pas une distribution de probabilité "propre" puisque le facteur de normalisation est ∞ . Cependant cette "entorse" ne conduit à aucune difficulté dans la suite de la dérivation de l'algorithme.

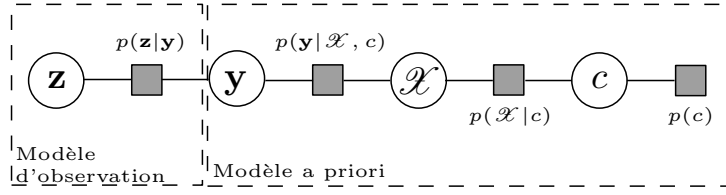


Figure 4.2 Illustration du modèle (4.4)-(4.6) et des relations entre les variables considérées.

Le modèle (4.4)-(4.6) est général. Le paramètre \mathbf{H} définit un bruit multiplicatif, par exemple dû aux appareils d'observation, et le paramètre Γ un bruit additif, comme ici par exemple, un bruit de quantification. On peut donc imaginer appliquer ce modèle dans des contextes d'utilisation plus larges, comme des problèmes d'amélioration de rendu, de débruitage - plus particulièrement de deblurring si \mathbf{H} modélise une convolution par un noyau par exemple Gaussien [65] - ou de superrésolution si \mathbf{H} est une matrice de sous-échantillonnage [70], et à d'autres types de signaux.

Dans notre cas, le seul bruit envisagé est celui dû à la prédiction et au codage des résidus de prédiction, il n'y a pas de bruit multiplicatif : $\mathbf{H} = \mathbf{I}_N$. Le contexte d'application est la prédiction spatiale du signal \mathbf{y}_m ; on pose donc $\sigma_m^2 \rightarrow +\infty$ pour modéliser l'indisponibilité de \mathbf{z}_m au décodeur.

4.2.2 Développement d'un nouvel algorithme

Dans ce contexte probabiliste, nous nous intéressons au problème d'estimation au sens de l'erreur quadratique moyenne minimale (MMSE pour minimum mean square error en anglais). Comme nous l'avons vu dans la section 2.2 du chapitre 2, l'estimation MMSE est une approche déjà exploitée dans la recherche d'approximations parcimonieuses. Et en effet, dans ce contexte, de récentes contributions [60, 95, 32] ont montré que l'estimation MMSE apportait de meilleures performances en terme de reconstruction que l'estimation MAP. Ces résultats encourageants motivent une utilisation plus générale de l'estimation MMSE dans des problèmes reposant sur des décompositions parcimonieuses. On peut ainsi espérer améliorer les performances en prédiction de la méthode "standard" (qui peut être interprétée, d'un point de vue Bayésien, comme une estimation MAP).

Plus précisément, nous nous intéressons ici à un problème d'estimation MMSE *approché*, exprimé de la façon suivante :

$$\hat{\mathbf{y}}_m^* = \underset{\hat{\mathbf{y}}_m}{\operatorname{argmin}} \int_{\mathbf{y}_m} \|\hat{\mathbf{y}}_m - \mathbf{y}_m\|_2^2 p(\mathbf{y}_m | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) d\mathbf{y}_m, \quad (4.8)$$

où

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}} \log p(\mathbf{z}, \mathcal{X}). \quad (4.9)$$

Connaissant \mathcal{X}^* , (4.8) est l'estimateur optimal au sens de l'erreur quadratique moyenne de reconstruction. L'approximation de l'estimation MMSE réside dans la maximisation sur \mathcal{X} , préférée à la marginalisation³.

La fonction objectif de (4.9) peut être réécrite comme

$$\begin{aligned} p(\mathbf{z}, \mathcal{X}) &= \sum_c \int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}, c) d\mathbf{y}, \\ &= \sum_c p(\mathbf{z}_0 | \mathcal{X}, c) p(\mathbf{z}_m | \mathcal{X}, c) p(\mathcal{X} | c) p(c), \end{aligned} \quad (4.10)$$

avec $p(\mathbf{z}_0 | \mathcal{X}, c = i) = \mathcal{N}(\mathbf{D}_i^o \mathbf{x}_i, (\sigma^2 + \sigma_o^2) \mathbf{I}_{N_o})$ et $p(\mathbf{z}_m | \mathcal{X}, c = i) = \mathcal{N}(\mathbf{D}_i^m \mathbf{x}_i, (\sigma^2 + \sigma_m^2) \mathbf{I}_{N_m})$ (ces résultats sont détaillés dans l'annexe de ce chapitre). Considérant le modèle (4.4)-(4.6), le i -ème terme de la somme sur c dépend seulement de \mathbf{x}_i . Ainsi, résoudre le problème d'optimisation joint (4.9) revient à résoudre P problèmes indépendants d'optimisation sur \mathbf{x}_i . Puisque nous avons posé l'hypothèse $\sigma_m^2 \rightarrow +\infty$, la solution de (4.9) s'exprime comme $\mathcal{X}^* = \{\mathbf{x}_i^*\}_{i=1}^P$ où

$$\forall i \in \{1, \dots, P\}, \quad \mathbf{x}_i^* = \underset{\mathbf{x}_i}{\operatorname{argmin}} \frac{1}{2(\sigma^2 + \sigma_o^2)} \|\mathbf{z}_0 - \mathbf{D}_i^o \mathbf{x}_i\|_2^2 + \lambda_i \|\mathbf{x}_i\|_0, \quad (4.11)$$

où $\mathbf{D}_i^o \in \mathbb{R}^{N_o \times M_i}$ est la restriction de \mathbf{D}_i aux lignes correspondant aux coefficients de \mathbf{z}_0 dans \mathbf{z} .

Notons que l'expression (4.11) a la forme du problème de décomposition parcimonieuse standard (\mathcal{P}_p^R). Plus précisément, \mathbf{x}_i^* peut être vu comme le vecteur de décomposition parcimonieuse de \mathbf{z}_0 dans le dictionnaire \mathbf{D}_i^o .

Par ailleurs, la théorie de l'estimation indique que la solution de (4.8) peut être réécrite comme

$$\hat{\mathbf{y}}_m^* = \int_{\mathbf{y}_m} \mathbf{y}_m p(\mathbf{y}_m | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) d\mathbf{y}_m, \quad (4.12)$$

où

$$p(\mathbf{y}_m | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) = \sum_{i=1}^P p(\mathbf{y}_m, c = i | \mathcal{X} = \mathcal{X}^*, \mathbf{z}), \quad (4.13)$$

$$= \sum_{i=1}^P p(c = i | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) p(\mathbf{y}_m | \mathcal{X} = \mathcal{X}^*, c = i, \mathbf{z}), \quad (4.14)$$

3. La marginalisation est ici tout à fait possible : si la distribution a priori $p(\mathcal{X} | c)$ est impropre, la distribution a posteriori $p(\mathcal{X} | \mathbf{y}, c)$ est, elle, bien définie. Mais par l'intégration, l'information de parcimonie disparaît, résultant en une approche peu intéressante.

avec $\mathbf{D}_i^m \in \mathbb{R}^{N_m \times M_i}$, restriction de \mathbf{D}_i aux lignes correspondant aux coefficients de \mathbf{z}_m dans \mathbf{z} . Soit, en reprenant alors le modèle (4.4)-(4.6),

$$p(\mathbf{y}_m | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) = \sum_{i=1}^P p(c = i | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) \mathcal{N}(\mathbf{D}_i^m \mathbf{x}_i^*, \sigma^2 I_{N_m}). \quad (4.15)$$

$p(\mathbf{y}_m | \mathcal{X} = \mathcal{X}^*, \mathbf{z})$ est donc un mélange de Gaussiennes et (4.12) se simplifie en

$$\begin{aligned} \hat{\mathbf{y}}_m^* &= \sum_{i=1}^P p(c = i | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) \int_{\mathbf{y}_m} \mathbf{y}_m \mathcal{N}(\mathbf{D}_i^m \mathbf{x}_i^*, \sigma^2 I_{N_m}), \\ &= \sum_{i=1}^P p(c = i | \mathcal{X} = \mathcal{X}^*, \mathbf{z}) \mathbf{D}_i^m \mathbf{x}_i^*. \end{aligned} \quad (4.16)$$

Selon les équations (4.2)-(4.1), $\mathbf{D}_i^m \mathbf{x}_i^*$ est l'estimateur de \mathbf{y}_m dans le seul dictionnaire \mathbf{D}_i . Ainsi, $\hat{\mathbf{y}}_m^*$ peut être interprété comme une combinaison pondérée des estimateurs calculés dans chaque dictionnaire. Les coefficients de pondération $p(c = i | \mathcal{X} = \mathcal{X}^*, \mathbf{z})$ donnent la probabilité que le vecteur d'observation \mathbf{z} ait été généré à partir du i -ème dictionnaire. Ces probabilités a posteriori sont calculées de la façon suivante :

$$p(c = i | \mathcal{X}^*, \mathbf{z}) \propto p(c = i, \mathcal{X}^*, \mathbf{z}), \quad (4.17)$$

$$\propto \int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}^*, c = i) d\mathbf{y}, \quad (4.18)$$

$$\propto p(\mathbf{z}_0 | c = i, \mathcal{X}^*) p(\mathbf{z}_m | c = i, \mathcal{X}^*) p(\mathcal{X}^* | c = i) p(c = i). \quad (4.19)$$

Soit, puisque $\sigma_m^2 \rightarrow +\infty$,

$$p(c = i | \mathcal{X}^*, \mathbf{z}) \propto \exp\left(-\frac{1}{2(\sigma^2 + \sigma_0^2)} \|\mathbf{z}_0 - \mathbf{D}_i^0 \mathbf{x}_i^*\|_2^2 - \lambda_i \|\mathbf{x}_i\|_0\right) p(c = i).$$

Le passage de (4.18) à (4.19) est expliqué dans l'annexe de ce chapitre.

L'implémentation des équations (4.8)-(4.9) est résumée dans l'algorithme 7.

La complexité de l'algorithme proposé est dominée par les P opérations (4.11). Elle est similaire à celle du problème de décompositions parcimonieuses (\mathcal{P}_p^R) utilisant un dictionnaire fait de la concaténation des P dictionnaires considérés.

Remarquons que si on fait l'hypothèse $P = 1$, l'équivalence entre (4.2)-(4.1) et (4.20)-(4.21) est immédiate en considérant le modèle (4.4)-(4.6). La formulation standard du problème de prédiction basé sur des décompositions parcimonieuses peut être ainsi vue comme un cas particulier de la méthode proposée ici.

Algorithme 7: Algorithme de prédiction basée sur un mélange de décompositions parcimonieuses

Etant donné $\Sigma = \sigma^2 + \sigma_o^2$:

1. Calcul des décompositions parcimonieuses dans chaque dictionnaire

$$\forall i \in \{1, \dots, P\}, \quad \mathbf{x}_i^* = \operatorname{argmin}_{\mathbf{x}_i} \frac{1}{2\Sigma} \|\mathbf{z}_o - \mathbf{D}_i^o \mathbf{x}_i\|_2^2 + \lambda_i \|\mathbf{x}_i\|_0, \quad (4.20)$$

2. Estimation du signal à prédire

$$\hat{\mathbf{y}}_m^* = \sum_{i=1}^P p(c=i | \mathcal{X}^* = \mathcal{X}^*, \mathbf{z}) \mathbf{D}_i^m \mathbf{x}_i^*, \quad (4.21)$$

$$\text{avec } p(c=i | \mathcal{X}^* = \mathcal{X}^*, \mathbf{z}) \propto \exp\left(-\frac{1}{2\Sigma} \|\mathbf{z}_o - \mathbf{D}_i^o \mathbf{x}_i^*\|_2^2 - \lambda_i \|\mathbf{x}_i\|_0\right) p(c=i). \quad (4.22)$$

4.3 Evaluation des performances

Dans cette section, on applique l'algorithme proposé au problème de prédiction intra d'image. On considère le contexte de prédiction illustré par la figure 4.3 : pour une image donnée, chaque bloc de taille 8×8 pixels (bloc blanc dans la figure 4.3) est prédit à partir des 4 blocs déjà décodés les plus proches (blocs gris dans la figure 4.3), formant le voisinage dit "causal" du bloc à prédire. On compare les performances de l'approche proposée avec deux autres algorithmes de prédiction : un schéma de prédiction similaire à celui utilisé pour la prédiction intra du format de compression vidéo H.264 [91] et le schéma de prédiction standard basée sur des décompositions parcimonieuses (4.2)-(4.1).

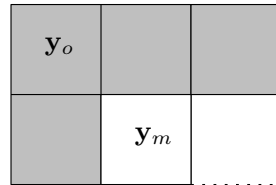


Figure 4.3 Illustration du contexte de prédiction spatiale utilisée : bloc à prédire \mathbf{y}_m et le voisinage causal considéré, \mathbf{y}_o .

Dans le reste de cette section, nous détaillons d'abord le choix des paramètres du modèle et le schéma d'encodage utilisé (sous-sections 4.3.1-4.3.2), puis illustrons les performances atteintes par les différents algorithmes (sous-section 4.3.3).

4.3.1 Paramètres du modèle

Les paramètres caractérisant le modèle (4.4)-(4.6) sont définis comme suit. On suppose que la distribution de la variable c est uniforme :

$$\forall i \in \{1, \dots, P\}, \quad p(c = i) = \frac{1}{P}. \quad (4.23)$$

Le calcul des probabilités a posteriori (4.17) se simplifie ainsi

$$p(c = i | \mathcal{X}^*, \mathbf{z}) \propto \exp\left(-\frac{1}{2\Sigma} \|\mathbf{z}_o - \mathbf{D}_i^o \mathbf{x}_i^*\|_2^2 - \lambda_i \|\mathbf{x}_i\|_0\right). \quad (4.24)$$

On utilise des DCT directionnelles introduites par Zeng et Fu dans [122] pour représenter les données de façon parcimonieuse. 7 DCT directionnelles sont générées selon les modes de prédiction H.264 (les modes "DC", "vertical" et "horizontal" sont réunis sous le mode "1", correspondant à la DCT conventionnelle). Rappelons que les DCT directionnelles sont des bases orthonormées.

On pose $\lambda_i = \log N$, $\forall i \in \{1, \dots, P\}$, en faisant l'hypothèse que le résultat établi par Donoho et Johnstone dans [29] est toujours valable lorsque l'on retire des lignes d'une base orthonormée.

Le choix de la valeur de Σ est liée très étroitement à la parcimonie des vecteurs \mathbf{x}_i^* . Or si la qualité d'approximation de \mathbf{y}_o par $\mathbf{D}_i^o \mathbf{x}_i^*$ croit lorsque la parcimonie de \mathbf{x}_i^* diminue, ce n'est pas le cas de la prédiction de \mathbf{y}_m par $\mathbf{D}_i^m \mathbf{x}_i^*$. Une information supplémentaire sur le "meilleur" niveau de parcimonie au sens de la prédiction de \mathbf{y}_m doit donc être transmise au décodeur. Dans les méthodes standard basées sur des décompositions parcimonieuses, le problème (4.1) est souvent remplacé par le problème (\mathcal{P}_0^A) :

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y}_o - \mathbf{D}^o \mathbf{x}\|_2^2 \quad \text{soumis à} \quad \|\mathbf{x}\|_0 \leq L, \quad (4.25)$$

qui peut être résolu de façon approchée par des algorithmes gloutons (cf. section 2.2 chapitre 2). L'information envoyée correspond alors au paramètre L , qui fixe le nombre de coefficients non nuls maximal de \mathbf{x} . Dans notre cas, envoyer la "meilleure" valeur de Σ est très coûteux puisque cette variable prend ses valeurs dans un ensemble continu. Par conséquent, on définit le paramètre Σ de la façon suivante. Pour un nombre de coefficients non-nuls donnés L , les vecteurs parcimonieux sont estimés en résolvant

$$\forall i \in \{1, \dots, P\}, \quad \mathbf{x}_i^* = \underset{\mathbf{x}_i}{\operatorname{argmin}} \|\mathbf{z}_o - \mathbf{D}_i^o \mathbf{x}_i\|_2^2 \quad \text{soumis à} \quad \|\mathbf{x}_i\|_0 \leq L, \quad (4.26)$$

puis Σ est calculé en fonction des \mathbf{x}_i^* comme

$$\Sigma = \frac{1}{N_o} \sum_{i=1}^P \frac{1}{P} \|\mathbf{z}_o - \mathbf{D}_i^o \mathbf{x}_i^*\|_2^2, \quad (4.27)$$

où N_o est le nombre de pixels dans \mathbf{z}_o .

Ainsi, définir Σ revient à connaître L , fixant le nombre de coefficients non nuls des \mathbf{x}_i^* (notons que nous considérons ici un même L pour tous les \mathbf{x}_i^* , $i \in \{1, \dots, P\}$). Dans le but de maximiser la qualité de reconstruction de \mathbf{y}_m , ce nombre d'itérations est ensuite optimisé sous un critère sur le bloc à prédire. Nous précisons dans les sous-sections suivantes cet aspect de l'implémentation (en particulier, deux types de critères sont considérés).

4.3.2 Schéma de prédiction et d'encodage

Pour initialiser la prédiction, la première rangée et la première colonne de blocs de taille 8×8 pixels sont encodées avec l'algorithme JPEG.

Remplaçant l'étape (4.20) par les deux étapes (4.26)-(4.27), on utilise l'algorithme OMP pour calculer les décompositions parcimonieuses \mathbf{x}_i^* dans l'approche proposée; de même dans l'approche standard basée sur des décompositions parcimonieuses pour résoudre (4.25). Le nombre d'itérations de l'algorithme glouton est optimisé sur l'intervalle $\{1, \dots, L_{max}\}$ selon deux critères différents :

1. Minimisation de l'erreur entre le bloc original et le bloc prédit correspondant,
2. Minimisation de l'erreur entre le bloc original et le bloc prédit incrémenté du résidu quantifié, codé puis transmis, sous contrainte de débit.

Le *premier* critère se formalise de la façon suivante : soit L^* le nombre optimal de coefficients non nuls dans chaque \mathbf{x}_i^* ,

$$L^* = \underset{L}{\operatorname{argmin}} \|\mathbf{y}_m - \hat{\mathbf{y}}_m^*(L)\|_2^2, \quad (4.28)$$

où $\hat{\mathbf{y}}_m^*(L)$ est prédit selon (4.2)-(4.1) (approche "standard") ou (4.20)-(4.22) (approche proposée), en fonction de $L \in \{1, \dots, L_{max}\}$. Par simplicité, nous appellerons ce critère "Minimisation MSE". Notons que le calcul de l'erreur se fait sur le bloc *prédit*.

