New sparse representation methods; application to image compression and indexing

Joaquin Zepeda Salvatierra

N° d'ordre : 4257 ANNÉE 2010

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

Joaquin ZEPEDA SALVATIERRA

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

---

Nouvelles méthodes de représentations parcimonieuses; Application à la compression et l'indexation d'images

Thèse soutenue à Rennes le 28 octobre 2010

devant le jury composé de :

Michel BARLAUD
Professeur, Université de Nice Sophia-Antipolis / président

Francis BACH
Chercheur, INRIA / rapporteur

Jean-Luc STARCK
Directeur de recherche, CEA / rapporteur

Onur GULERYUZ
Chercheur, DOCOMO / examinateur

Christine GUILLEMOT
Directeur de recherche, INRIA / directeur de thèse

Ewa KIJAK
Maître de conférences, Université de Rennes 1 / co-directeur de thèse

To my wife, for her unconditional support and motivation.

To my children, for being living motivation.

To my parents, for their hard work and example.

To Tatio, who put forth this challenge.

Acknowledgments

I would like to thank Francis Bach, Michel Barlaud, Onur Guleryuz and Jean-Luc Starck for accepting to be members of the PhD jury and for taking the time to carefully read this manuscript. Their suggestions have been taken into account and have significantly improved the final version.

I would like to thank my thesis supervisors, Christine Guillemot and Ewa Kijak, for making available all the resources needed to carry out the work in this thesis, for opening the doors to the various interesting topics treated in this work and for their counsel these past three years.

I would also like to thank all the members of the two projects (TEMICS and TEXMEX) I belonged to during this endeavour. In particular, from TEMICS, I thank Gagan Rath for various discussions on sparse representations and because reading his articles is what made me seek a position in IRISA; Jean-Jacques Fuchs for sharing his knowledge of sparse representations without reservation; and Cedric Herzet for various discussion that shaped the direction of my research.

From project TEXMEX, I would like to thank in particular Patrick Gros for adopting me in his team and making available computational resources without which most simulations in this work would not have been possible; and Hervé Jegoú for sharing, without reservation, his contagious enthusiasm and his knowledge of image description and indexing and for providing an admirable example of what can be accomplished during and after a PhD thesis.

Contents

List of Figures

List of Tables

Chapitre 1

Introduction

Les représentations parcimonieuses utilisant des dictionnaires redondants sont devenues le couteau suisse de la communauté de traitement d'image. Utilisant la simple notion de parcimonie, les recherches ont abouti à l'élaboration d'algorithmes performants dans des domaines aussi compétitifs et bien établis que la compression, le débruitage, la classification, la restauration et le filtrage d'images.

La parcimonie se prête aisément à une explication graphique et intuitive : imaginons un grand ensemble d'images que nous coupons en petits blocs de même taille (disons $0,5 \times 0,5$ cm pour les besoins de l'illustration). En triant tous ces blocs, nous trouverons facilement des groupes de blocs qui sont très semblables entre eux, et de chaque groupe de blocs similaires nous pouvons alors choisir un bloc-représentant unique. En examinant ensuite ces représentants, nous remarquons que certains d'entre eux présentent des motifs simples et d'autres des motifs plus complexes. Certains des blocs plus complexes ressemblent même à des combinaisons de blocs plus simples. Nous pouvons donc réduire notre ensemble de représentants en supprimant les plus complexes d'entre eux qui ne sont que des combinaisons de représentants plus simples. En répétant ce processus plusieurs fois, nous obtiendrons un ensemble réduit de blocs, appelé *dictionnaire*, contenant des blocs simples (pour la plupart), appelés *atomes*, et qui peuvent être combinés de façon à reproduire (raisonnablement bien) chacun des blocs découpés originellement du grand ensemble d'images. Il n'est pas difficile d'imaginer que le nombre d'atomes conservés est inversement proportionnel au nombre d'atomes nécessaires pour reproduire l'un des blocs d'origine.

L'exercice précédent permet d'introduire, en termes simples, les notions importantes de *parcimonie*, *redondance* et *compressibilité* qui ont donné lieu à une grande quantité d'activités de recherche ces dernières années. En effet, plus le nombre d'atomes conservés à la fin de l'exercice est grand (*i.e.,* plus notre dictionnaire est *redondant*), plus il sera facile de représenter les blocs d'origine avec seulement un petit ensemble des atomes (*i.e.,* un ensemble *parcimonieux*). On parle alors de *représentations parcimonieuses* des blocs d'origine. Nous pouvons aussi dire que les blocs d'origine sont *compressibles* dans le dictionnaire construit, puisque ce dictionnaire a été construit de façon à représenter parcimonieusement (et raisonnablement bien) l'ensemble des blocs d'origine. Cet exemple souligne également la nécessité d'algorithmes d'*apprentissage* de dictionnaires, étant donné un ensemble

d'images (comme nous l'avons fait manuellement à l'exercice précédent). **C'est l'un des grands thèmes explorés dans cette thèse : nous développons une nouvelle structure de dictionnaire et un algorithme d'apprentissage permettant de rendre une classe de signaux plus compressible que le dictionnaire de référence de l'état-de-l'art.**

Il n'est pas très difficile d'imaginer comment la parcimonie peut être utilisée pour compresser les images : si les représentations des blocs de l'image obtenues sont suffisamment parcimonieuses, chaque bloc d'une image que nous souhaitons compresser peut être représenté de façon compacte par les identifiants des atomes choisis et les coefficients qui définissent la combinaison de ces atomes. Apprendre un dictionnaire pour chaque classe d'images spécifiques (comme nous l'avons fait ci-dessus pour notre ensemble image) peut alors fournir un moyen de compresser les données image. **Il s'agit d'une approche que nous explorons dans cette thèse et qui produit de bons résultats en compression d'images, supérieurs à ceux du dernier standard (JPEG2000).**

Peut-être le plus important, la parcimonie offre aux chercheurs une description mathématique simple et efficace de ce à quoi peut ressembler une image : des données qui peuvent être représentées parcimonieusement à l'aide d'un dictionnaire bien choisi. Si nous combinons ce modèle simple avec une estimation de l'image en question, la parcimonie peut s'avérer un puissant outil de traitement d'images. Par exemple, supposons que nous ayons une image bruitée, dégradée par le vieillissement ou prise avec des capteurs de mauvaise qualité. Si notre modèle de représentation parcimonieuse de l'image est adéquat, l'image nette (indisponible) sera compressible tandis que la composante de bruit ne le sera pas. Alors une représentation parcimonieuse de l'image bruitée devrait produire une version de l'image plus nette et plus attrayante. **Nous considérons dans cette thèse l'application des représentations parcimonieuses au débruitage d'images. L'utilisation du dictionnaire que nous proposons dans un schéma de débruitage fournit de meilleurs résultats qu'un dictionnaire de l'état de l'art.**

Les signaux parcimonieux et compressibles ont aussi attiré l'attention de la communauté de recherche d'images, dont le souci est de trouver, dans une base de données d'images, des images similaires à une image requête donnée. Une raison importante de l'intérêt pour la parcimonie dans cette application particulière est la faible complexité des calculs de distances entre vecteurs, lorsque ceux-ci sont parcimonieux. Cependant un nouveau problème se pose dans ce contexte : les approches de représentations parcimonieuses ont été développées par la communauté de traitement d'image dans un souci de fidélité de la reconstruction, sans considérations pour savoir si les distances calculées entre les vecteurs parcimonieux représentent correctement la similarité visuelle entre les blocs d'image correspon-

dants. Cette dernière question est en effet une préoccupation de recherche très récente. **Nous attaquons ce sujet dans le dernier chapitre de cette thèse, où nous présentons un nouveau schéma de représentation parcimonieuse plus approprié pour la recherche d'image.**

1.1 Structure du manuscrit

Cette thèse est divisée en deux parties en dehors du présent chapitre d'introduction. La première partie, composée de deux chapitres, établi un état de l'art ainsi qu'un résumé des principales contributions de la thèse dans les domaines respectifs des représentations parcimonieuses (Chapitre 2), et de la description et recherche d'images (Chapitre 3). Dans la deuxième partie, composée des quatre chapitres restants (du Chapitre 4 au Chapitre 7), nous présentons en détails les contributions de cette thèse en structuration et apprentissage de dictionnaires, en débruitage et compression d'images, et en recherche d'images utilisant des représentations parcimonieuses.

Afin de ne pas alourdir la présentation des travaux, quelques-uns des calculs et des analyses mathématiques sont détaillés dans de petites annexes à la fin de certains chapitres. Le contenu de ces annexes incluses à la fin des chapitres fait partie des contributions de la thèse.

Nous avons également inclus deux annexes à la fin du manuscrit. Annexe A constitue un rappel concis des principaux outils utilisés dans la thèse, à savoir : (*i*) l'algèbre linéaire, (*ii*) la décomposition en valeurs singulières et (*iii*) la théorie de l'information. Dans Annexe B figurent deux tables de référence utiles qui présentent les conventions utilisées dans le manuscrit : Table B.1 (Pg. 180) résume les conventions de notation et Table B.2 (Pg. 181) les acronymes.

Dans le reste de ce chapitre, nous effectuons un résumé du manuscrit chapitre par chapitre.

1.2 Partie I : Etat de l'art

La Partie I de ce manuscrit est composée des deux chapitres, l'un sur les représentations parcimonieuses, l'autre sur la description et l'indexation d'images. Pour chacun des domaines, un état de l'art et les principales contributions de la thèse sont présentées.

1.2.1 Chapitre 2 : Représentations parcimonieuses

Le Chapitre 2 commence par introduire formellement le problème d'optimisation permettant d'obtenir une représentation parcimonieuse d'un vecteur signal. Ce problème est difficile à résoudre exactement et est donc généralement résolu en utilisant l'un des nombreux algorithme méthodes gloutons tels que le *Matching Pursuit* (MP), l'*Orthogonal Matching Pursuit* (OMP) et sa version optimisée l'*Optimized Orthogonal Matching Pursuit* (OOMP). Ces trois algorithmes de la famille des *Matching Pursuit* sont décrits ainsi que l'algorithme *Basis Pursuit* (BP).

Ensuite, nous passons en revue divers algorithmes d'apprentissage de dictionnaires, y compris trois algorithmes de l'état de l'art [Aharon 2006b, Rubinstein 2010a, Mairal 2010a] que nous utilisons par la suite comme référence pour l'évaluation de nos travaux. Nous donnons ensuite plusieurs exemples d'algorithmes qui exploitent la compressibilité du signal dans différentes tâches de traitement d'image.

À la fin du chapitre, nous résumons les contributions apportées dans cette thèse dans le domaine des représentations parcimonieuses.

1.2.2 Chapitre 3 : Description et indexation d'images

Le Chapitre 3 présente un état de l'art des domaines de la description (locale) et de l'indexation des images. La description locale d'une image peut être décomposée en trois blocs fonctionnels. Le premier bloc sélectionne les régions de l'image qui peuvent être détectées de façon répétable (stable) dans différentes prises de vues d'une même scène ou d'un même objet. Dans le deuxième bloc, les régions sélectionnées sont géométriquement normalisées pour obtenir une représentation canonique de chaque région [Mikolajczyk 2005a]. Enfin, dans le troisième bloc, les valeurs d'intensité des pixels de chaque région normalisée sont utilisées pour construire un vecteur de longueur fixe, selon l'un des nombreux algorithmes de description existants [Mikolajczyk 2005b]. Dans ce chapitre, nous décrivons pour chacun des trois blocs mentionnés les différents algorithmes de la littérature.

Lorsque des descripteurs locaux sont utilisés pour décrire de grandes bases d'images, il est nécessaire d'organiser les descripteurs de la base de données dans une structure d'*index* qui permet de réduire la complexité de la recherche dans la base. Plusieurs systèmes d'indexation actuels [Sivic 2003, Philbin 2008, Nister 2006, Jégou 2008, Zepeda 2009, Zepeda 2010e] reposent pour cela sur des représentations parcimonieuses des données signal. Nous présentons les *indices de matrices creuses*, qui sont une représentation des matrices creuses consistant à ne considérer que les coefficients non nuls, utilisés pour réduire la complexité des

calculs de produits scalaires entre des vecteurs parcimonieux.

Plusieurs travaux existants qui appliquent les représentations parcimonieuses à la description et l'indexation d'image sont ensuite présentés. L'un des travaux décrit est une mise en œuvre des indices de matrices creuses en utilisant la quantification vectorielle (qui est un cas particulier des représentations parcimonieuses) [Sivic 2003]. Dans un autre travail [Jost 2008], l'hypothèse que les coefficients obtenus par MP suivent une loi de puissance décroissante permet aux auteurs d'élaguer itérativement l'ensemble des correspondances possibles de la base de données afin de réduire la complexité de la recherche. Enfin, nous présentons une application qui utilise des représentations parcimonieuses pour modéliser le sous-espace des transformations d'une région image donnée [Kokiopoulou 2008]. Ce modèle permet de réaliser un calcul de faible complexité du plus proche voisin exact d'une région, d'une manière invariante à la transformation.

À la fin du Chapitre 3, nous résumons les contributions de cette thèse sur l'utilisation des représentations parcimonieuses dans le domaine de la description et de l'indexation d'images.

1.3 Partie II : Contributions

La Partie II de cette thèse est composée de quatre chapitres décrivant les contributions de cette thèse.

1.3.1 Chapitre 4 : Iteration-Tuned Dictionaries (ITD)

Le travail présenté dans le Chapitre 4 introduit une nouvelle façon de structurer des dictionnaires redondants pour les représentations parcimonieuses. Nous appelons ce nouveau schéma *dictionnaire adapté à l'itération* ou *Iteration-Tuned Dictionary* (ITD). La motivation sous-jacente aux ITDs provient de la nature itérative des algorithmes de poursuite tels que OMP et OOMP. Ces algorithmes contraignent le nouvel atome $\mathbf{d}^i$ sélectionné à l' itération i à ajouter de nouvelles informations à la représentation tout en étant linéairement indépendant des atomes $\left[\mathbf{d}^1, \ldots, \mathbf{d}^{i-1}\right]$ choisis dans les itérations précédentes. Cette exigence sur les atomes $\mathbf{d}^l$ a motivé l'approche proposée : les ITDs sont structurés en couches, chaque couche l se composant d'un dictionnaires $\mathbf{D}_l$.

1.3.1.1 Décompositions parcimonieuses utilisant les ITDs

Les décompositions basées sur les ITDs procèdent de la même façon que les algorithmes de poursuite tel que MP, en choisissant d'abord, au début de chaque

itération i, un dictionnaire candidat $\mathbf{D}_i$ de la i-ème couche ITD. Le choix du dictionnaire est réalisé en utilisant une *loi de sélection des candidats*. Un exemple de loi de sélection des candidats possible sera donné par la suite dans un cas particulier d'ITD. Puis, étant donné le dictionnaire $\mathbf{D}_i$ sélectionné (composé des atomes $\mathbf{d}^i$), les règles de sélection de l'atome et du coefficient pour la i-ème itération de la version ITD du MP (ITD-MP) sont données par :

$$\mathbf{d} = \underset{\mathbf{d}\in\mathbf{D}_i}{\operatorname{argmax}} \left|(\mathbf{d})^\mathsf{T} \cdot \mathbf{r}^{i-1}\right|, \tag{1.1a}$$

$$\gamma_i = (\mathbf{d}^i)^\mathsf{T} \cdot \mathbf{r}^{i-1}, \tag{1.1b}$$

où $\mathbf{r}^i$ représente le i-ème vecteur résiduel

$$\mathbf{r}^i = \mathbf{r}^{i-1} - \gamma^i \cdot \mathbf{d}^i. \tag{1.2}$$

Tout comme dans le cas des décompositions utilisant le même dictionnaire (fixé) dans toutes les itérations, le nombre d'atomes L utilisé dans la représentation ITD d'un vecteur signal $\mathbf{y}$ peut être choisi, par exemple, pour satisfaire un critère d'erreur quadratique moyenne (EQM) ε^2 pré-déterminée :

$$\underset{L}{\operatorname{argmin}}\, L \text{ s.t. } |\mathbf{y} - \hat{\mathbf{y}}^L|^2 \leq d \cdot \varepsilon^2, \tag{1.3}$$

où $\mathbf{y} \in \mathbb{R}^d$ est le vecteur signal original et $\hat{\mathbf{y}}^i$ sa i-ème approximation. Cette approximation peut être exprimée par $\hat{\mathbf{y}}^i = \mathbf{S}^i\mathbf{\Gamma}^i$, où

$$\mathbf{S}^i = \begin{bmatrix}\mathbf{d}^1 & \dots & \mathbf{d}^i\end{bmatrix} \tag{1.4}$$

est la *matrice des atomes sélectionnés* et $\mathbf{\Gamma}^i = \begin{bmatrix}\gamma_1 & \dots & \gamma_i\end{bmatrix}^\mathsf{T}$ regroupe les coefficients (non nuls) correspondants.

1.3.1.2 Apprentissage des ITDs

De la discussion qui précède, il s'ensuit que le signal d'entrée de la couche i d'un ITD est constituée des vecteurs résiduels $\mathbf{r}^{i-1}$ provenants de la couche précédente, et qu'un unique atome sera sélectionné par couche i. Nous utilisons ce fait pour formuler un programme d'apprentissage itératif des ITDs selon une approche descendante.

1.3.1.3 Comparaison avec l'état de l'art

Nous comparons l'algorithme ITD proposé aux dictionnaires de l'état-de-l'art [Aharon 2006b, Rubinstein 2010a], construit également par apprentissage, et nous

montrons que à la fois ITD peut significativement dépasser les performances des méthodes de reference en terme de compressibilité du signal. Nous montrons également que les ITD améliorent les performances en compression d'images, et (dans le chaptire suivant) en débruitage.

1.3.2 Chapitre 5 : Tree-Structured Iteration-Tuned Dictionaries

Nous proposons ensuite une approche contenant plusieurs dictionnaires candidats par couche, que nous appelons *Tree-Structured Iteration-Tuned Dictionary* (TSITD), dans laquelle les dictionnaires sont organisés selon une structure arborescente. Chaque noeud k de l'arbre contient un dictionnaire candidat $\mathbf{D}_k$, et chaque atome d'un dictionnaire candidat a un fils. Ainsi, la loi de sélection des atomes est également la loi de sélection des candidats : à une couche donnée l, l'indice k du dictionnaire choisi et le descandant de l'atome choisie dans la couche $l-1$. La combinaison de la loi de sélection des candidats pour TSITD décrite ci-dessus avec le schéma d'apprentissage des dictionnaires correspondant produit des matrices des atomes sélectionnés $\mathbf{S}^i$ orthogonales :

$$(\mathbf{S}^i)^\mathsf{T}\,\mathbf{S}^i = \mathbf{I} \quad \forall i, \forall \mathbf{S}^i. \tag{1.5}$$

L'un des avantages d'obtenir des matrices $\mathbf{S}^i$ orthogonales tient au fait que la pseudo-inverse d'une matrice orthogonale est donnée par sa transposée : $(\mathbf{S}^i)^+ = (\mathbf{S}^i)^\mathsf{T}$. On pourrait envisager des versions améliorées de (1.1) obtenues en utilisant des variantes de l'algorithme de poursuite OMP et sa version optimisée OOMP basées sur les ITDs [Pati 1993, Rebollo-Neira 2002]. Pourtant, le fait que ces algorithmes reposent sur $(\mathbf{S}^i)^+$ implique que, dans le cadre des TSITDs, ces deux algorithmes sont tous deux équivalents, en terme de complexité, à l'algorithme MP. Ainsi, TSITD profite à la fois de la faible complexité de l'algorithme MP et de la meilleure représentation offerte par les algorithmes OMP et OOMP (en terme d'erreur de reconstruction à parcimonie équivalente).

1.3.2.1 Iteration-Tuned and Aligned Dictionary

Nous introduisons en suite une représentation équivalente à la structure TSITD appelée *reduced Tree-Structured Iteration-Tuned Dictionary* (rTSITD). La structure rTSITD est basée sur l'observation que la propriété TSITD (1.5) ci-dessus implique que le rang de $\mathbf{D}_k$, dont k appartient au niveau l, est $d-l+1$. Alors, avec une matrice de rotation appropriée, les atomes de $\mathbf{D}_k$ (qui appartiennent à l'*espace signal* $\mathbb{R}^d$) peuvent être exprimés de manière équivalente dans un *espace réduit* $\mathbb{R}^{d-l+1}$. Nous notons $\mathbf{D}'_k$ les atomes exprimés dans $\mathbb{R}^{d-l+1}$.

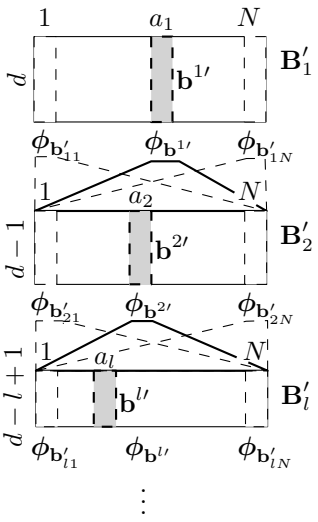

Figure 1.1: La structure ITAD : les indices a_i désignent les atomes $\mathbf{d}^{i\prime}$ (et leur matrice d'alignement correspondante $\boldsymbol{\phi}_i$) d'un même dictionnaire prototype $\mathbf{D}'_i$.

Nous en déduisons un cas particulier de la structure réduite TSITD que nous appelons *Iteration-Tuned and Aligned Dictionary* (ITAD). La structure ITAD est obtenue en contraignant les candidats $\mathbf{D}_k \in \mathbb{R}^{d\times N}$ d'une même couche l à partager la même représentation réduite $\mathbf{D}'_k = \mathbf{B}'_k \in \mathbb{R}^{(d-l+1)\times N}$ (pour tous les k de la couche l). $\mathbf{B}'_i$ est appelé *dictionnaire prototype* de la l-ème couche. A partir du dictionnaire prototype $\mathbf{B}'_l$ de la couche l, on retrouve un dictionnaire $\mathbf{D}_k$ donné par le biais d'une matrice de rotation orthonormale $\boldsymbol{\Phi}_l \in \mathbb{R}^{d\times(d-l+1)}$, factorisée sous la forme $\boldsymbol{\Phi}_k = \boldsymbol{\phi}_{\mathbf{b}^{1\prime}} \cdot \ldots \cdot \boldsymbol{\phi}_{\mathbf{b}^{l-1\prime}}$:

$$\mathbf{D}_i = (\boldsymbol{\phi}_{\mathbf{b}^{1\prime}} \cdot \ldots \cdot \boldsymbol{\phi}_{\mathbf{b}^{l-1\prime}})\, \mathbf{D}'_i, \tag{1.6}$$

$\mathbf{b}^{j\prime}$ etant l'atome choisie de la couche j. Chaque matrice $\boldsymbol{\phi}_{\mathbf{b}^{j\prime}} \in \mathbb{R}^{(d-j+1)\times(d-j)}$, $j = 1, \ldots, i-1$, appelée *matrice d'alignement*, est associée de façon unique (*cf.* Fig. 1.1) à l'un des atomes prototype parent $\mathbf{b}^{j\prime}$, $j = 1, \ldots, i-1$. Les matrices d'alignement ont pour fonction de réduire la dimension des vecteurs résiduels à la sortie des couches successives, et ceci est possible parce que chaque résidu est assuré d'être orthogonal à l'atome prototype sélectionné. De fait, n'importe quelle matrice $\boldsymbol{\phi}_{\mathbf{b}^{j\prime}}$ satisfaisant $(\boldsymbol{\phi}_{\mathbf{b}^{j\prime}})^{\mathsf{T}}\, \mathbf{b}^{j\prime} = 0$ permettra d'accomplir cette tâche. Nous choisissons par la suite comme matrices d'alignement ITAD, celles qui structurent le mieux les résidus des différents atomes d'un dictionnaire prototype donné.

La représentation compacte d'ITAD donnée dans la Fig. 1.1 illustre deux avantages d'ITAD par rapport à d'autres structures ITD. Tout d'abord, décomposer (nous montrons dans la suite comment la décomposition est réalisée) dans des couches i successivement réduites en dimension diminue la complexité combinée

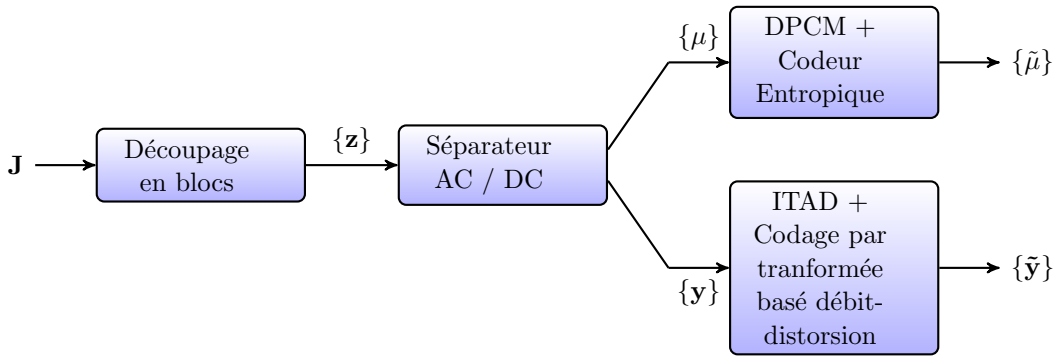

Figure 1.2: Le codec pour la compression d'images basé ITAD : L'image **J** est découpée en blocs **z** (sous forme vectorielle) sans chevauchements à l'aide d'un quadrillage régulier. Les composants DC μ et AC **y** de chaque bloc sont ensuite encodés séparément. L'opération réalisée par le bloc d'encodage en bas à droite est basée sur un nouveau critère débit-distortion permettant de déterminer la parcimonie de chaque bloc.

des opérations de décomposition/reconstruction pour des valeurs de i importantes. Cette complexité devient même plus faible que celle des systèmes traditionnels utilisant OMP avec des dictionnaires fixes. Ensuite, la structure ITAD illustrée dans la Fig. 1.1 bénéficie d'un coût de stockage plus petit que celles des autres structures ITD.

1.3.2.2 Evaluation

Nous montrons que TSITD/ITAD offrent plus d'avantages par rapport à ITD que cela soit en termes de performance, de complexité, de capacité stockage nécessaire, ou une combinaison des trois. Les évaluations sont faites en termes de (*i*) compressibilité du signal, (*ii*) compression d'image, et (*iii*) débruitage d'image.

1.3.2.3 Analyse débit-distorsion pour les dictionnaires redondants

Dans une annexe du Chapitre 5, nous introduisons une nouvelle analyse débit-distortion applicable aux codecs basés sur des dictionnaires redondants. Cette analyse facilite l'évaluation des ITDs en compression d'images, mais les expressions proposées ont des applications plus larges, comme par exemple dans le développement de nouveaux schémas d'apprentissage de dictionnaires mieux adaptés à la compression d'image.

1.3.3 Chapitre 6 : Compression d'images utilisant ITAD

Dans le Chapitre 6, nous présentons un nouveau codec d'images basé sur le dictionnaire ITAD présenté dans le Chapitre 5. Tout comme le codec JPEG, Le codec proposé encode séparément la valeur moyenne de chaque bloc (coefficient DC) et les écarts à la moyenne (coefficients AC).