Le *deuxième* nécessite l'introduction de nouvelles variables. On note $\mathbf{r}_m(L) = \mathbf{y}_m - \hat{\mathbf{y}}_m^*(L)$, le résidu entre le bloc original \mathbf{y}_m et le bloc prédit

$\hat{\mathbf{y}}_m^*(L)$, et $\hat{\mathbf{r}}_m(L)$ l'approximation (dûe à la quantification) de $\mathbf{r}_m(L)$ transmise au décodeur. Le bloc prédit incrémenté du résidu quantifié, codé puis transmis correspond au signal \mathbf{z}_m non-disponible au décodeur. On note, en fonction de L , $\mathbf{z}_m(L) = \hat{\mathbf{y}}_m^*(L) + \hat{\mathbf{r}}_m(L)$. On utilise pour l'estimation du débit le résultat de Mallat [68], montrant la relation de proportionnalité entre le débit et le nombre de coefficients de transformation quantifiés non nuls. Le deuxième critère se formalise finalement en

$$L^* = \underset{L}{\operatorname{argmin}} \|\mathbf{y}_m - \mathbf{z}_m(L)\|_2^2 + \lambda R(L), \quad (4.29)$$

où $R(L)$ est le coût de codage du résidu de prédiction $\mathbf{r}_m(L)$. Le multiplicateur Lagrangien λ fixant le compromis entre distorsion et débit est relié au pas de quantification du résidu selon l'expression (3.15) explicitée dans la sous-section 3.1.1 du chapitre 3. Par la suite, nous appellerons ce critère "Optimisation R-D".

Le nombre L^* est ensuite encodé par un code de Huffman.

Enfin, à chaque prédiction de bloc, le résidu $\mathbf{r}_m(L^*)$ entre le bloc original et sa prédiction est d'abord transformé par une DCT conventionnelle puis quantifié par une quantification scalaire uniforme. Notant t l'opérateur de transformation par DCT et q le quantificateur scalaire uniforme, on peut approcher [69] $R(L)$ par

$$R(L) \sim \gamma q(t(\mathbf{r}_m(L))), \quad (4.30)$$

avec $\gamma = 5.5$.

4.3.3 Analyse des performances

Nous évaluons les performances de notre algorithme de prédiction pour $L_{max} = 8$. Pour ce faire, nous le comparons à trois autres algorithmes :

- ♦ "H.264" correspond à une méthode de prédiction spatiale similaire à celle utilisée dans l'algorithme H.264 sur des blocs de 4×4 pixels (cf. sous-section 1.4.3 chapitre 1), mais appliquée ici à des blocs de taille unique, 8×8 pixels ; les 8 premiers modes de prédiction sont considérés ici (cf. sous-section 1.4.2 chapitre 1), le mode 8 étant dans notre cas "non-causal".
- ♦ "Standard $L_{max} = 8$ " qualifie la prédiction basée sur des décompositions parcimonieuses (4.2)-(4.1) avec un dictionnaire \mathbf{D} fait de la concaténation

des 7 DDCT générées selon les modes de prédiction de H.264 (cf. sous-section 4.3.1), *i.e.*, $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_7]$ où \mathbf{D}_i est une DCT directionnelle, et $L_{max} = 8$. Le coût de codage du degré de parcimonie est, dans ce cas, similaire à celui de l'algorithme que nous proposons. En revanche l'approximation parcimonieuse est évaluée sur un ensemble de combinaisons d'atomes possibles plus petit de sorte que sa qualité peut en être affectée.

- ◆ "Standard $L_{max} = 56$ " qualifie la prédiction basée sur des décompositions parcimonieuses (4.2)-(4.1) avec un dictionnaire \mathbf{D} fait de la concaténation des 7 DDCT générées selon les modes de prédiction de H.264 et $L_{max} = 56$. Dans ce cas, la dimension maximale des sous-espaces considérés dans le calcul de l'approximation parcimonieuse est identique à celle utilisée dans l'algorithme proposé. Mais c'est aux dépens du coût de codage du degré de parcimonie, possiblement supérieur.

On note "Mélange $L_{max} = 8$ " l'algorithme proposé avec $L_{max} = 8$. La valeur de L_{max} est ici choisie de façon à s'approcher du coût de spécification des modes de prédiction utilisés dans la méthode "H.264".

Les deux critères d'optimisation de la parcimonie de la décomposition de \mathbf{y} , (4.28) et (4.29), sont analysés séparément. Pour chacun, les performances des quatre algorithmes ci-dessus sont évaluées en termes d'erreur de prédiction (*i.e.*, $\|\mathbf{y}_m - \hat{\mathbf{y}}_m^*(L)\|_2^2$ avec les notations précédentes) *vs* débit, et en termes de distorsion finale (*i.e.*, $\|\mathbf{y}_m - \mathbf{z}_m(L)\|_2^2$ avec les notations précédentes) *vs* débit.

Critère "Minimisation MSE"

On s'intéresse dans un premier temps aux performances en prédiction seule. La figure 4.4 donne les courbes PSNR en prédiction *vs* débit pour chacun des quatre algorithmes "H.264", "Standard $L_{max} = 8$ ", "Standard $L_{max} = 56$ " et "Mélange $L_{max} = 8$ ". Si la méthode proposée se montre toujours meilleure que les deux méthodes "Standard" - jusqu'à un gain de 0.7 dB environ pour "Cameraman" - la comparaison avec l'algorithme "H.264" est en revanche moins tranchée. Pour certaines images comme "Barbara", l'approche proposée conduit à une meilleure prédiction - jusqu'à 0.4 dB à bas débits - pour d'autres, comme "Cameraman" ou "Roofs", la méthode "H.264" est la meilleure - jusqu'à 1 dB environ à hauts débits. Ces deux dernières images présentent en effet soit des zones homogènes ("Cameraman") avec des contours très contrastés et droits, soit des zones très texturées mais orientées

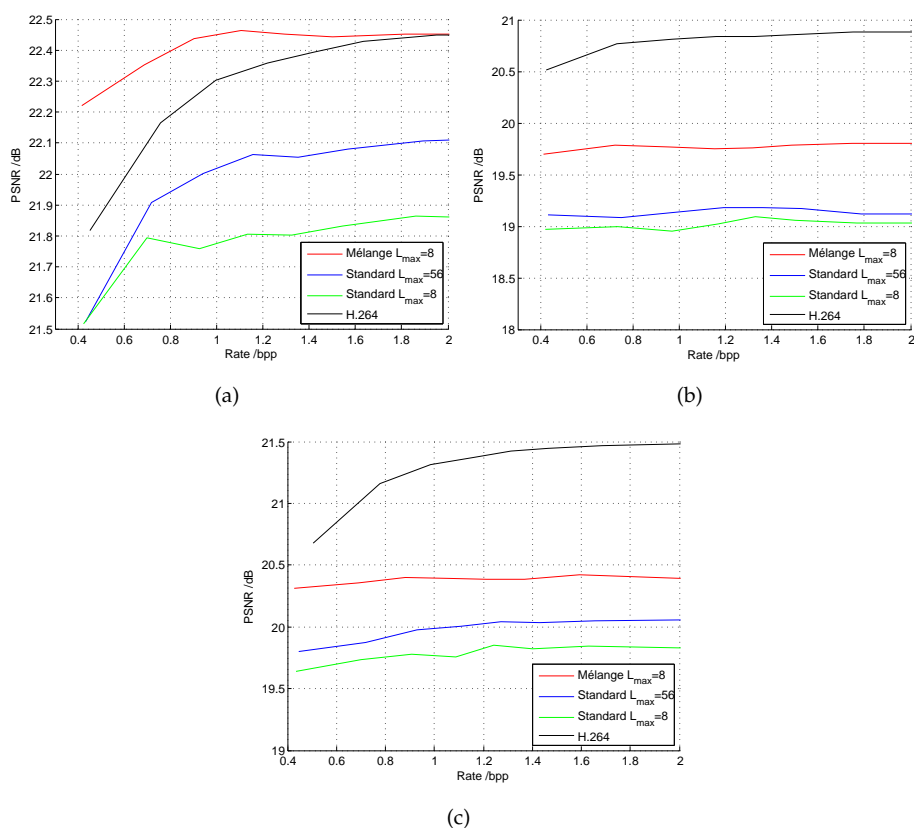


Figure 4.4 PSNR de prédiction vs débit pour “Barbara” (a), “Cameraman” (b) et “Roofs” (c).

de façon uniforme, *i.e.*, dans la même direction (les tuiles des toits de “Roofs”). Une prédiction basée sur les seuls pixels environnant le bloc à prédire, telle que la réalise l’algorithme “H.264”, constitue alors une méthode de prédiction efficace. Ce n’est pas le cas pour “Barbara” qui présente des zones plus diversifiées, avec des textures plus complexes et orientées différemment (les plis du foulard rayé, la nappe à carreaux, etc.).

Ces observations numériques sont également perceptibles visuellement sur les images prédites. Sur l’exemple de “Cameraman” donné en figure 4.5, on distingue ainsi une légère amélioration de la prédiction des structures géométriques lorsque l’on passe de $L_{max} = 8$ à $L_{max} = 56$ avec l’algorithme “Standard” : le pied de la caméra est mieux rendu. Toutefois cette amélioration ne permet pas d’atteindre le résultat obtenu par l’algorithme proposé où le pied de la caméra et le bras du “Cameraman” par exemple sont encore mieux reconstruits. Cependant, la supériorité de l’algorithme “H.264” est indiscu-

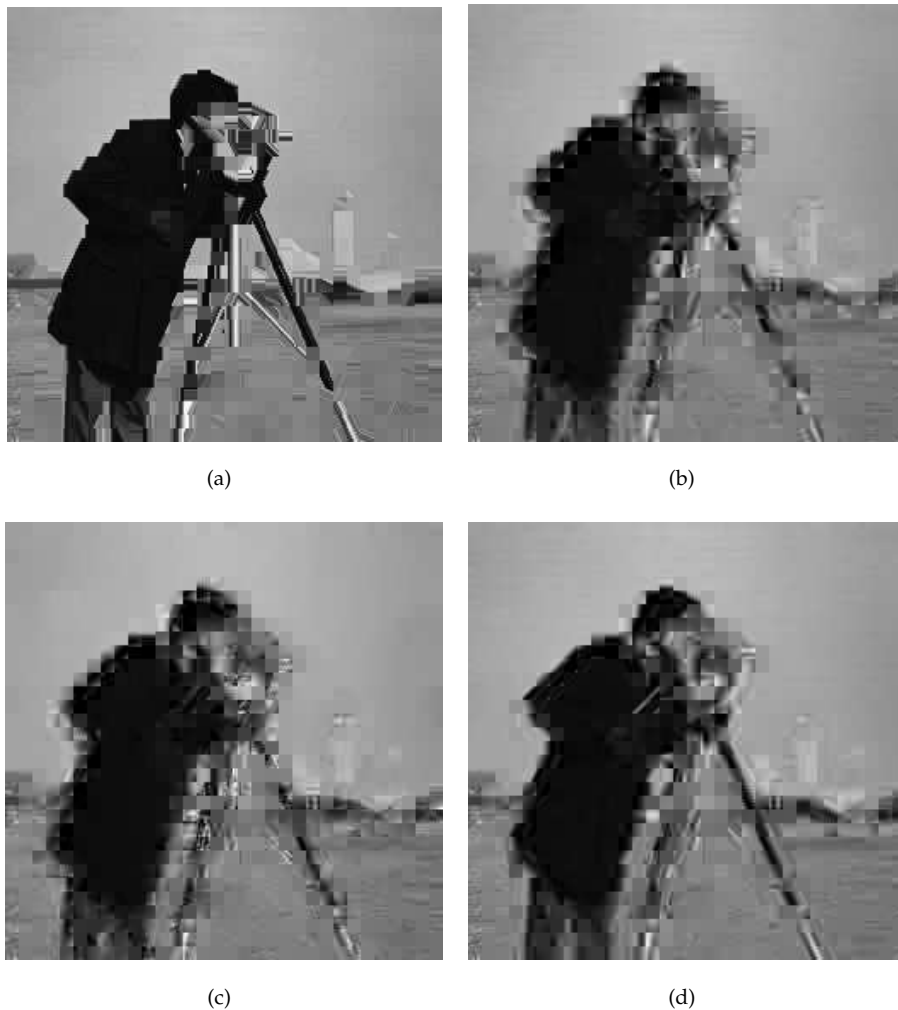
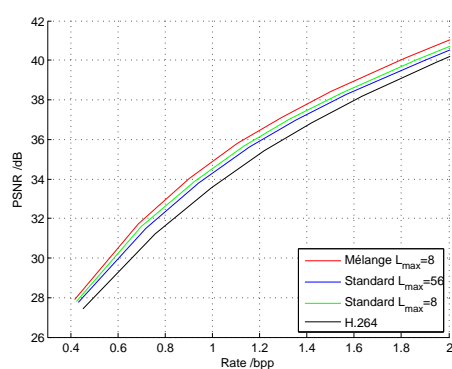


Figure 4.5 Résultat de la prédiction spatiale sur "Cameraman" avec la méthode de prédiction de type H.264 (a), les méthodes de prédiction "Standard" basée sur des décompositions parcimonieuses avec $L=8$ (b) et $L=56$ (c), et la méthode proposée avec $L=8$ (d), à 1.6 bpp environ.

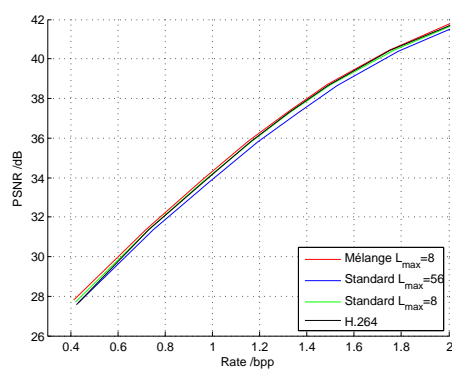
table : il conduit au rendu le plus fidèle, présentant des contours plus francs et mieux dessinés.

La figure 4.6 présente sur trois images considérées les performances PSNR final *vs* débit obtenues par les quatre algorithmes. Les performances sont comparées sous la forme de graphes et par leurs métriques de Bjontegaard calculées sur la référence de l'algorithme "H.264". Ces métriques, proposées par Bjontegaard en 2001 [7], permettent de calculer le gain moyen en PSNR et le pourcentage moyen en débit "économisé" entre deux courbes débit-



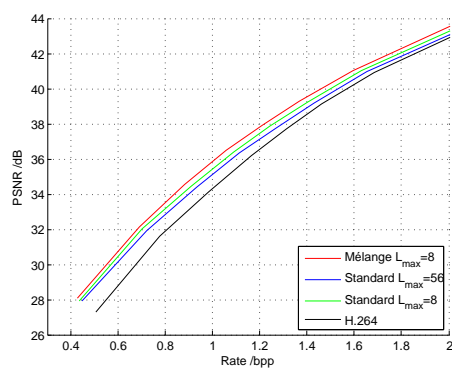
(a)

	dB	%
Mélange $L_{max} = 8$	1.05	-9.93
Standard $L_{max} = 56$	0.54	-5.13
Standard $L_{max} = 8$	0.75	-7.14



(b)

	dB	%
Mélange $L_{max} = 8$	0.15	-1.60
Standard $L_{max} = 56$	-0.21	2.08
Standard $L_{max} = 8$	-0.01	-0.03



(c)

	dB	%
Mélange $L_{max} = 8$	1.23	-10.55
Standard $L_{max} = 56$	0.66	-5.81
Standard $L_{max} = 8$	0.94	-8.09

Figure 4.6 PSNR final vs débit pour "Barbara" (a), "Cameraman" (b) et "Roofs" (c) : courbes et gains de Bjontegaard correspondants.

distorsion. Rappelons que pour les performances considérées, le PSNR est me-

suré sur l'image *finale*, obtenue au décodeur après prédiction, puis codage et transmission du résidu de prédiction.

D'une façon générale, la méthode proposée présente un bon comportement vis à vis des trois autres. On observe ainsi un gain de Bjontegaard de près de 1,23 dB (-10,55 % en débit) par rapport à l'algorithme "H.264" pour l'image "Roofs", tandis que les algorithmes "Standard $L_{max} = 8$ " et "Standard $L_{max} = 56$ " présentent un gain moindre - quelques 0,94 dB (-8,09 % en débit) apportés par l'algorithme "Standard $L_{max} = 8$ " pour "Roofs". Pour l'image "Cameraman" cependant, les courbes sont resserrées, avec des écarts peu significatifs. Si on observe toujours un léger avantage pour la méthode proposée - avec un gain de Bjontegaard de 0,15 dB (-1,60 % en débit), les méthodes "Standard" se trouvent dominées (de peu) par la méthode "H.264" - avec un déficit de près de -0,21 dB (2,08 % en débit) en métrique de Bjontegaard pour la méthode "Standard $L_{max} = 56$ ". Enfin, remarquons que quelle que soit l'image considérée, l'algorithme "Standard $L_{max} = 8$ " domine toujours l'algorithme "Standard $L_{max} = 56$ ", mais de peu (quelques 0.2 dB à moyens débits pour "Roofs").

Ces observations soulignent la complexité des schémas de compression par prédiction. De bonnes performances en prédiction n'assurent pas nécessairement de bonnes performances finales, en termes de PSNR sur l'image décodée *vs* débit. L'algorithme "H.264" en est une illustration : sur l'image "Roofs", il offre de très bonnes performances en prédiction (cf. figure 4.4), qui ne se retrouvent pas sur les courbes PSNR final *vs* débit de la figure 4.6. De dominant sur la courbe de la figure 4.4, il se trouve dominé par les deux méthodes "Standard" et "Mélange $L_{max} = 8$ ". Il en va de même pour l'algorithme "Standard $L_{max} = 56$ " qui présente une courbe de performance inférieure à celle de l'algorithme "Standard $L_{max} = 8$ " sur tous les graphes de la figure 4.6 alors même que ses performances en prédiction sont meilleures (cf. figure 4.4), et ce quelle que soit l'image considérée. Pour un débit donné, la reconstruction du résidu de prédiction au décodeur est de moins bonne qualité pour "Standard $L_{max} = 56$ " que pour "Standard $L_{max} = 8$ " : une plus grande partie du débit est allouée à la prédiction (en raison de l'augmentation du coût de codage de L^*) et même si cela permet une meilleure prédiction, les performances finales n'en sont pas améliorées.

Le deuxième critère d'optimisation de la parcimonie de la décomposition de y , "Optimisation R-D", permet une prise en compte du codage du résidu lors de la prédiction et devrait donc apporter de meilleures performances finales.

Critère "Optimisation R-D"

Puisque le critère "Optimisation R-D" ne minimise pas l'erreur de prédiction, les performances en prédiction des trois algorithmes seront moins bonnes, d'une façon générale, que celles observées précédemment avec le critère "Minimisation MSE".