1.3.3.1 L'encodeur

La Fig. 1.2 présente un diagramme en bloc illustrant les composants principaux de l'encodeur. La première opération consiste à découper l'image en blocs sans chevauchements de taille $b \times b$ appelés $\mathbf{z}$. Chaque bloc $\mathbf{z}$ est ensuite décomposé en ses coefficients DC μ (représentant la valeur d'intensité moyenne de chaque bloc) et AC $\mathbf{y}$ (écarts à la moyenne). Les séquences de coefficients DC $\{\mu\}$ et AC $\{\mathbf{y}\}$ sont ensuite encodées séparément. La séquence DC est encodée avec un code DPCM suivi d'un encodeur entropique, tandis que la séquence AC est encodée avec la transformée ITAD présentée au Chapitre 5. Un code à longueur fixe est utilisé pour coder les indices d'atomes et un quantificateur uniforme suivi d'un codage entropique pour coder les coefficients ITAD.

1.3.3.2 Détermination de la parcimonie basée sur un critère débit-distortion global

Le problème de la détermination du degré de parcimonie de chaque bloc est résolu avec une nouvelle méthode globale (*i.e.,* prenant en compte toute l'image) basée débit-distortion. Cette méthode consiste à distribuer (allouer) un atome supplémentaire à la fois à chaque bloc de l'image. Le bloc choisi à chaque itération est celui qui pour lequel l'"allocation" d'un nouvel atome offre la plus grande réduction d'erreur de reconstruction par bit. La méthode proposée est comparée avec une méthode communément utilisée qui consiste à choisir la plus petite parcimonie L (nombre d'atomes utilisés) satisfaisant une erreur quadratique moyenne ε^2 donnée :

$$\underset{L}{\operatorname{argmin}}\, L \text{ tel que } |\mathbf{y} - \hat{\mathbf{y}}^L|^2 \leq d \cdot \varepsilon^2. \tag{1.7}$$

On montre expérimentalement que la méthode proposée permet une amélioration en performance qui peut atteindre jusqu'à 0.6 dB.

1.3.3.3 Évaluation de la méthode

Nous comparons le codec basé ITAD proposé avec le codec de l'état de l'art JPEG2000 et son prédécesseur, le codec JPEG. Les évaluations montrent que

notre système peut offrir entre $2,5$ dB et $0,5$ dB de plus pour des débits entre $0,15$ et $0,45$ bits par pixel.

1.3.4 Chapitre 7 : Recherche approximative des plus proches voisins avec les représentations parcimonieuses

Dans le Chapitre 7 nous explorons l'utilisation des représentations parcimonieuses pour la recherche approximative des Plus Proches Voisins (PPV). Les représentations parcimonieuses sont intéressantes pour cette application parce que le calcul du produit scalaire entre deux vecteurs creux a une faible complexité. Supposons en effet que des vecteurs creux $\mathbf{x}$ de dimension N aient en moyenne l_0 coefficients non-nuls, le calcul du produit scalaire entre deux vecteurs ne nécessite que de l_0/N^2 multiplications en moyenne. Comme nous le montrons dans l'Annexe 7.B, cette complexité minimale est atteinte lorsque la distribution du support (*i.e.,* la position des coefficients non-nuls d'un vecteur creux) de l'ensemble des représentations creuses est uniforme (toutes les positions sont utilisées à fréquences égales). L'avantage de cette complexité réduite est exploitable en utilisant les indices de matrices creuses décrites dans le Chapitre 2.

Cette faible complexité du produit scalaire est la raison pour laquelle plusieurs travaux [Sivic 2003, Nister 2006, Philbin 2008, Jégou 2008, Zepeda 2009, Zepeda 2010e] se sont intéressés à l'utilisation de différentes variantes des re-pré-sen-ta-tions parcimonieuses pour la recherche d'images. Dans le Chapitre 7, nous proposons une extension de ces méthodes en considérant le cadre plus général des représentations parcimonieuses basées sur les dictionnaires redondants décrits dans le Chapitre 7. L'une des difficultés rencontrées dans cette tâche tient au fait que la formulation généralement établie de telles représentations parcimonieuses est basée sur un critère débit-distortion dans lequel l'erreur d'approximation (la distorsion) est minimisée sous une contrainte de parcimonie (le débit). Bien que la parcimonie soit importante dans le contexte de la recherche d'images, la distorsion ne l'est pas nécessairement.

1.3.4.1 Nouvelle formulation pour la sélection du support creux

Dans ce contexte, nous développons une nouvelle formulation produisant une nouvelle représentation parcimonieuse, appelée un *vecteur réduit*, obtenu en minimisant une *erreur d'approximation de distance*, donnée par le produit scalaire entre vecteurs creux, sous une contrainte de parcimonie (*i.e.,* de complexité de la recherche). Cette nouvelle formulation fait l'hypothèse que les données $\mathbf{y}$ sont compressibles et que leurs représentations parcimonieuses $\mathbf{x}$ sont disponibles et ont été obtenues avec un dictionnaire $\mathbf{D}$ tel que $\mathbf{y} = \mathbf{D}\mathbf{x}$. Notre but est d'approximer au

mieux le produit scalaire $\mathbf{y}_q^\mathsf{T}\,\mathbf{y}_b$ entre vecteur requête et vecteur de la base. Cette distance peut être écrite en fonction de $\mathbf{x}$ comme suit :

$$\mathbf{y}_q^\mathsf{T}\,\mathbf{y}_b = \mathbf{x}_q^\mathsf{T}\,\underbrace{\mathbf{D}^\mathsf{T}\mathbf{D}}_{\mathbf{C}_D}\mathbf{x}_b = \underbrace{\mathbf{x}_q\mathbf{C}_d}_{\boldsymbol{\chi}_q}\mathbf{x}_b = \boldsymbol{\chi}_q^\mathsf{T}\,\mathbf{x}_b. \tag{1.8}$$

On observe que la dernière expression est composée d'un produit scalaire entre un vecteur non-creux $\boldsymbol{\chi}_q$ et un vecteur creux $\mathbf{x}_b$. Le vecteur $\boldsymbol{\chi}_q$ étant non-creux, l'opération ne bénéficie plus d'une réduction en complexité. On considère donc une nouvelle représentation $\ddot{\mathbf{x}}_q$ équivalente à $\boldsymbol{\chi}_q$ aux l_0 positions $\mathcal{P} = \{p_1, \ldots, p_{l_0}\}$ du support creux et nulle ailleurs. La sélection du support creux $\mathcal{P}$ est réalisée en utilisant la formulation suivante :

$$\underset{\mathcal{P}}{\operatorname{argmin}} |(\boldsymbol{\chi}_q - \ddot{\mathbf{x}}_q)^t\mathbf{x}_b|^2 \text{ tel que } |\mathcal{P}| = l_0. \tag{1.9}$$

Nous présentons plusieurs méthodes pour résoudre ce problème.

1.3.4.2 Conditionnement des données pour une complexité minimum

Comme expliqué précédemment, la complexité du calcul du produit scalaire entre vecteurs creux est minimale lorsque le support creux des représentations est uniformément distribué sur toutes les positions possibles. Généralement les ensembles de données ne vérifient pas cette propriété, aussi, dans le Chapitre 7, nous introduisons aussi une nouvelle méthode de conditionnement des données pour traiter ce problème. Cette méthode est basée sur l'observation que, pour des dictionnaires suffisamment incohérents (*i.e.,* dont les atomes ne sont pas très corrélés entre eux), une meilleure distribution du support creux peut être obtenue avec des données uniformément distribuées sur l'hyper-sphère unitaire. Ainsi la méthode de conditionnement proposée a pour premier objectif de répartir les données le plus uniformément possible sur l'hyper-sphère unitaire. Cependant, dans le contexte de la recherche des PPVs, la méthode doit également préserver les distances relatives (produit scalaire) entre les vecteurs composant l'ensemble des données.

La méthode de conditionnement est basée sur une décomposition en valeurs singulières de la matrice de corrélation des données originales $\mathbf{y}$:

$$\mathrm{E}\left[\mathbf{y}\,\mathbf{y}^t\right] = \mathbf{U}_y^t\mathbf{D}_y\mathbf{U}_y. \tag{1.10}$$

Avec ceci, les données conditionnées $\mathbf{y}^c$ peuvent être exprimées comme suit :

$$\mathbf{y}^c = \frac{\mathbf{D}_y^{-1/2}\mathbf{U}_y\mathbf{y}}{|\mathbf{D}_y^{-1/2}\mathbf{U}_y\mathbf{y}|}. \tag{1.11}$$

1.3.4.3 Évaluation de la méthode

L'évaluation expérimentale de notre méthode de conditionnement montre que les données conditionnés sont effectivement mieux distribuées sur l'hyper-sphère unitaire et que leurs distances relatives sont préservées. En combinant le conditionnement avec la nouvelle représentation obtenue par le biais de (1.9), notre approche offre un avantage significatif sur les systèmes de recherche de PPVs utilisant des représentations parcimonieuses obtenues avec des critères débit-distortion.

Part I

Literature Review and Summary of Contributions

Chapter 2

Sparse Representations: Review and Contributions

Sparse representations have become a very active research topic in recent years. Many new algorithms have been developed that take advantage of sparse representations to achieve state-of-the-art results in a wide range of image processing applications including inpainting, denoising, and compression.

A sparse representation scheme consists of (*i*) a generally overcomplete basis (called the *dictionary*) and (*ii*) an algorithm that selects basis vectors (called the *atoms*) and weighting coefficients to produce a linear approximation of an input signal. The representation is termed sparse because only a small number of atoms / coefficients will be selected by the representation algorithm. Signal vectors that can be represented sparsely with acceptable error are termed *compressible*. Since signal compressibility depends on the dictionary used to obtain the representation, various authors have come up with new training algorithms that produce dictionaries adapted to a particular signal class. Trained dictionaries have become an important component of sparse representation schemes that has helped various authors achieve strong results.

In the current chapter we begin by formulating sparse representations as an optimization problem and then presenting various greedy algorithms used to tackle this problem. Next we carry out a survey of various dictionary training methods found in the literature. We then conclude the chapter by presenting concrete examples of algorithms that use sparse representations to achieve state-of-the-art results in image inpainting, denoising and compression.

2.1 Problem formulation

Let $\mathbf{D} \in \mathbb{R}^{d \times N}$ (with $N > d$) be the full-rank dictionary matrix $\mathbf{D}$ formed by N columns $\mathbf{d}$ called the atoms (assumed unit norm). A sparse representation $\mathbf{x}$ of a signal vector $\mathbf{y} \in \mathbb{R}^d$ is obtained under joint fidelity and sparsity criteria:

$$\underset{\mathbf{x}}{\operatorname{argmin}} |\mathbf{y} - \mathbf{D}\mathbf{x}| \text{ s.t. } |\mathbf{x}|_0 \leq L, \tag{2.1}$$

where the l-0 norm $|\cdot|_0$ counts the number of non-zero coefficients in a vector.

2.2 Sparse decomposition algorithms

The sparse decomposition problem in (2.1) is difficult to solve and in general requires a combinatorial approach wherein all possible atom combinations are tried. This approach suffers from an intractable complexity, and thus various authors have introduced an arsenal of algorithms that attack this problem in a computationally feasible manner.

2.2.1 The matching pursuit family

One well-known family of algorithms used to obtain approximate solutions of (2.1) is the *matching pursuit family* (MPF). The MPF algorithms proceed iteratively by selecting one new atom $\mathbf{d}_i$ at each iteration i. Let us concatenate all the atoms selected up to the i-th iteration to define the *selected-atoms matrix*

$$\mathbf{S}^i \triangleq \begin{bmatrix} \mathbf{d}_1 & \dots & \mathbf{d}_i \end{bmatrix}. \tag{2.2}$$

One can view the MPF algorithms as solving, at each iteration i, a simplification of the optimization in (2.1) given by

$$\underset{\mathbf{d}_i, \mathbf{\Gamma}^i}{\operatorname{argmin}} |\mathbf{y} - \begin{bmatrix} \mathbf{S}^{i-1} & | & \mathbf{d}_i \end{bmatrix} \mathbf{\Gamma}^i|, \tag{2.3}$$

where the vector $\mathbf{\Gamma}^i = \begin{bmatrix} \gamma_1 & \dots & \gamma_i \end{bmatrix}^\mathsf{T}$ contains the weighting coefficients corresponding to each atom $\mathbf{d}_i$. The resulting approximation of $\mathbf{y}$ at the output of the i-th iteration will be

$$\hat{\mathbf{y}}^i = \mathbf{S}^i \mathbf{\Gamma}^i.$$

We consider three MPF variants termed respectively matching pursuit (MP) [Mallat 1993], orthogonal matching pursuit (OMP) [Pati 1993] and optimized orthogonal matching pursuit (OOMP) [Rebollo-Neira 2002]. Each algorithm approximates (2.3) except OOMP, which solves it exactly.

2.2.1.1 Matching Pursuit (MP)

For the case of the matching pursuit (MP) algorithm, the new atom-index / coefficient pair is chosen to best approximate the residue vector

$$\mathbf{r}^{i-1} = \mathbf{y} - \hat{\mathbf{y}}^{i-1} \tag{2.4}$$

at the input of each iteration i (we assume that $\mathbf{r}^0 = \mathbf{y}$) using a single-atom representation $\gamma_i \cdot \mathbf{d}_i$. The MP atom / coefficient selection rules are:

$$\underset{\mathbf{d}_i}{\operatorname{argmax}} |\mathbf{d}_i^\mathsf{T} \cdot \mathbf{r}^{i-1}|, \tag{2.5a}$$

$$\gamma_i = \mathbf{d}_i^\mathsf{T} \cdot \mathbf{r}^{i-1}. \tag{2.5b}$$

Note that these rules solve (2.3) when the optimization is carried out only over the i-th coefficient γ_i rather than over all coefficients $\mathbf{\Gamma}_i$:

$$\underset{(\mathbf{d}_i,\gamma_i)}{\operatorname{argmin}} |\mathbf{r}^{i-1} - \gamma_i \cdot \mathbf{d}_i|. \tag{2.6}$$

While the MP atom/coefficient selection rules enjoy low complexity, they suffer from the problem that, unless there exists an atom that is collinear with $\mathbf{r}^{i-1}$ (and this happens with probability zero), the residual norm $|\mathbf{r}^i|$ at the output of the i-th iteration will never be zero even for $i \geq d$ (*i.e.,* even when there are more coefficients γ_i than there are dimensions in $\mathbf{y}$):

$$\hat{\mathbf{y}}^i = \mathbf{y} \iff i \to \infty. \tag{2.7}$$

2.2.1.2 Orthogonal Matching Pursuit (OMP)

Orthogonal matching pursuit (OMP) overcomes this problem by updating all coefficients $\mathbf{\Gamma}^i = [\gamma_1 \dots \gamma_i]^\mathsf{T}$ at iteration i using the projection unto the subspace spanned by all atoms selected up to the i-th iteration. Letting

$$\mathbf{S}^{i+} = \left((\mathbf{S}^i)^\mathsf{T} \mathbf{S}^i\right)^{-1} (\mathbf{S}^i)^\mathsf{T} \tag{2.8}$$

be the pseudo-inverse of the matrix of selected atoms $\mathbf{S}^i$ in (2.2), we can express the OMP atom / coefficient selection rules as follows:

$$\underset{\mathbf{d}_i}{\operatorname{argmax}} |\mathbf{d}_i^\mathsf{T} \cdot \mathbf{r}^{i-1}|, \tag{2.9a}$$

$$\mathbf{\Gamma}^i = \mathbf{S}^{i+} \mathbf{y}. \tag{2.9b}$$

While MP only considers an iteration's input residual $\mathbf{r}^{i-1}$ to obtain the i-th atom / coefficient pair, OMP looks back to previous iterations and optimizes the coefficients for previously selected atoms. The result is that OMP enjoys the property that

$$\hat{\mathbf{y}}^d = \mathbf{y}. \tag{2.10}$$

The property follows from that fact that $\mathbf{D}$ is assumed full rank and thus (2.9) guarantees that the selected atoms will be linearly independent. Thus $\mathbf{S}^i$ in (2.9b) is invertible for $i = d$, with $\mathbf{S}^{i+} = (\mathbf{S}^i)^{-1}$.

2.2.1.3 Optimized Orthogonal Matching Pursuit (OOMP)

The optimized orthogonal matching pursuit (OOMP) algorithm extends the idea of OMP and looks back to previous iterations not only to modify all coefficients

$\mathbf{\Gamma}^i$ but also to select a better atom $\mathbf{d}_i$ in each iteration i. The OOMP formulation solves (2.3) exactly using the following atom / coefficient selection rules:

$$\underset{\mathbf{d}_i}{\operatorname{argmax}} \left| \left[\mathbf{S}^{i-1} \;\middle|\; \mathbf{d}_i\right] \left[\mathbf{S}^{i-1} \;\middle|\; \mathbf{d}_i\right]^+ \mathbf{y} \right|, \tag{2.11a}$$

$$\mathbf{\Gamma}^i = \mathbf{S}^{i+}\,\mathbf{y}. \tag{2.11b}$$

Note that the atom $\mathbf{d}_i$ chosen at iteration i is now a function of atoms chosen in previous iterations (grouped to form $\mathbf{S}^{i-1} = \left[\mathbf{d}^1 \;\ldots\; \mathbf{d}^{i-1}\right]$). The coefficients $\mathbf{\Gamma}_i$ are chosen as for OMP and thus OOMP again enjoys the exact reconstruction property (2.10) for $i = d$.

2.2.2 Basis Pursuit

Another approach towards simplifying (2.1) consists of substituting the l-0 norm constraint by an l-1 norm constraint. The resulting algorithm, known as *basis pursuit* [Chen 2001] is usually expressed as the solution to the following problem

$$\underset{\mathbf{x}}{\operatorname{argmin}} |\mathbf{x}|_1 \text{ s.t. } \mathbf{y} = \mathbf{D}\mathbf{x}. \tag{2.12}$$

This formulation can be re-written as a linear program and thus solved with standard linear programming routines. The solution will generally consist of d (where $\mathbf{y} \in \mathbb{R}^d$) linearly independent vectors uniquely specified by $\mathbf{y}$. A sparse adaptation of this approach can be obtained by instead minimizing the reconstruction error subject to an inequality constraint on the sparsity (measured in terms of the l-1 norm); this can be formulated in Lagrangian form as

$$\underset{\mathbf{x}}{\operatorname{argmin}} |\mathbf{y} - \mathbf{D}\mathbf{x}|^2 + \lambda |x|_1. \tag{2.13}$$

2.3 Dictionary Training Methods

One common assumption made when employing sparse representations in signal processing tasks is that the signal vectors in question are compressible, meaning that they can be well-represented sparsely (*i.e.,* using only a small number of dictionary atoms). For this reason, various authors have come up with dictionary training schemes that will render a signal class (defined by a representative training set) compressible: The resulting dictionary $\mathbf{D}$ should thus produce low-error approximations $\mathbf{D}\mathbf{x}$ of the training signals $\mathbf{y}$ with sufficiently sparse $\mathbf{x}$.

Let $|\cdot|_F$ denote the Frobenius norm of a matrix. Assuming we are given a set of training vectors $\mathbf{y}$, we can write the cumulative representation error for all training vectors as follows:

$$|\mathbf{Y} - \mathbf{D}\mathbf{X}|_F^2, \tag{2.14}$$

where matrix $\mathbf{Y} = \text{cols}(\{\mathbf{y}\})$ contains the training vectors $\mathbf{y}$ as columns and matrix $\mathbf{X} = \text{cols}(\{\mathbf{x}\})$ likewise contains the corresponding sparse representations $\mathbf{x}$. Note that the coefficients matrix $\mathbf{X}$ itself depends on the dictionary $\mathbf{D}$: given $\mathbf{D}$, the columns of $\mathbf{X}$ can be calculated using, *eg.,* an algorithm of the matching pursuit family (*cf.* Section 2.2.1). Thus, dictionary training algorithms proceed iteratively by subsequently fixing either $\mathbf{D}$ or $\mathbf{X}$ while calculating the other quantity.

One of the earliest dictionary training schemes proposed is that of Olshausen and Field [Olshausen 1996]. Together with several subsequent publications, they considered maximum-likelihood estimates of the optimal dictionary when assuming a Gaussian or Laplace prior on the sparse representation coefficients. Both the coefficient and dictionary update stages used in [Olshausen 1996] are based on adaptations of steepest descent methods.

2.3.1 The method of optimal directions

The *Method of Optimal Directions* (MOD) [Engan 1999] instead used the fact that the best dictionary solution to the quadratic function (2.14) can be expressed in closed form by forcing the derivative to zero:

$$\mathbf{D} = \mathbf{Y}\mathbf{X}^{\mathsf{T}} \left(\mathbf{X}\mathbf{X}^{\mathsf{T}}\right)^{-1}, \tag{2.15}$$

thus solving in one step the dictionary update process. One drawback with this approach is the high complexity of the matrix inverse, particularly for large dictionaries.

2.3.2 Unions of orthogonal matrices

To simplify the training task, various researchers have instead considered dictionaries

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & | & \dots & | & \mathbf{D}_L \end{bmatrix} \tag{2.16}$$

that are unions of orthonormal bases $\mathbf{D}_k \in \mathbb{R}^{d \times d}$, $k = 1, \dots, L$. Here we present three such approaches.

2.3.2.1 Constrained sparse representations x

The first considered approach, proposed by Sezer *et al.* [Sezer 2008], assumes that the sparse decomposition process is constrained so that the representation uses a single basis $\mathbf{D}_k$ from $\mathbf{D}$ which is chosen based on a sparsity criteria. This constraint has the important consequence that the exact solution to the sparse representation problem (*i.e.,* under an l-0 norm constraint) is straightforward to obtain. Letting

$\mathbf{x}_k$ denote the representation that uses orthonormal bases $\mathbf{D}_k$, we can express the l-0-constrained sparse representation problem as follows:

$$\min_{\mathbf{x}_k} |\mathbf{y} - \mathbf{D}_k\mathbf{x}_k|^2 + \lambda|\mathbf{x}_k|_0. \tag{2.17}$$

The solution to the above problem is given below, where $\mathbf{d}_k^j$ denotes the j-th column of $\mathbf{D}_k$ and $\mathbf{x}_{k\langle j\rangle}$ denotes the j-th entry of $\mathbf{x}_k$:

$$\mathbf{x}_{k\langle j\rangle} = \begin{cases} 0 & \text{if } |\mathbf{y}^\mathsf{T} \cdot \mathbf{d}_k^j| < \sqrt{\lambda}, \\ \mathbf{y}^\mathsf{T} \cdot \mathbf{d}_k^j & \text{if } |\mathbf{y}^\mathsf{T} \cdot \mathbf{d}_k^j| \geq \sqrt{\lambda}. \end{cases} \tag{2.18}$$

The sparsest $\mathbf{x}_k$ (over all bases k) thus obtained specifies the selected basis $\mathbf{D}_k$ used in the decomposition.

If we now consider a large training set $\{\mathbf{y}\}$, the previous decomposition / basis selection process will partition the training matrix $\mathbf{Y} = \text{cols}(\{\mathbf{y}\})$ into *class matrices*

$$\mathbf{Y}_k = \text{cols}(\{\mathbf{y}|k = \underset{j}{\text{argmin}}(|\mathbf{x}_j|_0)\}) \tag{2.19}$$

according to the basis $\mathbf{D}_k$ used by each $\mathbf{y}$. We can likewise define the matrix $\mathbf{X}_k$ with columns consisting of sparse representations $\mathbf{x}_k$ (in $\mathbf{D}_k$) of the corresponding columns in $\mathbf{Y}_k$. Using this, each basis $\mathbf{D}_k$ can be updated by solving

$$\underset{\mathbf{D}_k}{\text{argmin}} |\mathbf{Y}_k - \mathbf{D}_k\mathbf{X}_k|_F. \tag{2.20}$$

Letting $\mathbf{U}_k$ and $\mathbf{V}_k$ denote the matrices of left and right singular vectors of $\mathbf{X}_k\mathbf{Y}_k^\mathsf{T}$ the solution [Sezer 2008] to the above is given by

$$\mathbf{D}_k = \mathbf{V}_k\mathbf{U}_k^\mathsf{T}. \tag{2.21}$$

This updated version of $\mathbf{D}_k$ is then used to re-classify the training set (by means of (2.19)) to obtain new $\mathbf{X}_k$ and $\mathbf{Y}_k$, thus defining an iterative procedure.

2.3.2.2 Relation to Gaussian Mixture Model

The work presented by Guoshen *et al.* [Yu 2010] establishes a relationship between the constrained representation scheme [Sezer 2008] outlined above and inverse problems based on Maximum a Posteriori (MAP) Expectation Maximization (EM) estimates using a Gaussian Mixture Model (GMM). The problem they consider is that of reconstructing a signal $\mathbf{f}$ from a degraded version $\mathbf{y} = \mathbf{H}\mathbf{f} + \mathbf{n}$, where $\mathbf{H}$ is a lossy operator and $\mathbf{n}$ is assumed to be Gaussian with covariance matrix $\sigma\mathbf{I}$. The GMM assumption means that the signal $\mathbf{f}$ is assumed to come from one of

K possible Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, $k = 1, \ldots, K$. Given some initial estimate $\tilde{\boldsymbol{\mu}}_k$ and $\tilde{\boldsymbol{\Sigma}}_k$ of the model parameters, the EM solution follows by iteratively *E-step* taking the estimate $\tilde{\mathbf{f}}$ to be the MAP estimate with highest confidence from all those obtained using each of the laws k and *M-step* estimating the parameters $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for each law k using the straightforward empirical estimates calculated from all the $\tilde{\mathbf{f}}$ of the corresponding law k.

The GMM / MAP-EM signal estimates $\tilde{\mathbf{f}}$ required in the M-step follow from

$$\underset{\mathbf{f}}{\operatorname{argmin}} \, |\mathbf{y} - \mathbf{H}\mathbf{f}|^2 + \sigma^2 \mathbf{f}^{\mathsf{T}} \tilde{\boldsymbol{\Sigma}}_k^{-1} \mathbf{f}, \tag{2.22}$$

which can be expressed in closed form by setting the derivative to zero. The important observation made by the authors of [Yu 2010] concerns this above expression when written in terms of the vector

$$\mathbf{x}_k = \mathbf{U}_k^{\mathsf{T}} \mathbf{f}, \tag{2.23}$$

where $\mathbf{U}_k$ is the (orthonormal) principal basis of $\tilde{\boldsymbol{\Sigma}}_k$ satisfying $\tilde{\boldsymbol{\Sigma}}_k = \mathbf{U}_k \boldsymbol{\Delta}_k \mathbf{U}_k^{\mathsf{T}}$ for a diagonal, non-negative $\boldsymbol{\Delta}_k$. Substituting (2.23) in (2.22) produces the following:

$$\underset{\mathbf{x}_k}{\operatorname{argmin}} \, |\mathbf{y} - \mathbf{H}\mathbf{U}_k\mathbf{x}_k|^2 + \sigma^2 \sum_l \frac{|\mathbf{x}_{k\langle l \rangle}|^2}{\boldsymbol{\Delta}_{k\langle l,l \rangle}}. \tag{2.24}$$

This expression has a form very similar to a sparse representation problem (*cf.* (2.1)) using an overcomplete dictionary $\mathbf{H}\big[\,\mathbf{U}_1 \;\big|\; \ldots \;\big|\; \mathbf{U}_K\,\big]$ and the same structuring constraint of [Sezer 2008] discussed above wherein a single basis $\mathbf{U}_k$ can be selected for each $\mathbf{y}$. Besides the obvious difference concerning the generalization of the dictionary structure through the loss operator $\mathbf{H}$, the coefficients $\mathbf{x}_k$ in (2.24) can be solved for each k by means of a linear operator (once again expressed in closed form by setting the derivative to zero), which is not the case in (2.18). The sparse coefficient calculation process in (2.18) is non-linear given the explicit sparsity norm used therein instead of the weighted l-2 proxy (*cf.* (2.24)) produced by the GMM assumption. Yet in both approaches, this initial coefficient calculation process is then followed by one that is non-linear consisting of selecting the basis k yielding the greatest estimation or approximation confidence.

Dictionary structuring by means of constrained decomposition schemes is a concept that we explore in the contributions of this manuscript. The GMM MAP-EM approach is further related to various approaches we present in its usage of concatenations of PCA bases obtained from partitions of the estimated data. We use PCA bases from training set partitions but retain only the first principal component of each basis when building our dictionary. Furthermore, we compute PCA bases from partitions of the training set $\{\mathbf{y}\}$ (or its i-th residual set $\{\mathbf{r}^i\}$) instead of using the set of estimates $\{\tilde{\mathbf{f}}\}$.

2.3.2.3 Unconstrained sparse representations x

An alternative approach [Lesage 2005] to the constrained sparse representation setups presented above again considers the structure in (2.16) but does not constraint the decomposition scheme to use a single $\mathbf{D}_k$ as above, instead using an algorithm such as matching pursuit applied over the entire dictionary $\mathbf{D}$. Again using $\mathbf{Y} = \text{cols}(\{\mathbf{y}\})$ and $\mathbf{X} = \text{cols}(\{\mathbf{x}\})$, the sparse approximation of all columns in $\mathbf{Y}$ can be written as

$$\mathbf{DX} = \begin{bmatrix}\mathbf{D}_1 & | & \dots & | & \mathbf{D}_L\end{bmatrix} \begin{bmatrix}\mathbf{X}_1 \\ \vdots \\ \mathbf{X}_L\end{bmatrix} = \sum_k \mathbf{D}_k \mathbf{X}_k. \tag{2.25}$$

The dictionary update then proceeds by modifying one $\mathbf{D}_l$ at a time while keeping all other bases $\mathbf{D}_k$, $k \neq l$, fixed. Let

$$\mathbf{E}_l = \mathbf{Y} - \sum_{k \neq l} \mathbf{D}_k \mathbf{X}_k \tag{2.26}$$

be the error matrix inside (2.14) with the term $\mathbf{D}_l \mathbf{X}_l$ removed. Each updated $\mathbf{D}_l$ is chosen to minimize

$$|\mathbf{E}_l - \mathbf{D}_l \mathbf{X}_l|_F^2. \tag{2.27}$$

2.3.3 The K-SVD dictionary

One could envision dictionary construction schemes based on (2.27) that instead use *undercomplete* orthonormal matrices $\mathbf{D}_l \in \mathbb{R}^{d \times n}$ with $n < d$ columns. Further considering optimizing $\mathbf{X}_l$ along with $\mathbf{D}_l$ when minimizing (2.27), the product $\mathbf{D}_l \mathbf{X}_l$ is seen to correspond to a rank n approximation of $\mathbf{E}_l$. Thus, the solution follows from the singular value decomposition (SVD) [Klema 1980] of $\mathbf{E}_l$. Letting $\mathbf{u}_k^l$, σ_k^l and $\mathbf{v}_k^l$ denote, respectively, the k-th left-singular vector, singular value, and right-singular vector of $\mathbf{E}_l$, we can express the solution as:

$$\mathbf{D}_l = \begin{bmatrix}\mathbf{u}_1^l & | & \dots & | & \mathbf{u}_n^l\end{bmatrix}, \tag{2.28}$$

$$\mathbf{X}_l = \text{diag}(\sigma_1, \dots, \sigma_n) \begin{bmatrix}\mathbf{v}_1^l & | & \dots & | & \mathbf{v}_n^l\end{bmatrix}^\mathsf{T}. \tag{2.29}$$

Indeed this is the approach taken by Aharon *et al.* to construct the K-SVD dictionary [Aharon 2006b] corresponding to the special case where $n = 1$.[1] Since this approach produces new estimates of the coefficients $\mathbf{X}_l$, Aharon *et al.* further employ a heuristic consisting of substituting the new coefficients $\mathbf{X}_l$ to obtain the

[1] Note that this general formulation for $1 < n < d$ is not proposed anywhere in the literature.

next error matrix $\mathbf{E}_{l+1}$. The sparsity of $\mathbf{X}_l$ is preserved by considering only those columns of $\mathbf{E}_l$ and $\mathbf{D}_l\mathbf{X}_l$ corresponding to non-zero columns of $\mathbf{X}_l$.

The K-SVD algorithm has originated a large amount of publications that adapt it in various ways. For example, in [Mairal 2008c], a multi-scale extension is considered that consists of forcing the atom to be null at all positions outside a small, atom-dependant square region corresponding to a quad-tree segmentation of the atom support. In [Mairal 2008b], K-SVD is used on color patches by applying the training procedure directly on the vector obtained by concatenating all three color channels. An application in image classification is also considered in [Mairal 2008a].

2.3.4 Learning schemes based on atom dependencies

2.3.4.1 Sparse dictionaries

A large number of *off-the-shelf* dictionaries exist that enjoy advantages in terms of both their negligible storage requirement and the low usage complexity relative to more recent learned (unstructured) dictionaries [Rubinstein 2010b]. The benefits of off-the-shelf dixtionaries follow since these dictionaries are defined analytically: storage of a large matrix $\mathbf{D}$ is not necessary and instead the dictionary structure is stored as a routine that carries the analysis and synthesis directly, taking advantage of the structured nature of the dictionary to lower the complexity of both operations. Many examples of such dictionaries exist in the literature, beginning with the Fast Fourier Transform (FFT) and the the Discrete Cosine Transform (DCT), all the way to the more recent wavelets and related structures (see [Rubinstein 2010b] for a chronological survey).

While learned dictionary matrices $\mathbf{D}$ suffer from an increased usage complexity and storage footprint, their adaptivity to the particular signal class is at the root of state-of-the-art results in various signal processing tasks. Thus the question comes to mind of whether it is possible to retain the best qualities of both approaches. A very recent dictionary learning formulation proposed by Rubinstein *et al.* successfully addresses this question [Rubinstein 2010a]; we refer to their method as the *Sparse Dictionary* (SD) when we compare against it in Chapter 5. Their approach considers a learned dictionary constrained to have the form

$$\mathbf{D} = \mathbf{B}\mathbf{A}, \tag{2.30}$$

where $\mathbf{B}$ is a fixed analytic dictionary and $\mathbf{A}$ is a square sparse matrix having L_a (*i.e.,* the atom sparsity) non-zero coefficients per column and is learned from the training data. Given a signal $\mathbf{y}$, one can calculate the projection coefficients via

$$\mathbf{A}^\mathsf{T}\mathbf{B}^\mathsf{T}\mathbf{y} \tag{2.31}$$

and hence use the analytic properties of $\mathbf{B}$ to compute $\mathbf{B}^\mathsf{T}\mathbf{y}$ efficiently; multiplication by $\mathbf{A}^\mathsf{T}$ also enjoys low complexity and storage as a consequence of the sparsity of $\mathbf{A}$.

As indicated by Rubinstein *et al.* the sparse matrix $\mathbf{A}$ can be seen as a model for the dependencies between the columns of the base dictionary $\mathbf{B}$: If one where to consider using $\mathbf{B}$ as a dictionary directly, one can expect that, for a redundant signal class, certain atom combinations would tend to be frequently chosen together by a decomposition algorithm. This idea of modelling dependencies between atoms is closely related to the work we present in Chapter 4 and Chapter 5 of this manuscript. The approach we follow to model the dependencies is nonetheless different, as we enforce the dependencies between atoms through a new dictionary structure and corresponding training scheme.

2.3.4.2 Tree-structured approaches

This concept of modelling atom dependencies is also at the root of the dictionary learning scheme proposed by Jenatton *et al.* [Jenatton 2009]. Their method proceeds by assuming that the atoms of the dictionary are organized into a tree, with each atom corresponding to a node $\mathcal{A}$ such that atoms can only be selected by the decomposition scheme if all ancestors are also selected. In order to use this structure to obtain a sparse representation of a signal, the authors use a sparsity measure (*i.e.,* in place of the l-0 or l-1 norms) [Zhao 2009] of the form

$$\Omega(\mathbf{x}) = \sum_{\mathcal{A}} w_{\mathcal{A}} |\mathbf{x}_{\mathcal{A}}|, \tag{2.32}$$

where $w_{\mathcal{A}}$ is a weight associated to each node $\mathcal{A}$ and $\mathbf{x}_{\mathcal{A}}$ is obtained by retaining from the sparse representation vector $\mathbf{x}$ only those positions corresponding to node $\mathcal{A}$ and its ancestors. This new sparsity metric thus enforces dependencies between atoms, and it can be substituted directly into the sparsity update stage of any dictionary learning algorithm (the authors use the training scheme described in [Mairal 2010a]).

A different approach was followed by Jost *et al.* [Jost 2006] to develop a tree-structuring scheme for arbitrary dictionaries (they focus on analytic dictionaries built from two mother functions, the 2-D Gaussian and one of its second-order derivatives). Their method learned dependencies between atoms by iteratively grouping those with high coherence to form a new atom called a *molecule*. The resulting tree structure was then useful in implementing a low complexity version of the MP algorithm.

Both of these tree-structured learned dictionary schemes are closely related to a tree-structured scheme that we present in Chapter 4. Our scheme is different

in that it consists instead of an entire dictionary in each tree-node and enjoys interesting properties such as the orthogonality of selected atoms, which is a useful property to have when considering coefficient quantization.

2.3.5 Online dictionary learning

Very recent research efforts have been directed at addressing the complexity of the dictionary training procedure in an attempt to make them faster and/or more suitable to online tasks such as processing of streaming data like video. These approaches generally rely on Stochastic Gradient Descent (SGD) methods that process a single (or a small number of) training example at a time instead of the entire training set as a batch. This endows SGD methods with lower memory requirements and can result in faster convergence rates [Mairal 2010a].

One interesting recent method is that termed the Image Signature Dictionary (ISD) [Aharon 2008]. The method sacrifices some of the dictionary redundancy in favor of compactness of the dictionary, which in turn reduces the training complexity and makes the scheme interesting for online tasks. The approach assumes that the dictionary, of size $d \times N$, can be represented as a small image of N pixels (*eg.,* of size $\sqrt{N} \times \sqrt{N}$). Each of the atoms $\mathbf{d}_k \in \mathbb{R}^d$ can be found as a $\sqrt{d} \times \sqrt{d}$ block centered around one of the N pixels of the ISD (periodically extended to deal with boundary issues). As before the dictionary update proceeds as a two-step process where the sparse representations $\mathbf{x}$ and the dictionary (or ISD, in this case) are optimized alternating between the two while keeping the other fixed. The dictionary update stage, however, is done via SGD by using a local linear model built from a single example $\mathbf{y}$ at a time.

The work of Mairal *et al.* [Mairal 2009, Mairal 2010a] considers a variation of this approach that employs a block-coordinate descent in the dictionary update stage (as done in the K-SVD algorithm). Rather than employing a first order linear model to carry out the SGD dictionary update, the approach instead minimizes the following cost using the exact block-coordinate (*i.e.,* one atom at a time while holding the remaining atoms and all sparse $\mathbf{x}$ fixed) minimum applied subsequently over each atom until convergence:

$$\mathbf{D}_{T+1} = \min_{\mathbf{D}} \frac{1}{t} \sum_{t=1}^{T} |\mathbf{y}_t - \mathbf{D}\mathbf{x}_t| + \lambda |\mathbf{x}_t|_1. \tag{2.33}$$

Note that this function differs from the empirical cost of the same form in that the representations $\mathbf{x}_t, t = 1, \ldots, T$ were calculated using older dictionaries $\mathbf{D}_{t-1}$: in keeping with the on-line spirit, at the current time T a single sample $\mathbf{y}_T$ is drawn from the training set (or training stream), decomposed using $\mathbf{D}_T$ to produce

$\mathbf{x}_T$, and then both $\mathbf{x}_T$ and $\mathbf{y}_T$ are used to update the constant coefficients of the quadratic expression (2.33), solved for $\mathbf{D}$ using the iterative block-coordinate method above described. We will refer to this approach as the *Online Learned Dictionary* (ONLD) when we compare our method against it in Chapter 5.

The contributions presented in this thesis do not consider online implementations of the proposed training methods. However, the learning schemes derived are based on adaptations of the K-means algorithm for soft-coefficient, single atom representation schemes. It is not to difficult to imagine similar adaptations of existing *online* K-means algorithms [Ripley 1996], and doing so is certainly an interesting direction that in our view offers opportunities for very competitive, low-complexity performance in the online scenario.

2.4 Applications of sparse representations in image processing

Sparse representations have a wide range of applications and in this section we provide several examples of these.

2.4.1 Inpainting

Image inpainting, which consists of filling in missing pixels of an image, is useful in several scenarios. In the context of data transmission, image inpainting can provide an alternative to channel codes for bursty erasure channels [Zepeda 2006, Rath 2004] commonly used to model packet-based network transmissions. Examples of inpainting in image manipulation include the removal of superposed text, road-signs or publicity logos [Elad 2005].

Let us consider an image patch $\mathbf{y} = \begin{bmatrix} \mathbf{y}_a^\mathsf{T} & \mathbf{y}_m^\mathsf{T} \end{bmatrix}^\mathsf{T}$ consisting of two sub-vectors: vector $\mathbf{y}_a$ contains the available pixels while $\mathbf{y}_m$ contains the missing pixels that we wish to estimate. Guleryuz [Guleryuz 2006] proposes using a concatenation of orthonormal bases that render $\mathbf{y}$ compressible to estimate the missing $\mathbf{y}_m$. Considering one such orthonormal basis $\mathbf{D}$, the compressibility assumption means that, for the affected $\mathbf{y}$, there exists some sparse vector $\mathbf{x}$ satisfying $\mathbf{y} = \mathbf{D}\mathbf{x}$. Without loss of generality we assume that $\mathbf{x} = \begin{bmatrix} \mathbf{\Gamma}^\mathsf{T} & \mathbf{0}^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and, using the orthonormality of $\mathbf{D}$, we thus write

$$\mathbf{x} = \begin{bmatrix} \mathbf{\Gamma} \\ \mathbf{0} \end{bmatrix} = \mathbf{D}^\mathsf{T}\mathbf{y} \tag{2.34}$$

$$= \begin{bmatrix} \mathbf{D}_{a,n} & \mathbf{D}_{m,n} \\ \mathbf{D}_{a,z} & \mathbf{D}_{m,z} \end{bmatrix} \begin{bmatrix} \mathbf{y}_a \\ \mathbf{y}_m \end{bmatrix}, \tag{2.35}$$

where the quantities labeled with subscripts n and z produce, respectively, the *non-zero* and *zero* components of $\mathbf{x}$. The sparsity constraint

$$\mathbf{0} = \mathbf{D}_{a,z}\mathbf{y}_a + \mathbf{D}_{m,z}\mathbf{y}_m \tag{2.36}$$

can then be used to obtain one possible estimate of the missing data

$$\hat{\mathbf{y}}_m = -(\mathbf{D}_{m,z})^+ \mathbf{D}_{a,z}\mathbf{y}_a. \tag{2.37}$$

The problem with this approach lies in the estimation of the support of $\mathbf{x}$ without which it is impossible to correctly partition $\mathbf{D}^\mathsf{T}$ as in (2.35). Instead one can build an estimate $\hat{\mathbf{y}}_m$ iteratively by subsequently enforcing (*i*) an estimate of the sparsity constraint (*i.e.,* of the support of $\mathbf{x}$) and (*ii*) the available data constraint. Let the diagonal matrices $\mathbf{\Delta}_n$, $\mathbf{\Delta}_a$ and $\mathbf{\Delta}_m$ have 1/0-valued diagonals built to select, respectively, the non-zero entries of $\mathbf{x}$ and the available and missing entries of $\mathbf{y}$. We thus express the i-th estimate of $\mathbf{y}$ as

$$\hat{\mathbf{y}}^i = \begin{bmatrix} \mathbf{y}_a \\ \hat{\mathbf{y}}^i_m \end{bmatrix} = \mathbf{\Delta}_a \hat{\mathbf{y}}^{i-1} + \mathbf{\Delta}_m \left(\mathbf{D}\mathbf{\Delta}_n \mathbf{D}^\mathsf{T}\right) \hat{\mathbf{y}}^{i-1}. \tag{2.38}$$

It can be shown [Guleryuz 2006] that the estimate $\hat{\mathbf{y}}^i_m$ thus obtained converges to that in (2.37). Yet the iterative nature of the approach in (2.38) permits an adaptive selection of the support of $\mathbf{x}$ wherein the support is re-considered after every few iterations.

2.4.2 Denoising

Sparse representations have also been used to carry out image and video denoising [Elad 2006b, Mairal 2008b, Elad 2006a, Protter 2009] by using the sparse prior on the data to formulate the denoising problem as a MAP estimation problem. The numerical solution of this MAP estimation amounts to obtaining sparse approximations of overlapping image blocks (3-D spatio-temporal blocks, for the case of video) and then averaging over all blocks to obtain the denoised data.

Consider, for example, the vector representation $\mathbf{N} \in \mathbb{R}^{r \cdot c}$ of a noisy image of size $r \times c$. We can stack all unique, overlapping $b \times b$ blocks $\mathbf{y}$ (in vectorized form) of the noisy image to form a large vector $\mathbf{Y} \in \mathbb{R}^{(r-b+1)\cdot(c-b+1)}$ and accordingly write the 1/0-valued matrix $\mathbf{\Omega}$ with i-th row specifying all positions in $\mathbf{Y}$ where pixel $\mathbf{N}_{\langle i \rangle}$ occurs. We then build $\hat{\mathbf{Y}}$ by replacing each block $\mathbf{y}$ in $\mathbf{Y}$ by its sparse approximation $\hat{\mathbf{y}} = \mathbf{D}\mathbf{x}$ chosen, *eg.,* so that the error norm $|\mathbf{y} - \hat{\mathbf{y}}|$ is similar to the noise variance. The denoised image estimate $\hat{\mathbf{N}}$ proposed in [Elad 2006b, Elad 2006a] is thus built as:

$$\hat{\mathbf{N}} = \frac{\mathbf{\Omega}\hat{\mathbf{Y}} + \lambda \mathbf{N}}{\mathbf{\Omega}\mathbf{\Omega}^\mathsf{T} + \lambda}. \tag{2.39}$$

Note that the denominator of this expression is a diagonal matrix with i-th diagonal entry given by $n_i + \lambda$, where n_i is the number of unique, overlapping blocks in the image that contain the i-th pixel. Hence the above expression just specifies an averaging of all sparse approximations of any given pixel with a relaxation using the noisy image $\mathbf{N}$ [Elad 2006b].

2.4.3 Texture separation and classification

The application of sparse representations to texture separation has been explored in a series of works [Starck 2004b, Starck 2004a, Peyré 2010] that assume that image blocks $\mathbf{y}$ consist of a mixture of overlapping component layers $\mathbf{u}_k$:

$$\mathbf{y} = \sum_k \alpha_k \mathbf{u}_k. \tag{2.40}$$

This model coincides with the case of mixtures between multiple images (*eg.,* , as can occur when taking an image through glass) or to natural images consisting of *cartoon*-like shape layers with superposed texture [Elad 2005].

Sparse representations can be adapted to this source separation problem by assuming that dictionaries $\mathbf{D}_k$ are available that render $\mathbf{u}_k$ compressible but not $\mathbf{u}_j \forall j \neq k$. Two issues come up regarding the usage of this tool: (*i*) obtaining the $\mathbf{D}_k$ and (*ii*) using them to separate the various image layers. We will focus on the more recent approach presented in [Peyré 2010] which uses a combination of off-the-shelf dictionaries (fixed for all the image patches) to model the cartoon layers, while using learned, adaptive (to the neighborhood of each image patch) dictionaries for the more complex texture layer. The formulation of the approach uses the sparse constrain to induce the image layering yet estimates the layers using a separate optimization ($\mathbf{H}$ denotes a loss matrix as in (2.22) and is set to identity for the texture separation application):

$$\begin{aligned} &\underset{\{\mathbf{u}_k, \mathbf{u}_l\}}{\operatorname{argmin}} |\mathbf{y} - \mathbf{H}\left(\sum_k \mathbf{u}_k + \sum_l \mathbf{u}_l\right)|^2 + \\ &\sum_k \min_{\mathbf{x}_k} \left(|\mathbf{u}_k - \mathbf{D}_k \mathbf{x}_k|^2 + \lambda |\mathbf{x}_k|\right) \; + \\ &\sum_l \min_{\mathbf{D}_l} \left(\sum_j \min_{\mathbf{x}_j} \left(|\mathbf{R}_j(\mathbf{u}_l) - \mathbf{D}_l \mathbf{x}_j|^2 + \lambda |\mathbf{x}_j|\right)\right). \end{aligned} \tag{2.41}$$

Here we have used subscript k to denote cartoon layers that use a fixed $\mathbf{D}_k$ and subscript l to denote the layers using adaptive dictionaries $\mathbf{D}_l$. The adaptivity of this approach hence concerns the third term in the expression: rather than carrying

out a sparse decomposition of a given texture layer $\mathbf{u}_l$ directly, the approach decomposes each such layer into many overlapping sub-blocks $\mathbf{R}_j(\mathbf{u}_l)$, $j = 1, 2, \ldots$, which will serve as a training set for the adaptive dictionary $\mathbf{D}_l$. The above problem is solved using three iterative stages (*i*) solving for the $\mathbf{x}$ using linear programming methods, (*ii*) solving for the $\mathbf{u}$ using conjugate gradient descent and (*iii*) solving for the $\mathbf{D}_l$ using gradient descent with projection unto a convex optimization domain of bounded-norm possibilities.

The learned MCA approach above described shares similarities with a new dictionary structure presented in this manuscript (Chapter 4 and Chapter 5). The novel structure proposed also assumes that the signal patches are composed of layers that become more textured with the layer index. Hence the dictionary structure is composed of multiple dictionaries (one or more per *layer* of the structure), each concerned with one of the different layers of the signal being decomposed.

2.4.4 Image compression

Yet another application of sparse representations is that of image compression, where sparse representations consisting of only a few non-zero coefficients conveniently produce compact representations of image blocks. Indeed the JPEG [Wallace 1991] standard is based on the premise that natural images are compressible in the DCT basis. Its successor, the JPEG2000 standard, instead substitutes a wavelet basis that better satisfies the compressibility requirement for natural images. An adaptation of the JPEG standard is still very much in use for intra-coding of video frames in the H.264 standard [Sullivan 2005].

2.4.4.1 The K-SVD facial image encoder

Recent successful attempts have been made towards using learned overcomplete dictionaries adapted to a signal class for the purpose of image compression. By using a learned dictionary, the image encoder can benefit from the resulting greater compressibility of the considered signal class. An example of this approach is embodied in a facial image codec introduced by Bryt and Elad [Bryt 2008] based on the learned K-SVD dictionary [Aharon 2006b] (*cf.* Section 2.3.3). Their approach nonetheless employs a piecewise-affine warping of the face that ensures that the various facial features coincide with those of a pre-specified face template. Each non-overlapping block (taken over a regular grid) of the face template (corresponding roughly to a facial feature such as the nose) thus defines a class of signals that is then represented with a corresponding K-SVD dictionary. Thus the compressibility of the image blocks in that approach relies, to a large extent, not on the K-SVD dictionary but rather on the affine warping procedure. This warping pro-

cedure in turn increases the codec complexity and is further sensitive to image variations encountered in practical scenarios (*eg.,* in lighting conditions, pose and particularities of the subject). The approach nonetheless paid off and enjoyed a wide improvement in PSNR over the state-of-the-art JPEG2000 [Adams 2005] algorithm, although this comes at the expense of a large storage overhead related to the feature-dependent dictionaries.

2.4.4.2 Sparse orthonormal transforms for image encoding

Another example of an image compression system based on trained overcomplete dictionaries is that developed by Sezer *et al.* [Sezer 2008]. Their dictionary structure (*cf.* Section 2.3.2) consists of a concatenation of orthonormal bases. Each image block is encoded using a single one of these bases. This approach reduces the overhead related to coding the atom indices at the expense of reducing the effective size of the dictionary.

2.5 Contributions (1 of 2)

In this section we summarize the contributions of this thesis in the field of sparse representations for image processing (presented in Chapter 4, Chapter 5 and Chapter 6). See Section 3.4 for a summary of contributions in the field of image description and indexing.

2.5.1 Iteration-Tuned Dictionaries (ITDs): A new overcomplete dictionary framework

In Chapter 4 we introduce a new learned overcomplete dictionary framework applicable to sparse representations called an *Iteration-Tuned Dictionary* (ITD). We show how ITDs can be used to outperform the state-of-the-art trained dictionary K-SVD (*cf.* Section 2.3.3) in terms of sparsity versus PSNR and obtain state-of-the art results in image compression and denoising.

ITDs are layered dictionaries where each layer $i = 1, 2, \ldots, L_M$ contains a set $\{\mathbf{D}_i\}$ of dictionary matrices called the *candidate dictionaries*. One $\mathbf{D}_i$ is chosen for use during the i-th iteration of a pursuit decomposition using a *candidate selection law*; the chosen $\mathbf{D}_i$ will be better adapted to the structural components of the residues at the input of the corresponding iteration.

Chapter 4 begins by formalizing the concept of ITDs and then deriving a general ITD training scheme applicable to any given candidate selection law. In the same chapter we then introduce two ITD instances. The first one, the *Basic Iteration-Tuned Dictionary* (BITD) consists of a single candidate dictionary per

layer and hence the candidate selection law is trivial. The second one, *Tree-Structured Iteration-Tuned Dictionary* (TSITD), consists of candidates $\mathbf{D}_i$ that are arranged into a tree structure across all layers. We carry out evaluations of the new dictionary structures against the state-of-the-art K-SVD dictionary and the (over)complete DCT dictionary and show that they outperform the reference dictionaries in terms of (*i*) PSNR versus sparsity and in (*ii*) image denoising and (*iii*) image compression applications.

2.5.2 The Iteration-Tuned and Aligned Dictionary (ITAD)

In Chapter 5 we first develop a new equivalent representation of the TSITD structure called the *reduced Tree-Structured Iteration-Tuned Dictionary* (rTSITD). The rTSITD structure takes advantage of the orthogonality of the TSITD selected-atoms matrix to represent candidate dictionaries in spaces of dimensionality $d - i + 1$ that reduces with the layer index i. We show how rTSITD can enjoy both reduced complexity and a smaller storage footprint than the equivalent TSITD representation. We then propose a particular rTSITD dictionary called the *Iteration-Tuned and Aligned Dictionary* (ITAD). ITAD offers several advantages over both BITD and TSITD. For one thing, the ITAD storage footprint is smaller than that of either TSITD or BITD. This in turn implies that the ITAD training process is accordingly less complex. Yet ITAD retains several (r)TSITD properties including the orthogonality of its selected-atoms matrices and the large dictionary redundancy implicit in the tree-structuring of the candidate dictionaries. As a result of these, ITAD will outperform TSITD in the vast majority of the practical (*eg.,* regarding the size of the training set) setups considered and BITD in all setups considered.