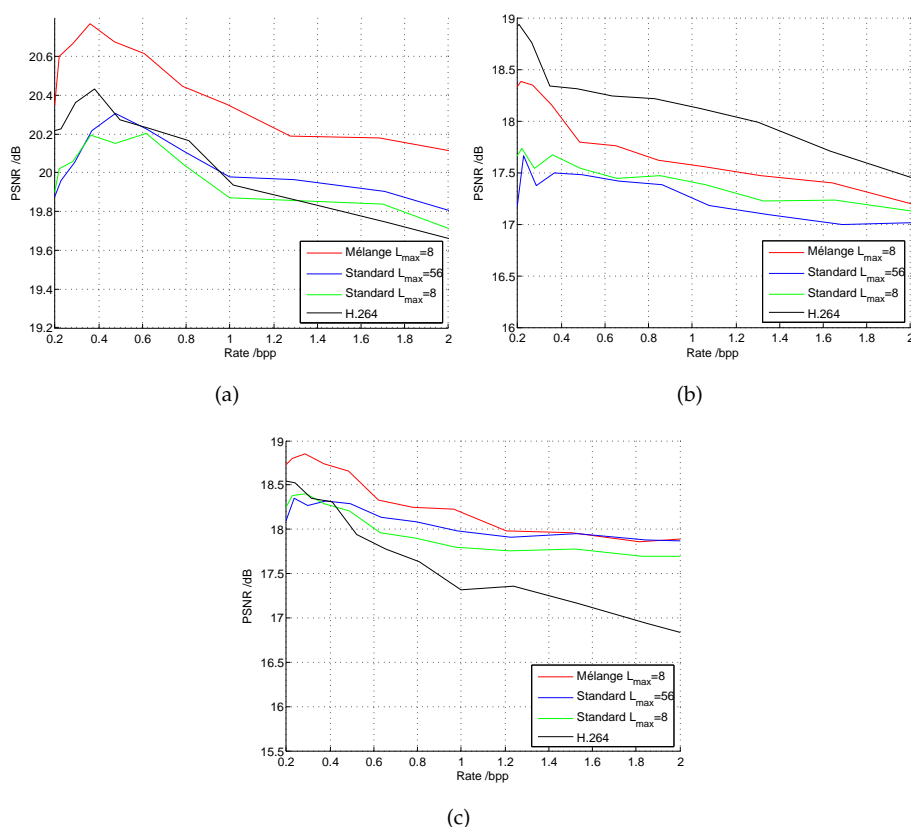
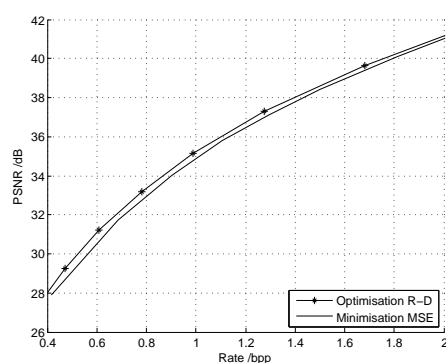


Figure 4.7 PSNR de prédiction vs débit pour "Barbara" (a), "Cameraman" (b) et "Roofs" (c).

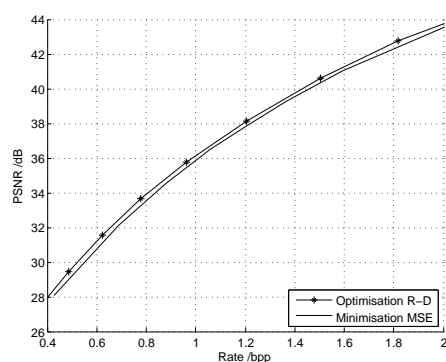
La figure 4.7 représente les courbes PSNR en prédiction *vs* débit pour chaque algorithme et pour les images "Barbara", "Cameraman" et "Roofs". On constate d'abord que comme nous l'avons prévu, les PSNR atteints ne dépassent pas 21 dB pour "Barbara" (contre 22.5 dB avec le critère "Minimisation MSE", cf. figure 4.4), et 19.5 dB pour "Cameraman" et "Roofs" (contre 21 dB avec le critère "Minimisation MSE"). L'allure des courbes est ensuite beaucoup moins monotone et leurs positions les unes par rapport aux

autres moins généralisables que celles de la figure 4.4. L'algorithme "Mélange $L_{max} = 8$ " semble toujours apporter de meilleures performances que les algorithmes "Standard", et ce quelle que soit l'image considérée. En revanche, les algorithmes "H.264" et "Standard" ont des comportements plus chaotiques : pour "Barbara", "Standard $L_{max} = 8$ " présente les plus mauvaises performances, pour "Cameraman", c'est "Standard $L_{max} = 56$ ", et pour "Roofs", "H.264".



(a)

	dB	%
"Optimisation R-D"	0.80	-5.09



(b)

	dB	%
"Optimisation R-D"	0.31	-3.09

Figure 4.8 Comparaison des deux critères en termes de PSNR final *vs* débit pour "Barbara" (a) et "Roofs" (b) : courbes et gains de Bjontegaard.

Les figures 4.8 et 4.9 considèrent les performances PSNR final *vs* débit, mesurées après codage et transmission des résidus de prédiction au décodeur. Ainsi que nous l'attendions, le critère "Optimisation R-D" permet une amélioration sensible des performances. Ceci est illustré sur la figure 4.8 pour la compression des images "Barbara" et "Roofs" par la méthode proposée.

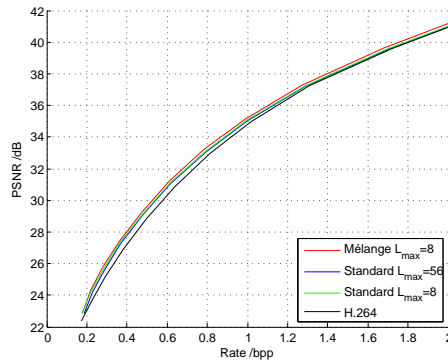
Pour l'image "Barbara", le critère "Optimisation R-D" permet d'obtenir un gain de Bjontegaard de 0.8 dB (-5.09 % en débit) par rapport au critère "Minimisation MSE".

Cependant l'amélioration apportée par le critère "Optimisation R-D" concerne les trois algorithmes. Comme on peut le voir sur la figure 4.9, si la méthode proposée surpasse toujours les autres algorithmes pour la compression des images "Barbara" et "Roofs", l'écart entre les courbes est moins important que sur la figure 4.6 avec un gain de Bjontegaard de 0.71 dB (-7.19 % en débit), obtenu pour "Roofs", par rapport à l'algorithme "H.264". Les deux méthodes "Standard" présentent des performances tout à fait similaires, leurs courbes de performances sont pratiquement superposées, avec un léger avantage cependant pour "Standard $L_{max} = 8$ ". Le débit total est la somme du coût de codage de la prédiction et du coût de codage du résidu. Comme nous l'avons pressenti déjà avec le critère "Minimisation MSE", le premier est plus important pour "Standard $L_{max} = 56$ " et n'est apparemment pas compensé par une amélioration de la qualité de reconstruction "finale" (*i.e.*, en prenant en compte le résidu). De son côté, "Standard $L_{max} = 8$ " conduit à un coût de codage de prédiction moindre et peut donc se permettre un plus grand coût de codage du résidu, *i.e.*, améliore la qualité de reconstruction du résidu au décodeur. Pour la compression de "Cameraman", la méthode proposée, tout comme les méthodes "Standard", présente des performances légèrement inférieures à celles de l'algorithme "H.264" - quelques -0.04 dB (0.55 % en débit) en métrique de Bjontegaard par rapport à l'algorithme "H.264". Notons toutefois, que vis à vis des deux méthodes "Standard", la méthode "Mélange $L_{max} = 8$ " maintient son bon comportement et atteint toujours de meilleures performances.

4.3.4 Perspectives

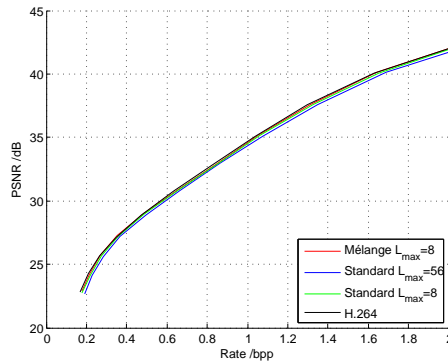
Il est important de souligner que toutes les techniques tendant à adapter le support de la prédiction (voisinage causal, en gris sur la figure 4.3) développées dans le cadre des méthodes de prédiction basées sur des décompositions parcimonieuses (cf. par exemple [110]) peuvent être également appliquées à l'algorithme proposé et conduire possiblement à des améliorations des performances.

En outre, les DCT directionnelles choisies permettent ici une appréhension "H.264" du problème de prédiction basée sur des décompositions parcimonieuses, mais d'autres dictionnaires peuvent être également considérés. Par



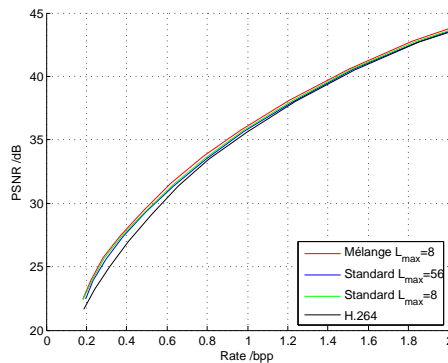
(a)

	dB	%
Mélange $L_{max} = 8$	0.53	-5.77
Standard $L_{max} = 56$	0.26	-2.84
Standard $L_{max} = 8$	0.37	-4.06



(b)

	dB	%
Mélange $L_{max} = 8$	-0.04	0.55
Standard $L_{max} = 56$	-0.37	4.40
Standard $L_{max} = 8$	-0.15	1.83



(c)

	dB	%
Mélange $L_{max} = 8$	0.71	-7.19
Standard $L_{max} = 56$	0.40	-4.00
Standard $L_{max} = 8$	0.56	-5.77

Figure 4.9 PSNR final vs débit pour "Barbara" (a), "Cameraman" (b) et "Roofs" (c) : courbes et gains de Bjontegaard correspondants.

exemple, un ensemble de dictionnaires bien adaptés d'un côté aux contenus texturés comme des transformées de Gabor ou des paquets d'onde-

lettres [87], de l'autre aux contenus "cartoon" comme des curvelets [10] ou des bandelettes [61] pourrait conduire à une meilleure représentation des caractéristiques locales de l'image et ainsi à une amélioration possible des performances en prédiction (quoique nous ayons vu que le lien entre performances en prédiction et performances en codage n'était pas évident).

Enfin, comme nous l'avons mentionné dans l'introduction, il serait nécessaire, pour valider la pertinence de l'algorithme proposé, de l'intégrer dans un codeur vidéo de type H.264. Cette intégration peut être envisagée par exemple en remplacement d'un mode de prédiction intra de H.264 peu utilisé, ainsi que Martin le propose dans [74].

4.4 Annexe : calcul de l'intrégrale $\int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}, c) d\mathbf{y}$

L'intrégrale $\int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}, c) d\mathbf{y}$ peut s'écrire sous la forme

$$\begin{aligned} \int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}, c) d\mathbf{y} &= \int_{\mathbf{y}} p(\mathbf{z}|\mathbf{y}) p(\mathbf{y}|\mathcal{X}, c) p(\mathcal{X}|c) p(c) d\mathbf{y}, \\ &= p(\mathcal{X}|c) p(c) \\ &\int_{\mathbf{y}_o} p(\mathbf{z}_o|\mathbf{y}_o) p(\mathbf{y}_o|\mathcal{X}, c) d\mathbf{y}_o \int_{\mathbf{y}_m} p(\mathbf{z}_m|\mathbf{y}_m) p(\mathbf{y}_m|\mathcal{X}, c) d\mathbf{y}_m. \end{aligned} \quad (4.31)$$

Les calculs des intégrales $\int_{\mathbf{y}_o} p(\mathbf{z}_o|\mathbf{y}_o) p(\mathbf{y}_o|\mathcal{X}, c) d\mathbf{y}_o$ et $\int_{\mathbf{y}_m} p(\mathbf{z}_m|\mathbf{y}_m) p(\mathbf{y}_m|\mathcal{X}, c) d\mathbf{y}_m$ sont analogues. Nous détaillons ici le premier en notant que le résultat du second est obtenu en remplaçant l'indice "o" par l'indice "m".

D'après le modèle (4.4)-(4.6),

$$\begin{aligned} p(\mathbf{z}_o|\mathbf{y}_o) p(\mathbf{y}_o|\mathcal{X}, c) &= \frac{1}{\sqrt{2\pi\sigma_o^2}} \exp\left(-\frac{1}{2\sigma_o^2} \|\mathbf{z}_o - \mathbf{y}_o\|_2^2\right) \\ &\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y}_o - \mathbf{D}_i^o \mathbf{x}_i\|_2^2\right), \\ &= \frac{1}{\sqrt{2\pi\sigma^2\sigma_o^2}} \\ &\exp\left(-\frac{1}{2} \left(\frac{\sigma^2 + \sigma_o^2}{\sigma^2\sigma_o^2}\right) \mathbf{y}_o^T \mathbf{y}_o + \left(\frac{1}{\sigma^2} \mathbf{x}_i^T \mathbf{D}_i^{oT} + \frac{1}{\sigma_o^2} \mathbf{z}_o^T\right) \mathbf{y}_o\right) \\ &\exp\left(-\frac{1}{2\sigma^2} \mathbf{x}_i^T \mathbf{D}_i^{oT} \mathbf{D}_i^o \mathbf{x}_i - \frac{1}{2\sigma_o^2} \mathbf{z}_o^T \mathbf{z}_o\right). \end{aligned} \quad (4.32)$$

On note $A = \exp\left(-\frac{1}{2} \left(\frac{\sigma^2 + \sigma_o^2}{\sigma^2\sigma_o^2}\right) \mathbf{y}_o^T \mathbf{y}_o + \left(\frac{1}{\sigma^2} \mathbf{x}_i^T \mathbf{D}_i^{oT} + \frac{1}{\sigma_o^2} \mathbf{z}_o^T\right) \mathbf{y}_o\right)$. A peut être identifiée à une distribution Gaussienne en \mathbf{y}_o de moyenne $\mathbf{m}_{\mathbf{y}_o}$ et de matrice

de covariance $\Gamma_{\mathbf{y}_o} \mathbf{I}_{N_o}$ telles que

$$\Gamma_{\mathbf{y}_o} = \frac{\sigma^2 + \sigma_o^2}{\sigma^2 \sigma_o^2}, \quad (4.34)$$

$$\mathbf{m}_{\mathbf{y}_o} = \frac{\sigma_o^2}{\sigma^2 + \sigma_o^2} \mathbf{D}_i^o \mathbf{x}_i + \frac{\sigma^2}{\sigma^2 + \sigma_o^2} \mathbf{z}_o. \quad (4.35)$$

On note $B = \exp\left(-\frac{1}{2}(\mathbf{y}_o - \mathbf{m}_{\mathbf{y}_o})^T \Gamma_{\mathbf{y}_o} (\mathbf{y}_o - \mathbf{m}_{\mathbf{y}_o})\right)$. Insérant B dans notre calcul initial, on a

$$p(\mathbf{z}_o | \mathbf{y}_o) p(\mathbf{y}_o | \mathcal{X}, c) = B \frac{1}{\sqrt{2\pi(\sigma^2 + \sigma_o^2)}} \exp\left(-\frac{1}{2(\sigma^2 + \sigma_o^2)} \left(\mathbf{z}_o^T \mathbf{z}_o - 2\mathbf{x}_i^T \mathbf{D}_i^{oT} \mathbf{z}_o + \mathbf{x}_i^T \mathbf{D}_i^{oT} \mathbf{D}_i^o \mathbf{x}_i\right)\right). \quad (4.36)$$

On reconnait alors une distribution Gaussienne en \mathbf{z}_o de moyenne $\mathbf{m}_{\mathbf{z}_o}$ et de variance $\Gamma_{\mathbf{z}_o} \mathbf{I}_{N_o}$ telles que

$$\Gamma_{\mathbf{z}_o} = \sigma^2 + \sigma_o^2, \quad (4.37)$$

$$\mathbf{m}_{\mathbf{z}_o} = \mathbf{D}_i^o \mathbf{x}_i. \quad (4.38)$$

On note $p(\mathbf{z}_o | \mathcal{X}, c)$ cette distribution.

Finalement on a

$$p(\mathbf{z}_o | \mathbf{y}_o) p(\mathbf{y}_o | \mathcal{X}, c) = \mathcal{N}_{\mathbf{y}_o}(\mathbf{m}_{\mathbf{y}_o}, \Gamma_{\mathbf{y}_o} \mathbf{I}_{N_o}) p(\mathbf{z}_o | \mathcal{X}, c), \quad (4.39)$$

et l'intégration sur \mathbf{y}_o donne

$$\int_{\mathbf{y}_o} p(\mathbf{z}_o | \mathbf{y}_o) p(\mathbf{y}_o | \mathcal{X}, c) d\mathbf{y}_o = p(\mathbf{z}_o | \mathcal{X}, c). \quad (4.40)$$

L'intégrale $\int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}, c) d\mathbf{y}$ devient alors

$$\begin{aligned} \int_{\mathbf{y}} p(\mathbf{z}, \mathbf{y}, \mathcal{X}, c) d\mathbf{y} &= p(\mathcal{X} | c) p(c) \\ &= \int_{\mathbf{y}_o} p(\mathbf{z}_o | \mathbf{y}_o) p(\mathbf{y}_o | \mathcal{X}, c) d\mathbf{y}_o \int_{\mathbf{y}_m} p(\mathbf{z}_m | \mathbf{y}_m) p(\mathbf{y}_m | \mathcal{X}, c) d\mathbf{y}_m \\ &= p(\mathbf{z}_o | \mathcal{X}, c) p(\mathbf{z}_m | \mathcal{X}, c) p(\mathcal{X} | c) p(c), \end{aligned}$$

avec $p(\mathbf{z}_o | \mathcal{X}, c) = \mathcal{N}(\mathbf{D}_i^o \mathbf{x}_i, (\sigma^2 + \sigma_o^2) \mathbf{I}_{N_o})$ et $p(\mathbf{z}_m | \mathcal{X}, c) = \mathcal{N}(\mathbf{D}_i^m \mathbf{x}_i, (\sigma^2 + \sigma_m^2) \mathbf{I}_{N_m})$.

Algorithmes gloutons Bayésiens

5

Dans le chapitre 3, l'étude de l'algorithme d'apprentissage de bases proposé par Sezer *et al.* dans [96] et en particulier des performances en codage atteintes par les bases apprises a mis en évidence l'intérêt d'une décomposition parcimonieuse privilégiant un certain ordonnancement des atomes. Cet ordonnancement doit permettre de regrouper les coefficients non nuls de façon à diminuer le coût de codage par RLE des indices des atomes utilisés dans les décompositions parcimonieuses. Or, dans le chapitre 3, les atomes appris semblent être sélectionnés de façon uniforme : il n'existe pas d'ordonnement unique qui soit optimal pour tous les blocs.

Une solution envisageable réside dans l'utilisation d'un modèle probabiliste permettant de prendre en compte des probabilités d'occurrence des atomes différentes. Cette condition est par exemple satisfaite par les modèles Bernoulli-Gaussiens reposant sur des variables Gaussiennes liées à des variables de Bernoulli de paramètres différents.

Dans ce chapitre, nous étudions un modèle Bernoulli-Gaussien en particulier, également utilisé dans [103, 119, 120, 121]. Nous justifions la pertinence du choix d'un tel modèle dans le problème de recherche de décompositions parcimonieuses et en dérivons de nouveaux algorithmes Bayésiens. Pour ce faire, nous considérons un problème d'estimation au sens du maximum a posteriori (MAP) et proposons pour le résoudre plusieurs approches basées sur des maximisations séquentielles. Nous le verrons, ces approches rappellent les processus itératifs des algorithmes gloutons de la littérature (cf. état de l'art section 2.2 chapitre 2).

Ces algorithmes ont fait l'objet d'un article dans les proceedings de la conférence EUSIPCO 2010 [B].