2.5.3 Rate-distortion analysis for overcomplete dictionaries

In order to avoid codec design challenges (and the ensuing comparison biases) related to evaluating an overcomplete dictionary's performance in image compression, Chapter 4 includes a new rate-distortion analysis applicable to codecs based on overcomplete dictionaries. The analysis accounts both for the increased complexity of the rate control mechanism (which involves both sparsity selection and quantizer resolution) as well as the non-orthogonality of the selected-atoms matrix. The derived expressions have greater application, for example, in the design of new training algorithms.

2.5.4 New ITD-based image codec

In Chapter 6 we further implement a complete codec based on the ITAD structure and show how it outperforms the state-of-the art JPEG2000 and its predecessor JPEG. The codec uses the ITAD transform to encode the high-frequency components of image blocks. We also introduce a novel architecture for the selection of the sparsity of each image block based on a global (image-wide) rate-distortion criterion.

Chapter 3

Image Description and Indexing: Review and Contributions

The aim of *image description* [Jia 2008] is to allow a user to automatically establish visual similarities between a query image and one or more reference images. Yet the process of establishing matches between images is complicated because different users can perceive different amounts of similarity between the same pair of images. This ambiguity in turn translates into a difficulty in expressing an image similarity function mathematically. In order to cast the problem into a familiar framework, the various approaches found in the literature generally consist of extracting one or multiple fixed-length *image descriptor* vectors from each image. The descriptors are built so that their distance under some metric roughly translates to visual similarity. Thus the problem of establishing matches between images is cast into a more familiar vector-space search problem [Zezula 2006].

One can roughly classify description schemes into approaches that produce multiple *local descriptors* extracted from sub-regions of the image and those that produce a single *global descriptor* extracted from the entirety of the image. Local descriptors are more flexible than their global counterparts because they can be used to launch queries locally (*i.e.,* from a small, user selected image area). Their local nature also makes it easier to achieve invariance to image transformations arising when shots are taken at different times or from different perspectives. Yet the increased flexibility of local descriptors comes at the expense of search complexity, as many more descriptors have to be processed at query time. This increased complexity is addressed by means of adequate indexing structures that permit low complexity, approximate distance calculations.

3.1 Local image description

As illustrated in the block diagram of Fig. 3.1, local image description can be seen as a a three-step process consisting of (*i*) detecting transformation-covariant regions, (*ii*) normalizing the transformation undergone by these regions and (*iii*) describing the resulting normalized regions [Mikolajczyk 2005a]. In the following

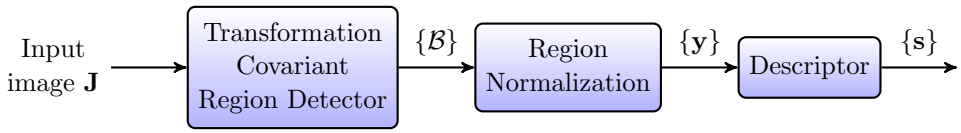

Figure 3.1: A set of affine covariant regions are detected in an input image. Each region is then affine normalized. A descriptor $\mathbf{s}$ is then calculated on each normalized region.

discussion we will review various algorithms currently used to implement the task of these three components of image description systems.

3.1.1 Transformation-covariant region detectors

A *transformation-covariant region* is a set of connected pixel positions $\mathcal{B}$ that varies along with the set of transformations that concern the particular image description system. For example, if the description system is only concerned with invariance to scale changes, a transformation-covariant region detector aims to detect the same set of (correspondingly scaled) regions $\{\mathcal{B}\}$ in any given scaled versions of the same image. The region detection component of the SIFT algorithm is a good example of a scale invariant region detector [Lowe 2004]. The algorithm proceeds by first building a difference-of-Gaussian pyramid with each pyramid level given by the difference of two Gaussian-filtered versions of the input image produced with filters of increasing standard deviation σ. Each level of the pyramid is assigned an image scale obtained from the two σ values of the corresponding filters (*eg.,* the smallest of the two). Then a corner detection is carried across the three-dimensional space of the pyramid defined by horizontal and vertical pixel positions h and v, and level scale σ. Each detected corner is thus specified by an (h, v, σ)-tuple. In turn each tuple is interpreted as defining a circle in the original image $\mathbf{J}$ (*i.e.,* with (h, v) defining the circle center and $\beta \cdot \sigma$ its radius for some fixed β) corresponding to the scale-covariant regions $\mathcal{B}$.

Rather than simple invariance to scale changes, image description systems are instead commonly concerned with invariance to affine transformations. The reason is that affine transformations provide a good model for rigid movements of planar surfaces in images and, in turn, 3-D objects can be modeled locally as a planar surface. Thus local description schemes based on affine-covariant region detectors can be used to represent complex scenes involving natural objects.

One common way [Mikolajczyk 2005a, Sivic 2003] of obtaining an affine-covariant region detection algorithm consists of taking a corner detector such as the Harris corner detector [Harris 1998] and then fitting an ellipse around each detected corner to thus produce the regions $\mathcal{B}$. The Harris detector in fact already includes all the elements necessary to define the required ellipse: Each Harris corner

is chosen to locally maximize a *corner saliency measure* assigned to each pixel position $\mathbf{p} = \begin{bmatrix} h & v \end{bmatrix}^{\mathsf{T}}$ using the pixel's second moment matrix $\mathbf{H}(\mathbf{p})$. Letting $G(\sigma; \mathbf{p})$ denote the Gaussian function of standard deviation σ; $I(\sigma_d; \mathbf{p})$ denote the smoothed version of the pixel intensity function obtained using $G(\sigma_d; \mathbf{p})$; and subscripts h and v denote differentiation along the related orientation, the second moment matrix is given by

$$\mathbf{H} = G(\sigma_I; \mathbf{p}) * \begin{bmatrix} I_x^2(\sigma_d; \mathbf{p}) & I_x(\sigma_d; \mathbf{p}) I_y(\sigma_d; \mathbf{p}) \\ I_y(\sigma_d; \mathbf{p}) I_x(\sigma_d; \mathbf{p}) & I_y^2(\sigma_d; \mathbf{p}) \end{bmatrix}, \tag{3.1}$$

where the convolution with the matrix is carried out entry-wise and over the spatial parameter $\mathbf{p}$. The Harris corner saliency measure is defined in terms of the singular values σ_i, $i = 1, 2$, of $\mathbf{H}$ as follows

$$|\sigma_1 \sigma_2 - \alpha (\sigma_1 + \sigma_2)^2 |$$

for some user-selected parameter α that enforces the degree of *cornerness* required. To obtain the ellipse defining an affine invariant representation $\mathcal{B}$ of this corner, one further uses the singular vectors of $\mathbf{H}$ to define the two principal axes of the desired ellipse, with the length of axes $i = 1, 2$ given by

$$\beta / \sigma_i$$

for some user-selected constant β.

A more recent affine-covariant region detector, the Maximally-Stable Extremal Region Detector (MSER) [Matas 2002], instead takes advantage of the property that any given region of an image will retain its intensity relative to its neighborhood when subject to geometrical transformations. Thus the MSER detector chooses those regions that display large contrast relative to their neighborhood. It does this by first increasing an intensity threshold τ starting from the minimum intensity value (*i.e.,* $\tau = 0$ to 255 for 8-bit images) and then selecting connected groups of pixels that remain relatively constant along the way, subsequently repeating the process with a decreasing τ (*eg.,* from 255 down to 0) to obtain still other pixel groups. The detected image regions $\mathcal{B}$ are then obtained from these pixel groups by fitting an ellipse to each pixel group.

Mikolajczyk *et al.* [Mikolajczyk 2005a] have carried out extensive experimental evaluations of the various region detectors including those discussed above and have found that the MSER and the Hessian-affine detectors tend to outperform others (depending on the particular image and distortion considered) in terms of their ability to detect the same affine-covariant regions from two versions of the same scene. Nonetheless, practical schemes use multiple region detectors [Sivic 2003, Jégou 2008] because some images will not result in a suitable

number of regions of a specific type. For example, region detectors based on corner detectors will tend to produce high-contrast regions, while others such as the MSER detector will produce smooth regions.

3.1.2 Region Normalization

The second step of the image description process illustrated in Fig. 3.1 consists of obtaining a geometrically normalized version (denoted in vectorized form $\mathbf{y} \in \mathbb{R}^d$) of each of the detected regions $\mathcal{B}$. A common implementation of this process takes advantage of the fact that the region detectors in the first block of Fig. 3.1 return regions $\mathcal{B}$ expressed in terms of simple geometrical shapes such as circles or ellipses. Thus the region normalization process will consists of first calculating the homography mapping the region $\mathcal{B}$ to a canonical instance (*eg.,* the unit circle) of the geometrical shape used by the detector. This homography is then used to extract the pixel intensities of the normalized version of $\mathcal{B}$ and a pre-established neighborhood, thus producing a normalized patch $\mathbf{y}$ that is usually square in shape.

This process can be expressed as follows for the case of affine covariant region detectors that produce elliptical regions: Each region is assumed represented by $\mathcal{B} = \{\mathbf{B}, \mathbf{p}_c\}$, where $\mathbf{p}_c$ is the ellipse center and $\mathbf{B}$ is a matrix defining its shape as follows:

$$\mathbf{p}^\mathsf{T} \mathbf{B} \mathbf{p} = 1. \tag{3.2}$$

The region normalization process requires a homography defined as mapping the detected ellipse to some canonical shape chosen by convention to be the unit circle. Letting $\mathbf{p}'$ denote the coordinates of the normalized region, this canonical shape satisfies

$$(\mathbf{p}')^\mathsf{T} \mathbf{p}' = 1 \tag{3.3}$$

and hence the relationship to the original coordinate system is expressed from (3.2) as

$$\mathbf{p} = \mathbf{B}^{-\frac{1}{2}} \mathbf{p}', \tag{3.4}$$

thus showing that the required homography is given by $\mathbf{B}^{-\frac{1}{2}}$.

3.1.3 Region Description

A wide variety of methods exist to implement the third block of the image description process illustrated in Fig. 3.1. The most successful of these methods are based on some variation of the description component of the SIFT algorithm [Lowe 2004], and we thus now provide a review of this method.

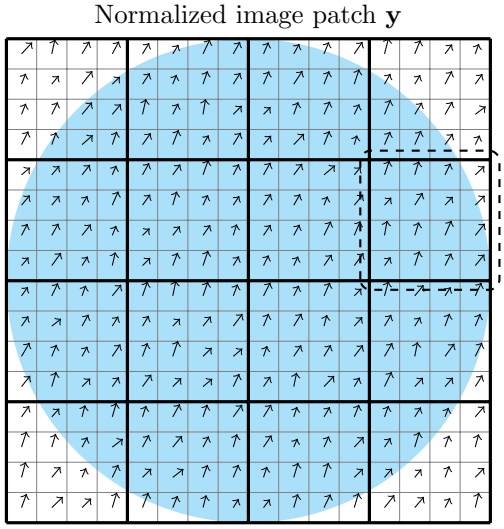

Figure 3.2: The SIFT descriptor: The arrows denote the differential gradients at each pixel position. A dominant orientation (*i.e.,* the histogram peak) is first obtained from a gradient-angle histogram built using the entire patch. Each of the 4×4 sub-regions is then used to build one 8-bin histogram using gradient-angles taken relative to the previously computed dominant orientation.

The SIFT description algorithm consists of first taking the differential gradients of the intensity level at each pixel as given by

$$\left[\frac{\Delta \mathbf{y}_{\langle h,v\rangle}}{\Delta h} \quad \frac{\Delta \mathbf{y}_{\langle h,v\rangle}}{\Delta v}\right]^{\mathsf{T}}. \tag{3.5}$$

where Δh and Δv denote, respectively, horizontal and vertical displacement; $\mathbf{y}_{\langle h,v\rangle}$ is the intensity level of the normalized patch at pixel position (h, v). Being a two dimensional vector, each gradient can be expressed in polar form with an angle given by

$$\arctan\left(\frac{\Delta \mathbf{y}_{h,v}}{\Delta v} \Big/ \frac{\Delta \mathbf{y}_{h,v}}{\Delta h}\right). \tag{3.6}$$

These angles are first used to build a a single, patch-wide histogram of angles; the position of the peak of this histogram will define the patch's dominant orientation. Subsequently, all gradient angles are expressed relative to this dominant angle (*i.e.,* the dominant angle is subtracted from all gradient angles). The normalized patch is then split into 4×4 regions, and each of the 16 regions is used to build an 8-bin histogram using the relative gradient angles previously computed. These 16 histograms are then concatenated to produce the SIFT descriptor $\mathbf{s} \in \mathbb{R}^{128}$.

Various authors have proposed using variations of the SIFT method to derive new descriptors resulting in a mix of improved (*i*) descriptor distinctiveness,

(*ii*) descriptor construction complexity, and (*iii*) descriptor search complexity. For example, the Gradient Locality and Orientation Histogram (GLOH) descriptor [Mikolajczyk 2005b] uses a log-polar division of the image patch instead of the 4×4 arrangement shown in Fig. 3.2. The experiments carried out by the authors of that work shows that GLOH can yield a distinctiveness advantage over SIFT, albeit at increased computational cost related to a required dimensionality-reducing PCA projection. Another approach, PCA-SIFT [Ke 2004], discards the division into sub-regions altogether and instead aggregates gradient-angles using a projection unto the first 36 principal component vectors.The reduced dimensionality of the descriptor results in reduced matching complexity, although this comes at the expense of distinctiveness. Yet a third approach, Speeded-Up Robust Features (SURF) [Bay 2008], was conceived with a keen eye for descriptor construction complexity: Using integral images (these allow computing the sum of intensities in a rectangular area of arbitrary size with only 3 additions) combined with a modification of the differential gradient computation using the Haar wavelet response (which is constant over two rectangular areas) results in a very low-complexity descriptor construction process. The integral image representation is further exploited to build a low complexity scale-invariant region detector.

3.2 Image searches using local descriptor

In order to carry out image searches using local descriptors, one needs to devise an aggregate similarity score applicable to descriptor sets (*i.e.,* the query set and the set of a potentially matching image). In this section we first present an example of a local descriptor voting mechanism commonly used as an aggregate similarity score. This scheme requires the computation of pairwise distances between each query descriptor and the large quantity of index descriptors. To reduce the complexity of the process, Approximate Nearest-Neighbor (ANN) search schemes are commonly used to carry out this pairwise descriptor matching. We thus present next the *sparse-matrix index* which is an ANN search scheme used by various local-descriptor-based image search algorithms.

3.2.1 Local descriptor voting mechanisms

Voting mechanisms offer one common possibility [Lowe 2004, Sivic 2003, Jégou 2008] of devising an aggregate similarity score applicable for locally-described image sets. A voting mechanism takes each of the N_q query descriptor and assigns a vote to the underlying images of each of the K database descriptors that are (approximately) nearest to each query descriptor. In the end, a total of $K \times N_q$ votes are

distributed amongst all database images, and the cumulative votes for each image yields its score. Various possible flavors of this algorithm include forcing the K nearest descriptors to be taken from different images, or counting at most one vote when multiple K-NN descriptors belong to the same database image (note in this case KN_q will be rather a closed upper bound on the number of distributed votes).

3.2.2 Approximate Nearest-Neighbor (ANN) searches using the sparse-matrix index

In order to obtain estimates of the nearest neighbors of the query vectors required by voting mechanisms, many Approximate Nearest-Neighbor (ANN) methods have been proposed very recently [Sivic 2003, Nister 2006, Philbin 2008, Zepeda 2009, Jégou 2008, Muja 2009] that rely on a sparse representation of the images' local descriptors. The sparse representation allows the search scheme to benefit from the low complexity of inner-product computations between sparse vectors. These methods consist of adaptations of the *inverted file index* from the text-search community [Zobel 2006]. We refer to the structure employed by these methods as a *sparse-matrix index*.

Let us assume that we are dealing with sparse representations $\mathbf{x} \in N$ of the local descriptors having only a few non-zero entries γ. The *sparse-matrix index* will be the matrix

$$\mathbf{X} = \text{cols}(\{\mathbf{x}_b\}) \tag{3.7}$$

built by placing the index vectors $\mathbf{x}_b$ side-by-side as columns. $\mathbf{X}$ is used in its row-major representation consisting of N bins $\mathcal{V}_r$ storing each the non-zero coefficients γ along the r-th row of $\mathbf{X}$ and their identifier b:

$$\mathcal{V}_r = \{(\gamma, b) | \mathbf{x}_{b\langle r\rangle} = \gamma \neq 0, \forall b\}. \tag{3.8}$$

Sparse-matrix indices are useful to carry out queries using the inner-product distance: Given a query vector $\mathbf{x}_q$, only the bins $\mathcal{V}_r$ with r specifying a non-zero position of $\mathbf{x}_q$ need to be processed. The complexity of the query operation $\mathbf{x}_q^\mathsf{T}\,\mathbf{X}$ can be measured by the total number of multiplications carried out in the process:

$$\sum_{\substack{r \text{ s.t.} \\ \mathbf{x}_{q\langle r\rangle} \neq 0}} |\mathcal{V}_r|. \tag{3.9}$$

Taking the expectation over $\mathbf{x}_q$ of this expression produces the following mean complexity:

$$\sum_r p(\mathbf{x}_{q\langle r\rangle} \neq 0) |\mathcal{V}_r|. \tag{3.10}$$

If we assume that the non-zero positions of $\mathbf{x}_q$ are equally distributed along all positions, then the above mean complexity will be minimized when all bins $\mathcal{V}_r$ have the same number of elements.[1] If this is the case, then we can drop the constant term $|\mathcal{V}_r| = \beta$ in (3.9) and instead express the complexity per query vector as the number of non-zero coefficients in $\mathbf{x}_q$ given by its l-0 norm $|\mathbf{x}_q|_0$:

$$\sum_{\substack{r \text{ s.t.} \\ \mathbf{x}_{q\langle r \rangle} \neq 0}} |\mathcal{V}_r| = \beta |\mathbf{x}_q|_0 \;\propto\; |\mathbf{x}_q|_0. \tag{3.11}$$

3.3 Applications of sparse representations in image description and search

In the current section we will present several recent efforts that use the sparse representation tools discussed in Chapter 2 in the context of image description or image indexing. The first work presented is an implementation of the sparse-matrix index above discussed. The second method presented likewise aims to reduce query complexity but using a different approach based on iteratively pruning the possible matches using a power-decay model of the sparse coefficients. These first two methods deal directly with exploiting the low complexity of computations between sparse vectors. The third method can instead be seen as an alternative way of describing and comparing image patches in a transformation-invariant manner.

3.3.1 Semi-local searches using Bag-of-Features (BOF)

The first approach [Sivic 2003] to demonstrate the feasibility of low-complexity large-scale image searches was that carried out by Sivic and Zisserman. The approach consisted of representing the local descriptors $\mathbf{s}$ of an image or a user-selected area of the image as a single sparse vector $\mathbf{w}$ called a *bag-of-features*. The *bag-of-features* vector is built by first vector quantizing each of the $\mathbf{s}$ using a set of code-vectors (called *visual words* in this context) learned using K-means (a variant presented in [Nister 2006] uses a tree-structured K-means approach). The resulting quantized version of $\mathbf{s}$ can be represented using the 1/0-valued sparse vectors $\mathbf{x}$ given by

$$\mathbf{x}_{\langle i \rangle} = \begin{cases} 1 & \text{if } i = \text{argmin}_k \, |\mathbf{d}_k - \mathbf{s}| \\ 0 & \text{otherwise} \end{cases}. \tag{3.12}$$

[1] We show this formally in Appendix 7.B.

In the second step, the resulting sparse vectors $\mathbf{x}$ are summed and then normalized by the number N_I of SIFT descriptors in the image to obtain

$$\mathbf{f} = \frac{1}{N_I} \sum_{l=1}^{N_I} \mathbf{x}_l. \tag{3.13}$$

Note that $\mathbf{f}_{\langle i \rangle}$ is just the frequency of occurrence of codeword $\mathbf{d}_i$ in the image. We can likewise define the vector $\mathbf{f}^{DB}$ with i-th entry giving the frequency of occurrence across all the database of images with descriptors $\mathbf{s}$ that produce code-vector $\mathbf{d}_i$. The *bag-of-features* $\mathbf{w}$ follows by applying the predetermined weights $\log \frac{1}{\mathbf{f}^{DB}_{\langle i \rangle}}$ to corresponding entries of $\mathbf{f}$ and then normalizing by the l-1 norm as follows:

$$\mathbf{w} = \frac{1}{|\mathbf{w}|_1} \ \mathrm{diag}(\log \frac{1}{\mathbf{f}^{DB}_{\langle 1 \rangle}}, \log \frac{1}{\mathbf{f}^{DB}_{\langle 2 \rangle}}, \ldots)\ \mathbf{f}. \tag{3.14}$$

These weighting scheme is taken directly from the text-search community and is known as Text Frequency-Inverse Document Frequency (TF-IDF) [Zobel 2006].

The *bag-of-features* approach has been extended in [Philbin 2008] by instead computing the vector $\mathbf{f}$ in (3.13) using sparse vectors $\mathbf{x}$ with $K \geq 1$ non-zero coefficients at positions i_k corresponding to the K code-vectors $\mathbf{d}_{i_k}, k = 1, \ldots, K$, nearest to $\mathbf{s}$ (as shown in (3.12) for the case $k = 1$). The value of the non-zero coefficients of $\mathbf{x}$ is given by

$$\mathbf{x}_{\langle i \rangle} = \beta \exp(-|\mathbf{s} - \mathbf{d}_i|^2/\sigma^2), \quad i = 1, \ldots, k, \tag{3.15}$$

where σ^2 is a learned, codebook-dependent parameter [Philbin 2008], and β is an l-1 normalization constant ensuring that $|\mathbf{x}|_1 = 1$. The resulting *bag-of-features* vector $\mathbf{w}$ is again built from (3.13) (using the new $\mathbf{x}$) and (3.14).

3.3.2 Exact and approximate searches

Jost and Vandergheynst [Jost 2008] instead proposed using the MP decomposition algorithm (*cf.* Section 2.2.1.1) to obtain a low-complexity search scheme for compressible signals. MP decompositions build the sparse representation $\mathbf{x}$ iteratively, selecting a single non-zero coefficient γ_i in each iteration i. The algorithm proposed in [Jost 2008] uses an assumed power decay law on the norm of the γ_i,

$$|\gamma_i| \leq \alpha \cdot i^{-\beta_i}, \tag{3.16}$$

to prune the set of possible database matches $\mathbf{y}_b$ of a given query vector $\mathbf{y}_q$ under the absolute inner product-similarity measure. The parameter α and β_i in (3.16) are learned from the database vectors.

Let us assume that the signal vectors $\mathbf{y}$ are compressible and can hence be represented using a selected-atoms matrix $\mathbf{S} = \begin{bmatrix} \mathbf{d}^1 & \dots & \mathbf{d}^L \end{bmatrix}$ and a corresponding vector of MP coefficients $\mathbf{\Gamma} = \begin{bmatrix} \gamma_1 & \dots & \gamma_L \end{bmatrix}^\mathsf{T}$: $\mathbf{y} = \mathbf{S}\mathbf{\Gamma}$. Using this, we express the similarity measure between a query vector $\mathbf{y}_q$ and any given database vector $\mathbf{y}_b$ as

$$|\langle \mathbf{y}_q, \mathbf{y}_b \rangle| = |\mathbf{\Gamma}_q^\mathsf{T} \mathbf{S}_q^\mathsf{T} \mathbf{S}_b \mathbf{\Gamma}_b|. \tag{3.17}$$

The right-hand term of the above expression can be expressed as the sum of all entries c_{kl} of the matrix

$$\mathbf{C} = \mathrm{diag}(\mathbf{\Gamma}_q) \mathbf{S}_q^\mathsf{T} \mathbf{S}_b \, \mathrm{diag}(\mathbf{\Gamma}_b). \tag{3.18}$$

as follows

$$\big|\langle \mathbf{y}_q, \mathbf{y}_b \rangle\big| = \big| \sum_{k,l} c_{kl} \big| \tag{3.19}$$

$$= \big| \overbrace{\sum_{\substack{k,l \\ k+l \leq i}} c_{kl}}^{a_i} + \sum_{\substack{k,l \\ k+l > i}} c_{kl} \big| \tag{3.20}$$

The pruning scheme proceeds iteratively by considering up to the i-th representation coefficient γ_i at each iteration i. As illustrated in (3.20), the coefficients c_{kl} along the first i anti-diagonals define the current similarity estimate a_i. The estimation error (corresponding to the summation over the remaining anti-diagonals $i+1, i+2, \dots$) is upper-bounded using the power decay law (3.16):

$$\big| \sum_{\substack{k,l \\ k+l > i}} c_{kl} \big| \leq \big| \overbrace{\sum_{\substack{k,l \\ k+l > i}} \alpha^2 \cdot k^{-\beta_k} l^{-\beta_l}}^{e_i} \big|. \tag{3.21}$$

Using this result and the triangle inequality produces the following lower and upper bounds on the similarity (3.20):

$$|a_i| - |e_i| \leq \langle \mathbf{y}_q, \mathbf{y}_b \rangle \leq |a_i| + |e_i|. \tag{3.22}$$

The approach proceeds iteratively where, in each iteration i, the contribution of the i-th anti-diagonal of $\mathbf{C}$ is removed from e_i (where it appeared in upper-bound form) and added (in exact form) to the term a_i to thus improve the bounds in (3.22). The new bounds can be used to prune the possible query responses $\mathbf{y}_b$ at each iteration i. For example, if the N nearest-neighbors are sought, the minimum of the highest N lower bounds $|a_i| - |e_i|$ will define a threshold. All $\mathbf{y}_b$ having an

upper bound $|a_i|+|e_i|$ below this threshold can be discarded from the computations in subsequent iterations $i+1, i+2, \ldots$ as we are certain that they are not one of the N nearest neighbors.

For completeness we note that a reduced-complexity version of the above scheme is also presented in [Jost 2008] that uses a probabilistic model to make the upper-bound in (3.21) tighter as a function of a user-selected probability that the bound is satisfied. Thus this latter probabilistic approach is an approximate search scheme, unlike the former deterministic approach which is exact.

3.3.3 A manifold descriptor and similarity measure

Kokiopoulou and Frossard [Kokiopoulou 2008] proposed using sparse representations to compute a transformation invariant similarity measure between two patches, *eg.,* a query patch and a reference patch. Since all possible transformations of the query patch define a non-linear manifold, they proposed using the *manifold distance* as their transformation invariant measure. The manifold distance is given by the minimum distance between the reference patch and any point along the transformation manifold of the query patch. Their system can be seen as a joint implementation of the region normalization and description blocks in Fig. 3.1 that takes each selected region $\mathcal{B}$ and produces a manifold that serves as the corresponding descriptor.

To produce the manifold model of a selected region, the method relies on a parametric dictionary consisting of atoms that are seen as continuous 2-D surfaces $\mathbf{d}_{\mathcal{T}}$ obtained by applying transformations $\mathcal{T}$ to a template function $\boldsymbol{\alpha}$ such as the 2-D Gaussian or one of its derivatives,

$$\mathbf{d}_{\mathcal{T}} = f(\mathcal{T}, \boldsymbol{\alpha}). \tag{3.23}$$

A dictionary comprised of sampled versions of these atoms is first used to obtain a sparse representation of a given region by means of OMP,

$$\sum_i \gamma_i \cdot \mathbf{d}_{\mathcal{T}_i}. \tag{3.24}$$

The manifold of all possible deformations $\mathcal{N}$ of $\mathcal{B}$ is then represented analytically as

$$f\left(\mathcal{N}, \sum_i \gamma_i \cdot \mathbf{d}_{\mathcal{T}_i}\right) = \sum \gamma_i \cdot \mathbf{d}_{\mathcal{N}\circ\mathcal{T}_i}, \tag{3.25}$$

for an adequate transformation combination law $\circ$. For example, if we only consider isotropic scale changes by a factor σ, then $\mathcal{T} = \{\sigma_T\}$, $\mathcal{N} = \{\sigma_N\}$ and $\mathcal{T} \circ \mathcal{N} = \{\sigma_T \sigma_N\}$.

The authors show that the inner product similarity measure between an image patch and the manifold representation (3.25) of another image patch can be expressed as the difference of functions that are convex (*i.e.,* a DC function) in $\mathcal{N}$. Algorithms exist that produce the global minimizer of DC functions, and thus the authors exploited this fact to produce the global manifold distance.

3.4 Contributions (2 of 2): ANN searches using sparse representations

In this section we summarize the contributions of this thesis in the field of image description and indexing, which we present in Chapter 7. See Section 2.5 for a summary of contributions in the field of sparse representations.

3.4.1 New formulation for sparse representations

We propose a novel sparse representation scheme for image description and indexing. Referring to the block diagram in Fig. 3.1, several image search methods [Sivic 2003, Nister 2006, Philbin 2008, Zepeda 2009, Jégou 2008] rely on a sparse representation $\mathbf{x}$ of the local descriptors $\mathbf{s}$. One problem with this approach pertains to the stability of the support of $\mathbf{x}$ as a function of normalization errors at the output of the geometrical normalization block. Sparse representation schemes such as vector quantization or the MPF algorithms in Section 2.2.1 choose the support of $\mathbf{x}$ in order to maximize the fidelity of the resulting representation (*i.e.,* to minimize the distortion $|\mathbf{y} - \mathbf{D}\mathbf{x}|$). This is the correct approach to follow in the context of image compression, but not in the context of approximate nearest-neighbor searches. Thus the first contribution is to propose a new formulation for the selection of the sparse support of $\mathbf{x}$. The formulation aims to reduce the distance approximation error under a constraint on the l-0 norm of $\mathbf{x}$ which, as we saw in Section 3.2.2, is representative of search complexity.

3.4.2 Data conditioning for sparse-matrix indices

Since search complexity is minimized for a uniform distribution of the sparse support of $\mathbf{x}$ (*cf.* Section 3.2.2), a second contribution is a new conditioning transform that enforces this property for our new sparse representation scheme. The new transform is used in a pre-processing step applied to the data vectors to be conditioned and is aimed at (*i*) more uniformly distributing them on the unit sphere while (*ii*) retaining their relative angular positions . We show experimentally that our new conditioning transform indeed succeeds in carrying out this

task. The evaluation of the uniformity of the conditioned data is complicated in high-dimensional spaces, and thus we proceed by comparing rather the distribution of the projections of the data unto various lines. To do so, we also derive an exact analytical expression for the distribution of projections from data uniformly-distributed on the unit hyper-sphere.

As we show in our evaluations, the combination of our new sparse representation scheme along with the data conditioning transform results in a significant improvement in distance approximation for a fixed complexity when compared to searches based on traditional sparse representation schemes.

Part II

Contributions

Chapter 4

The Iteration-Tuned Dictionary

4.1 Introduction

In this chapter we introduce a novel sparse representation approach that incorporates the iterative nature of greedy pursuit algorithms in a new overcomplete dictionary framework called an *Iteration Tuned Dictionary* (ITD). ITDs consist of a layered structure with each layer composed of a different dictionary matrix. ITD decompositions proceed by using the dictionary from the i-th layer in the i-th pursuit iteration. We propose an ITD training scheme that relies on single-atom sparse representations which can be solved exactly with any approximate algorithm such as Matching Pursuit (unlike higher sparsity representations used by training schemes found in the literature).

We compare the proposed ITD to the K-SVD dictionary [Aharon 2006b] (*cf.* Section 2.3.3) and to the Sparse Dictionary (SD) [Rubinstein 2010a] (*cf.* Section 2.3.4.1) which is designed to reduce the decomposition complexity. We first consider a concatenated ITD (cITD) setup where the trained layers are concatenated to form a single dictionary. The cITD scheme is shown experimentally to provide a better sparsity / approximation error tradeoff than the reference dictionaries. In a second experiment, we compute the experimental rate / distortion bound and show that both cITD and ITD can likewise offer improved rate / distortion tradeoffs. ITD further achieves this with a complexity that is much lower than that of the reference dictionaries, and this reduced complexity makes it possible to tap into the large rate / distortion gains achievable with high overcompleteness factors.

For convenience, we have define some constants used throughout this chapter in Table 4.1.

4.2 Background

We will begin the present chapter by providing a brief review of sparse representations, matching pursuit, and matching pursuit complexity. The intent is to make the chapter self-contained and keep references to the more relevant equations at hand.

Symbol	Definition
C	Number of ITD layers.
N	Total number of atoms across all ITD layers (or in a reference, fixed dictionary such as K-SVD).
n	Number of atoms in an ITD layer (with $N = C \cdot n$) .
$\mathbf{d}^l$	An atom from the l-th layer that is the l-th chosen atom from a sequence $\mathbf{d}^1, \mathbf{d}^2, \mathbf{d}^3, \ldots$.

Table 4.1: List of notational conventions.

4.2.1 Sparse representations using overcomplete dictionaries

Let $\mathbf{D} \in \mathbb{R}^{d \times N}$ (with $N > d$) be the full-rank dictionary matrix formed by N columns $\mathbf{d}_a$, $a = 1, \ldots, N$, called the atoms (we assume all atoms are unit norm). A sparse representation $\mathbf{x}$ of a signal vector $\mathbf{y} \in \mathbb{R}^d$ is obtained under joint fidelity and sparsity criteria:

$$\underset{\mathbf{x} \in \mathbb{R}^N}{\operatorname{argmin}} |\mathbf{y} - \mathbf{D}\mathbf{x}| \text{ s.t. } |\mathbf{x}|_0 \leq L, \tag{4.1}$$

where the l-0 norm $|\cdot|_0$ counts the number of non-zero coefficients in a vector.

Without loss of generality, we assume that $\mathbf{D}$ is full-rank and hence the minimum of (4.1) will occur at the boundary $|\mathbf{x}|_0 = L$ (assuming $L \leq d$) with probability one. Hence (4.1) consists of selecting the L atoms $\mathbf{d}^1, \ldots, \mathbf{d}^L \in \mathbf{D}$ that produce the best approximation of $\mathbf{y}$. We can thus re-write (4.1) in the following equivalent form that is more similar to expressions we cover in subsequent sections:

$$\underset{\substack{\mathbf{d}^1, \ldots, \mathbf{d}^L \in \mathbf{D}; \\ \mathbf{\Gamma} \in \mathbb{R}^L}}{\operatorname{argmin}} \left| \mathbf{y} - \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^L \end{bmatrix} \mathbf{\Gamma} \right|. \tag{4.2}$$

To make explicit the relationship between (4.2) and (4.1), we let $a_l \in \{1, \ldots, N\}$ denote the index of the l-th atom chosen in (4.2), *i.e.,* $\mathbf{d}^l = \mathbf{d}_{a_l}$. The solution $\mathbf{x}$ of (4.1) can hence be obtained from the solution of the above problem as follows (we use $\mathbf{v}[l]$ to denote the l-th coefficient of a vector $\mathbf{v}$):

$$\mathbf{x}[k] = \begin{cases} \mathbf{\Gamma}[l] & \text{if } k = a_l, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

4.2.2 Matching pursuit

Many algorithms exist in the literature that produce approximations of the NP-hard problem in (4.1) (see Section 2.2). In our work we will use the Matching Pursuit (MP) algorithm [Mallat 1993]. MP proceeds by selecting one atom / coefficient pair at each iteration i. Let $\mathbf{r}^{i-1}$ denote the residue at the output of the previous iteration (with $\mathbf{r}^0 = \mathbf{y}$) and $(\cdot)^\mathsf{T}$ denote the matrix or vector transpose. The MP atom and coefficient selection rules are given by

$$\mathbf{d}^i = \underset{\mathbf{d}\in\mathbf{D}}{\operatorname{argmax}} |\mathbf{d}^\mathsf{T} \cdot \mathbf{r}^{i-1}|, \tag{4.4a}$$

$$\gamma_i = (\mathbf{d}^i)^\mathsf{T} \cdot \mathbf{r}^{i-1}, \tag{4.4b}$$

with $\mathbf{r}^i = \mathbf{r}^{i-1} - \gamma_i \cdot \mathbf{d}^i$. The decomposition is carried out until L atom / coefficient pairs are selected, where the value of L is either defined beforehand or selected along the iterations to satisfy an RMSE threshold ε as follows:

$$L = \min_{i\in\{1,2,\dots\}} i \text{ s.t. } |\mathbf{r}^i|^2 \leq d \cdot \varepsilon^2. \tag{4.5}$$

We can group the coefficients produced up to the i-th iteration to form the i-th coefficients vector

$$\mathbf{\Gamma}^i = \begin{bmatrix}\gamma_1 & \dots & \gamma_i\end{bmatrix}^\mathsf{T} \tag{4.6}$$

and likewise group the selected atoms to form the *selected-atoms matrix*

$$\mathbf{S}^i = \begin{bmatrix}\mathbf{d}^1 & \dots & \mathbf{d}^i\end{bmatrix}. \tag{4.7}$$

With this, the i-th approximation can be written as $\mathbf{S}^i\mathbf{\Gamma}^i$ and the i-th residue vector can be written as

$$\mathbf{r}^i = \mathbf{y} - \mathbf{S}^i\mathbf{\Gamma}^i. \tag{4.8}$$

4.2.3 Complexity

In this paper we will develop structured dictionaries that result in low complexity decompositions, and thus we will briefly review MP complexity. Note that each MP iteration in (4.4) requires the computation of the *dictionary-vector product* $\mathbf{D}^\mathsf{T}\mathbf{r}^{i-1}$, a total of N inner-products. Hence $N \cdot L$ gives an exact complexity measure (expressed in terms of inner-products) of the MP decomposition process. However, one can trade storage space for reduced decomposition complexity by pre-computing and storing the dictionary's Gram matrix $\mathbf{G} = \mathbf{D}^\mathsf{T}\mathbf{D}$. The result is that only the first iteration will require the computation of a dictionary-vector product [Cotter 1999]:

$$\mathbf{D}^\mathsf{T}\mathbf{r}^{i-1} = \mathbf{D}^\mathsf{T}\left(\mathbf{y} - \mathbf{S}^{i-1}\mathbf{\Gamma}^{i-1}\right), \tag{4.9}$$

$$= \mathbf{D}^\mathsf{T}\mathbf{y} - \left(\mathbf{D}^\mathsf{T}\mathbf{S}^{i-1}\right)\mathbf{\Gamma}^{i-1} \tag{4.10}$$

where we note that $\mathbf{D}^\mathsf{T}\mathbf{S}^{i-1}$ is a sub-matrix of $\mathbf{G}$ and does not need to be computed. In this scenario, N inner-products is a lower bound on the complexity of all L iterations.

Rubinstein *et al.* [Rubinstein 2010a] proposed a Sparse Dictionary (SD) that reduces the cost of the dictionary-vector product operation. Their dictionary is constrained to have the form $\mathbf{D} = \mathbf{\Phi}\mathbf{A}$, where $\mathbf{\Phi} \in \mathbb{R}^{d\times N}$ is the overcomplete DCT dictionary and $\mathbf{A} \in \mathbb{R}^{N\times N}$ is a learned sparse matrix having g non-zero coefficients per column. The dictionary-vector product then has the form $\mathbf{A}^\mathsf{T}(\mathbf{\Phi}^\mathsf{T}\mathbf{y})$. Since the DCT dictionary is separable, the term in parenthesis can be computed using exactly $M(2b-1)b + M(2b-1)M$ operations, where $M = \sqrt{N}$ and $b = \sqrt{d}$; left multiplication by $\mathbf{A}^\mathsf{T}$ will require at least $(2g-1)N$. Since a single inner-product requires $2d-1$ operations, we can express the SD dictionary-vector product complexity in terms of inner-products as follows:

$$\left(M(2b-1)b + M(2b-1)M + (2g-1)N\right)/(2d-1). \tag{4.11}$$

4.3 The Iteration-Tuned Dictionary

In the current section we will begin by introducing the proposed Iteration-Tuned Dictionary (ITD) structure establishing, in particular, the related notation. We will then discuss the ITD signal decomposition strategy and show that it corresponds to a constrained case of traditional, fixed-dictionary decomposition schemes. As is the case with the fixed-dictionary sparse representation problem in (4.2) , the ITD decomposition problem is NP-hard, and hence we will rely on matching pursuit to decompose signals using ITD. The ITD constraint that each atom comes from a different component dictionary implies sub-optimality relative to the unconstrained, fixed-dictionary approach. Yet the ITD approach offers several interesting advantages that we discuss at the end of the section.

4.3.1 ITD structure

The main idea of our proposed algorithm is that, when carrying out an iterative sparse decomposition of a signal $\mathbf{y}$, the dictionary matrix used is replaced by a different dictionary matrix at each iteration of a matching pursuit decomposition. We refer to the set of all dictionary matrices available to the decomposition algorithm as an Iteration-Tuned Dictionary (ITD). Formally, an ITD consists of set of C dictionary matrices $\{\mathbf{D}_k\}_{k=1}^{K}$ with each $\mathbf{D}_k \in \mathbb{R}^{d\times n}$. ITDs can be seen as layered structures, with the k-th layer containing the k-th component dictionary $\mathbf{D}_k$. Each ITD component dictionary need not be overcomplete (*i.e.,* n can be less than d). Yet the total number of atoms $N = C \cdot n$ in the ITD structure is

generally greater than d. The parameters C and n are design variables that need to be chosen experimentally.

4.3.2 Signal decomposition using ITDs

The problem of obtaining a sparse decomposition of a given signal using an ITD setup can be formulated optimally as follows:

$$\underset{\substack{\mathbf{d}^k \in \mathbf{D}_k, k=1,\ldots,L; \\ \boldsymbol{\Gamma} \in \mathbb{R}^L}}{\operatorname{argmin}} \left| \mathbf{y} - \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^L \end{bmatrix} \boldsymbol{\Gamma} \right|. \tag{4.12}$$

This expression can be seen to correspond to a constrained version of the standard fixed-dictionary formulation in (4.2), the constraint being that at most one atom be chosen from each layer and that the layers be used successively from the top.

As is the case for the fixed-dictionary formulation in (4.2), the ITD sparse decomposition problem in (4.12) is difficult to solve and generally requires a combinatorial approach. We thus employ a straight forward adaptation of the MP algorithm wherein the i-th chosen atom is constrained to come from the i-th ITD layer, as written below:

$$\mathbf{d}^i = \underset{\mathbf{d} \in \mathbf{D}_i}{\operatorname{argmax}} \left| \mathbf{d}^\mathsf{T} \cdot \mathbf{r}^{i-1} \right|, \tag{4.13a}$$

$$\gamma_i = (\mathbf{d}^i)^\mathsf{T} \cdot \mathbf{r}^{i-1}. \tag{4.13b}$$

Two strategies can be used to deal with iterations after the last layer C: The first is to avoid them altogether by forcing the maximum sparsity of representations to be C. The second is to propagate the last layer in subsequent iterations, *i.e.,* $\forall i > C, \mathbf{D}_i = \mathbf{D}_C$.

4.3.3 Advantages of the ITD approach

The ITD atom selection constraint that results in sub-optimal sparse decomposition is nonetheless also responsible for three important advantages that we now discuss.

The first advantage is that the ITD approach using MP decompositions can handle very large dictionary over-completeness factors (*i.e.,* $N \gg d$) with low decomposition complexity. As discussed in Section 4.2.3, for fixed-dictionaries, a large over-completeness implies either a large complexity penalty when computing the dictionary-matrix product $\mathbf{D}^\mathsf{T} \cdot \mathbf{r}^{i-1}$ in each iteration i, or a large storage penalty if the Gram matrix $\mathbf{G} = \mathbf{D}^\mathsf{T}\mathbf{D} \in \mathbb{R}^{N \times N}$ is pre-computed. On the other hand, an ITD with $N = C \cdot n$ atoms (n per layer), will require, for all sparsities

L, only $L \cdot n$ inner-products. This is just a fraction of a full dictionary-vector product, which is the least one can hope for in the fixed-dictionary case. This reduced complexity further does not rely on the Gram matrix and is hence attained without the extra storage penalty.

A second advantage of ITDs concerns the sparse decomposition algorithm used in the dictionary training process. We discuss ITD training in the next section and for now only point out that the training procedure relies on single-atom sparse representations. Note that all training methods found in the literature rely on representations using higher sparsity values, and in these situations, and particularly for large (high-coherence) dictionaries, the approximate schemes used (*eg.,* OMP or basis pursuit) are not guaranteed to recover the optimal sparse representation [Tropp 2004]. A single-atom sparse representation, on the other hand, will be recovered exactly with MP regardless of the dictionary size. Hence one cannot conclude that using a practical, approximate sparse decomposition method to train a fixed-dictionary globally should perform better than a fixed dictionary built by concatenating ITD layers trained each using the optimal (single-atom) decomposition. We explore this concatenated ITD (cITD) approach in the results section and show that indeed it can outperform fixed-dictionary training methods.

Yet a third important advantage of the ITD scheme concerns the image compression application. In this scenario, it is not the sparsity of the representation that is important, but rather the number of bits required to represent, amongst other things, the indices of the selected atoms. We can expect ITD to offer an advantage in this situation since each ITD atom is selected from only n atoms, whereas each fixed-dictionary atom is chosen from $N = C \cdot n$ possibilities. We will see in the results section that this advantage can be sufficient to overcome the distortion penalty resulting from the sub-optimality of the constrained atom selection of ITD over a wide range of coding bit-rates used in general purpose applications.

4.4 Construction of Iteration Tuned Dictionaries

In this section we develop an ITD training algorithm. We begin by formulating the ITD training problem optimally. To simplify the problem, we then propose a top-down training scheme in which the ITD layers are trained on the residues of the training vectors obtained at the output of the previous layer. We also propose an iterative variant of the top-down approach. The algorithm used to update each layer is presented next. Both the iterative top-down algorithm and the layer update algorithm are shown to converge. We present an illustration of the trained

ITD layers at the end of the section.

Throughout our discussion we will assume that some representative training set $\{\mathbf{y}_t\}_{t=1}^{T}$ is available. When necessary, we will thus use subscript t to indicate quantities related to the t-th training vector. For example, training vector $\mathbf{y}_t$ will have a selected atoms matrix, i-th selected atom, i-th coefficients vector, and i-th coefficient denoted by $\mathbf{S}_t^i, \mathbf{d}_t^i, \mathbf{\Gamma}_t^i$, and $\gamma_{i,t}$, respectively.

4.4.1 Problem formulation

The construction of the set of ITD candidate dictionaries can be expressed as the minimization of the cumulative representation error of all training vectors. Using the optimal ITD sparse decomposition formulation in (4.12), we can express the optimal ITD $\{\mathbf{D}_k\}_{k=1}^{C}$ as follows:

$$\underset{\substack{\mathbf{D}_k\in\mathcal{N},\\ k=1,\ldots,C}}{\operatorname{argmin}} \left(\sum_{t=1}^{T} \min_{\substack{\mathbf{d}^k\in\mathbf{D}_k,\\ k=1,\ldots,L_t;\\ \mathbf{\Gamma}\in\mathbb{R}^{L_t}}} \left| \mathbf{y}_t - \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^{L_t} \end{bmatrix} \mathbf{\Gamma} \right|^2 \right), \tag{4.14}$$

where the optimization domain of each layer dictionary $\mathbf{D}_1, \ldots, \mathbf{D}_C$ is the subset

$$\mathcal{N} = \{\mathbf{D} \in \mathbb{R}^{d\times N} : \forall \mathbf{d} \in \mathbf{D}, \mathbf{d}^{\mathsf{T}}\,\mathbf{d} = 1\} \tag{4.15}$$

consisting of matrices in $\mathbb{R}^{d\times N}$ with unit-norm columns. This makes the solution unambiguous as otherwise, scaling the a-th atom by β and $\mathbf{\Gamma}[a]$ by $1/\beta$ produces the same cost. In the above problem, the sparsity L_t of each training vector $\mathbf{y}_t$ is assumed fixed beforehand. A different approach consists of modifying the inner minimization to instead optimize simultaneously over L_t with an added sparsity penalty term of the form $\lambda \cdot L_t$. Regardless of the method used to select L_t, the exact ITD sparse representation problem can not be solved in general, and thus we need to consider alternative approaches.

4.4.1.1 Top-down training

To simplify the ITD construction problem, we instead consider building the component dictionaries one layer at a time using a top-down approach. When building the k-th dictionary, we thus assume that all previous dictionaries $\mathbf{D}_1, \ldots, \mathbf{D}_{k-1}$ are readily available. Letting

$$\mathcal{T}_k = \{t = 1, \ldots, T : L_t \geq k\} \tag{4.16}$$

denote the set of indices of those training vectors $\mathbf{y}_t$ that use atoms from the k-th layer, the resulting formulation for the construction of the k-th dictionary can be expressed as

$$\mathbf{D}_k = \underset{\mathbf{D}\in\mathcal{N}}{\operatorname{argmin}} \sum_{t\in\mathcal{T}_k} \min_{\mathbf{d}\in\mathbf{D};\boldsymbol{\Gamma}\in\mathbb{R}^k} \left\| \mathbf{y}_t - \left[\mathbf{S}_t^{k-1} \;\middle|\; \mathbf{d}\right]\boldsymbol{\Gamma} \right\|^2. \tag{4.17}$$

When using the above top-down training approach, the sparsities L_t of each training vector $\mathbf{y}_t$ can be set during the training process using the approach in (4.5). Alternatively, L_t can be set to the number of layers C for all training vectors, which is equivalent to setting $\varepsilon = 0$ in (4.5). Unlike fixed dictionary training schemes, the selection of the sparsities L_t is not too critical when training ITDs, as ITD decompositions will use a single atom from each layer. Hence our approach is to set $\forall t, L_t = C$.

4.4.1.2 Iterative top-down training

The top-down training procedure described above neglects the joint nature of the dictionary construction specified by the optimal formulation in (4.14). One way to address this issue is to update previously constructed dictionaries using an iterative top-down approach. Thus the k-th layer can be updated after construction of layers $k+1, k+2, \ldots, C$. Letting $\bar{\mathbf{S}}_t^{k+1}$ contain the atoms selected for $\mathbf{y}_t$ from layers $k+1, \ldots, L_t$ such that $\mathbf{S}_t^{L_t} = \left[\mathbf{S}_t^k \;\middle|\; \bar{\mathbf{S}}_t^{k+1}\right]$, this can be expressed as follows:

$$\underset{\mathbf{D}_k\in\mathcal{N}}{\operatorname{argmin}} \sum_{t\in\mathcal{T}_k} \min_{\mathbf{d}\in\mathbf{D}_k;\boldsymbol{\Gamma}\in\mathbb{R}^{L_t}} \left\| \mathbf{y}_t - \left[\mathbf{S}_t^{k-1} \;\middle|\; \mathbf{d} \;\middle|\; \bar{\mathbf{S}}_t^{k+1}\right]\boldsymbol{\Gamma} \right\|^2. \tag{4.18}$$

When traversing the first iteration of the iterative top-down construction, one can assume that all matrices $\bar{\mathbf{S}}_t^{k+1}$ in (4.18) are initialized to the zero matrix, and likewise assume that $\forall l > k, \gamma_{l,t} = 0$. This is equivalent to running the non-iterative top-down construction scheme in (4.17). One can also select the sparsities L_t of the training vectors during this first top-down iteration using (4.5), keeping all L_t fixed in subsequent top-down iterations.

We summarize the resulting ITD training algorithm consisting of J top-down iterations in Fig. 4.1.

4.4.2 Layer update process

The solution of the layer update expression in (4.18) consists of two minimizations and hence each can be solved alternately to define an iterative training procedure. When solving one minimization, the optimization variables of the other are fixed

1: **Input**: Training vectors $\{\mathbf{y}_t\}_{t=1}^{T}$
2: **Output**: ITD $\{\mathbf{D}_k\}_{k=1}^{C}$.

3: **Initialization**:
4: **for** $t = 1, \ldots, T$ **do**
5: $\quad L_t \leftarrow C$
6: $\quad \mathbf{S}_t^{L_t} \leftarrow \mathbf{0} \in \mathbb{R}^{d \times L_t}$
7: $\quad \mathbf{\Gamma}_t^{L_t} \leftarrow \mathbf{0} \in \mathbb{R}^{L_t}$
8: **end for**

9: **Algorithm**:
10: **for** $j = 1, \ldots, J$ **do**
11: $\quad$ **for** $k = 1, \ldots, C$ **do**
12: $\quad\quad$ **if** $j == 1$ **then**
13: $\quad\quad\quad \left(\mathbf{D}_k, \{\mathbf{d}_t^k, \mathbf{\Gamma}_t^{L_t}\}_{t \in \mathcal{T}_k}\right) \leftarrow$ LayerUpdate $\left(\{\mathbf{y}_t, \mathbf{S}_t^{L_t}, \mathbf{\Gamma}_t^{L_t}\}_{t \in \mathcal{T}_k}\right)$
14: $\quad\quad$ **else**
15: $\quad\quad\quad \left(\mathbf{D}_k, \{\mathbf{d}_t^k, \mathbf{\Gamma}_t^{L_t}\}_{t \in \mathcal{T}_k}\right) \leftarrow$ LayerUpdate $\left(\{\mathbf{y}_t, \mathbf{S}_t^{L_t}, \mathbf{\Gamma}_t^{L_t}\}_{t \in \mathcal{T}_k}, \mathbf{D}_k\right)$
16: $\quad\quad$ **end if**
17: $\quad\quad$ **for** $t \in \mathcal{T}_k$ **do**
18: $\quad\quad\quad \mathbf{S}_t^{L_t}[*, k] \leftarrow \mathbf{d}_t^k$
19: $\quad\quad\quad$ **if** $(\mathrm{RMSE}(\mathbf{y}_t, \mathbf{S}_t^{L_t}\mathbf{\Gamma}_t^{L_t}) \leq \varepsilon) \& (j == 1) \& (k == 1)$ **then**
20: $\quad\quad\quad\quad L_t \leftarrow k$
21: $\quad\quad\quad\quad \mathbf{S}_t^{L_t} \leftarrow \mathbf{S}_t^{L_t}[*, 1 : L_t]$
22: $\quad\quad\quad$ **end if**
23: $\quad\quad$ **end for**
24: $\quad$ **end for**
25: **end for**

Figure 4.1: The ITD training algorithm using J top-down iterations. The LayerUpdate function in line 13 and line 15 solves (4.18) and returns the new dictionary $\mathbf{D}_k$ as well as the atom $\mathbf{d}_t^k \in \mathbf{D}_k$ selected for each $\mathbf{y}_t$ and the corresponding coefficients vector $\mathbf{\Gamma}_t^{L_t}$. The operation in line 21 truncates all columns of $\mathbf{S}_t^{L_t}$ after column L_t. The operation in line 19 computes the root mean squared error between the two input vectors.

to the most recent values. Hence we now address first the inner minimization and next the outer minimization of (4.18).