5.1 Introduction

Dans la section 2.2 du chapitre 2, nous avons introduit quelques algorithmes, dits de poursuite ou gloutons, de la littérature. Ces algorithmes reposent sur des procédures “forward” : partant d’un vecteur de support \mathbf{s} nul, les atomes de la décompositions parcimonieuses sont ajoutés petit à petit (un à un [71, 84] ou plusieurs à la fois [30, 22, 80]). À l’inverse, Couvreur et Bresler [19] ont proposé un algorithme “backward” annulant au fur et à mesure de la procédure les coefficients du vecteur support \mathbf{s} . Cet algorithme n’est toutefois réservé qu’aux dictionnaires (sous-)complets. On trouve enfin des algorithmes mêlant les deux approches, “forward” et “backward”, dans lesquels insertion et retrait d’atomes sont rendus possibles. Les contributions [48, 123, 103] ont montré l’intérêt de ces algorithmes, qui permettent la correction d’une “mauvaise” insertion par un retrait ultérieur.

Nous étudions ici la conception d’algorithmes de type “backward-forward” dans un contexte Bayésien. Cette étude a été réalisée en parallèle et indépendamment d’un travail de Soussen *et al.* sur une problématique similaire [103]. Soussen *et al.* proposent une extension “backward-forward” de l’algorithme Orthogonal Least Square (OLS) [14], tandis que selon une approche identique, nous introduisons quatre nouveaux algorithmes, pouvant être interprétés comme des extensions des algorithmes MP, OMP, StOMP et SP/CoSaMP. Dans l’approche suivie par Soussen *et al.* comme dans la nôtre, le même modèle Bernoulli-Gaussien et le même problème d’estimation MAP sont considérés.

Dans la suite de ce chapitre, nous adopterons les notations suivantes. On s’intéresse à la décomposition parcimonieuse d’un signal $\mathbf{y} \in \mathbb{R}^N$ dans un dictionnaire redondant $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_M]$ avec $\mathbf{d}_i \in \mathbb{R}^N, \forall i \in \{1, \dots, M\}$, vecteur normé à 1. On note $\mathbf{x} = [x_1, \dots, x_M]^T$ le vecteur de décomposition parcimonieuse et $\mathbf{s} = [s_1, \dots, s_M]^T$, avec $s_i \in \{0, 1\}, \forall i \in \{1, \dots, M\}$, le vecteur caractérisant le *support* de la décomposition parcimonieuse, *i.e.*, le sous-ensemble de colonnes de \mathbf{D} utilisé pour générer le vecteur \mathbf{y} . On rappelle la “convention” fixée dans le chapitre 2 : si $s_i = 1$ (respectivement $s_i = 0$), la i -ème colonne \mathbf{d}_i de \mathbf{D} est (respectivement n’est pas) utilisé dans la décomposition de \mathbf{y} . Lorsque le contexte le permet, on assimilera variable aléatoire et réalisation.

5.2 Définition d'un cadre probabiliste

On considère le modèle d'observation suivant :

$$\mathbf{y} = \sum_{i=1}^M s_i x_i \mathbf{d}_i + \mathbf{n}, \quad (5.1)$$

où \mathbf{n} est un bruit Gaussien de moyenne nulle et de variance σ^2 . Les distributions sur \mathbf{x} , \mathbf{s} et \mathbf{n} définissent la distribution sur \mathbf{y} :

$$p(\mathbf{n}) = p(\mathbf{y}|\mathbf{x}, \mathbf{s}) = \mathcal{N}(\mathbf{D}_s \mathbf{x}_s, \sigma^2 \mathbf{I}_N), \quad (5.2)$$

où \mathbf{D}_s (respectivement \mathbf{x}_s) est une matrice (respectivement vecteur) faite des \mathbf{d}_i (respectivement x_i) tels que $s_i = 1$. On suppose que \mathbf{x} et \mathbf{s} obéissent au modèle probabiliste suivant :

$$p(\mathbf{x}) = \prod_{i=1}^M p(x_i), \quad p(\mathbf{s}) = \prod_{i=1}^M p(s_i), \quad (5.3)$$

avec

$$p(x_i) = \mathcal{N}(0, \sigma_x^2), \quad (5.4)$$

$$p(s_i) = \text{Ber}(p_i), \quad (5.5)$$

et $\text{Ber}(p_i)$ correspond à une distribution de Bernoulli de paramètre p_i . Rappelons qu'une distribution de Bernoulli est telle que, en reprenant les notations ci-dessus,

$$p(s_i = s) = \begin{cases} p_i & \text{si } s = 1, \\ 1 - p_i & \text{si } s = 0. \end{cases} \quad (5.6)$$

Cette distribution peut encore se formaliser de la façon suivante :

$$p(s_i) = (1 - p_i)^{1 - \|s_i\|_0} p_i^{\|s_i\|_0}, \quad (5.7)$$

$$= \exp((1 - \|s_i\|_0) \log(1 - p_i) + \|s_i\|_0 \log p_i),$$

$$= \exp\left(\log(1 - p_i) - \|s_i\|_0 \log \frac{1 - p_i}{p_i}\right),$$

$$\propto \exp(-\lambda_i \|s_i\|_0), \quad (5.8)$$

avec

$$\lambda_i = \log \frac{1 - p_i}{p_i}. \quad (5.9)$$

Dans la suite de cette section, c'est cette dernière notation que nous adopterons préférentiellement.

Il est important de noter que les expressions (5.2)-(5.5) constituent un *modèle* sur \mathbf{y} et ne prétendent pas décrire sa distribution réelle. Cependant, elles sont très bien adaptées à la modélisation de situations où \mathbf{y} dérive d'un processus parcimonieux. En effet, si $p_i \ll 1, \forall i \in \{1, \dots, M\}$, les réalisations de la variable aléatoire définie par $p(\mathbf{s})$ auront une grande probabilité d'avoir peu de coefficients non nuls et donc le vecteur d'observation \mathbf{y} d'être généré à partir d'un sous-ensemble des colonnes de \mathbf{D} de petite dimension. En particulier, si $p_i = p, \forall i \in \{1, \dots, M\}$, et M est très grand, des réalisations typiques de \mathbf{y} seront des combinaisons de pM colonnes de \mathbf{D} (par la loi des grands nombres).

Notons enfin que des probabilités p_i différentes selon leurs indices permettent de pondérer la sélection des atomes et ainsi de privilégier certaines formes de décompositions parcimonieuses.

La figure 5.1 schématise par un graphe factoriel [59] les dépendances entre les variables considérées.

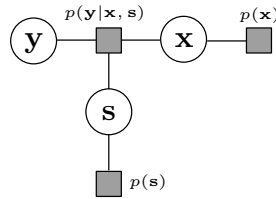


Figure 5.1 Illustration du modèle (5.2)-(5.5) et des relations entre les variables considérées.

5.3 Formulation du problème

Le modèle (5.2)-(5.5) ou des variantes de ce modèle ont déjà été utilisés dans de nombreux algorithmes Bayésiens de la littérature ([103, 119, 120, 121]). Cependant, à notre connaissance, leur connexion avec les problèmes de recherche d'approximations parcimonieuses (cf. sous-section 2.1.3, chapitre 2), et en particulier le problème de régularisation (\mathcal{P}_0^R), n'a pas été prouvée. Le résultat suivant¹ interprète le problème de régularisation (\mathcal{P}_0^R) comme un cas limite d'un problème d'estimation au sens du maximum a posteriori (MAP) s'appuyant sur le modèle Bernoulli-Gaussien (5.2)-(5.5). Corollaire de

1. Le même résultat a été obtenu parallèlement par Soussen *et al.* dans [103].

ce résultat, les algorithmes proposés pour résoudre (approximativement) le problème MAP considéré peuvent être vus comme des généralisations des algorithmes de poursuite “standard”.

Théorème 10 *Considérons le problème d'estimation au sens du maximum a posteriori suivant :*

$$(\hat{\mathbf{x}}, \hat{\mathbf{s}}) = \underset{\mathbf{x}, \mathbf{s}}{\operatorname{argmax}} \log p(\mathbf{y}, \mathbf{x}, \mathbf{s}), \quad (5.10)$$

où $p(\mathbf{y}, \mathbf{x}, \mathbf{s}) = p(\mathbf{y}|\mathbf{x}, \mathbf{s})p(\mathbf{x})p(\mathbf{s})$ est défini par le modèle Bernoulli-Gaussien (5.2)-(5.5). Si,

1. $\|\mathbf{D}_s^+ \mathbf{y}\|_0 = \|\mathbf{s}\|_0, \forall \mathbf{s} \in \{0, 1\}^M,$
2. $\sigma_x^2 \rightarrow +\infty, p_i = p \forall i \in \{1, \dots, M\}$ et $\mu = 2\sigma^2 \log(\frac{1-p}{p}),$

alors,

$$\mathbf{x}^* = \hat{\mathbf{x}}, \quad (5.11)$$

où

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \mu \|\mathbf{x}\|_0. \quad (\mathcal{P}_0^R)$$

Une preuve de ce résultat est donnée en annexe 5.6.1 de ce chapitre.

Le résultat établi dans le théorème 10 place le problème d'approximation parcimonieuse (\mathcal{P}_0^R) dans un cadre Bayésien plus général. En particulier, il révèle les hypothèses statistiques implicitement posées lorsqu'on considère le problème (\mathcal{P}_0^R). Par exemple, toute information a priori sur les probabilités d'occurrence des atomes (p_i) ou sur l'amplitude des coefficients non nuls (σ_x^2) peuvent être explicitement prises en compte. Le cas particulier $\sigma_x^2 = +\infty$ correspond à une distribution a priori $p(\mathbf{x})$ non-informative.

La première condition du théorème est purement “technique”. Elle est satisfaite dans la plupart des cas rencontrés dans la pratique. En particulier, cette condition est satisfaite avec une probabilité 1 tant que \mathbf{y} est une variable aléatoire continue dans \mathbb{R}^N . En effet, l'ensemble des points ne satisfaisant pas cette condition est égal à l'intersection d'un ensemble fini de sous-espaces de \mathbb{R}^N . Cet ensemble est par conséquent de mesure nulle dans \mathbb{R}^N .

La formulation du problème (5.10) n'offre pas d'intérêt en termes de complexité par rapport à la formulation (\mathcal{P}_0^R). Ainsi, le calcul pratique des solutions de (5.10) requiert des approches sous-optimales, de la même façon que

pour son homologue (\mathcal{P}_0^R). Dans la section suivante, nous explicitons plusieurs procédures, qui, en raison de l'équivalence (5.11), auront quelques similarités avec les algorithmes gloutons exposés dans la sous-section 2.2.2 du chapitre 2.

5.4 Développement de nouveaux algorithmes

Dans cette section, nous nous appuyons sur le cadre probabiliste défini dans la sous-section 5.2 pour dériver quatre algorithmes gloutons.

Ainsi que nous l'avons mentionné, nous verrons que ces algorithmes peuvent être vus comme des extensions des algorithmes de poursuite "standard", présentés dans le chapitre 2, section 2.2 de ce manuscrit. Ils offrent une plus haute flexibilité et une plus grande précision dans le calcul du support et des valeurs des vecteurs de décompositions parcimonieuses. En particulier,

- ◆ l'information a priori sur la fréquence d'apparition des atomes dans les décompositions parcimonieuses, constituée par les valeurs des p_i selon les notations admises dans la sous-section 5.2, peut être explicitement prise en compte dans le processus d'estimation,
- ◆ le problème de la désélection des atomes est résolu de façon "naturelle",
- ◆ le cadre Bayésien permet une estimation des paramètres du modèle. Nous verrons ainsi que l'estimation itération après itération de la variance de bruit σ^2 joue un rôle très important dans les performances des algorithmes.

Les algorithmes proposés sont des procédures relativement simples maximisant $\log p(\mathbf{y}, \mathbf{x}, \mathbf{s})$ de façon itérative.

5.4.1 Bayesian Matching Pursuit (BMP)

Comme nous l'avons vu dans la section 2.2 du chapitre 2, l'algorithme MP met à jour à chaque itération *le* coefficient du vecteur de décomposition parcimonieuse qui conduit à la diminution maximale de la norme du résidu. Ici, une séquence $\{(\hat{\mathbf{s}}^{(n)}, \hat{\mathbf{x}}^{(n)})\}$ est construite de façon à augmenter la fonction $\log p(\mathbf{y}, \mathbf{x}, \mathbf{s})$. Pour ce faire, une approche similaire à celle utilisée par MP peut être suivie : l'algorithme Bayesian Matching Pursuit (BMP) que nous proposons met à jour un unique couple $(\hat{s}_j^{(n)}, \hat{x}_j^{(n)})$ à chaque itération, les autres restant fixés à leurs valeurs à l'itération précédente.

Pour décrire formellement cette procédure, nous définissons $\forall i \in \{1, \dots, M\}$

$$F(s_i, x_i) \triangleq \log p(\mathbf{y}, \hat{\mathbf{x}}_i^{(n-1)}, \hat{\mathbf{s}}_i^{(n-1)}), \quad (5.12)$$

où $\hat{\mathbf{x}}_i^{(n-1)}$ (respectivement $\hat{\mathbf{s}}_i^{(n-1)}$) est un vecteur égal à $\hat{\mathbf{x}}^{(n-1)}$ (respectivement $\hat{\mathbf{s}}^{(n-1)}$) mais pour lequel le i -ème élément est libre de varier. Exploitant le modèle Bernoulli-Gaussien (5.2)-(5.5), on obtient

$$F(s_i, x_i) \propto -\frac{1}{2} \left(\frac{s_i}{\sigma^2} + \frac{1}{\sigma_x^2} \right) x_i^2 + \frac{1}{\sigma^2} s_i (\hat{x}_i^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle) x_i - \lambda_i \|s_i\|_0.$$

A la n -ième itération, le couple modifié $(\hat{s}_j^{(n)}, \hat{x}_j^{(n)})$ est celui pour lequel on obtient la plus grande variation de la fonction objectif, *i.e.*,

$$j = \operatorname{argmax}_i \max_{(s_i, x_i)} F(s_i, x_i). \quad (5.13)$$

Il est calculé comme solution du problème d'estimation

$$(\hat{s}_j^{(n)}, \hat{x}_j^{(n)}) = \operatorname{argmax}_{(s_j, x_j)} F(s_j, x_j). \quad (5.14)$$

L'annexe 5.6.2 donne les expressions de $\hat{s}_j^{(n)}$ et $\hat{x}_j^{(n)}$ résultantes.

En pratique, à chaque itération, on optimise tous les couples (s_i, x_i) , pour $i \in \{1, \dots, M\}$, indépendamment des autres selon (5.14). Le couple (s_j, x_j) qui sera effectivement modifié est choisi ensuite, par (5.13), les autres ne sont pas modifiés par rapport à l'itération précédente. L'algorithme 8 résume et formalise la procédure.

Plusieurs caractéristiques importantes de l'algorithme BMP méritent d'être soulignées :

- ◆ $\hat{s}_i^{(n)}$ est une décision sur s_i optimale localement, *i.e.*, la décision maximisant l'augmentation de la fonction objectif étant donné l'estimateur courant. L'estimation de $\hat{s}_i^{(n)}$ repose sur la comparaison d'une énergie de signal résiduel dans la direction de \mathbf{d}_i avec un seuil T_i (cf. équation (5.15)). Ce seuil dépend de la probabilité d'apparition p_i de chaque atome : plus p_i est grande (*i.e.*, plus λ_i est petit), plus T_i est petit, et donc plus l'atome \mathbf{d}_i a de chances d'être sélectionné dans la décomposition parcimonieuse. Notons que si $\hat{s}_i^{(n)} = 0$ alors que $\hat{s}_i^{(n-1)} = 1$, la décision optimale localement consiste à retirer l'atome \mathbf{d}_i du support de la décomposition parcimonieuse. De cette façon, l'algorithme BMP implémente "naturellement" le processus de désélection d'atomes dans la décomposition courante.

Algorithme 8: Bayesian Matching Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$ et $\hat{\mathbf{x}}^{(0)} = 0$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Calcul de (5.14) $\forall i \in \{1, \dots, M\}$

$$\forall i \in \{1, \dots, M\} \quad \hat{s}_i^{(n)} = \begin{cases} 1 & \text{si } \langle \mathbf{r}^{(n-1)} + \hat{\mathbf{x}}_i^{(n-1)} \mathbf{d}_i, \mathbf{d}_i \rangle^2 > T_i, \\ 0 & \text{sinon,} \end{cases} \quad (5.15)$$

avec

$$T_i \triangleq 2\sigma^2 \frac{\sigma^2 + \sigma_x^2}{\sigma_x^2} \lambda_i, \quad (5.16)$$

et

$$\forall i \in \{1, \dots, M\} \quad \hat{\mathbf{x}}_i^{(n)} = \hat{s}_i^{(n)} \left(\hat{\mathbf{x}}_i^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} \right). \quad (5.17)$$

2. Choix du couple modifié : $j = \operatorname{argmax}_i F(\hat{s}_i^{(n)}, \hat{\mathbf{x}}_i^{(n)})$.

3. Mise à jour du support de la décomposition parcimonieuse

$$\forall i \in \{1, \dots, M\} \quad \hat{s}_i^{(n)} = \begin{cases} \hat{s}_i^{(n)} & \text{si } i = j, \\ \hat{s}_i^{(n-1)} & \text{sinon.} \end{cases} \quad (5.18)$$

4. Mise à jour du vecteur parcimonieux

$$\forall i \in \{1, \dots, M\} \quad \hat{\mathbf{x}}_i^{(n)} = \begin{cases} \hat{\mathbf{x}}_i^{(n)} & \text{si } i = j, \\ \hat{\mathbf{x}}_i^{(n-1)} & \text{sinon.} \end{cases} \quad (5.19)$$

4. Mise à jour du résidu

$$\mathbf{r}^{(n)} = \mathbf{y} - \sum_{i=1}^M \hat{s}_i^{(n)} \hat{\mathbf{x}}_i^{(n)} \mathbf{d}_i. \quad (5.20)$$

- ◆ L'estimation de l'amplitude des coefficients (5.17) est réalisée en prenant en compte une information a priori sur la distribution de \mathbf{x} , *i.e.*, σ_x^2 . Notons que si $\hat{s}_i^{(n)} = 1$ et $\sigma_x^2 \rightarrow +\infty$, l'équation (5.17) devient

$$\hat{\mathbf{x}}_i^{(n)} \stackrel{\sigma_x^2 \rightarrow +\infty}{=} \hat{\mathbf{x}}_i^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle, \quad (5.21)$$

résultant en l'estimation réalisée dans l'algorithme MP (cf. équation (2.17) dans l'algorithme 1, section 2.2, chapitre 2).

Dans la sous-section précédente, nous avons montré l'équivalence entre le problème d'estimation au sens du maximum a posteriori (5.10) et le problème de régularisation (\mathcal{P}_0^R) si $\sigma_x^2 \rightarrow +\infty$ et $p_i = p$, $\forall i \in \{1, \dots, M\}$. Ces conditions ne sont pas suffisantes pour assurer l'équivalence entre les algorithmes BMP et MP en raison de la désélection des atomes, permise par BMP mais non implémentée dans la procédure MP. Empêchant cette possibilité (en forçant $\hat{s}_i^{(n)} = 1$, $\forall i \in \{1, \dots, M\}$), *i.e.*, en ne considérant que l'ajout d'atomes dans

le support, on retrouve l'implémentation de l'algorithme MP. Ce dernier peut donc être considéré comme un cas particulier de son pendant Bayésien.