The solution of the inner minimization of (4.18) is straightforward if one notes

that the optimal $\mathbf{\Gamma}$ can be expressed in terms of the optimal $\mathbf{d}$ as follows:

$$\mathbf{\Gamma} = \left[\mathbf{S}_t^{k-1} \;\mid\; \mathbf{d} \;\mid\; \bar{\mathbf{S}}_t^{k+1}\right]^+ \mathbf{y}_t, \tag{4.19}$$

where $(\cdot)^+$ denotes the pseudo-inverse. Hence the inner minimization of (4.18) in fact depends on the single variable $\mathbf{d}$ and thus it can be solved by trying all possible $\mathbf{d} \in \mathbf{D}_k$, one at a time:

$$\min_{\mathbf{d}\in\mathbf{D}_k} \left|\mathbf{y}_t - \left[\mathbf{S}_t^{k-1} \;\mid\; \mathbf{d} \;\mid\; \bar{\mathbf{S}}_t^{k+1}\right] \cdot \left[\mathbf{S}_t^{k-1} \;\mid\; \mathbf{d} \;\mid\; \bar{\mathbf{S}}_t^{k+1}\right]^+ \mathbf{y}_t\right|. \tag{4.20}$$

This problem has the same form as the atom selection method of the greedy sparse decomposition algorithm called Optimized OMP (OOMP). The inconvenience of that approach pertains to the complexity of the pseudo-inverse computation, and indeed one can consider instead a simpler, MP-based solution which consists of carrying out the inner minimization of (4.18) only over the k-th coefficient γ_k (*i.e.,* the k-th element of $\mathbf{\Gamma}$) while using, for other $\gamma_l, l \neq k$, the values previously computed when building other layers $\mathbf{D}_l$ (recall that, during the first top-down iteration, we assume that $\forall l > k, \gamma_l = 0$). The optimal γ_k can again be expressed in terms of the optimal $\mathbf{d}$ as follows: $\gamma_k = \mathbf{y}_t^{\mathsf{T}} \cdot \mathbf{d}$, and hence the inner minimization can again be solved by trying all $\mathbf{d} \in \mathbf{D}_k$ one at a time:

$$\operatorname*{argmin}_{\mathbf{d}\in\mathbf{D}_k} \left|\mathbf{y}_t - \left[\mathbf{S}_t^{k-1} \;\mid\; \bar{\mathbf{S}}_t^{k+1}\right] \cdot \left[\, (\mathbf{\Gamma}_t^{k-1})^{\mathsf{T}} \;\mid\; (\bar{\mathbf{\Gamma}}_t^{k+1})^{\mathsf{T}} \,\right]^{\mathsf{T}} - (\mathbf{y}_t^{\mathsf{T}} \cdot \mathbf{d})\mathbf{d}\right|, \tag{4.21}$$

where $\bar{\mathbf{\Gamma}}_t^{k+1}$ contains the coefficients previously selected at layers $k+1, \ldots, L_t$. Since we will use MP decompositions when using ITDs (to enjoy reduced decomposition complexity), we will likewise use this simpler, MP-based approach in (4.21) in the training procedure.

To now devise a solution to the outer minimization of (4.18) we first note that, after solving the inner minimization, the squared term inside expression (4.18) can be written as follows:

$$\begin{aligned} &\left|\mathbf{y}_t - \left[\, \mathbf{S}_t^{k-1} \mid \bar{\mathbf{S}}_t^{k+1} \,\right] \left[\, (\mathbf{\Gamma}_t^{k-1})^{\mathsf{T}} \;\mid\; (\bar{\mathbf{\Gamma}}_t^{k+1})^{\mathsf{T}} \,\right]^{\mathsf{T}} - \gamma_{k,t} \cdot \mathbf{d}_t^k\right|^2 \\ &\quad = |\mathbf{q}_t^k - \gamma_{k,t} \cdot \mathbf{d}_t^k|^2, \end{aligned} \tag{4.22}$$

where $\mathbf{\Gamma}_t^{k-1}$ / $\bar{\mathbf{\Gamma}}_t^{k+1}$, $\gamma_{k,t}$, and $\mathbf{d}_t^k$ are the coefficients vectors, the k-th coefficient, and the atom selected for $\mathbf{y}_t$. Note that, in the first top-down iteration, $\mathbf{q}_t^k$ will equal the k-th approximation residue $\mathbf{r}^k$. For latter iterations, $\mathbf{q}_t^k$ can be though of as the residue resulting from approximations using atoms from layers $1, \ldots, k-1, k+1, \ldots, L_t$.

If we let $a_t \in \{1, \ldots, N\}$ denote the index of the chosen atom $\mathbf{d}_t^k \in \mathbf{D}_k$, then each vector $\mathbf{q}_t^k$ in (4.22) can be thought of as belonging to class number a_t out of N possible classes, one per each of N atoms $\mathbf{d}_{k1}, \ldots, \mathbf{d}_{kN} \in \mathbf{D}_k$ of the k-th layer. Let $f(\mathbf{y}_t)$ denote the atom selection method (either of (4.20) or (4.21)). The class corresponding to the a-th atom will be comprised of those vectors with indices

$$\mathcal{Q}_{ka} = \{t \in \mathcal{T}_k : f(\mathbf{y}_t) = \mathbf{d}_{ka}\}. \tag{4.23}$$

This classification process defines the first step of a two-step iterative solution to the outer minimization of (4.18). To obtain the second step of each iteration, we use the partition defined by the classes $\mathcal{Q}_{ka}, a = 1, \ldots, N$, to write the outer optimization of (4.18) as follows:

$$\mathbf{D}_k = \underset{(\mathbf{d}_a)_{a=1}^N \in \mathcal{N}}{\operatorname{argmin}} \sum_{a=1}^{N} \left| (\mathbf{q}_t^k)_{t \in \mathcal{Q}_{ka}} - \mathbf{d}_a \cdot (\gamma_{k,t})_{t \in \mathcal{Q}_{ka}} \right|_F^2, \tag{4.24}$$

where $(\mathbf{q}_t^k)_{t \in \mathcal{Q}_{ka}}$ (respectively, $(\gamma_{k,t})_{t \in \mathcal{Q}_{ka}}$) is the matrix with columns $\mathbf{q}_t^k$ (row-vector with coefficients $\gamma_{k,t}$) such that $t \in \mathcal{Q}_{ka}$. Since the cost of the above optimization consists of a sum of squares and each column $\mathbf{d}_a$ intervenes in a single summation term, the problem can be split into N optimizations, one per atom $\mathbf{d}_{ka}, a = 1, \ldots, N$, each having the form

$$\mathbf{d}_{ka} = \underset{\mathbf{d} \in \mathbb{R}^d, \mathbf{d}^\mathsf{T}\mathbf{d}=1}{\operatorname{argmin}} \; \min_{\boldsymbol{\omega} \in \mathbb{R}^{1 \times |\mathcal{Q}_{ka}|}} \sum_{a=1}^{N} \left| (\mathbf{q}_t^k)_{t \in \mathcal{Q}_{ka}} - \mathbf{d} \cdot \boldsymbol{\omega} \right|_F^2. \tag{4.25}$$

Note that, unlike what is specified in (4.24), we have further considered simultaneously updating the coefficients $(\gamma_{k,t})_{t \in \mathcal{Q}_{ka}}$ related to $\mathbf{d}_{ka}$ (by means of the optimization variable $\boldsymbol{\omega}$). The reason we do this is that, since the term $\mathbf{d} \cdot \boldsymbol{\omega}$ is a rank-1 approximation of $(\mathbf{q}_t^k)_{t \in \mathcal{Q}_{ka}}$, the solution $\mathbf{d}_{ka}$ in the joint optimization is just the first singular vector of $(\mathbf{q}_t^k)_{t \in \mathcal{Q}_{ka}}$ [Klema 1980]. The coefficients obtained as a consequence of this joint optimization are further exploited in future iterations of the top-down optimization procedure.

We summarize the layer update algorithm in Fig. 4.2.

The layer update procedure above described in fact corresponds to the K-SVD algorithm [Aharon 2006b] for the case when the sparse representations used in the K-SVD training process are set to be unit l-0 norm and the input training set is $\{\mathbf{q}_t^k\}_{t \in \mathcal{T}_k}$. Using unit l-0 norm decompositions in the training process has the important advantage that the exact optimal atom can be selected every time. Every dictionary training scheme available in the literature (*eg.,* [Aharon 2006b, Rubinstein 2010a, Mairal 2010a]) employs sparse representations with sparsity greater than one, and these can only be approximated (*eg.,* using one of the matching

1: **Input**: $\{\mathbf{y}_t, \mathbf{S}_t^{L_t}, \mathbf{\Gamma}_t^{L_t}\}_{t\in\mathcal{T}_k}$, $\mathbf{D}_k$ *(optional)*
2: **Output**: $\mathbf{D}_k$, $\{\mathbf{d}_t^k, \mathbf{\Gamma}_t^{L_t}\}_{t\in\mathcal{T}_k}$,

3: **Initialization**:
4: **if** input $\mathbf{D}_k$ not provided **then**
5: $\quad\mathcal{I} \leftarrow \textsc{RandSelect}(N, T)$
6: $\quad\mathbf{D}_k \leftarrow (\mathbf{y}_t)_{t\in\mathcal{I}}$
7: **end if**

8: **Algorithm**:
9: **repeat**
10: $\quad$**for** $t \in \mathcal{T}_k$ **do**
11: $\quad\quad\mathbf{d}_t^k, \mathbf{q}_t^k, \mathbf{\Gamma}_t^{L_t} \leftarrow \textsc{Decomp}(\mathbf{y}_t, \mathbf{S}_t^{L_t}, \mathbf{\Gamma}_t^{L_t})$
12: $\quad\quad a = \textsc{Index}(\mathbf{d}_t^k)$
13: $\quad\quad\mathcal{Q}_{ka} \leftarrow \mathcal{Q}_{ka} \bigcup t$
14: $\quad$**end for**
15: $\quad$**for** $a = 1$ **to** N **do**
16: $\quad\quad(\mathbf{u}, \sigma, \mathbf{v}) \leftarrow \mathrm{SVD1}\big((\mathbf{q}_t^k)_{t\in\mathcal{Q}_{ka}}\big)$
17: $\quad\quad\mathbf{d}_{ka} \leftarrow \mathbf{u}$ (where $\mathbf{D}_k = (\mathbf{d}_{ka})_{a=1}^N$)
18: $\quad\quad(\gamma_{k,t})_{t\in\mathcal{Q}_{ka}} \leftarrow \sigma \cdot \mathbf{v}^\mathsf{T}$
19: $\quad$**end for**
20: **until** convergence of $\mathbf{D}_k$

Figure 4.2: The layer update algorithm. The function $\textsc{RandSelect}(N, T)$ selects N integers in $[1, N]$ without repetition. Function $\textsc{Decomp}$ implements one of (4.20) or (4.21) and returns the selected atom $\mathbf{d}_t^k$, the coefficients $\mathbf{\Gamma}_t^{L_t}$, and the vector $\mathbf{q}_t^k$ appearing in (4.22) computed using the new coefficients $\mathbf{\Gamma}_t^{L_t}$. Function $\textsc{Index}$ in line 12 returns the atom index $a \in \{1, \ldots, n\}$ of the atom $\mathbf{d}_t^k$ such that, for $\mathbf{D}_k = (\mathbf{d}_{k\alpha})_{\alpha=1}^N$, $\mathbf{d}_t^k = \mathbf{d}_{ka}$. Function SVD1 in line 16 produces the first left-singular vector, first singular value and first right-singular vector, where we assume ordering according to decreasing magnitude of singular value.

pursuit methods or basis pursuit). Optimality guarantees for approximate algorithms only exist for low sparsity values and when the dictionary has very low coherence [Tropp 2004], meaning that the over-completeness factor needs to be small.

4.4.3 Convergence

The ITD training procedure in Fig. 4.1 and the layer update procedure in Fig. 4.2 are both guaranteed to reduce the optimization cost given by the approximation

error of the training vectors

$$\sum_{t=1}^{T} |\mathbf{y}_t - \mathbf{S}_t^{L_t} \mathbf{\Gamma}_t^{L_t}|^2. \tag{4.26}$$

Hence both procedures are guaranteed to converge. Regarding the ITD training procedure in Fig. 4.1, the approximation error of the training vectors will only be affected by the layer update procedure and hence it will converge if the layer update procedure ensures a reduction in the cumulative error of the training vectors. This is indeed the case, as can be seen by noting that the layer update procedure itself consists of two steps that each are guaranteed to reduce the representation error. The first step produces the classes in (4.23). It depends on single-atom sparse representations, and these are always solved exactly and hence a reduction in the approximation error of each vector $\mathbf{y}_t$ is guaranteed. In the second step, all atoms of the dictionary are updated following (4.25). This update replaces the rank-1 approximation of the class matrices $(\mathbf{q}_t^k)_{t \in \mathcal{Q}_{ka}}$ with the optimal rank-1 approximation using the SVD. This update ensures a reduction in the cumulative approximation error of each class and hence likewise ensures the convergence of the procedure.

The convergence of the iterative top-down training procedure is illustrated for experimental data (described in the results section) in Fig. 4.3 for an ITD having $C = 16$ layers and $n = 16$ atoms per layer (*i.e.,* $N = 256$), trained using $J = 3$ iterations of the top-down procedure. Most of the reduction of the cost is achieved in the first iteration of the top-down procedure, and this is expected since the layers are built for the first time in the first top-down iteration. Note, however, that running the top-down training for two more iterations succeeds in reducing the cost by more than the reduction achieved by the last two layers in the first iteration (as emphasized by the three horizontal lines). The last two layers (and any other layer), however, require storage space and exploiting them incurs decomposition complexity. In general, using iterative top-down training as opposed to the non-iterative variant will bring noticeable benefits only for some ITD configurations.

4.4.4 Example of ITD structure

In Fig. 4.4 we illustrate ITD dictionaries of layers $i = 1, 3, 5, \ldots, 15$ trained on a large set of 8×8 image patches and using $J = 1$ iterations of the top-down training algorithm in Fig. 4.1. Note how the spatial frequencies comprising the dictionaries increase along with the layer index i. The spatial correlation of atom pixels is also seen to decrease with increasing layer index (*i.e.,* the atoms become less structured and more like noise).

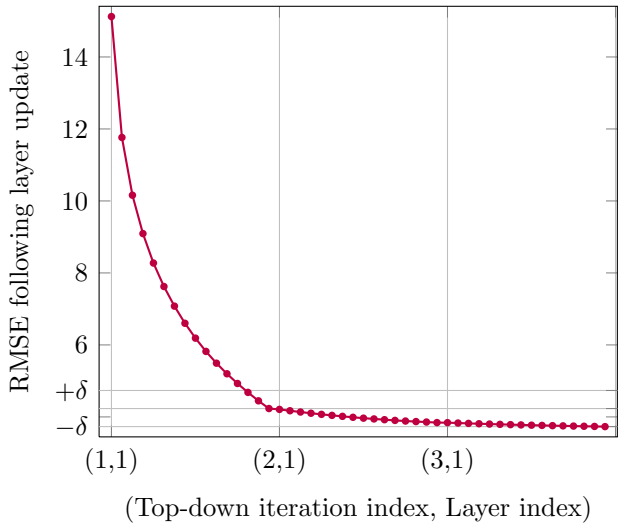

Figure 4.3: Minimization cost in (4.26) (expressed in RMSE) throughout $J = 3$ consecutive iterations of the top-down approach in Fig. 4.1 for an ITD having $C = 16$ layers and $n = 16$ atoms per layer. The top-down iteration index and layer index in the abscissa correspond, respectively, to indices j and k in Fig. 4.1. Two horizontal grid lines are drawn at $\pm\delta$ from a center line crossing the RMSE at indices $(1, 16)$, where δ is the RMSE difference from this point to that at indices $(3, 16)$.

4.5 Results

In this section we evaluate our proposed ITD dictionary and the related concatenated ITD (cITD) fixed-dictionary by comparing them against the K-SVD dictionary of Aharon *et al.* [Aharon 2006b] and the Sparse Dictionary (SD) of Rubinstein *et al.* [Rubinstein 2010a]. We carry out two experiments in our evaluation: In the first experiment we evaluate the PSNR of the sparse approximations obtained with the various dictionaries as a function of sparsity. Sparser approximations of signal vectors are likely to result in better performance in various real-life applications [Guleryuz 2006, Mallat 2008]. In the second experiment we thus compute experimentally the rate-distortion bound achievable with the various dictionaries. We will see that cITD outperforms the reference dictionaries for all rates and can even perform comparably with only half the number of atoms. ITD can offer small gains in rate-distortion trade-off, but these come at an important complexity advantage. This complexity advantage makes it possible to tap into the rate-distortion gains offered by large dictionary over-completeness factors without incurring the related complexity penalty.

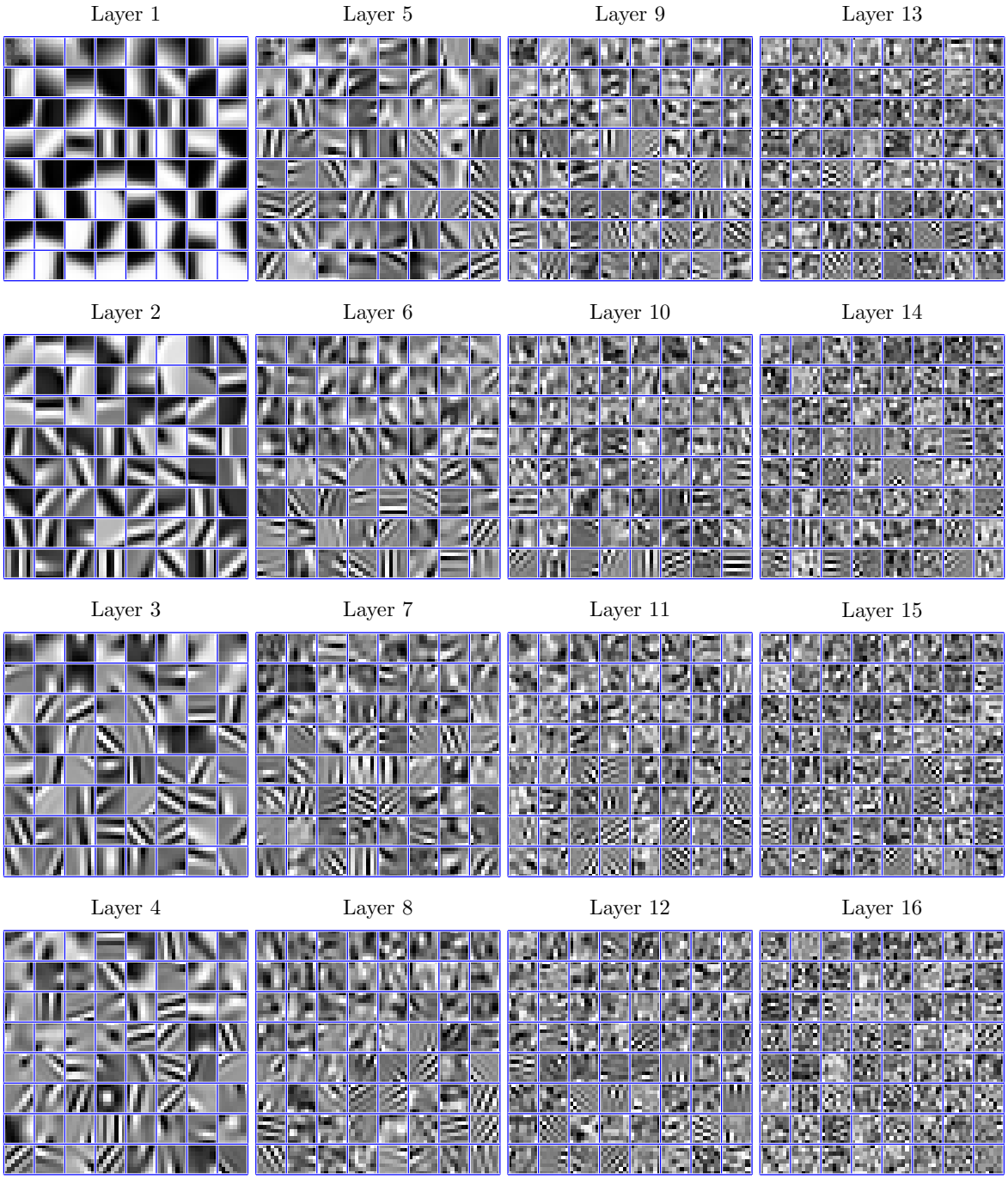

Figure 4.4: An example of ITD layers with $C = 16$, $n = 64$ trained on 8×8 image blocks using non-iterative top-down training.

Figure 4.5: Sample images from the FERET dataset.

4.5.1 Experimental setup

4.5.1.1 Dataset

Throughout our experiments we will use image blocks of size 8×8 to form signal vectors $\mathbf{y}$. The blocks are taken (without overlap) from frontal pose face images of 545 different subjects taken from the FERET dataset [Phillips 2000]: the first 445 images comprise the training set and the remaining 100 images comprise the test set. The size of each image is 768×512, for a total of 2.7×10^6 possible training blocks and 6.1×10^5 possible test blocks. Since all images contain a large background, we remove from each set the 30% least energetic (following mean removal) blocks and randomly select 9.11×10^5 training vectors and 2.05×10^5 testing vectors. The FERET dataset is similar to the one used in [Aharon 2006b] (see the sample thumbnail images in Fig. 4.5) and has the advantage of consisting of a large set of uncompressed images.

4.5.1.2 ITD / cITD

In order to keep the comparison fair, we will use ITD structures having C layers each of n atoms so that the total number of atoms $N = C \cdot n$ in the ITD structure equals the number of atoms N of the reference dictionaries. Decompositions using the ITD scheme will be carried out using MP, where the atom selected in the i-th iteration is constrained to be one of the n atoms of the i-th layer. For simplicity, we do not propagate the last layer $\mathbf{D}_C$ and force all sparsities (for ITD decompositions) to be at most C.

We will also consider a concatenated ITD (cITD) scheme wherein the i-th chosen atom is selected from all $N = C \cdot n$ ITD atoms. The training procedure used to obtain cITD is the same as that described in Fig. 4.1. To keep the training complexity low, we use the MP-based layer update process in (4.21). Unless stated otherwise, we will run $J = 1$ top-down iterations (*cf.* Fig. 4.2), and we always use

	ITD		cITD	
N	C	n	C	n
256	16	16	16	16
512	16	32	16	32
1024	16	64	16	64
2048	16	128	8	256
4096	16	256	8	512

Table 4.2: Number of layers C and atoms per layer n for a given cumulative number of atoms $N = C \cdot n$.

$\varepsilon = 0$ (*i.e.,* $\forall t, L_t = C$). We specify the number of layers C and atoms per layer n used for a given $N = C \cdot n$ for both ITD and cITD in Table 4.2; these values where chosen experimentally.

4.5.1.3 Matching pursuits

While we use MP to decompose signals with ITD, we will use OMP to decompose signals using the all reference fixed-dictionaries. This will favor the reference dictionaries since OMP updates all coefficients at each iteration i using $\mathbf{\Gamma}^i = (\mathbf{S}^i)^+ \mathbf{y}$, thus producing sparser representations. Since OMP uses the same atom-selection rule as MP, its complexity can be lower-bounded by the N inner-product computations of the atom selection step (see the background discussion in Section 4.2) if we neglect the extra complexity of the pseudo-inverse computation.

Note that we also use OMP to obtain cITD dictionary decompositions. In this case the decomposition complexity at a given sparsity will be equal to that of K-SVD using the same number of atoms.

4.5.1.4 Training of reference dictionaries

We train K-SVD dictionaries using a modified version of the software made publicly available by the authors [Aharon 2006a]. K-SVD is not guaranteed to converge when using practical solutions (*i.e.,* OMP) to the sparse decomposition step of the K-SVD training procedure. This is problematic in particular when training large (high coherence) K-SVD dictionaries as we do here. Hence, as suggested in [Aharon 2006b], at each K-SVD training iteration, we update the sparse decompositions of a given training vector only when the update reduces the representation error.

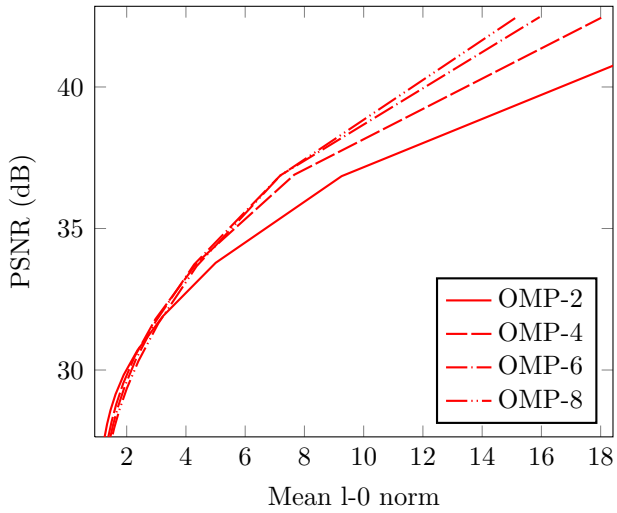

Figure 4.6: Performance of K-SVD (N=512) in terms of PSNR versus sparsity when using various OMP training sparsity thresholds.

As in [Aharon 2006b, Bryt 2008], we use a fixed-sparsity OMP decomposition to train K-SVD dictionaries. To make the comparisons fair, we will consider multiple sparsity values ($L = 2, 4, 6, 8$) for each dictionary size N. The results plotted for K-SVD in both experiments corresponds to the maximum PSNR obtained using any of the 4 dictionaries available for that N value. See Fig. 4.6 for an example of the performance of the four resulting K-SVD dictionaries (in terms of PSNR versus sparsity) when using $N = 512$ atoms.

The SD scheme is likewise trained using the software made publicly available by the authors. We do not modify the SD scheme to attempt to ensure convergence as this issue is not covered in [Rubinstein 2010a]. Note that SD employs a sparse decomposition procedure in two different steps of each training iteration, and hence the convergence issues are more difficult to address. When training SD, we consider 24 different dictionaries per value of $N = 256, 1024, 4096$ corresponding to training vector sparsities from $L = 2, 4, 6, 8, 10$ and dictionary atom sparsities from $g = 8, 12, 15, 16, 17, 24$ (*cf.* Section 4.2.3). Again we plot, in both experiments, the maximum PSNR from that produced by any of the 24 dictionaries available for that N.