5.4.2 Bayesian Orthogonal Matching Pursuit (BOMP)

On considère maintenant l'implémentation d'un OMP Bayésien, que nous noterons BOMP par simplicité. De la même façon que son homologue "standard" OMP, BOMP procède en modifiant (ajoutant ou, comme nous le verrons dans la suite, retirant) un unique atome du support de la décomposition parcimonieuse, mais en réestimant à chaque itération le vecteur complet de pondération \mathbf{x} . Les trois premières étapes sont réalisées à la manière de la procédure BMP précédente. L'indice j de l'atome modifié est choisi selon l'équation (5.13) et la valeur du $\hat{s}_j^{(n)}$ correspondant est calculée par (5.15) et (5.18). L'étape de mise à jour du vecteur parcimonieux, en revanche, se formalise différemment. Après la mise à jour du support de la décomposition, $\hat{\mathbf{x}}^{(n)}$ est calculé comme

$$\hat{\mathbf{x}}^{(n)} = \underset{\mathbf{x}}{\operatorname{argmax}} \log p(\mathbf{y}, \mathbf{x}, \hat{\mathbf{s}}^{(n)}), \quad (5.22)$$

ayant pour solution $\hat{\mathbf{x}}^{(n)}$ défini par

$$\hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y}, \quad (5.23)$$

et $\hat{x}_j^{(n)} = 0$ si $\hat{s}_j^{(n)} = 0$. Rappelons que $\hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)}$ est constitué des éléments $\hat{x}_i^{(n)}$ de $\hat{\mathbf{x}}^{(n)}$ tels que $\hat{s}_i^{(n)} = 1$. La procédure est résumée dans l'algorithme 9. Notons que, similairement à BMP, BOMP met à jour les coefficients non nuls du vecteur de décomposition en prenant en compte l'information a priori sur l'amplitude des coefficients, σ_x^2 . Le détail du calcul de (5.23) est donné en annexe 5.6.3 de ce chapitre.

Comme nous l'avons dit, l'étape de mise à jour du support de la décomposition parcimonieuse est inchangée par rapport à l'algorithme BMP. Ainsi, comme BMP, BOMP autorise également la désélection d'atomes. Pour cette raison, similaire à celle mentionnée précédemment pour l'équivalence BMP/MP, BOMP ne s'identifie pas à OMP lorsque $\sigma_x^2 \rightarrow +\infty$ et $p_i = p$, $\forall i \in \{1, \dots, M\}$.

Algorithme 9: Bayesian Orthogonal Matching Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$ et $\hat{\mathbf{x}}^{(0)} = 0$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Calcul de (5.14) $\forall i \in \{1, \dots, M\}$

$$\forall i \in \{1, \dots, M\} \quad \hat{s}_i^{(n)} = \begin{cases} 1 & \text{si } \langle \mathbf{r}^{(n-1)} + \hat{\mathbf{x}}_i^{(n-1)} \mathbf{d}_i, \mathbf{d}_i \rangle^2 > T_i, \\ 0 & \text{sinon,} \end{cases}$$

avec

$$T_i = 2\sigma^2 \frac{\sigma^2 + \sigma_x^2}{\sigma_x^2} \lambda_i,$$

et

$$\forall i \in \{1, \dots, M\} \quad \hat{\mathbf{x}}_i^{(n)} = \hat{s}_i^{(n)} \left(\hat{\mathbf{x}}_i^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_i \rangle \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} \right).$$

2. Choix du couple modifié : $j = \operatorname{argmax}_i F(\hat{s}_i^{(n)}, \hat{\mathbf{x}}_i^{(n)})$.

3. Mise à jour du support de la décomposition parcimonieuse

$$\forall i \in \{1, \dots, M\} \quad \hat{s}_i^{(n)} = \begin{cases} \hat{s}_i^{(n)} & \text{si } i = j, \\ \hat{s}_i^{(n-1)} & \text{sinon.} \end{cases}$$

4. Mise à jour du vecteur parcimonieux

$$\hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y}, \quad (5.23)$$

et

$$\forall i \in \{1, \dots, M\} \quad \hat{\mathbf{x}}_i^{(n)} = 0 \text{ si } \hat{s}_i^{(n)} = 0. \quad (5.24)$$

4. Mise à jour du résidu

$$\mathbf{r}^{(n)} = \mathbf{y} - \sum_{i=1}^M \hat{s}_i^{(n)} \hat{\mathbf{x}}_i^{(n)} \mathbf{d}_i.$$

5.4.3 Bayesian Stagewise Orthogonal Matching Pursuit (BStOMP)

A la manière de StOMP, l'algorithme Bayesian Stagewise Orthogonal Matching Pursuit, noté BStOMP dans la suite, est une variante de BOMP où plusieurs composantes du vecteur de support \mathbf{s} peuvent être modifiées à chaque itération. L'idée est donc de relâcher la condition (5.18).

A la n -ième itération, les coefficients $\hat{s}_j^{(n)}$ du vecteur support sont calculés comme

$$\hat{s}_j^{(n)} = \begin{cases} 1 & \text{si } \langle \mathbf{r}^{(n-1)} + \hat{\mathbf{x}}_j^{(n-1)} \mathbf{d}_j, \mathbf{d}_j \rangle^2 > T_j, \\ 0 & \text{sinon,} \end{cases} \quad (5.25)$$

où T_j est défini par (5.16). La mise à jour des coefficients du vecteur de décomposition parcimonieuse, reste, elle, inchangée par rapport à la procédure BOMP et est exprimée par (5.23). L'algorithme 10 détaille les étapes

Algorithme 10: Bayesian Stagewise Orthogonal Matching Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$ et $\hat{\mathbf{x}}^{(0)} = 0$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Mise à jour du support de la décomposition parcimonieuse

$$\forall i \in \{1, \dots, M\} \quad \hat{s}_i^{(n)} = \begin{cases} 1 & \text{si } \langle \mathbf{r}^{(n-1)} + \hat{\mathbf{x}}_i^{(n-1)} \mathbf{d}_i, \mathbf{d}_i \rangle^2 > T_i, \\ 0 & \text{sinon,} \end{cases}$$

avec

$$T_i = 2\sigma^2 \frac{\sigma^2 + \sigma_x^2}{\sigma_x^2} \lambda_i,$$

1. Mise à jour du vecteur parcimonieux

$$\hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y},$$

et

$$\forall i \in \{1, \dots, M\} \quad \hat{\mathbf{x}}_i^{(n)} = 0 \text{ si } \hat{s}_i^{(n)} = 0.$$

3. Mise à jour du résidu

$$\mathbf{r}^{(n)} = \mathbf{y} - \sum_{i=1}^M \hat{s}_i^{(n)} \hat{\mathbf{x}}_i^{(n)} \mathbf{d}_i.$$

de la procédure BStOMP proposée.

Remarquons que si le j -ème atome n'a pas été sélectionné à l'itération précédente $n-1$, *i.e.*, $(\hat{s}_j^{(n-1)}, \hat{x}_j^{(n-1)}) = (0, 0)$, la condition (5.25) devient similaire à celle utilisée dans l'étape de mise à jour du support de décomposition parcimonieuse de StOMP (cf. équation (2.22), algorithme 3, section 2.2, chapitre 2) :

$$\hat{s}_j^{(n)} = \begin{cases} 1 & \text{si } \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle^2 > T_j, \\ \hat{s}_j^{(n-1)} & \text{sinon.} \end{cases} \quad (5.26)$$

Cependant, dans le cas général, les deux opérations ne sont pas équivalentes, (5.25) permettant la désélection d'atomes interdite par (5.26).

Autre différence entre les deux algorithmes StOMP et BStOMP : le seuil T_j est ici défini "naturellement" en fonction des paramètres du modèle. Ce n'est pas le cas de la définition donnée par Donoho *et al.* dans [30], qui requiert des hypothèses supplémentaires.

Enfin, notons que les performances de BStOMP peuvent être grandement améliorées par l'ajout de l'estimation de la variance de bruit σ^2 dans le processus itératif. Comme nous l'avons mentionné précédemment, l'estimation des paramètres du modèle sont facilitées par le contexte Bayésien. En particulier, l'estimateur au sens du maximum de vraisemblance de σ^2 s'écrit, à l'itération

n ,

$$\begin{aligned}
(\sigma^2)^{(n)} &= \operatorname{argmax}_{\sigma^2} \log p(\mathbf{y}, \hat{\mathbf{x}}^{(n-1)}, \hat{\mathbf{s}}^{(n-1)}), \\
&= \frac{1}{N} \|\mathbf{y} - \mathbf{D}\hat{\mathbf{x}}^{(n-1)}\|_2^2, \\
&= \frac{\|\mathbf{r}^{(n-1)}\|_2^2}{N}.
\end{aligned} \tag{5.27}$$

Insérant cette expression dans la définition du seuil (5.16), on obtient

$$T_j^{(n)} \triangleq 2 \frac{\|\mathbf{r}^{(n-1)}\|_2^2}{N} \frac{\sigma_x^2 + N^{-1}\|\mathbf{r}^{(n-1)}\|_2^2}{\sigma_x^2} \lambda_j. \tag{5.28}$$

Ainsi, le seuil devient une fonction du numéro d'itération n . Notons que si $\sigma_x^2 \rightarrow +\infty$,

$$T_j^{(n)} \stackrel{\sigma_x^2 \rightarrow +\infty}{=} 2 \frac{\|\mathbf{r}^{(n-1)}\|_2^2}{N} \lambda_j. \tag{5.29}$$

$T_j^{(n)}$ est alors proportionnel à l'énergie du résidu $\mathbf{r}^{(n-1)}$, avec un facteur de proportionnalité dépendant de la probabilité d'apparition de chaque atome (cf. équation (5.9)). Dans les premières itérations, la norme du résidu $\|\mathbf{r}^{(n-1)}\|_2$ est grande, le seuil $T_j^{(n)}$ est lui-même haut, de sorte que les atomes insérés sont peu nombreux. L'insertion aura tendance à s'accélérer au fur et à mesure des itérations lorsque $\|\mathbf{r}^{(n-1)}\|_2$ diminuant, le seuil $T_j^{(n)}$ sera plus bas.

5.4.4 Bayesian Subspace Pursuit (BSP)

Enfin, on peut proposer un algorithme de recherche "ressemblant" à l'algorithme CoSaMP/SP ([80, 22]). Nous appellerons cet algorithme Bayesian Subspace Pursuit, et l'abrégerons par BSP. L'algorithme BSP présente une structure similaire à celle de son homologue "standard". L'idée est de calculer la solution optimale $(\tilde{\mathbf{s}}^{(n)}, \tilde{\mathbf{x}}^{(n)})$ dans un sous-espace de dimension supérieure puis de l'utiliser pour déterminer la solution optimale $(\hat{\mathbf{s}}^{(n)}, \hat{\mathbf{x}}^{(n)})$ dans un sous-espace de dimension fixée, *i.e.*, le nombre de coefficients non nuls du vecteur de décomposition parcimonieuse est fixé. Nous notons L ce nombre de coefficients.

Ainsi, l'estimation du support à l'itération n est réalisée de la façon suivante :

$$\hat{\mathbf{s}}^{(n)} = \operatorname{argmax}_{\mathbf{s}} \sum_i \max_{x_i} \log p(\mathbf{y}, \tilde{\mathbf{x}}_i^{(n)}, \tilde{\mathbf{s}}_i^{(n)}) \text{ soumis à } \|\mathbf{s}\|_0 = L, \tag{5.30}$$

Algorithme 11: Bayesian Subspace Pursuit

0. Initialisation : $\mathbf{r}^{(0)} = \mathbf{y}$ et $\hat{\mathbf{x}}^{(0)} = 0$.

Tant que le critère d'arrêt n'est pas atteint, répéter :

1. Calcul de (5.32)

$$\forall i \in \{1, \dots, M\} \quad \tilde{s}_i^{(n)} = \begin{cases} 1 & \text{si } \langle \mathbf{r}^{(n-1)} + \hat{\mathbf{x}}_i^{(n-1)} \mathbf{d}_i, \mathbf{d}_i \rangle^2 > T_i, \\ 0 & \text{sinon,} \end{cases}$$

avec

$$T_i = 2\sigma^2 \frac{\sigma^2 + \sigma_x^2}{\sigma_x^2} \lambda_i,$$

2. Projection sur le sous-espace ainsi défini

$$\tilde{\mathbf{x}}_{\tilde{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\tilde{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\tilde{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\tilde{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\tilde{\mathbf{s}}^{(n)}}^T \mathbf{y}, \quad (5.31)$$

et

$$\forall i \in \{1, \dots, M\} \quad \tilde{\mathbf{x}}_i^{(n)} = 0 \text{ si } \tilde{s}_i^{(n)} = 0.$$

3. Mise à jour du support de la décomposition parcimonieuse

$$\hat{\mathbf{s}}^{(n)} = \operatorname{argmax}_{\mathbf{s}} \sum_i \max_{x_i} \log p(\mathbf{y}, \tilde{\mathbf{x}}_i^{(n)}, \tilde{s}_i^{(n)}) \text{ soumis à } \|\mathbf{s}\|_0 = L. \quad (5.30)$$

4. Mise à jour du vecteur parcimonieux

$$\hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y}, \quad (5.33)$$

et

$$\forall i \in \{1, \dots, M\} \quad \hat{\mathbf{x}}_i^{(n)} = 0 \text{ si } \hat{s}_i^{(n)} = 0.$$

5. Mise à jour du résidu

$$\mathbf{r}^{(n)} = \mathbf{y} - \sum_{i=1}^M \hat{s}_i^{(n)} \hat{\mathbf{x}}_i^{(n)} \mathbf{d}_i.$$

s'appuyant sur l'estimation de $\tilde{\mathbf{x}}^{(n)}$ telle que

$$\tilde{\mathbf{x}}_{\tilde{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\tilde{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\tilde{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\tilde{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\tilde{\mathbf{s}}^{(n)}}^T \mathbf{y}, \quad (5.31)$$

avec

$$\tilde{\mathbf{s}}^{(n)} = \operatorname{argmax}_{\mathbf{s}} \sum_i \max_{x_i} \log p(\mathbf{y}, \hat{\mathbf{x}}_i^{(n-1)}, \hat{s}_i^{(n-1)}). \quad (5.32)$$

Le vecteur de décomposition parcimonieuse $\hat{\mathbf{x}}^{(n)}$ est finalement obtenu par

$$\hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y}. \quad (5.33)$$

L'algorithme 11 résume la procédure BSP proposée.

Il est intéressant de noter que, à l'inverse de CoSaMP/SP, l'algorithme que nous proposons n'impose aucune contrainte sur le nombre d'éléments non nuls de $\tilde{\mathbf{s}}^{(n)}$. Ainsi, $\|\tilde{\mathbf{s}}^{(n)}\|_0$ peut être plus grand ou plus petit que L . $\tilde{\mathbf{s}}^{(n)}$ est calculé en prenant la meilleure décision locale pour chaque atome du dictionnaire. Cette règle est équivalente à l'opération (5.25) implémentée dans BS-tOMP. En raison de cette similarité, nous implémentons également l'estimation de la variance de bruit σ^2 (5.27), permettant via l'expression du seuil T_j un meilleur contrôle de l'insertion des atomes.

De même, l'estimation (5.30) est réalisée de la façon suivante. $\forall i \in \{1, \dots, M\}$, on effectue (5.25) avec $\tilde{\mathbf{x}}^{(n)}$. Puis, si le nombre d'éléments mis à 1 est supérieur ou égal à L , les modifications ne sont conservées que pour K indices menant aux plus grandes valeurs de $\max_{(x_i, s_i)} \log p(\mathbf{y}, \tilde{\mathbf{x}}_i^{(n)}, \tilde{\mathbf{s}}_i^{(n)})$ et tels que $\|\mathbf{s}\|_0 = L$. Notons en effet que (5.25) permettant la désélection d'atomes, K n'est pas nécessairement égal à L mais supérieur ou égal. Si par contre, le nombre d'éléments mis à 1 est inférieur à L , les modifications sont conservées pour tous les éléments mis à 1 et pour les indices menant aux plus petites valeurs de $\max_{(x_i, s_i)} \log p(\mathbf{y}, \tilde{\mathbf{x}}_i^{(n)}, \tilde{\mathbf{s}}_i^{(n)})$ jusqu'à ce que $\|\mathbf{s}\|_0 = L$. De cette façon, on minimise l'impact de l'introduction d'éléments pénalisants.

5.5 Evaluation des performances

Dans cette section, nous étudions les performances des algorithmes proposés. Nous suivons pour cela la même méthodologie que celle utilisée dans [22] : la probabilité de reconstruction exacte du vecteur \mathbf{x} est calculée en fonction du nombre de coefficients non nuls de \mathbf{x} , L . Nous supposons que le vecteur $\hat{\mathbf{x}}$ a été correctement reconstruit si l'erreur de reconstruction sur chaque coefficient non nul est inférieure à 10^{-4} (*i.e.*, $\forall i \in \{1, \dots, M\}, |x_i - \hat{x}_i|^2 < 10^{-4}$).

Les algorithmes proposés sont comparés à leurs homologues "standard" et à l'algorithme Basis Pursuit Denoising (BPD) [16], qui implémente une solution du problème (\mathcal{P}_1^R) .

MP et OMP s'arrêtent lorsque la norme ℓ_2 du résidu atteint $\sqrt{N\sigma^2}$. Les algorithmes Bayésiens itèrent tant que $\max_i \rho_i^{(n)} - \log p(\mathbf{y}, \hat{\mathbf{x}}^{(n-1)}, \hat{\mathbf{s}}^{(n-1)}) > 10^{-6}$, *i.e.*, tant que la variation de la fonction objectif est suffisamment grande. Pour StOMP, SP et BPD, les implémentations choisies sont celles disponibles respectivement sur les pages <http://sparselab.stanford.edu/> (SparseLab), <http://igorcarron.googlepages.com/cscodes> et <http://www.acm.caltech.edu/l1magic/>

(ℓ_1 -Magic). StOMP est utilisé avec le critère de seuillage CFDR ([30]), tandis que pour les algorithmes BStOMP et BSP, le seuillage (5.28) est considéré.

De façon à tester le comportement des algorithmes dans divers scénarios, les performances sont évaluées sur des signaux générés synthétiquement selon des modèles différents.

5.5.1 Modèle Bernoulli-Gaussien, dictionnaire aléatoire

Le premier modèle considéré est celui utilisé dans les algorithmes Bayésiens, comme produit bruité d'un vecteur parcimonieux avec un dictionnaire. Les positions des coefficients non nuls du vecteur parcimonieux sont déterminées aléatoirement de façon uniforme. L'amplitude des coefficients non nuls est construite à partir d'une distribution Gaussienne de moyenne nulle et de variance $\sigma_x^2 = 10$. Pour les variables de Bernoulli $s_i \forall i \in \{1, \dots, M\}$, on choisit des paramètres p_i identiques et égaux à $p_i = L/M$. Les éléments du dictionnaire sont des réalisations indépendantes et identiquement distribuées selon une loi Gaussienne de variance N^{-1} , où N est la dimension des données fixée à chaque expérimentation (cf. ci-après). La taille du dictionnaire est prise égale à $M = 256$ atomes. Enfin, un bruit Gaussien de variance $\sigma^2 = 10^{-5}$ est ajouté.