Both K-SVD and SD training procedures are run during 70 iterations.

4.5.2 Experiment 1: Sparsity vs. PSNR

In the first experiment we evaluate how well ITD and cITD can approximate the test set signal vectors by plotting the average PSNR versus average sparsity, when

varying an RMSE threshold used as the OMP stopping criterion.

The results for ITD are plotted against those for K-SVD in Fig. 4.7. Generally the ITD setup under-performs K-SVD by a wide margin when both schemes have the same total number of atoms N. This is expected given that the ITD scheme is structured and forces the i-th selected atom to come from the subset of n atoms that make up the i-th layer. In other words, the effective number of atoms made available by the ITD schemes is somewhere between the number of atoms per layer n and the total number of atoms N. In the second experiment (on rate-distortion tradeoffs achievable with a given dictionary) we will be more concerned by physical quantities (complexity and rate) that better reflect this effective number of atoms. We will see then that the ITD structure offers important advantages in this application.

The results for the cITD scheme, illustrated in Fig. 4.9 tell a different story. By removing the structured atom selection constraint, the cITD scheme uniformly outperforms K-SVD for all N. This result seems unexpected given the fact that, unlike the ITD training formulation, the K-SVD training formulation does not constrain the atom selection process. Yet, as discussed in Section 4.3.3, practical sparse decomposition algorithms can only approximate the true solution to the sparse representation problem (4.1). K-SVD uses this low-quality approximation in its training procedure. On the other hand, the ITD training scheme relies on single-atom representations, which are solved exactly with a single MP iteration. Theoretical results [Tropp 2004] show that the quality of the OMP approximation decreases with increasing dictionary coherence which, in turn, increases with increasing dictionary size. This observation is in line with the fact that the cITD / K-SVD performance gap increases with N.

In Fig. 4.8 we compare ITD to the SD scheme. Note that, for the larges dictionary size $N = 4096$, ITD can outperform SD. For $N = 256, 1024$, ITD can perform comparably to SD for low sparsity values. The advantage of the SD scheme, however, lies in the low complexity of the dictionary-vector product computation. We will address the issue of decomposition complexity in the second experiment.

In Fig. 4.10 we evaluate the influence of the number of top-down iterations J used in the ITD training procedure by plotting the experimental sparsity versus PSNR computed on the test data. We only present results for the cITD dictionary with $N = 256$ atoms, as the gain for ITD and for larger cITD dictionaries was marginal for $J > 1$. The reduction in the training cost for the cITD $N = 256$ training procedure when using $J = 3$ was previously illustrated in Fig. 4.3. The results plotted in Fig. 4.10 illustrate that using $J = 3$ also results in improved performance on the test set relative to non-iterative top-down training (with $J = 1$). At the higher sparsity values shown, using $J = 3$ instead of $J = 1$ increases

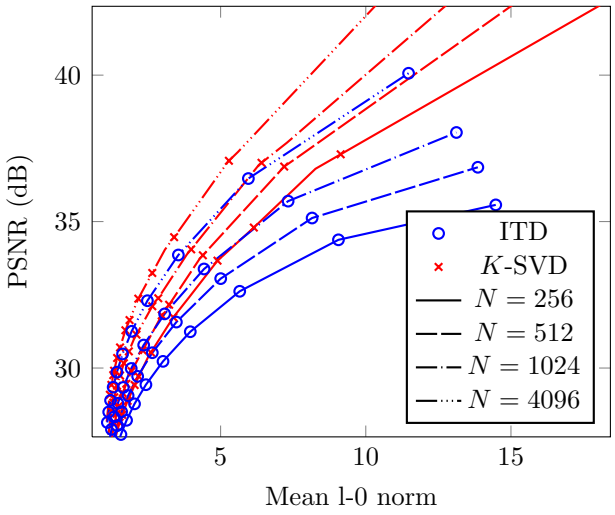

Figure 4.7: PSNR versus sparsity: K-SVD versus ITD.

the performance by 0.27 dB. The gain over K-SVD reaches 0.38 dB.

4.5.3 Experiment 2: Rate-distortion performance

In the second experiment we evaluate the performance of ITD in image compression. There is a large range of design parameters that define a full image codec. An overcomplete dictionary is only a possible transform that can be used by the codec. Other components of an image codec include the coefficient quantization mechanism, the selection of the representation sparsity of each image block, methods such as deblocking filters used to deal with blocking artifacts, prediction mechanism and other more elaborate methods such as the affine warping scheme used in [Bryt 2008]. Since we are only interested in evaluating the performance of the dictionaries in this application, we will proceed by computing experimentally the rate-distortion bound achievable by each dictionary. This will avoid any biases in the comparison resulting from external factors related to the codec design. By establishing that a dictionary produces improved rate-distortion bounds, we establish which dictionary is a better candidate for use in an image codec based on overcomplete dictionaries. Indeed we will see that our approach can achieve better rate-distortion tradeoffs at a significantly reduced decomposition complexity and this without the overhead of storing the Gram matrix as required to achieve low complexity with fixed-dictionary approaches (*cf.* Section 4.2).

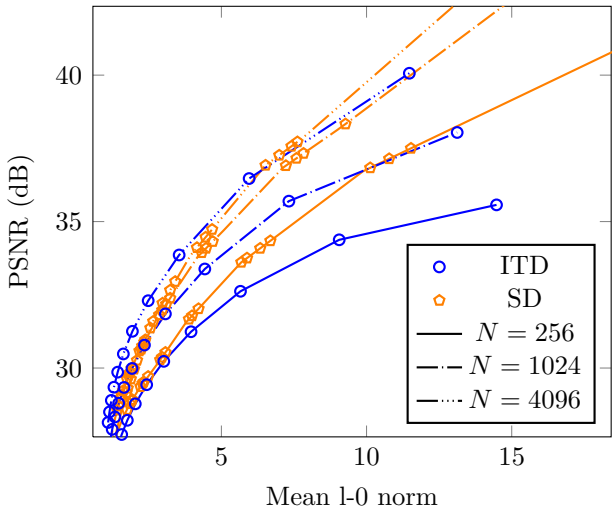

Figure 4.8: PSNR versus sparsity: SD versus ITD.

4.5.3.1 Computation of rate-distortion curves

We illustrate our method for computing the rate-distortion curve of a given dictionary in Fig. 4.11: First we compute the sparse representations of the test set vectors using multiple RMSE thresholds $\varepsilon = 2, 4, 6 \ldots, 22$. For each ε, we quantize all coefficients using a quantization step Δ:

$$\hat{\gamma}_i = \Delta \cdot \text{round}(\frac{\gamma_i}{\Delta}). \tag{4.27}$$

The value of Δ is varied to obtain one rate-distortion sub-curve per each ε. The final rate-distortion curve is given by the envelope of all these sub-curves.

For each (ε, Δ) pair in the sub-curves of Fig. 4.11, the distortion plotted corresponds to the reconstruction error averaged over all test vectors. The value of rate plotted is computed as follows: Let $(a_1, \hat{\gamma}_1), \ldots, (a_L, \hat{\gamma}_L)$, denote the quantized sparse representation. This can be represented using the following rate (in bits-per-pixel)

$$B = \left(\mathrm{R}\left(L\right) + \sum_{i=1}^{L} \left(\mathrm{R}\left(a_i\right) + \mathrm{R}\left(\hat{\gamma}_i\right) \right) \right) / d, \tag{4.28}$$

where $\mathrm{R}\left(\cdot\right)$ returns the number of bits required to represent the input symbol. Letting $\mathcal{H}\left(\cdot\right)$ denote source entropy and assuming that $\mathcal{H}\left(a_i\right)$ and $\mathcal{H}\left(\hat{\gamma}_i\right)$ are both independent of i, the expectation of the above expression is lower bounded as follows:

$$\mathrm{E}\left(B\right) \geq \left(\mathcal{H}\left(L\right) + \mathrm{E}\left(L\right) \cdot \left(\mathcal{H}\left(a\right) + \mathcal{H}\left(\hat{\gamma}\right) \right) \right) / d. \tag{4.29}$$

The rates plotted throughout are obtained using the above expression with expectations and entropies estimated over the test set. We present results for bit rates

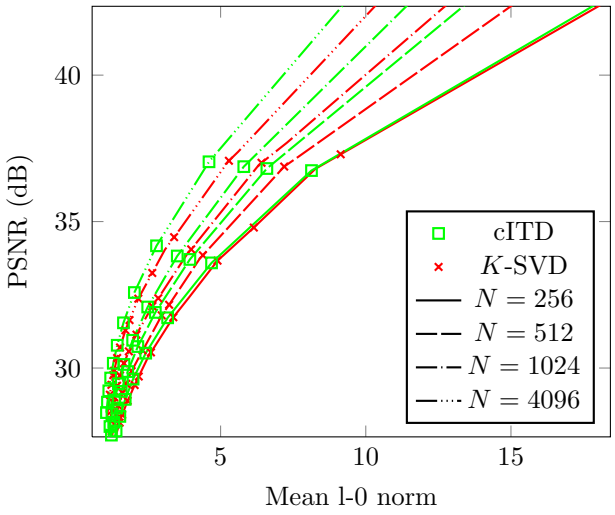

Figure 4.9: PSNR versus sparsity: K-SVD versus concatenated ITD.

between 0.2 bpp and 1.3 bpp which covers all general purpose scenarios.

As done in the first experiment, for the reference dictionaries we plot the maximum PSNR at each rate from all the rate-distortion bounds produced by each of the reference dictionaries available for each N. Recall that there are 4 K-SVD dictionaries available for each N and 24 SD dictionaries corresponding to various training parameters.

Since each point on the envelope curve is associated to a single ε, we can likewise associate a single experimental mean sparsity per point of the experimental rate-distortion bound; we illustrate this sparsity L in the legend of the example in Fig. 4.11.

4.5.3.2 Concatenated ITD

In Fig. 4.12 we evaluate the rate-distortion performance of cITD versus K-SVD for three different dictionary sizes $N = 256, 512, 4096$. Note that the cITD curves for $N = 256$ and 512 perform comparably to the K-SVD curves having, respectively, $N = 512$ and 1024 (*i.e.,* twice as many atoms).

When both dictionaries have the same size, cITD offers important rate-distortion gains (particularly for $N > 256$). The cITD scheme at $N = 512$ outperforms K-SVD by 0.30 dB at 0.4 bpp. This gain over K-SVD increases with the rate, reaching 0.53 dB at 1 bpp. A similar behavior is observed for $N = 1024$, with cITD's gain over KSVD rising from 0.51 dB to 0.92 dB between 0.3 bpp and 1 bpp. The increased rate-distortion performance of cITD over K-SVD is not unexpected given the increased sparsity versus PSNR performance of cITD (*cf.* Fig. 4.9).

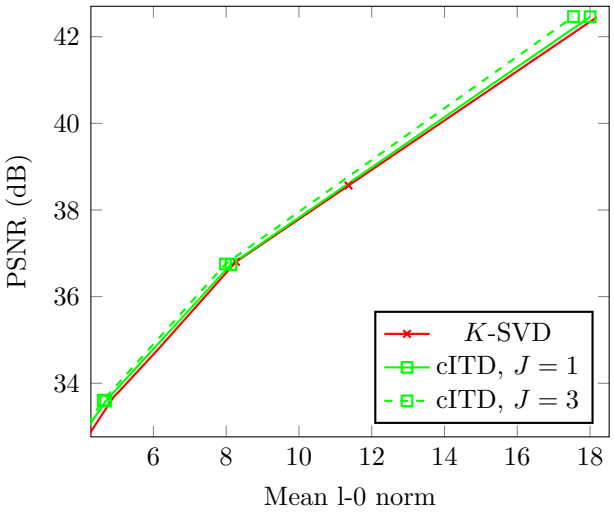

Figure 4.10: PSNR versus sparsity: K-SVD versus cITD for $N = 256$; cITD trained using $J = 1, 3$ top-down training iterations.

4.5.3.3 ITD

In Fig. 4.13 and Fig. 4.14 we plot, respectively, decomposition complexities as a function of rate and experimental rate-distortion bounds. In both curves we compare ITD and K-SVD for $N = 256, 1024$ and 4096. The corresponding comparisons with SD are shown in Fig. 4.15 and Fig. 4.16.

Complexity We measure decomposition complexity in terms of the number of inner-product computations required by the atom selection process (*cf.* Section 4.2). For an average decomposition sparsity L, the measured ITD complexity will hence be given by $L{\cdot}n \leq N$ (n the number of atoms per layer). For fixed dictionaries, two possible complexities are illustrated: If the pre-computed Gram matrix $\mathbf{G}$ is assumed to be available, then the complexity is fixed, for all L, to the N inner-product computations of the dictionary-vector product $\mathbf{D}^{\mathsf{T}}\mathbf{y}$, denoted by the light horizontal lines at $256, 1024$ and 4096 in Fig. 4.13 and at $92, 337$ and 1286 (these values are given by (4.11) using $b = 8, g = 12, d = 64$) in Fig. 4.15. Otherwise, the measured complexity, as illustrated by the curves labeled K-SVD in Fig. 4.13 and SD in Fig. 4.15 is given by L times the dictionary-vector product, where L is the average sparsity at a given rate. As discussed in Section 4.2 these complexity values represent the real complexity for MP-based decompositions (ITD uses MP) whereas they are only a lower bound for OMP-based decompositions (K-SVD uses OMP).

The first observation to make regarding the results in Fig. 4.13 is that for all N, the ITD decomposition complexity at any rate is more than one order of

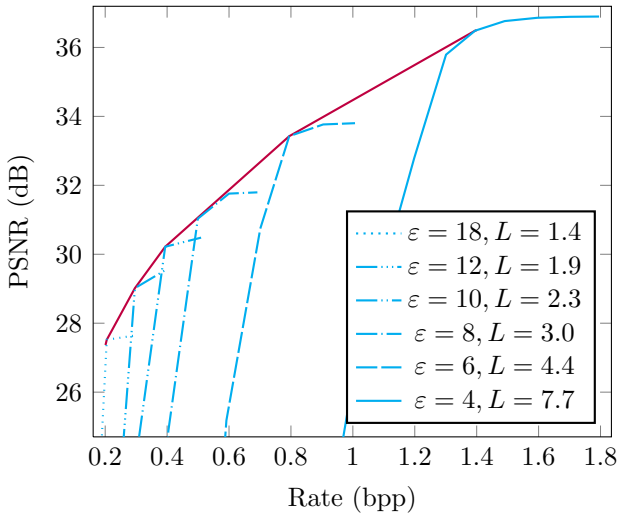

Figure 4.11: Each sub-curve is obtained for a fixed ε by varying the quantization step size. The rate-distortion curve of a given dictionary is given by the envelope of these sub-curves.

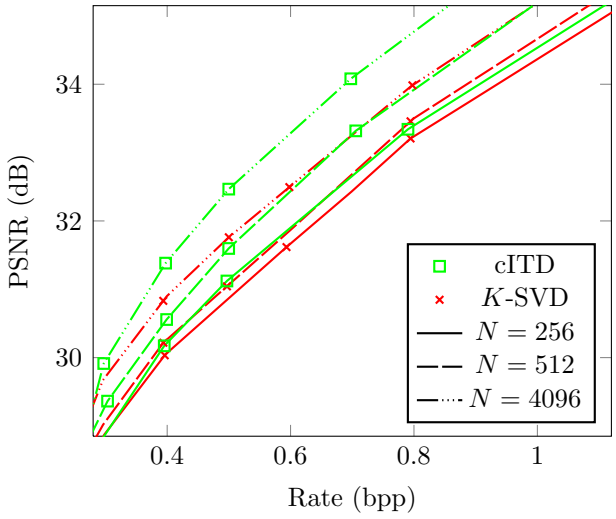

Figure 4.12: Experimental rate-distortion bounds: cITD versus K-SVD .

magnitude below the K-SVD complexity for the same N. In fact, ITD decomposition complexities for all N and all rates are always below the K-SVD , $N = 256$ complexity curve. This is important for memory sensitive applications where one cannot afford to store the dictionary's Gram matrix.

The ITD curves for all N even offer an important complexity advantage, at all rates, relative to the dictionary-vector product at the corresponding N (recall that this is a lower bound for the atom selection complexity when using Gram

matrix acceleration). The complexity advantage is large enough so that, for all rates below 0.65 bpp, ITD for $N = 1024$ and $N = 4096$ incurs a complexity that is even lower than that of the dictionary-vector product respectively at $N = 256$ and $N = 1024$ (*i.e.,* half the number of atoms).

The ITD complexity gains are more modest when compared to the SD scheme which was designed with complexity in mind. Yet even here the ITD complexity for all N is close to an order of magnitude below the corresponding complexity curve of SD. For a large range of bit-rates (up to 0.7 bpp for the ITD, $N = 256$ curve) the complexities are also below the dictionary-vector product complexities of the corresponding SD dictionary.

Rate-distortion The rate-distortion bounds for the SD approach, illustrated in Fig. 4.15, are poor, yet this is expected given the poor performance of SD in the sparsity versus PSNR experiment. Thus we will focus in the remainder of the section on the comparison against K-SVD .

The ITD, $N = 256$ setup does not outperform the K-SVD , $N = 256$ setup, yet, for all rates below 0.5 bpp, the distortion penalty is less than 0.16 dB. Note that the corresponding complexity gain of the ITD, $N = 256$ setup relative to the $N = 256$ dictionary-vector product is $10\times$ at 0.2 bpp and $4\times$ at 0.5 bpp, and hence a rate-distortion penalty of 0.16 dB is justified.

The ITD, $N = 1024$ curve does enjoy an important distortion gain over the K-SVD , $N = 256$ setup. The gain reaches 0.58 dB at 0.6 bpp, and yet the required complexity is also below the $N = 256$ dictionary-vector product for all rates below 0.6 bpp. When compared to the K-SVD , $N = 1024$ setup, a distortion gain is observed for all rates below 0.7 bpp; the gain reaches 0.24 dB at 0.5 bpp.

The ITD, $N = 4096$ curve enjoys a distortion gain over K-SVD , $N = 4096$ for all rates below 1.1 bpp. The gain reaches 0.3 dB at 0.6 bpp. Relative to the K-SVD, $N = 256$ curve, the distortion gain reaches 1.15 dB. The required complexity is above that of the $N = 256$ dictionary-vector product, yet it is uniformly below the K-SVD , $N = 256$ curve.

In light of the poor PSNR versus sparsity results of the ITD scheme (*cf.* Fig. 4.7) the ITD rate-distortion bounds illustrated in Fig. 4.14 might be unexpected. Yet one must keep in mind that ITD offers an important advantage as all its atom indices are in $[1, n]$. With $n = 16$ (*cf.* Table 4.2) one needs at most 4 bits to represent each atom index. For K-SVD , one will need at most 8 bits for $N = 256$ and 12 bits for $N = 4096$.

The large distortion gains offered by the ITD setup for $N = 1024$ and $N = 4096$ relative to K-SVD , $N = 256$ are the result of a much larger dictionary overcompleteness. While comparable gains are achievable with K-SVD dictionaries using $N = 1024$ and $N = 4096$, it is important to note that the large decomposi-

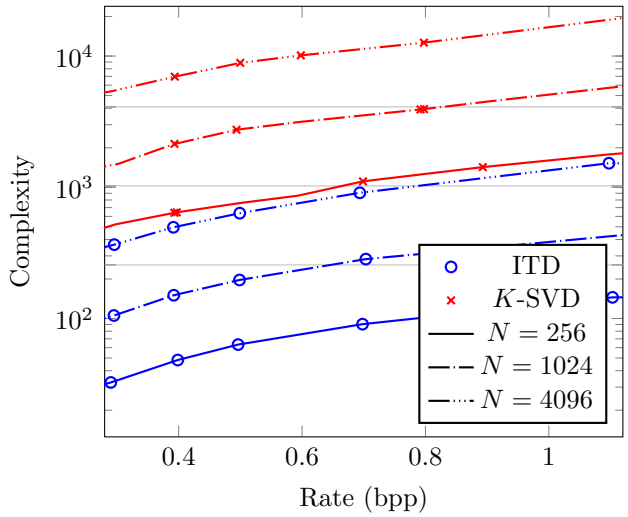

Figure 4.13: Decomposition complexity: ITD versus K-SVD . The complexity is given by the atom selection cost expressed as the number of d-dimensional inner products. The horizontal lines display the single dictionary-vector product cost which equals N.

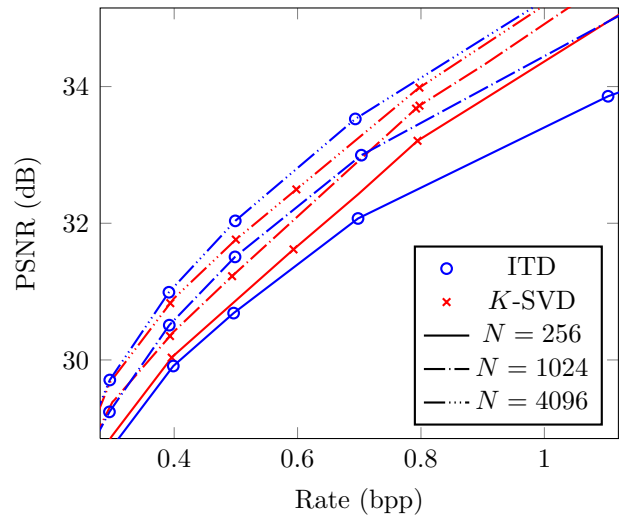

Figure 4.14: Rate-distortion curves: ITD versus K-SVD

tion complexities required by fixed-dictionary schemes in general (including cITD) makes such large over-completeness prohibitive. Hence the ITD scheme provides an affordable mechanism, memory and complexity-wise, to tap into the benefits of large over-completeness.

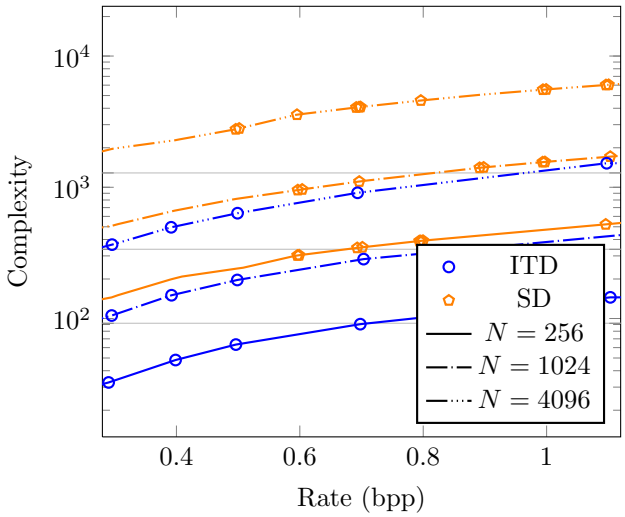

Figure 4.15: Decomposition complexity: ITD versus SD. The complexity is given by the atom selection cost expressed as the number of d-dimensional inner products. The horizontal lines display the single dictionary-vector product cost given by (4.11).

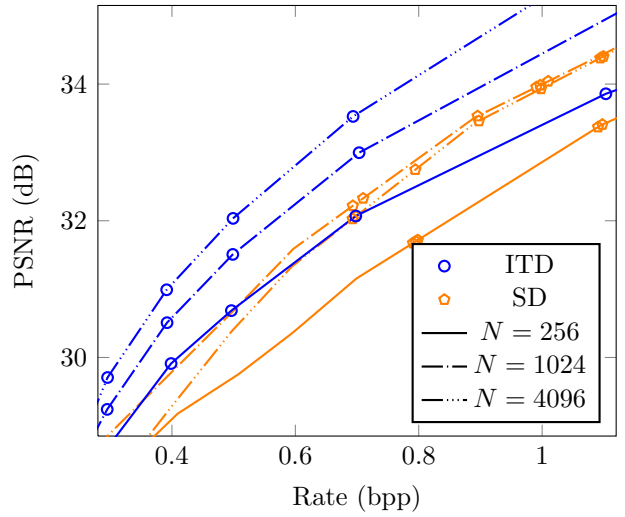

Figure 4.16: Rate-distortion curves: ITD versus SD

4.6 Conclusion

In this paper we introduced a novel learned, structured dictionary called an *Iteration Tuned Dictionary* (ITD). ITDs are layered structures, with each layer i consisting of a dictionary matrix that is used in the i-th iteration of a matching pursuit decomposition. We presented an ITD training procedure that, unlike other training schemes found in the literature, relies on single atom sparse repre-

sentations. A concatenated ITD (cITD) dictionary was also presented, consisting of a concatenation of the dictionary matrices of all the ITD layers. The cITD dictionary was shown to outperform other methods found in the literature in two different evaluations, PSNR versus sparsity and achievable rate-distortion tradeoffs. The ITD scheme likewise offered gains in achievable rate-distortion tradeoffs, while at the same time requiring a decomposition complexity well below that of traditional, fixed-dictionary approaches using the same total number of atoms. This complexity gain enables the ITD scheme to tap into the important rate-distortion gains achievable with increased dictionary overcompleteness without paying a complexity penalty, or an extra storage penalty in the form of the Gram matrix commonly used to reduce decomposition complexity.

Chapter 5

Tree-Structured Iteration Tuned Dictionaries

5.1 Introduction

In the last chapter, we showed that Iteration-Tuned Dictionaries (ITDs) enable the use of highly overcomplete dictionaries with nonetheless low decomposition complexity. The ITD approach consists of substituting the dictionary matrix $\mathbf{D}_i$ in each iteration i of a Matching Pursuit (MP) decomposition. The matrices $\mathbf{D}_i$ are trained to make them suitable to the residues at the input of the i-th MP iteration.

In this chapter we introduce a new ITD variant, the Tree-Structured Iteration-Tuned Dictionary, which is a constrained case of the ITD setup previously introduced. The constrain consists of making the dictionary used in the i-th (called here a *candidate* dictionary) iteration depend not only on the iteration index i but also on the sequence of previously selected *ancestor* atoms. This approach ensures that TSITDs having larger overcompleteness factors require the same decomposition complexity as smaller ITDs. The resulting TSITD scheme enjoys the interesting property that a given dictionary is orthogonal to all of its ancestor atoms, meaning that they can be represented in spaces of reduced dimensionality. This property is exploited to reduce the TSITD storage footprint in the form of the reduced TSITD (rTSITD) structure.

We then present a TSITD variant called the *Iteration-Tuned and Aligned Dictionary* (ITAD) which constrains the reduced representations of a given layer of the rTSITD tree to be the same. ITAD hence enjoys a storage footprint much smaller than that of TSITD, while retaining nonetheless the large redundancy related to the TSITD tree structure. Experiments show that, for comparable complexity, ITAD indeed outperforms ITD uniformly and TSITD in practical scenarios both in terms of approximation error versus sparsity and in image compression and denoising applications. We also compare both TSITD and ITAD to the K-SVD algorithm as well as other, recently proposed approaches.