A chaque point d'expérimentation, 500 simulations (*i.e.*, décompositions parcimonieuses) sont réalisées et moyennées. De façon à ne pas favoriser les méthodes proposées par une quelconque information a priori, on utilise $\sigma_x^2 = 1000$ dans les algorithmes Bayésiens.

La figure 5.2 illustre les performances en reconstruction (probabilité de reconstruction exacte de \mathbf{x} vs nombre de coefficients non nuls de \mathbf{x}) obtenues pour trois dimensions de signaux : $N = 77$, $N = 128$ et $N = 180$. En regard des courbes de performances sont représentées les complexités correspondantes, en temps de calcul moyen par simulation (*i.e.*, par décomposition parcimonieuse) et pour chaque algorithme.

Les positions des courbes de performance les unes par rapport aux autres sont relativement insensibles à la redondance du dictionnaire utilisé : ainsi, les algorithmes BOMP et BSP présentent les meilleures performances, et à l'inverse, l'algorithme MP les plus mauvaises, sur les trois figures (a), (c) et (e). Notons toutefois que l'algorithme BStOMP réalise de bonnes performances vis à vis de l'algorithme BPD (pour $N = 77$ et $L = 25$, on observe une probabilité de reconstruction exacte de 0.3 pour BPD contre 0.65 pour BStOMP, soit un gain de 35%) et des performances similaires à celles de l'algorithme SP

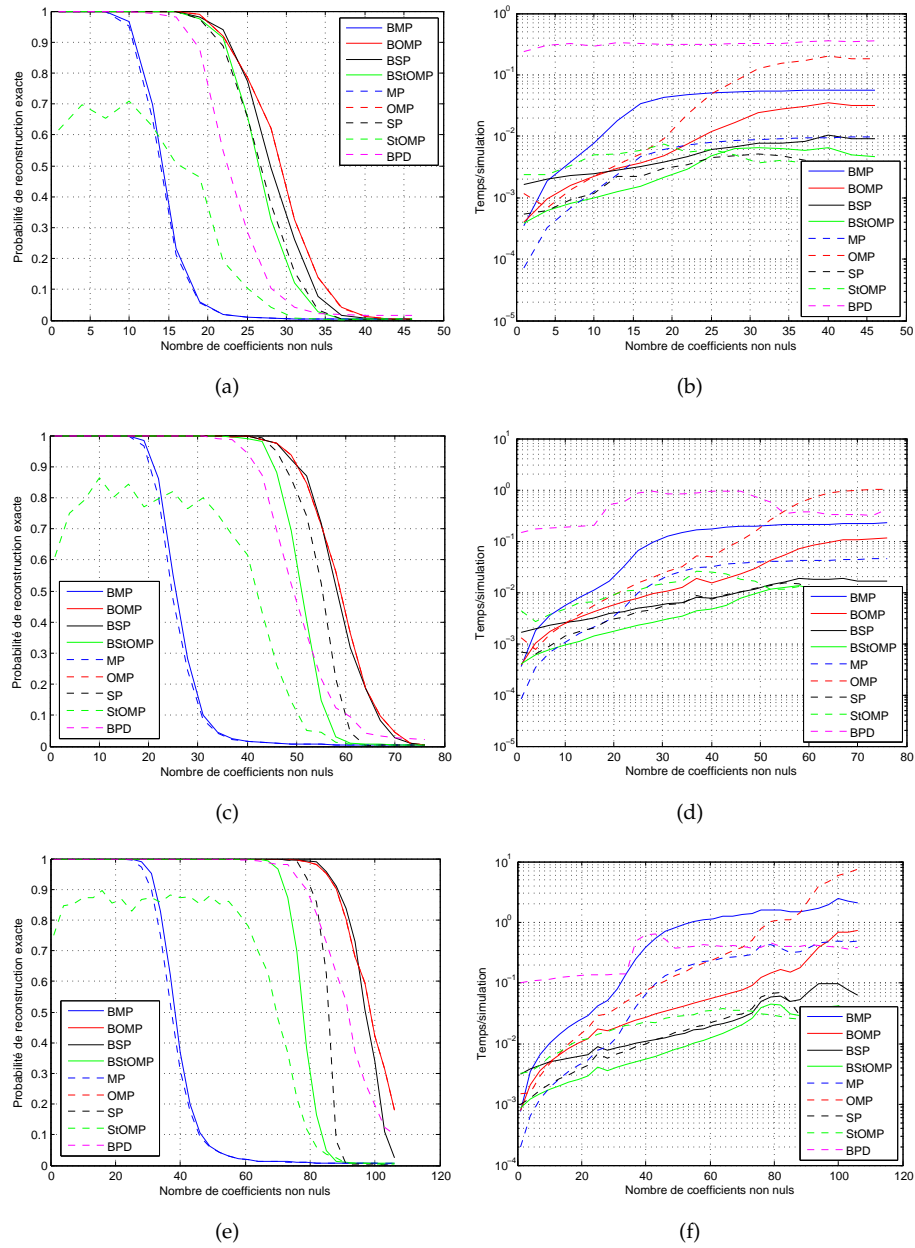


Figure 5.2 Modèle Bernoulli-Gaussien, dictionnaire aléatoire - Performances en reconstruction obtenues par différents algorithmes et complexités correspondantes, pour des dimensions de signaux différentes : $N = 77$ (a)-(b), $N = 128$ (c)-(d) et $N = 180$ (e)-(f).

avec des dictionnaires très redondants ($N = 77$, figure 5.2 (a)), mais est supplanté par ces deux derniers avec des dictionnaires peu redondants ($N = 180$,

figure 5.2 (e)).

On observe par ailleurs que les algorithmes gloutons Bayésiens reconstruisent les vecteurs parcimonieux utilisés pour la génération des données sinon mieux du moins aussi bien que leurs homologues “standard”. Si BMP et BOMP n’améliorent que peu (10% de plus environ avec une dimension $N = 128$ pour BMP) ou pas du tout (pour BOMP) les performances atteintes par respectivement MP et OMP, les performances obtenues avec les algorithmes BStOMP et BSP sont nettement supérieures à celles de StOMP et SP respectivement (jusqu’à 30% de gain pour BSP et 50% pour BStOMP avec une dimension $N = 128$).

Avant de poursuivre par l’analyse des complexités, il peut être intéressant de se pencher plus attentivement sur le couple BOMP/OMP et les raisons pour lesquelles les deux algorithmes présentent exactement les mêmes performances. Reprenant l’équation de mise à jour du vecteur parcimonieux \hat{x} dans BOMP (5.23), on peut voir que si $\sigma_x^2 \gg \sigma^2$ alors $\frac{\sigma^2}{\sigma_x^2} \rightarrow 0$ et (5.23) est équivalente à l’équation de mise à jour du vecteur parcimonieux \hat{x} dans OMP (2.20). D’autre part, cette équation montre que quel que soit le support de la décomposition parcimonieuse obtenue, il suffit que les “bons” atomes (*i.e.*, ceux qui ont été utilisés pour générer le vecteur d’observation y) aient été sélectionnés pour retrouver le “bon” vecteur parcimonieux (*i.e.*, celui qui a été utilisé pour générer le vecteur d’observation y) : les valeurs des coefficients x_i ne correspondant pas aux “bons” atomes seront mises à zéro ou à des valeurs très proches de zéro (dû à la présence de bruit). Ainsi, une analyse supplémentaire pourrait être de mesurer les performances en reconstruction des *supports* des décompositions parcimonieuses, qui a priori, doivent, eux, être différents selon l’algorithme utilisé, BOMP ou OMP.

Il est délicat de comparer les complexités de chacun des algorithmes considérés. Comme l’illustrent les deux algorithmes BMP et MP, les critères d’arrêt utilisés jouent un rôle très important : ainsi si BMP et MP présentent la même complexité par itération (cf. sous-section précédente), le premier itère plus longtemps que le dernier avant d’atteindre le critère d’arrêt, résultant en une complexité d’exécution globale plus élevée (et pouvant d’autre part expliquer la légère amélioration des performances). Pour ce cas précis, une modification du critère d’arrêt pourrait être intéressante.

Cependant, cette critique perd de sa pertinence pour les autres algorithmes Bayésiens. On peut ainsi observer que BOMP présente une complexité très inférieure à celle de son homologue “standard” OMP, pour, comme nous l’avons vu, des performances en reconstruction équivalentes. Ce comporte-

ment peut s'expliquer par la désélection d'atomes, autorisée dans BOMP et interdite dans OMP, qui permet de réduire le coût de l'inversion matricielle (5.23) sans pour autant perdre en qualité de reconstruction. On peut également supposer que la désélection d'atomes a un effet positif sur la convergence de l'algorithme, "accéléralant" le processus de résolution. Un comportement similaire régit l'algorithme BStOMP vis à vis de son homologue "standard" StOMP.

L'algorithme BSP présente une complexité plus élevée que l'algorithme SP, pour, rappelons-le, de meilleures performances en reconstruction. Cette double observation peut s'expliquer par le relâchement de la contrainte sur la dimension de l'espace de recherche lors de la première mise à jour du support de la décomposition parcimonieuse (5.32) : ce degré de liberté supplémentaire est susceptible d'améliorer les performances mais exige davantage de calculs dans (5.31) (le vecteur support \mathbf{s} peut avoir un nombre de coefficients non nuls très élevé).

Notons enfin que l'algorithme BPD présente une complexité nettement plus élevée que les autres algorithmes. Nous l'avions évoqué dans le chapitre 2, cet algorithme présente une complexité en $\mathcal{O}(N^3)$ peu compétitive en comparaison des algorithmes de poursuite "standard". Il en va de même avec les algorithmes gloutons Bayésiens proposés.

5.5.2 Modèle 0 – 1, dictionnaire aléatoire

Dans un deuxième temps, on étudie un autre modèle de génération des données, ne respectant pas le modèle considéré dans les algorithmes gloutons Bayésiens proposés. Cette fois, le vecteur parcimonieux n'est autorisé à prendre ses valeurs que dans l'ensemble $\{0, 1\}^M$, de sorte que $\mathbf{x} = \mathbf{s}$. Les positions des coefficients non nuls restent déterminées aléatoirement de façon uniforme. De même, la construction du dictionnaire et les conditions d'exécution des algorithmes sont inchangées par rapport aux expérimentations précédentes.

Ici encore, on étudie les performances obtenues pour trois dimensions de signaux : $N = 77$, $N = 128$ et $N = 180$. La figure 5.3 montre les probabilités de reconstruction atteintes en fonction du nombre de coefficients non nuls.

Une première observation met en évidence les bons comportements des algorithmes "standard" : quelle que soit la redondance du dictionnaire considérée, BPD présente toujours les meilleures performances en reconstruction, suivi par SP. Les algorithmes BMP et MP sont pour leur part toujours

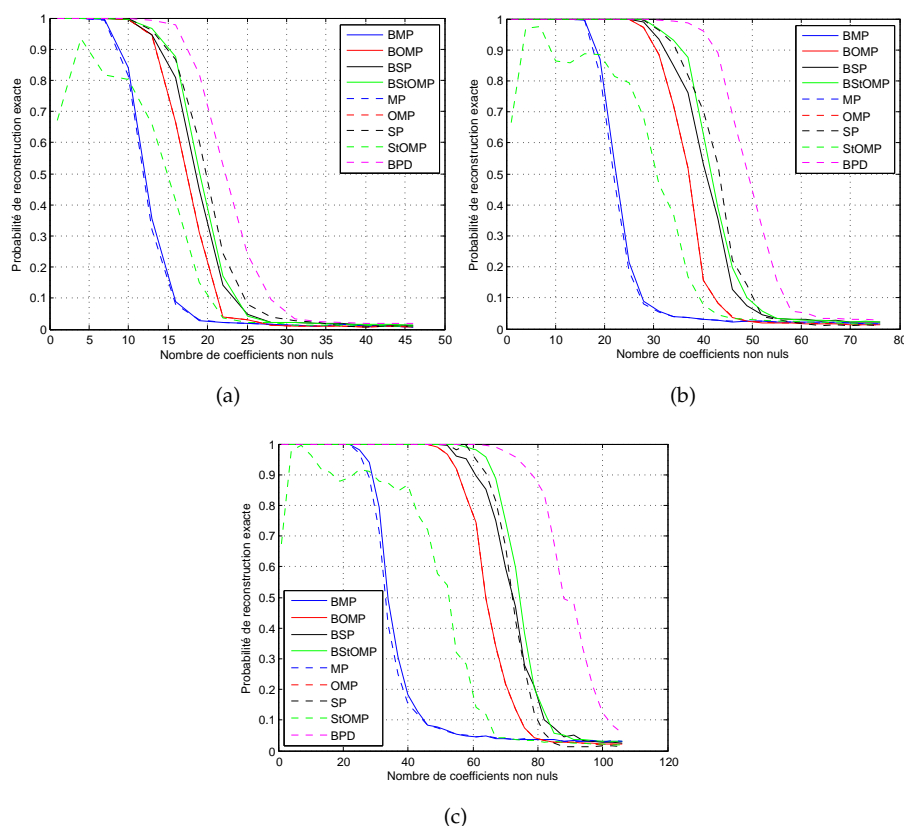


Figure 5.3 Modèle 0 – 1, dictionnaire aléatoire - Performances en reconstruction obtenues par différents algorithmes et pour des dimensions de signaux différentes : $N = 77$ (a), $N = 128$ (b) et $N = 180$ (c).

en queue de peloton, avec des comportements similaires à ceux observés lors des expérimentations précédentes (BMP légèrement meilleur que MP). Cependant, il est intéressant de constater que lorsque la redondance diminue (*i.e.*, pour des dimensions de signaux plus grandes), les performances atteintes par les algorithmes gloutons Bayésiens s’améliorent (jusqu’à dépasser SP de 10% à $N = 180$ pour BStOMP).

Comme nous l’avons mentionné pour le couple BMP/MP, les comportements des algorithmes Bayésiens vis à vis de leurs homologues “standard” sont très similaires à ceux observés lors des expérimentations précédentes : BOMP et OMP ont des courbes de performances confondues et BStOMP surpasse StOMP de près de 50% pour $N = 128$. Toutefois, le couple BSP/SP fait entorse à la règle : alors que les performances atteintes par BSP présentaient un gain de 30% environ par rapport à celles de SP lors des expérimentations

précédentes, on observe ici l'inverse, SP domine BSP avec une différence de près de 20% pour $N = 128$.

5.5.3 Modèle Bernoulli-Gaussien, dictionnaire en cosinus

Dans une troisième et dernière étude, la construction du dictionnaire est modifiée : les atomes sont des fonctions en cosinus discrets. Positions et amplitudes des coefficients non nuls du vecteur parcimonieux sont déterminées selon le premier modèle que nous avons considéré. Enfin, le scénario d'expérimentation est inchangé.

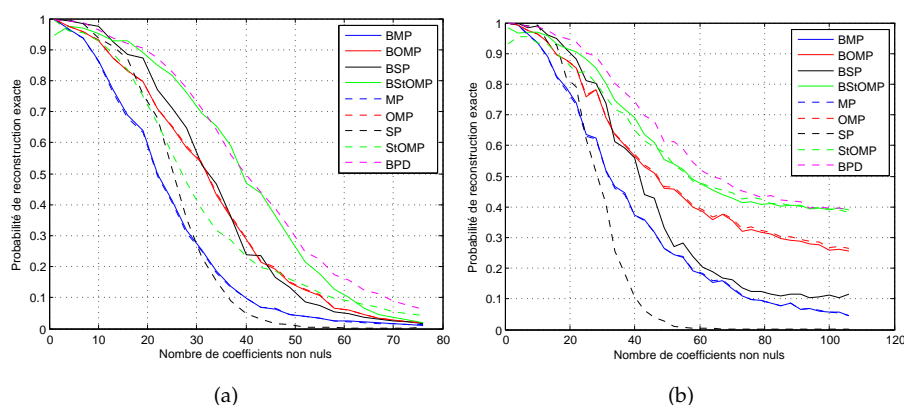


Figure 5.4 Modèle Bernoulli-Gaussien, dictionnaire en cosinus - Performances en reconstruction obtenues par différents algorithmes et pour des dimensions de signaux différentes : $N = 128$ (a) et $N = 180$ (b).

La figure 5.4 illustrent les performances en reconstruction obtenues pour deux dimensions de signaux : $N = 128$ et $N = 180$.

De même que lors des deuxièmes expérimentations, l'algorithme BPD présente les meilleures performances en reconstruction. Il est cependant suivi de près par l'algorithme BStOMP si le dictionnaire utilisé est très redondant ($N = 128$, figure 5.4 (a)). Le couple BMP/MP atteint cette fois-ci encore de faibles performances, mais il réussit à surpasser l'algorithme SP jusqu'à 25% pour des dictionnaires peu redondants ($N = 180$, figure 5.4 (b)) et 5% pour des dictionnaires plus redondants ($N = 128$, figure 5.4 (a)).

D'une façon générale, on observe une plus grande similarité entre les performances atteintes par les algorithmes Bayésiens et celles obtenues par leurs homologues "standard" lorsque la redondance du dictionnaire diminue. Ainsi, les courbes de performances de BMP et MP, BOMP et OMP, BS-

tOMP et StOMP, sont presque confondues sur la figure 5.4 (b). Cette tendance générale ne semble pas affecter l'algorithme BSP, qui reste bien meilleur que SP, et ce, quelle que soit la redondance du dictionnaire (jusqu'à 25% de gain pour $N = 128$ et 45% pour $N = 180$).

5.6 Annexes

5.6.1 Preuve du théorème (10)

Soit $f(\mathbf{x}) \triangleq \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \mu\|\mathbf{x}\|_0$ et $\mathbf{x}^*(\mathbf{s})$ la solution de

$$\mathbf{x}^*(\mathbf{s}) = \underset{\mathbf{x}}{\operatorname{argmin}} f(\mathbf{x}) \text{ soumis à } x_i = 0 \text{ si } s_i = 0, \quad \forall i \in \{1, \dots, M\}. \quad (5.34)$$

$\mathbf{x}^*(\mathbf{s})$ est la solution du problème d'approximation parcimonieuse (\mathcal{P}_p^R) si la position des coefficients non nuls est forcée. Notons que si \mathbf{D}_s n'est pas de rang plein, la solution $\mathbf{x}^*(\mathbf{s})$ du problème de minimisation n'est pas unique. Dans la suite de la preuve, nous nous plaçons dans le cas où $\|\mathbf{s}\|_0 \leq N$ et tout sous-groupe de N colonnes de \mathbf{D} forme une famille de vecteurs linéairement indépendants. Ces deux conditions impliquent que \mathbf{D}_s est de rang plein $\forall \mathbf{s}$. Le cas général, quoique plus complexe, se traite de façon similaire.

La solution du problème d'approximation parcimonieuse (\mathcal{P}_0^R) peut alors être reformulée comme

$$\mathbf{x}^* = \mathbf{x}^*(\mathbf{s}^*) \text{ avec } \mathbf{s}^* = \underset{\mathbf{s} \in \{0,1\}^M}{\operatorname{argmin}} f(\mathbf{x}^*(\mathbf{s})). \quad (5.35)$$

De façon similaire, soit $g(\mathbf{x}) \triangleq -\log p(\mathbf{y}, \mathbf{x}, \mathbf{s})$ et $\hat{\mathbf{x}}(\mathbf{s})$ la solution de

$$\hat{\mathbf{x}}(\mathbf{s}) = \underset{\mathbf{x}}{\operatorname{argmax}} \log p(\mathbf{y}, \mathbf{x}, \mathbf{s}). \quad (5.36)$$

Le problème d'estimation (5.10) peut être alors reformulé comme

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}(\hat{\mathbf{s}}) \text{ avec } \hat{\mathbf{s}} = \underset{\mathbf{s} \in \{0,1\}^M}{\operatorname{argmin}} g(\hat{\mathbf{x}}(\mathbf{s})). \quad (5.37)$$

Le théorème 10 est prouvé en montrant que $\hat{\mathbf{x}}(\mathbf{s}) = \mathbf{x}^*(\mathbf{s})$ et $g(\hat{\mathbf{x}}(\mathbf{s})) = f(\mathbf{x}^*(\mathbf{s}))$, $\forall \mathbf{s}$ sous les hypothèses considérées.

Pour faciliter l'écriture, on suppose que les L premiers coefficients de \mathbf{s} sont non nuls (cette hypothèse ne limite pas le cadre de travail défini précédemment). Si \mathbf{D}_s est la matrice faite des L premières colonnes de \mathbf{D} et

\mathbf{D}_s^+ est sa pseudo-inverse, on a

$$x_i^*(\mathbf{s}) = \begin{cases} (\mathbf{D}_s^+ \mathbf{y})_i & \text{si } i \in \{1, \dots, L\}, \\ 0 & \text{sinon.} \end{cases} \quad (5.38)$$

D'un autre côté, la solution de (5.37) s'écrit :

$$\hat{x}_i(\mathbf{s}) = \begin{cases} \left(\mathbf{D}_s^T \mathbf{D}_s + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_L \right)^{-1} \mathbf{D}_s^T \mathbf{y}_i & \text{si } i \in \{1, \dots, L\}, \\ 0 & \text{sinon.} \end{cases} \quad (5.39)$$

Il est clair que $\lim_{\sigma_x^2 \rightarrow +\infty} \hat{\mathbf{x}}(\mathbf{s}) = \mathbf{x}^*(\mathbf{s})$. Utilisant ce résultat et prenant en compte le modèle (5.2)-(5.5), on obtient

$$\lim_{\sigma_x^2 \rightarrow +\infty} g(\hat{\mathbf{s}}) = \frac{\|\mathbf{y} - \mathbf{D}\mathbf{x}^*(\mathbf{s})\|_2^2}{2\sigma^2} + \log p(\mathbf{s}) + \lim_{\sigma_x^2 \rightarrow +\infty} \frac{\sum_{i=1}^M (x_i^*(\mathbf{s}))^2}{2\sigma^2}. \quad (5.40)$$

Le dernier terme tend vers 0 lorsque σ_x^2 tend vers $+\infty$. De plus, si $p_i = p$, $\forall i \in \{1, \dots, M\}$, alors $\lambda_i = \lambda$, $\forall i \in \{1, \dots, M\}$ et $p(\mathbf{s}) \propto \exp(\lambda \|\mathbf{s}\|_0)$.

Enfin, nous avons posé par hypothèse que $\|\mathbf{x}^*(\mathbf{s})\|_0 \triangleq \|\mathbf{D}_s^+ \mathbf{y}\|_0 = \|\mathbf{s}\|_0$, $\forall \mathbf{s}$. Alors puisque $\mu = 2\sigma^2\lambda$, on a $g(\hat{\mathbf{x}}(\mathbf{s})) = f(\mathbf{x}^*(\mathbf{s}))$, $\forall \mathbf{s}$ et nécessairement $\hat{\mathbf{x}} = \mathbf{x}^*$.

5.6.2 Calcul de l'expression du seuil (5.16) et de l'équation de mise à jour des coefficients (5.17) dans BMP

Pour un j , solution de (5.13) à l'itération n , on cherche le couple $(\hat{s}_j^{(n)}, \hat{x}_j^{(n)})$ tel que

$$(\hat{s}_j^{(n)}, \hat{x}_j^{(n)}) = \underset{(s_j, x_j)}{\operatorname{argmax}} \log p(\mathbf{y}, \hat{\mathbf{x}}_j^{(n-1)}, \hat{\mathbf{s}}_j^{(n-1)}). \quad (5.14)$$

Soit $F(s_j, x_j) \triangleq \log p(\mathbf{y}, \hat{\mathbf{x}}_j^{(n-1)}, \hat{\mathbf{s}}_j^{(n-1)})$. Exploitant le modèle Bernoulli-Gaussien (5.2)-(5.5), on obtient

$$F(s_j, x_j) \propto -\frac{1}{2} \left(\frac{s_j}{\sigma^2} + \frac{1}{\sigma_x^2} \right) x_j^2 + \frac{1}{\sigma^2} s_j (\hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle) x_j - \lambda_j \|s_j\|_0. \quad (5.41)$$

Supposons $\hat{s}_j^{(n)} = 0$, alors

$$\begin{aligned} \hat{x}_j^{(n)}(0) &= \underset{x_j}{\operatorname{argmax}} F(0, x_j), \\ &= \underset{x_j}{\operatorname{argmax}} -\frac{1}{2\sigma_x^2} x_j^2, \\ &= 0, \end{aligned} \quad (5.42)$$

et

$$F(0, \hat{x}_j^{(n)}(0)) = 0. \quad (5.43)$$

Supposons $\hat{s}_j^{(n)} = 1$, alors

$$\begin{aligned} \hat{x}_j^{(n)}(1) &= \operatorname{argmax}_{x_j} F(1, x_j), \\ &= \operatorname{argmax}_{x_j} -\frac{\sigma^2 + \sigma_x^2}{2\sigma^2\sigma_x^2} x_j^2 + \frac{1}{\sigma^2} (\hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle) x_j. \end{aligned} \quad (5.44)$$

La fonction objectif étant strictement concave, l'optimum peut être obtenu en dérivant par rapport à x_j et égalisant l'expression obtenue à 0 :

$$-\frac{\sigma^2 + \sigma_x^2}{\sigma^2\sigma_x^2} \hat{x}_j^{(n)}(1) + \frac{1}{\sigma^2} (\hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle) = 0, \quad (5.45)$$

soit

$$\hat{x}_j^{(n)}(1) = \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} (\hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle). \quad (5.46)$$

Pour $(\hat{s}_j^{(n)}, \hat{x}_j^{(n)}) = (1, \hat{x}_j^{(n)}(1))$,

$$F(1, \hat{x}_j^{(n)}(1)) = \frac{1}{2\sigma^2} \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2} \left(\hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle \right)^2 - \lambda_j. \quad (5.47)$$

Les deux expressions (5.42) et (5.46) peuvent s'écrire sous une forme unique dépendant de $\hat{s}_j^{(n)}$. On retrouve alors la formule (5.17) :

$$\hat{x}_j^{(n)} \triangleq \hat{s}_j^{(n)} \left(\hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle \right) \frac{\sigma_x^2}{\sigma^2 + \sigma_x^2}. \quad (5.17)$$

Le seuil T_j est, quant à lui, défini par la condition d'activation de $\hat{s}_j^{(n)}$: $\hat{s}_j^{(n)}$ sera égal à 1 si

$$F(1, \hat{x}_j^{(n)}(1)) > F(0, \hat{x}_j^{(n)}(0)),$$

soit

$$\left(\hat{x}_j^{(n-1)} + \langle \mathbf{r}^{(n-1)}, \mathbf{d}_j \rangle \right)^2 > 2\sigma^2 \frac{\sigma^2 + \sigma_x^2}{\sigma_x^2} \lambda_j. \quad (5.48)$$

On peut alors poser la définition (5.16) : $T_j \triangleq 2\sigma^2 \frac{\sigma^2 + \sigma_x^2}{\sigma_x^2} \lambda_j$.

5.6.3 Calcul de l'équation de mise à jour des coefficients (5.23) dans BOMP

Développons le problème d'estimation (5.22) :

$$\begin{aligned}
 \hat{\mathbf{x}} &= \operatorname{argmax}_{\mathbf{x}} \log p(\mathbf{y}, \mathbf{x}, \hat{\mathbf{s}}^{(n)}). & (5.22) \\
 &= \operatorname{argmax}_{\mathbf{x}} -\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} \mathbf{x}_{\hat{\mathbf{s}}^{(n)}}\|_2^2 - \frac{1}{2\sigma_x^2} \mathbf{x}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{x}_{\hat{\mathbf{s}}^{(n)}}, \\
 &= \operatorname{argmax}_{\mathbf{x}} -\frac{1}{2} \mathbf{x}_{\hat{\mathbf{s}}^{(n)}}^T \left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right) \mathbf{x}_{\hat{\mathbf{s}}^{(n)}} + \mathbf{x}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y}.
 \end{aligned}$$

La fonction objectif étant strictement concave, l'optimum peut être obtenu en dérivant par rapport à $\mathbf{x}_{\hat{\mathbf{s}}^{(n)}}$ et égalisant l'expression obtenue à 0 :

$$-\left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right) \hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)} + \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y} = 0, \quad (5.49)$$

soit l'expression (5.23),

$$\hat{\mathbf{x}}_{\hat{\mathbf{s}}^{(n)}}^{(n)} = \left(\mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{D}_{\hat{\mathbf{s}}^{(n)}} + \frac{\sigma^2}{\sigma_x^2} \mathbf{I}_{\|\hat{\mathbf{s}}^{(n)}\|_0} \right)^{-1} \mathbf{D}_{\hat{\mathbf{s}}^{(n)}}^T \mathbf{y}. \quad (5.23)$$

Conclusion

Les travaux réalisés pendant cette thèse ont permis de creuser des voies et, peut-être, d'en inspirer d'autres. Ce dernier chapitre synthétise les contributions de cette thèse, et pour chacune, définit quelques axes de recherche qu'il semble pertinent d'explorer dans le cadre de travaux futurs.

Durant ces trois ans, nous nous sommes intéressés à l'utilisation de décompositions parcimonieuses dans les schémas de compression. Notre étude nous a amenés à considérer des modèles probabilistes, ces modèles permettant l'emploi de méthodes d'estimation Bayésiennes.

Transformations adaptatives

Le chapitre 3 s'est penché plus particulièrement sur le lien entre parcimonie et codage par transformation. Une étude des coûts de codage des coefficients transformés puis quantifiés montrent que si la redondance du dictionnaire utilisé dans la transformation peut être intéressante du point de vue de l'approximation du signal à compresser, le coût de codage des coefficients correspondant peut être rédhibitoire. L'idée de structuration du dictionnaire est alors introduite et envisagée sous la forme d'un ensemble de bases. Dans ce contexte, un schéma de compression est proposé, optimisant la transformée appliquée sur l'image dans le domaine spatial, par une adaptation du support via un bintree, et dans le domaine transformé, par la sélection de bases locales parmi des ensembles finis. Deux cas sont alors étudiés : dans l'un, les ensembles de bases locales sont prédéfinis (nous considérons des DCT directionnelles de Zeng et Fu [122]), dans l'autre, il s'agit d'ensembles appris avec l'algorithme de Sezer *et al.* [96]. Dans ces deux cas, des comparaisons de performances en termes de débit *vs* distorsion sont réalisées entre les schémas proposés et les standards de compression JPEG et JPEG2000.

Partant d'une analyse approfondie de l'algorithme de Sezer, une extension

probabiliste est ensuite développée, sur la base d'un algorithme EM. Nous montrons que les bases apprises avec ce nouvel algorithme améliorent la parcimonie des décompositions par rapport à celles apprises par l'algorithme de Sezer. En revanche, elles ne résolvent pas une faiblesse mise en évidence déjà dans l'algorithme de Sezer : les atomes des décompositions parcimonieuses sont choisis "uniformément" de sorte qu'un regroupement des coefficients non nuls est difficile et inefficace dans le codage RLE des indices des coefficients non nuls.

Perspectives

Dans de futurs travaux, il serait intéressant de s'interroger sur l'optimisation du nombre de bases apprises. Ici, nous avons proposé des ensembles de 7 bases, en raison de l'initialisation de l'algorithme d'apprentissage par les DCT directionnelles. Mais rien ne prouve que, dans le contexte des schémas de compression considérés, ce nombre de bases conduise aux meilleures performances en termes de débit *vs* distorsion.

Prédiction et parcimonie

Le chapitre 4 s'intéresse au codage prédictif, toujours d'un point de vue parcimonieux. Un nouvel algorithme de prédiction est élaboré, reposant sur un mélange de décompositions parcimonieuses. Pour ce faire, on considère un modèle probabiliste similaire à celui utilisé dans le chapitre précédent, pour le développement d'un nouvel algorithme d'apprentissage de bases. L'algorithme considéré ici se propose alors de résoudre un problème d'estimation MMSE approché. Ses performances sont évaluées en termes de débit *vs* distorsion sur un schéma de codage simple, reposant sur une segmentation de l'image en blocs de taille fixe et utilisant un ensemble de DCT directionnelles. Par ailleurs, on considère deux critères d'optimisation de la prédiction différents. L'algorithme est comparé à des méthodes de prédiction existantes, deux basées également sur des décompositions parcimonieuses et une reprenant le principe de la prédiction intra utilisée dans le standard de compression vidéo H.264. Nous montrons que l'algorithme proposé présente un bon comportement général, dans la limite des images testées. Afin de valider totalement la pertinence de notre algorithme, il serait intéressant, dans des travaux futurs, d'envisager son intégration dans un codeur vidéo complet de type H.264.

Perspectives

Dans [118], Yu *et al.* proposent un algorithme de débruitage basé sur des approximations parcimonieuses. Le débruitage est réalisé sur un découpage de l'image en blocs de taille fixe et chaque bloc débruité est estimé par son approximation parcimonieuse dans une base choisie dans un ensemble appris sur l'image bruitée. L'apprentissage des bases est réalisé par des analyses en composantes principales (PCA). S'inspirant de leurs travaux, nous pouvons envisager un nouvel algorithme de débruitage. Notre contribution serait double. D'abord, il pourrait être intéressant de remplacer les PCA par l'algorithme d'apprentissage proposé dans le chapitre précédent. En effet, de même que dans l'algorithme proposé par Sezer *et al.*, les PCA sont calculées sur des ensembles de blocs de l'image et non sur l'image entière; l'algorithme proposé dans le chapitre précédent permettrait une prise en compte de l'ensemble de l'image dans l'apprentissage de toutes les bases, via des calculs de probabilités a posteriori. Ensuite, l'algorithme de prédiction proposé dans ce chapitre pourrait être utilisé en lieu et place de la sélection d'une base d'approximation. L'estimation de chaque bloc ne serait alors plus son approximation parcimonieuse dans une base parmi un ensemble possible, mais la somme pondérée des approximations parcimonieuses dans toutes les bases.

Algorithmes gloutons Bayésiens

Enfin, le dernier chapitre de contributions revient sur les observations faites dans le chapitre 3 concernant l'"uniformisation" de la sélection des atomes utilisés dans les décompositions parcimonieuses. Ici, on s'intéresse en particulier à l'étude d'un modèle Bernoulli-Gaussien : celui-ci permet en effet, via des paramètres de Bernoulli différents, la prise en compte d'une certaine hiérarchisation des atomes. L'utilisation de ce modèle dans la dérivation d'algorithmes de recherche de décompositions parcimonieuses est étudiée, on montre ainsi l'équivalence d'un problème d'estimation MAP basé sur ce modèle avec un des problèmes de décompositions parcimonieuses standard. Plusieurs algorithmes sont alors proposés, rappelant les algorithmes de poursuite de la littérature. Leurs performances en reconstruction sont comparées sur des données synthétiques générées selon différents modèles. Pour simplifier, on ne considère ici que le cas où les paramètres de Bernoulli

sont égaux. Les résultats obtenus, évalués en termes de taux de détection manquée *vs* SNR, sont encourageants et montrent un bon comportement de certains d'entre eux.

Perspectives

Compte tenu des motivations qui nous ont conduits à considérer un modèle Bernoulli-Gaussien, une perspective de ce travail serait la conception d'un algorithme d'apprentissage de dictionnaires basé sur ce même modèle. Considérant d'abord des paramètres de Bernoulli égaux, il faudrait ensuite se pencher sur une hiérarchisation des atomes. Il peut s'agir dans un premier temps de paramètres de Bernoulli différents (on force a priori une certaine sélection des atomes) ou suivant une certaine loi de probabilités (Zhou *et al.* envisagent dans [124] une loi Beta permettant le recours à des méthodes d'approximation variationnelles Bayésiennes). Dans un deuxième temps, on peut envisager des modèles plus complexes, liant le choix d'un atome à un autre par des probabilités conditionnelles. Ces travaux vont dans le sens d'une parcimonie plus structurée. Et en effet, compte tenu des résultats de cette thèse et de récentes contributions de la littérature ([118]), cet axe de recherche semble particulièrement prometteur.

Bibliographie

- [1] M. Aharon, M. Elad, and A. Bruckstein. K-svd : An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. On Signal Processing*, 54(11) :4311–4322, November 2006.
- [2] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Trans. On Computers*, C-23 :90–93, January 1974.
- [3] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-based compressive sensing. To appear in *IEEE Trans. On Information Theory*, 2010.
- [4] D. Baron, S. Sarvotham, and R. G. Baraniuk. Bayesian compressive sensing via belief propagation. Technical report, available at <http://arxiv.org:0812.4627v2.pdf>, June 2009.
- [5] J. O. Berger. *Statistical decision theory and Bayesian analysis*. Springer-Verlag, 2nd edition, 1985.
- [6] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley & Sons Ltd., 1994.
- [7] G. Bjontegaard. Calculation of average psnr differences between rd-curves. In *VCEG-M33*, 2001.
- [8] T. Blumensath and M. E. Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5-6) :629–654, December 2008.
- [9] J. Bobin, J.-L. Starck, J. M. Fadili, Y. Moudden, and D. L. Donoho. Morphological component analysis : An adaptive thresholding strategy. *IEEE Trans. On Image Processing*, 16(11) :2675–2681, 2007.
- [10] E. J. Candès and D. L. Donoho. Curvelets : A surprisingly effective nonadaptive representation for objects with edges. Technical report, Stanford University CA, Dept. of Statistics, 2000.

- [11] G. Casella and C. P. Robert. *Monte Carlo Statistical Methods*. Springer-Verlag New York, LLC, 2nd edition, 2004.
- [12] V. Chappelier and C. Guillemot. Oriented wavelet transform for image compression and denoising. *IEEE Trans. On Image Processing*, 15(10) :2892–2903, October 2006.
- [13] C.-T. Chen. Adaptive transform coding via quadtree-based variable blocksize dct. In *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 23-26 May 1989.
- [14] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *Int'l Journal of Control*, 50(5) :1873–1896, 1989.
- [15] S. S. Chen and D. L. Donoho. Basis pursuit. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, volume 1, pages 41–44, 1994.
- [16] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20 :33–61, 1998.
- [17] R. R. Coifman, Y. Meyer, and M. V. Wickerhauser. *Wavelet analysis and signal processing*. Wavelets and their applications, 1992.
- [18] P. L. Combettes and J.-C. Pesquet. A proximal decomposition method for solving convex variational inverse problems. *Inverse problems*, 24(6) :1–31, 2008.
- [19] C. Couvreur and Y. Bresler. On the optimality of the backward greedy algorithm for the subset selection problem. *SIAM Journal on Matrix Analysis and Applications*, 21(3) :797–808, February 2000.
- [20] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications, 1991.
- [21] C. Dai, O. D. Escoda, X. Yin, L. Peng, and C. Gomila. Geometry-adaptive bloc partitioning for intra prediction in image and video coding. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, volume 6, pages VI – 85 – VI – 88, 2007.
- [22] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Trans. On Information Theory*, 55(5) :2230–2249, May 2009.
- [23] Y. Dai, Q. Zhang, A. Tourapis, and C.-C. J. Kuo. Efficient block-based intra prediction for image coding with 2d geometrical manipulations. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, San Diego, CA, 2008.

- [24] I. Daubechies, M. Defrise, and C. DeMol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11) :1413–1457, 2004.
- [25] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39 :1–38, 1977.
- [26] M. N. Do and M. Vetterli. Contourlets : a directional multiresolution image representation. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, volume 1, pages 357–360, 2002.
- [27] D. L. Donoho. De-noising by soft-thresholding. *IEEE Trans. On Information Theory*, 41(3) :613–627, May 1995.
- [28] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proc. Nat. Acad. Sci.*, 100(5) :2197–2202, March 2003.
- [29] D. L. Donoho and I. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81 :425–455, 1993.
- [30] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Technical report, Stanford University, March 2006.
- [31] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Journal on Applied and Computational Harmonic Analysis (ACHA)*, 19 :340–358, November 2005.
- [32] M. Elad and I. Yavneh. A plurality of sparse representations is better than the sparsest one alone. *IEEE Trans. On Information Theory*, 55(10) :4701–4714, October 2009.
- [33] Y. C. Eldar, P. Kuppinger, and H. Bolcskei. Compressed sensing of block-sparse signals : uncertainty relations and efficient recovery. Submitted to *IEEE Trans. On Signal Processing*, 2010.
- [34] Y. C. Eldar and M. Mishali. Robust recovery of signals from a structured union of subspaces. *IEEE Trans. On Information Theory*, 55(11) :5302–5316, November 2009.
- [35] K. Engan, S. O. Aase, and J. H. Husoy. Method of optimal directions for frame design. In IEEE Computer Society, editor, *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, pages 2443–2446, 1999.

- [36] H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3) :399–417, May-June 1963.
- [37] M. J. Fadili and J.-L. Starck. An em algorithm for sparse representation-based image inpainting. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, volume 2, pages II – 61–4, September 2005.
- [38] A. C. Faul and M. E. Tipping. Analysis of sparse bayesian learning. *Advances in Neural Information Processing Systems*, 14 :383–389, 2002.
- [39] J.-J. Fuchs. Detection and estimation of superimposed signals. In *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 3, pages 1649 – 1652, 1998.
- [40] J.-J. Fuchs. On the application of the global matched filter to doa estimation with uniform circular arrays. *IEEE Trans. On Signal Processing*, 49(4) :702–709, 2001.
- [41] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.
- [42] H. Gish and J.N. Pierce. Asymptotically efficient quantizing. *IEEE Trans. On Informatio Theory*, 14(5) :676–683, September 1968.
- [43] R. Gribonval and M. Nielsen. Sparse representations in unions of bases. *IEEE Trans. On Information Theory*, 49(12) :3320–3325, December 2003.
- [44] R. Gribonval and P. Vandergheynst. On the exponential convergence of matching pursuits in quasi-incoherent dictionaries. *IEEE Trans. On Information Theory*, 52(1) :255–261, 2006.
- [45] W. I. Grosky and R. Jain. Optimal quadtrees for image segments. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, PAMI-5(1) :77–83, January 1983.
- [46] O. G. Guleryuz. Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising : Part i - theory. *IEEE Trans. On Image Processing*, 15 :555–571, 2004.
- [47] O. G. Guleryuz. Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising : Part ii - adaptive algorithms. *IEEE Trans. On Image Processing*, 15 :555–571, 2004.
- [48] D. Haugland. A bidirectional greedy heuristic for the subspace selection problem. *Lecture Notes in Computer Science*, 4638 :162–176, 2007.

- [49] L. He and L. Carin. Exploiting structure in wavelet-based bayesian compressive sensing. *IEEE Trans. On Signal Processing*, 57(9) :3488–3497, September 2009.
- [50] M. Helsingius, P. Kuosmanen, and J. Astola. Image compression using multiple transforms. *Signal Processing. Image communication*, 15(6) :513–529, 2000.
- [51] D. A. Huffman. A method for the construction of minimum redundancy codes. *Proc. Institute of Radio Engineers (IRE)*, 40(9) :1098–1101, September 1952.
- [52] R. M. Figueras i Ventura, L. Granai, and P. Vandergheynst. R-d analysis of adaptive edge representations. In *Proc. IEEE Int'l Workshop on Multimedia Signal Processing (MMSP)*, pages 130–133, December 2002.
- [53] R. M. Figueras i Ventura, P. Vandergheynst, and P. Frossard. Low rate and scalable image coding with redundant representations. Technical report, TR-ITS-03.02, June 2003.
- [54] L. K. Jones. On a conjecture of huber concerning the convergence of projection pursuit regression. *Annals of statistics*, 15(2) :880–882, 1987.
- [55] N. G. Kingsbury and T. H. Reeves. Overcomplete image coding using iterative projection-based noise shaping. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, volume 3, pages 597–600, 2002.
- [56] J. J. Kormylo and J. M. Mendel. Maximum likelihood detection and estimation of bernoulli-gaussian processes. *IEEE Trans. On Information Theory*, IT-28(3) :482488, May 1982.
- [57] H. P. Kramer and M. V. Mathews. A linear coding for transmitting a set of correlated signals. *IRE Trans. On Information Theory*, IT-23 :41–46, September 1956.
- [58] K. Kreutz-Delgado and B. D. Rao. Focuss-based dictionary learning algorithms. In *Proc. SPIE*, volume 4119 : "Wavelet Applications in Signal and Image Processing VIII, July-August 2000.
- [59] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. On Information Theory*, 47(2) :498–519, February 2001.
- [60] E. G. Larsson and Y. Selen. Linear regression with a sparse parameter vector. *IEEE Trans. On Signal Processing*, 55(2) :451–460, January 2007.
- [61] E. LePennec and S. Mallat. Sparse geometric image representations with bandelets. *IEEE Trans. On Image Processing*, 14(4) :423–438, April 2005.

- [62] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya. Learning unions of orthonormal bases with thresholded singular value decomposition. In *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, pages v293–v296, 18-23 March 2005.
- [63] M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Comp.*, 12 :337–365, 2000.
- [64] S. P. Lloyd. Least squares quantization in pcm. Technical report, Bell Laboratories, 1957.
- [65] Y. Lou, A. Bertozzi, and S. Soatto. Direct sparse deblurring. Technical report, CAM-UCLA, 2009.
- [66] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3) :415–447, May 1992.
- [67] S. Mallat. A theory for multiresolution signal decomposition : the wavelet representation. *IEEE Trans. On Pattern Analysis Machine Intelligence*, 11(7) :674–693, July 1989.
- [68] S. Mallat. *A Wavelet Tour of Signal Processing - The Sparse Way*. Academic Press, third edition, December 2008.
- [69] S. Mallat and F. Falzon. Analysis of low bit rate image transform coding. *IEEE Trans. On Signal Processing*, 46(4) :1027–1042, April 1998.
- [70] S. Mallat and G. Yu. Super-resolution with mixing sparse estimators. Submitted to *IEEE Trans. On Image Processing*, 2009.
- [71] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. On Signal Processing*, 41(12) :3397–3415, December 1993.
- [72] M. W. Marcellin, M. Gormish, A. Bilgin, and M. Boliek. An overview of jpeg2000. In *Proc. Data Compression Conference*, pages 523–541, Snowbird, March 2000.
- [73] A. Martin, J.-J. Fuchs, C. Guillemot, and D. Thoreau. Sparse representations for image prediction. In *Proc. European Signal Processing Conference (EUSIPCO), Poznan, Poland.*, September 2007.
- [74] A. Martin, J.-J. Fuchs, C. Guillemot, and D. Thoreau. Atomic decomposition dedicated to avc and spatial svc prediction. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, pages 2492 – 2495, October 2008.
- [75] S. Matsuo, S. Takamura, K. Kamikura, and Y. Yashima. Extension of intra prediction using reference lines. In *ITU-T VCEG AF05*, 2007.

- [76] J. Max. Quantizing for minimum distortion. *IEEE Trans. On Information Theory*, 6(1) :7–12, March 1960.
- [77] J. M. Mendel. *Optimal Seismic Deconvolution*. Academic Press, 1983.
- [78] F. G. Meyer. Image compression with adaptive local cosines : A comparative study. *IEEE Trans. On Image Processing*, 11(6) :616–629, June 2002.
- [79] J. L. Murray and K. Kreutz-Delgado. An improved focuss-based learning algorithm for solving sparse linear inverse problems. In *Proc. Asilomar Conf. On Signals, Systems and Computers*, volume 1, pages 347 – 351, 2001.
- [80] D. Needell and J. A. Tropp. Cosamp : iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3) :301–321, May 2009.
- [81] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set : a strategy employed by v1 ? *Vision Research*, 37(23) :3311–3325, 1997.
- [82] A. Ortega and K. Ramchandran. Rate-distortion methods for image and video compression. *IEEE Signal Processing Magazine*, 15(6) :23–50, November 1998.
- [83] F. O’Sullivan. A statistical perspective on ill-posed inverse problems. *Statistical science*, 1(4) :502–518, 1986.
- [84] Y. C. Pati, R. Rezaifar, Y. C. Pati R. Rezaifar, and P. S. Krishnaprasad. Orthogonal matching pursuit : Recursive function approximation with applications to wavelet decomposition. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, pages 40–44, 1993.
- [85] R. Penrose. A generalized inverse for matrices. *Proc. of the Cambridge Philosophical Society*, 51 :406–413, July 1955.
- [86] L. Peotta, L. Granai, and P. Vandergheynst. Very low bit rate image coding using redundant dictionaries. In *Proc. SPIE*, volume 5207 "Wavelets : applications in signal and image processing", pages 228–239, San Diego, CA, November 2003.
- [87] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Trans. On Image Processing*, 2(2) :160–175, April 1993.
- [88] G. Rath and C. Guillemot. A complementary matching pursuit algorithm for sparse approximation. In *Proc. European Signal Processing Conference (EUSIPCO)*, 2008.

- [89] G. Rath and C. Guillemot. Sparse approximation with an orthogonal complementary matching pursuit algorithm. In *Proc. IEEE Int'l Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3325 – 3328, 2009.
- [90] L. Rebollo-Neira and D. Lowe. Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters*, 9(4) :137–140, April 2002.
- [91] I. E. G. Richardson. *H.264 and MPEG-4 Video Compression*. British Library, 2003.
- [92] J. Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of research and development*, 20(3) :198–203, May 1976.
- [93] A. Robert, I. Amonou, and B. Pesquet-Popescu. Improving intra mode coding in h.264/avc through bloc oriented transforms. In *Proc. IEEE Int'l Workshop on Multimedia Signal Processing (MMSP)*, pages 382–386, 2006.
- [94] S. Sardy, A. G. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries. *Journal of computational and graphical statistics*, 9 :361–379, 2000.
- [95] P. Schniter, L. C. Potter, and J. Ziniel. Fast bayesian matching pursuit. In *Proc. Workshop on Information Theory and Applications (ITA)*, pages 326 – 333, La Jolla, CA, January 2008.
- [96] O. G. Sezer, O. Harmanci, and O. G. Guleryuz. Sparse orthonormal transforms for image compression. In *Proc. IEEE Int'l Conference on Image Processing (ICIP), San Diego, CA., October 2008*.
- [97] C. A. Shaffer, R. Juvvadi, and L. S. Health. Generalized comparison of quadtree and bintree storage requirements. *Image Vision Computing*, 11(7) :402–412, 1993.
- [98] K. S. Shanmugam. Comments on “discrete cosine transform”. *IEEE Trans. On Computers*, C-24 :759, July 1975.
- [99] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :379–423 and 623–656, July and October 1948.
- [100] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. On Signal Processing*, 41(12) :3445–3462, December 1993.
- [101] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Trans. On Acoustics, Speech and Signal Processing*, 36(9) :1445–1453, September 1988.

- [102] E. Shusterman and M. Feder. Image compression via improved quadtree decomposition algorithms. *IEEE Trans. On Image Processing*, 3(2) :207–215, March 1994.
- [103] C. Soussen, J. Idier, D. Brie, and J. Duan. From bernoulli-gaussian deconvolution to sparse signal restoration. Technical report, CRAN/IRCCyN, January 2010.
- [104] G. J. Sullivan and R. L. Baker. Efficient quadtree coding of images and video. *IEEE Trans On Image Processing*, 3(3) :327–331, May 1994.
- [105] T. K. Tan, C. S. Boon, and Y. Suzuki. Intra prediction by template matching. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, pages 1693–1696, 2006.
- [106] D. Taubman. High performance scalable image compression with ebcot. *IEEE Trans. On Image Processing*, 9(7) :1158–1170, July 2000.
- [107] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58 :267–288, 1996.
- [108] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1 :211–244, 2001.
- [109] M. E. Tipping and A. C. Faul. Fast marginal likelihood maximisation for sparse bayesian models. In *Proc. Int'l Workshop on Artificial Intelligence and Statistics*, 2003.
- [110] M. Türkan and C. Guillemot. Sparse approximation with adaptive dictionary for image prediction. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, pages 25–28, November 2009.
- [111] J. Tropp. Greed is good : algorithmic results for sparse approximation. *IEEE Trans. On Information Theory*, 50(10) :2231–2242, October 2004.
- [112] M. Wien. Variable block-size transforms for h.264/avc. *IEEE Trans. On Circuits and Systems for Video Technology*, 13(7) :604–613, July 2003.
- [113] D. P. Wipf. *Bayesian methods for finding sparse representations*. PhD thesis, University of California, 2006.
- [114] I. A. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6) :520–540, June 1987.
- [115] M. A. Woodbury. Inverting modified matrices. In *Princeton University, Statistical Research Group, Memorandum Report*, number 42, June 1950.
- [116] X. Wu. Image coding by adaptive tree-structured segmentation. *IEEE Trans. On Information Theory*, 38(6) :1755–1767, November 1992.

- [117] Y. Ye and M. Karczewicz. Improved h.264 intra coding based on bi-directional intra prediction, directional transform and adaptive coefficient scanning. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, San Diego, CA, pages 2116–2119, October 2008.
- [118] G. Yu, G. Sapiro, and S. Mallat. Image modeling and enhancement via structured sparse model selection. In *Proc. IEEE Int'l Conference on Image Processing (ICIP)*, September 2010.
- [119] H. Zayyani, M. Babaie-Zadeh, and C. Jutten. Sparse component analysis in presence of noise using em-map. In *Proc. Int'l Conf. on Independent Component Analysis and Signal Separation*, London, 2007.
- [120] H. Zayyani, M. Babaie-Zadeh, and C. Jutten. Bayesian pursuit algorithm for sparse representation. In *Proc. IEEE Int'l Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009.
- [121] H. Zayyani, M. Babaie-Zadeh, and C. Jutten. An iterative bayesian algorithm for sparse component analysis in presence of noise. *IEEE Trans. On Signal Processing*, 57(11) :4378–4390, November 2009.
- [122] B. Zeng and J. Fu. Directional discrete cosine transforms - a new framework for image coding. *IEEE Trans. On Circuits and Systems for Video Technology*, 18(3) :305–313, March 2008.
- [123] T. Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. Technical report, Rutgers Statistics Department, April 2008.
- [124] M. Zhou, H. Chen, J. Paisley, L. Ren, G. Sapiro, and L. Carin. Non-parametric bayesian dictionary learning for sparse image representations. In *Proc. Neural and Information Processing Systems (NIPS)*, 2009.

Publications

- [A] A. Drémeau, M. Türkan, C. Herzet, C. Guillemot and J.-J. Fuchs. Spatial intra-prediction based on mixtures of sparse representations. In *Proc. IEEE Int'l Workshop on Multimedia Signal Processing (MMSP)*, Saint-Malo, France, October, 2010.
- [B] C. Herzet and A. Drémeau. Bayesian Pursuit Algorithms. In *Proc. European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, August, 2010.
- [C] A. Drémeau, C. Herzet, C. Guillemot and J.-J. Fuchs. Sparse Optimization with Directional DCT Bases for Image Compression. In *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1290-1293, Dallas, USA, March, 2010.
- [D] A. Drémeau and C. Herzet. An EM-Algorithm Approach for the Design of Orthonormal Bases adapted to Sparse Representations. In *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2046-2049, Dallas, USA, March, 2010.
- [E] C. Herzet and A. Drémeau. Sparse Representation Algorithms based on Mean-Field Approximations. In *Proc. IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2034-2037, Dallas, USA, March, 2010.
- [F] A. Drémeau, C. Herzet, C. Guillemot and J.-J. Fuchs. Anisotropic Multi-scale Sparse Learned Bases for Image Compression. In *Proc. Int'l Conference IS&T/SPIE Electronic Imaging*, volume 7543, San Jose, USA, January, 2010.

Résumé

Cette thèse s'intéresse à différentes techniques de compression d'image combinant à la fois des aspects Bayésiens et des aspects "décompositions parcimonieuses".

Deux types de compression sont en particulier examinés. Le codage par transformation, d'abord, est traité sous l'angle de l'optimisation de la transformation. L'étude de bases prédéfinies puis apprises par un algorithme de la littérature constitue une introduction à la conception d'un nouvel algorithme d'apprentissage Bayésien, favorisant la parcimonie de la décomposition. Le codage prédictif ensuite est abordé. Inspiré de contributions récentes s'appuyant sur des décompositions parcimonieuses, un nouvel algorithme de prédiction Bayésien reposant sur un mélange de décompositions parcimonieuses est proposé.

Enfin, ces travaux ont permis de mettre en évidence l'intérêt de structurer la parcimonie des décompositions. Par exemple, une pondération des atomes de la décomposition peut être envisagée via l'utilisation d'un modèle Bernoulli-Gaussien de paramètres différents. Ce modèle est considéré dans une dernière partie, pour le développement d'algorithmes de décompositions parcimonieuses.

Abstract

This thesis interests in different methods of image compression combining both Bayesian aspects and "sparse decompositions" aspects.

Two compression methods are in particular investigated. Transform coding, first, is addressed from a transform optimization point of view. The optimization is considered at two levels : in the spatial domain by adapting the support of the transform, and in the transform domain by selecting local bases among finite sets. The study of bases learned with an algorithm from the literature constitutes an introduction to a novel learning algorithm, which encourages the sparsity of the decompositions. Predictive coding is then addressed. Motivated by some recent contributions based on sparse decompositions, we propose a novel Bayesian prediction algorithm based on mixtures of sparse decompositions.

Finally, these works allowed to underline the interest of structuring the sparsity of the decompositions. For example, a weighting of the decomposition atoms can be considered by the use of a Bernoulli-Gaussian model with different parameters. This model is studied in the last part of this thesis, for the development of sparse decomposition algorithms.