Symbol	Definition
K	Total number of dictionaries in a (r)TSITD.
C	Number of layers in an ITD/(r)TSITD/ITAD structure.
N	Number of columns in a dictionary matrix.
$\mathcal{L}_l$	Subset of $\{1, \ldots, K\}$ specifying the indices k of all dictionaries $\mathbf{D}_k$ of layer l.
$\mathbf{d}^l$	An atom from the l-th layer that is the l-th descendant atom from a sequence $\mathbf{d}^1, \mathbf{d}^2, \mathbf{d}^3, \ldots$ found along a given tree-path.
k_l	A tree-node index that is the l-th such index in a sequence of descendant nodes $k_1, k_2, k_3, \ldots$. By convention, $k_1 = 1$ specifies the root node.

Table 5.1: TSITD, rTSITD and ITAD: List of notational conventions.

5.2 Notation

We will use the notation $(\mathbf{d}_a)_{a=1}^N$ to denote the matrix $\mathbf{D}$ consisting of N ordered columns, with the a-th column being $\mathbf{d}_a$. We use the notation $\mathbf{d} \in \mathbf{D}$ to denote that $\mathbf{d}$ is a column of $\mathbf{D}$. In our work we consider sets of matrices $\{\mathbf{D}_k\}_k$, with $\mathbf{D}_k = (\mathbf{d}_{ka})_{a=1}^N$. Hence $\mathbf{d}_{k_l a_l} \in \mathbf{D}_{k_l}$, $l = 1, 2, \ldots$ specifies a sequence of columns having column indices a_l taken from the matrices with indices k_l. To simplify the notation, we will often specify $\mathbf{d}_{k_l a_l}$ using only the sequence index l as a superscript, *i.e.,* we let $\mathbf{d}^l = \mathbf{d}_{k_l a_l}$.

For convenience, we have also summarized several constants used throughout our work in Table 5.1.

5.3 Background

We will begin the present chapter by providing a brief review of sparse representations, matching pursuit, and the iteration-tuned dictionary. The intent is to make the chapter self contained and keep references to the more relevant equations at hand. See, respectively, Section 2.2, Section 2.2.1.1 and Chapter 4 for a more detailed discussion of these three topics.

5.3.1 Sparse representations using overcomplete dictionaries

Let $\mathbf{D} \in \mathbb{R}^{d \times N}$ (with $N > d$) be the full-rank dictionary matrix $\mathbf{D} = (\mathbf{d}_a)_{a=1}^{N}$ formed by N columns $\mathbf{d}_a$ called *atoms* (throughout this paper we assume that all atoms are unit norm). A sparse representation $\mathbf{x}$ of a signal vector $\mathbf{y} \in \mathbb{R}^d$ is obtained under joint fidelity and sparsity criteria:

$$\underset{\mathbf{x} \in \mathbb{R}^N}{\operatorname{argmin}} |\mathbf{y} - \mathbf{D}\mathbf{x}| \text{ s.t. } |\mathbf{x}|_0 \leq L, \tag{5.1}$$

where the l-0 norm $|\cdot|_0$ counts the number of non-zero coefficients in a vector.

When $\mathbf{D}$ is assumed to be full-rank, the minimum of (5.1) will occur at the boundary $|\mathbf{x}|_0 = L$ (assuming $L \leq d$) with probability one. Hence (5.1) consists of selecting the L atoms $\mathbf{d}^1, \ldots, \mathbf{d}^L \in \mathbf{D}$ that produce the best approximation of $\mathbf{y}$.[1] We can thus re-write (5.1) in the following equivalent form that is more similar to expressions we cover in subsequent sections:

$$\underset{\substack{\mathbf{d}^1, \ldots, \mathbf{d}^L \in \mathbf{D}; \\ \boldsymbol{\Gamma} \in \mathbb{R}^L}}{\operatorname{argmin}} \left| \mathbf{y} - \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^L \end{bmatrix} \boldsymbol{\Gamma} \right|. \tag{5.2}$$

5.3.2 The matching pursuit family

In practice, the solution to (5.1) is approximated using a greedy pursuit algorithm such as those of the *Matching Pursuit Family* (MPF). For example, the matching pursuit (MP) [Mallat 1993] algorithm proceeds by selecting one atom / coefficient pair at each iteration i. Let $\mathbf{r}^{i-1}$ denote the residue at the output of the previous iteration (with $\mathbf{r}^0 = \mathbf{y}$) and $(\cdot)^\mathsf{T}$ denote the matrix or vector transpose. The MP atom and coefficient selection rules at iteration i are given by

$$\mathbf{d}^i = \underset{\mathbf{d} \in \mathbf{D}}{\operatorname{argmax}} |\mathbf{d}^\mathsf{T} \cdot \mathbf{r}^{i-1}|, \tag{5.3a}$$

$$\gamma_i = (\mathbf{d}^i)^\mathsf{T} \cdot \mathbf{r}^{i-1}, \tag{5.3b}$$

with the iteration's output residue given by

$$\mathbf{r}^i = \mathbf{r}^{i-1} - \gamma_i \mathbf{d}^i. \tag{5.4}$$

The *Orthogonal Matching Pursuit* (OMP) [Pati 1993] variant uses the same atom selection rule but instead optimally updates all the coefficients

$$\boldsymbol{\Gamma}^i = \begin{bmatrix} \gamma_1 & \ldots & \gamma_i \end{bmatrix}^\mathsf{T} \tag{5.5}$$

[1] We use the notation $\mathbf{d} \in \mathbf{D}$ to denote that vector $\mathbf{d}$ is one of the columns of matrix $\mathbf{D}$.

produced up to the i-th iteration. By first defining the *selected-atoms matrix*

$$\mathbf{S}^i = \begin{bmatrix} \mathbf{d}^1 & \dots & \mathbf{d}^i \end{bmatrix}, \tag{5.6}$$

the OMP coefficients at iteration i can be expressed as

$$\mathbf{\Gamma}^i = (\mathbf{S}^i)^+ \mathbf{y}, \tag{5.7}$$

where superscript $+$ denotes the pseudo-inverse operation. Yet a third MPF variant known as *Optimized Orthogonal Matching Pursuit* (OOMP) [Rebollo-Neira 2002] further considers the above coefficient computation rule when choosing each atom.

Regardless of the MPF variant used, the iterative procedure is carried out either for a number of iterations L which is either pre-determined or selected on-line to satisfy an RMSE fidelity criterion ε using

$$L = \underset{i \in \{1,2,\dots\}}{\operatorname{argmin}} \, i \text{ s.t. } |\mathbf{r}^i|^2 \leq d \cdot \varepsilon^2. \tag{5.8}$$

5.3.3 The iteration-tuned dictionary

An iteration-tuned dictionary $\{\mathbf{D}_k\}_{k=1}^C$ presented in Chapter 4 is a structured dictionary consisting of a sequence of C layers, with the k-th layer containing a single dictionary matrix $\mathbf{D}_k \in \mathbb{R}^{d \times N}$. ITD-based signal decompositions force the L selected atoms to come one from each of the first L ITD layers:

$$\underset{\substack{\mathbf{d}^l \in \mathbf{D}_l, l=1,\dots,L; \\ \mathbf{\Gamma} \in \mathbb{R}^L}}{\operatorname{argmin}} \left| \mathbf{y} - \begin{bmatrix} \mathbf{d}^1 & \dots & \mathbf{d}^L \end{bmatrix} \mathbf{\Gamma} \right|. \tag{5.9}$$

The ITD decomposition expression above can be seen as a constrained version of the sparse decomposition problem in (5.2): While in (5.2) the selected atoms $\mathbf{d}^1, \dots, \mathbf{d}^L$ are chosen without restriction from all the atoms available (in matrix $\mathbf{D}$), in (5.9), each $\mathbf{d}^l, l = 1, \dots, L$ is forced to come from the subset of the available atoms found in $\mathbf{D}_l$. Hence the ITD constrain means that a diminished sparsity versus approximation error tradeoff is attainable compared to an approach that allows $\mathbf{d}^l \in \begin{bmatrix} \mathbf{D}_1 & | & \dots & | & \mathbf{D}_L \end{bmatrix}, l = 1, \dots L$. Yet the constrain also results in an important reduction in decomposition complexity that makes it possible to employ large overcompleteness factors with tractable complexity. Furthermore, the ITD layers are learned (see Chapter 4) so as to mitigate losses resulting from the ITD constrain.

Similarly to (5.2), the ITD decomposition problem in (5.9) is NP-hard and is hence solved using the matching pursuit algorithm in (5.3). The ITD constrain implies that the atom selection in (5.3a) is carried out over the i-th layer $\mathbf{D}_i$.

5.4 Tree-structured iteration-tuned dictionary (TSITD)

In this section we will consider a generalization of the ITD scheme of Chapter 4 that will give us access to even larger overcompleteness factors with the same decmoposition complexity. The approach consists of organizing the ITD layers into a tree-structure; we refer to the resulting setup as the *Tree-Structured Iteration Tuned Dictionary* (TSITD). We will begin the current section by presenting a generalization of the ITD scheme wherein each layer is allowed to contain several *candidate* dictionary matrices. The TSITD scheme follows by letting the candidate dictionary selection consist of using assigning a unique matrix from layer l to the atom chosen from the previous layer $l-1$.

5.4.1 A more general ITD framework

Like the ITD scheme, we consider a structure consisting of C layers, yet we let each layer contain more than one dictionary matrix $\mathbf{D}_k \in \mathbb{R}^{d\times N}$. The dictionary matrices of the l-th layer are called the *candidate dictionaries* of that layer. We let $\{\mathbf{D}_k\}_{k=1}^K$ denote the set of all candidate dictionaries found in all layers $l = 1, \ldots, C$. The dictionaries of the l-th layer are those with indices $k \in \mathcal{L}_l$,

$$\{\mathbf{D}_k\}_{k\in\mathcal{L}_l}, \tag{5.10}$$

where the subsets $\mathcal{L}_1, \ldots, \mathcal{L}_C$ define a partition of $\{1, \ldots, K\}$:

$$\bigcup_{i=1}^{C} \mathcal{L}_l = \{1, \ldots, K\};\ \forall l \neq j, \mathcal{L}_l \cap \mathcal{L}_j = \emptyset. \tag{5.11}$$

5.4.2 The TSITD candidate selection law

The underlying observation motivating the TSITD candidate selection law is that similar signals (*i.e.,* those that result in the same selected atom given some overcomplete dictionary) tend to produce residues that are also similar. As an example we consider the case of images, where similar edges at a given orientation are likely to result in residues displaying ringing parallel to the edge. Hence, the atom selection process can be seen as a residue classification step since the residues at the output of a given atom are likely to share a common structure and be thus well represented by a common dictionary. Thus the proposed TSITD candidate selection law consists of selecting, from a given layer $l \geq 2$, a dictionary $\mathbf{D}_k$ that is uniquely associated to the atom $\mathbf{d}^{l-1}$ chosen from the previous layer $l-1$ (by

convention, the root dictionary $\mathbf{D}_1$ is chosen for $l = 1$):

$$\mathbf{D}_k = child(\mathbf{d}^{l-1}). \tag{5.12}$$

We illustrate the tree-structure induced by the above TSITD candidate selection law on the top of Fig. 5.3. Each node with index $k \in \mathcal{L}_l$ is found in the l-th tree layer and contains a single candidate dictionary $\mathbf{D}_k$. Node k gives rise to N children nodes, one per atom of $\mathbf{D}_k$. In the figure we have illustrated a sequence $k_1, \ldots, k_l$ of descendant node indices satisfying (5.12) such that $k_1 = 1$ is the root node and, for $j \geq 2$, $\mathbf{D}_{k_l} = child(\mathbf{d}^{l-1})$, with $k_j \in \mathcal{L}_j$ and $\mathbf{d}^j \in \mathbf{D}_{k_j}$. In subsequent discussions we will use this same notation k_j / $\mathbf{D}_{k_j}$ / $\mathbf{d}^j$ to denote the j-th element of a sequence of descendant nodes / dictionaries / atoms.

5.4.3 Signal decomposition using TSITD

The problem of obtaining a sparse decomposition of a given signal using the TSITD candidate selection law can be formulated as follows, where $k_1 = 1$, $\mathbf{D}_1$ is the root dictionary and, for $l \geq 2$, $\mathbf{D}_{k_l}$ is the child dictionary of $\mathbf{d}^{l-1}$:

$$\min_{\substack{\mathbf{d}^l \in \mathbf{D}_{k_l}, l=1,\ldots,L; \\ \mathbf{\Gamma} \in \mathbb{R}^L}} \left| \mathbf{y} - \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^L \end{bmatrix} \mathbf{\Gamma} \right|. \tag{5.13}$$

Note that the TSITD decomposition expression above can be seen as a constrained version of (5.9): in (5.9), any combination of atoms is allowed as long as one is taken from each layer, whereas in the TSITD expression above only atom combinations corresponding to TSITD tree paths are allowed. Adequate training of the TSITD candidate dictionaries, combined with the large overcompleteness of the TSITD structure, will result in TSITD schemes that overcome the sub-optimality related to the TSITD atom selection constrain. Note, however, that this added TSITD overcompleteness comes at no extra decomposition complexity when the TSITD and ITD candidate dictionaries all have the same number of atoms N.

As was the case for ITDs the exact TSITD decomposition expression is difficult to solve and hence we will use the MP algorithm in (5.3). The TSITD atom selection constrain means that the atom $\mathbf{d}^i$ chosen in the i-th iteration using (5.3a) will be forced to come from the child dictionary $\mathbf{D}_{k_i} = child(\mathbf{d}^{i-1})$ (or the root dictionary $\mathbf{D}_1$ for $i = 1$):

$$\mathbf{d}^i = \underset{\mathbf{d} \in \mathbf{D}_{k_i}}{\operatorname{argmax}} |\mathbf{d}^\mathsf{T} \cdot \mathbf{r}^{i-1}|, \tag{5.14a}$$

$$\gamma_i = (\mathbf{d}^i)^\mathsf{T} \cdot \mathbf{r}^{i-1}, \tag{5.14b}$$

5.4.4 TSITD training

Given some representative finite training set $\{\mathbf{y}_t\}_{t=1}^T$, the construction of the set of TSITD candidate dictionaries can be expressed optimally as the minimization of the cumulative representation error of the training vectors $\mathbf{y}$. Using the TSITD sparse decomposition formulation in (5.13), we can express the optimal ITD construction formulation as follows:

$$\underset{\substack{\mathbf{D}_k\in\mathcal{N},\\ k=1,\ldots,K}}{\operatorname{argmin}} \left(\sum_{t=1}^{T} \min_{\substack{\mathbf{d}^l\in\mathbf{D}_{k_l},\\ l=1,\ldots,L_t;\\ \boldsymbol{\Gamma}\in\mathbb{R}^{L_t}}} \left| \mathbf{y}_t - \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^{L_t} \end{bmatrix} \boldsymbol{\Gamma} \right|^2 \right). \tag{5.15}$$

where the optimization domain of each candidate dictionary is the subset

$$\mathcal{N} = \{\mathbf{D} \in \mathbb{R}^{d\times N} : \forall \mathbf{d} \in \mathbf{D}, \mathbf{d}^\mathsf{T}\mathbf{d} = 1\} \tag{5.16}$$

consisting of matrices with unit norm columns.

5.4.4.1 Top-down training

To simplify the above problem, we instead consider a top-down training procedure wherein a given dictionary $\mathbf{D}_k$, $k \in \mathcal{L}_l$ is trained after all its ancestors $\mathbf{D}_{k_1}, \ldots, \mathbf{D}_{k_{l-1}}$, $k_j \in \mathcal{L}_j$. With this assumption, the above TSITD training problem becomes

$$\underset{\mathbf{D}_k\in\mathcal{N}}{\operatorname{argmin}} \left(\sum_{t\in\mathcal{T}_k} \min_{\substack{\mathbf{d}\in\mathbf{D}_k\\ \boldsymbol{\Gamma}\in\mathbb{R}^{L_t}}} \left| \mathbf{y}_t - \begin{bmatrix} \mathbf{S}_t^{l-1} & | & \mathbf{d} \end{bmatrix} \boldsymbol{\Gamma} \right|^2 \right), \tag{5.17}$$

where $\mathbf{S}_t^{l-1} = (\mathbf{d}_t^j)_{j=1}^{l-1}$, $\mathbf{d}_t^j \in \mathbf{D}_{k_j} \forall t$, contains the atoms $\mathbf{d}_t^j$ selected for $\mathbf{y}_t$ from the ancestor dictionaries $\mathbf{D}_{k_j}$. Note that to obtain $\mathbf{S}_t^{l-1}$ it is necessary to carry out a TSITD-MP decomposition of the signals $\mathbf{y}_t$ across the nodes $k_1, \ldots, k_{l-1}$. When using the RMSE based MP stopping criterion in (5.8), it is possible that some signals $\mathbf{y}_t$ require $L_t \leq l-1$ atoms; such signals need to be excluded from the training set of dictionary $\mathbf{D}_k$ (found in layer l). This explains the summation range $t \in \mathcal{T}_k$, defined as follows (recall that $k = 1$ is the index of the root dictionary):[2]

$$\mathcal{T}_1 = \{t = 1, \ldots, T : |\mathbf{y}_t|^2 \geq d\kappa^2\}, \text{ and} \tag{5.18a}$$

$$\mathcal{T}_k = \{t \in \mathcal{T}_{k_{l-1}} : |\mathbf{r}_t^{l-1}|^2 \geq d\kappa^2, \mathbf{d}_t^{l-1} = \mathbf{p}_k\}, k > 1. \tag{5.18b}$$

[2]We use the symbol κ (instead of ε) to denote the RMSE stopping criterion used during training.

where $\mathbf{p}_k \in \mathbf{D}_{k_{l-1}}$ refers to the atom chosen to be the parent of node k. Note that the top-down training approach means that all the ancestor dictionaries of $\mathbf{D}_k$ are trained prior to $\mathbf{D}_k$. Hence, it is possible to determine the $(l-1)$-th selected atom $\mathbf{d}_t^{l-1}$ of each training vector $\mathbf{y}_t$ before training $\mathbf{D}_k$.

5.4.4.2 MP-based top-down training

We consider now a second simplification of the TSITD training procedure. Inspection of (5.17) reveals that the inner minimization corresponds to an OOMP atom selection step. One can likewise use a simplified MP atom selection step wherein only the l-th coefficient of $\mathbf{\Gamma}$ is updated; all other $l-1$ coefficients are fixed to the values computed when training ancestor dictionaries $\mathbf{D}_{k_1}, \ldots, \mathbf{D}_{k_{l-1}}$. Let $\mathbf{\Gamma}_t^{l-1}$ denote these fixed coefficients computed from training vector $\mathbf{y}_t$. The term inside the absolute value signs in (5.17) can then be written as $\mathbf{y}_t - \mathbf{S}_t^{l-1}\mathbf{\Gamma}_t^{l-1} - \gamma\mathbf{d} = \mathbf{r}_t^{l-1} - \gamma\mathbf{d}$. Accordingly, we can write the simplified version of (5.17) as follows:

$$\underset{\mathbf{D}_k \in \mathcal{N}}{\operatorname{argmin}} \left(\sum_{t \in \mathcal{T}_k} \min_{\substack{\mathbf{d} \in \mathbf{D}_k \\ \gamma \in \mathbb{R}}} \left| \mathbf{r}_t^{l-1} - \gamma \mathbf{d} \right|^2 \right). \tag{5.19}$$

The above expression can be solved using, *eg.,* the K-SVD algorithm using an $L = 1$ sparsity constrain. The procedure can be described as a K-means under the projection distance and consists of iteratively (*i*) fixing $\mathbf{D}_k$ and classifying the training residues according to which atom $\mathbf{d} \in \mathbf{D}_k$ is closest (under the angular distances) and (*ii*) updating each $\mathbf{d}$ so that it minimizes the cumulative representation error $|\mathbf{r}_t^{l-1} - \gamma\mathbf{d}|^2$ of those training residues $\mathbf{r}_t^{l-1}$ assigned to $\mathbf{d}$.

For completeness, we outline the resulting TSITD candidate training procedure in Fig. 5.1. The TSITD training algorithm based on the top-down training approach and (5.19) is outlined in Fig. 5.2.

5.4.5 Orthogonality of selected-atoms matrices and consequences

In (5.6) we defined the MP *selected-atoms matrix* of the i-th iteration as having, in the j-th of i columns, the j-th selected atom $\mathbf{d}^j$. Given the TSITD candidate selection rule, the TSITD selected-atoms matrices will contain a sequence of descendant atoms $\mathbf{d}^1, \ldots, \mathbf{d}^i$ so that, given a sequence of descendant nodes $k_1, \ldots, k_i$, $\mathbf{d}^j \in \mathbf{D}_{k_j}$. Note that any TSITD node with index $k \in \mathcal{L}_l$ will have a unique selected-atoms matrix $\mathbf{S}^{l-1}$ consisting of all the ancestor atoms of that node.

1: **Definition**: $\mathbf{D}_k = \textsc{Candi}(\{\mathbf{r}_t^{l-1}\}_{t\in\mathcal{T}_k}, N)$

2: **Initialization**:
3: $\mathcal{R} \leftarrow \textsc{RandSelect}(N, |\mathcal{T}_k|)$
4: $(\mathbf{d}_{ka})_{a=1}^N \leftarrow (\mathbf{y}_t)_{t\in\mathcal{R}}$

5: **Algorithm**:
6: **repeat**
7: $\quad\mathcal{Q}_a \leftarrow \emptyset, a = 1, \ldots, N.$
8: $\quad$**for** $t \in \mathcal{T}_k$ **do**
9: $\quad\quad a \leftarrow \text{argmax}_{\alpha\in\{1,\ldots,N\}} |\mathbf{d}_{k\alpha}^{\mathsf{T}} \mathbf{r}_t^{l-1}|$
10: $\quad\quad\mathcal{Q}_a \leftarrow \mathcal{Q}_a \bigcup t$
11: $\quad$**end for**
12: $\quad$**for** $a = 1$ **to** N **do**
13: $\quad\quad(\mathbf{u}_{aj})_j \leftarrow \text{LSV}\big((\mathbf{r}_t^{l-1})_{t\in\mathcal{Q}_a}\big)$
14: $\quad\quad\mathbf{d}_{ka} \leftarrow \mathbf{u}_{a1}$
15: $\quad$**end for**
16: **until** convergence of $\mathbf{D}$

Figure 5.1: The candidate construction algorithm $\mathbf{D}_k = \textsc{Candi}(\{\mathbf{r}_t^{l-1}\}_{t\in\mathcal{T}_k}, N)$ for a node $k \in \mathcal{L}_l$. The function $\textsc{RandSelect}(N, T)$ selects N integers in $[1, T]$ without repetition. Function LSV in line 13 computes the left singular vectors, assumed ordered by decreasing singular value.

One interesting TSITD property is that the tree-structuring of TSITD candidates, combined with the top-down ITD construction approach of (5.19), results in TSITDs with selected-atoms matrices $\mathbf{S}^l = \begin{bmatrix}\mathbf{d}^1 & \ldots & \mathbf{d}^l\end{bmatrix}$ that are orthogonal. We summarize this result in the following theorem:

Theorem 5.4.1 (Orthogonality of TSITD selected-atoms matrices $\mathbf{S}^l$)**.** *Assume that a TSITD structure is trained using the top-down training approach outlined in Fig. 5.2. Let $k_1, \ldots, k_l$ denote a sequence of TSITD descendant node indices, $\mathbf{D}_{k_1}, \ldots \mathbf{D}_{k_l}$ denote the candidate dictionaries in those nodes, and $\mathbf{d}^1, \ldots, \mathbf{d}^l$ be any sequence of atoms such that $\mathbf{d}^j \in \mathbf{D}_{k_j}, j = 1, \ldots, l$. All possible selected-atoms matrices $\mathbf{S}^l = \begin{bmatrix}\mathbf{d}^1 & \ldots & \mathbf{d}^l\end{bmatrix}$ will satisfy*

$$(\mathbf{S}^l)^{\mathsf{T}} \mathbf{S}^l = \mathbf{I}. \tag{5.20}$$

Proof. The proof is based on three observations. The first observation is that the residual training set of a candidate $\mathbf{D}_{k_j}$ is contained in the orthogonal complement of the parent atom:

$$\text{span}(\{\mathbf{r}_t^{j-1}\}_{t\in\mathcal{T}_{k_j}}) \subseteq \text{span}^{\perp}(\mathbf{d}^{j-1}). \tag{5.21}$$

1: **Definition**: $k_{out} = \text{SUBTREE}(k, l, \{\mathbf{r}_t^{l-1}\}_{t\in\mathcal{T}_k})$ ($k \in \mathcal{L}_l$, k_{out} is the index of the last node in the subtree of k)
2: **Global parameters**:

1. N: nominal number of atoms per candidate
2. M: minimum number of training vectors per atom
3. C: maximum layer index
4. κ: minimum residual RMSE

3: **Algorithm**:
4: $N_k = \min(N, \left\lfloor \frac{|\mathcal{T}_k|}{M} \right\rfloor)$
5: **if not** ($l > C$ **or** $N_k = 0$) **then**
6: $\quad \mathbf{D}_k = \text{CANDI}(\{\mathbf{r}_t^{l-1}\}_{t\in\mathcal{T}_k}, N_k)$
7: $\quad$ **for** $t \in \mathcal{T}_k$ **do**
8: $\quad\quad \mathbf{d}_t^l = \text{argmax}_{\mathbf{d}\in\mathbf{D}_k} |\mathbf{d}^T \mathbf{r}_t^{l-1}|$
9: $\quad\quad \mathbf{r}_t^l = \mathbf{r}_t^{l-1} - (\mathbf{d}_t^{lT} \mathbf{r}_t^{l-1})\mathbf{d}_t^l$
10: $\quad$ **end for**
11: $\quad p = k$
12: $\quad$ **for** $a = 1, \ldots, N$ **do**
13: $\quad\quad k \leftarrow k + 1$
14: $\quad\quad \mathbf{h}[k] = p$
15: $\quad\quad \mathcal{T}_k = \{t \in \mathcal{T}_p : |\mathbf{r}_t^l|^2 \geq d\kappa^2, \mathbf{d}_t^l = \mathbf{d}_{ka}\}$
16: $\quad\quad k \leftarrow \text{SUBTREE}(k, l+1, \{\mathbf{r}_t^l\}_{t\in\mathcal{T}_k})$
17: $\quad$ **end for**
18: **end if**

Figure 5.2: Function that constructs a subtree of the TSITD structure recursively. The entire TSITD tree is constructed using inputs $k = 1$ and training set $\{\mathbf{y}_t\}_{t\in\mathcal{T}_1}$. Function CANDI() in line 6 is implemented by the algorithm in Fig. 5.1. Node ancestry is recorded in array $\mathbf{h}$, with $\mathbf{h}[k]$ producing the index of the parent node of node k. Here N denotes a nominal number of atoms, the actual number of atoms N_k per candidate $\mathbf{D}_k$ will depend on the availability of training vectors.

This is a well known consequence of the MP rules in (5.3).

The second observation is that the span of $\mathbf{D}_{k_j}$ is contained in the span of its training set:

$$\text{span}(\mathbf{D}_{k_j}) \subseteq \text{span}(\{\mathbf{r}_t^{j-1}\}_{t\in\mathcal{T}_{k_j}}) \tag{5.22}$$

This relation follows from the SVD-based atom computation (*cf.* Fig. 5.1), as only singular vectors corresponding to zero-valued singular values fall outside the span of the data set on which the SVD is computed.

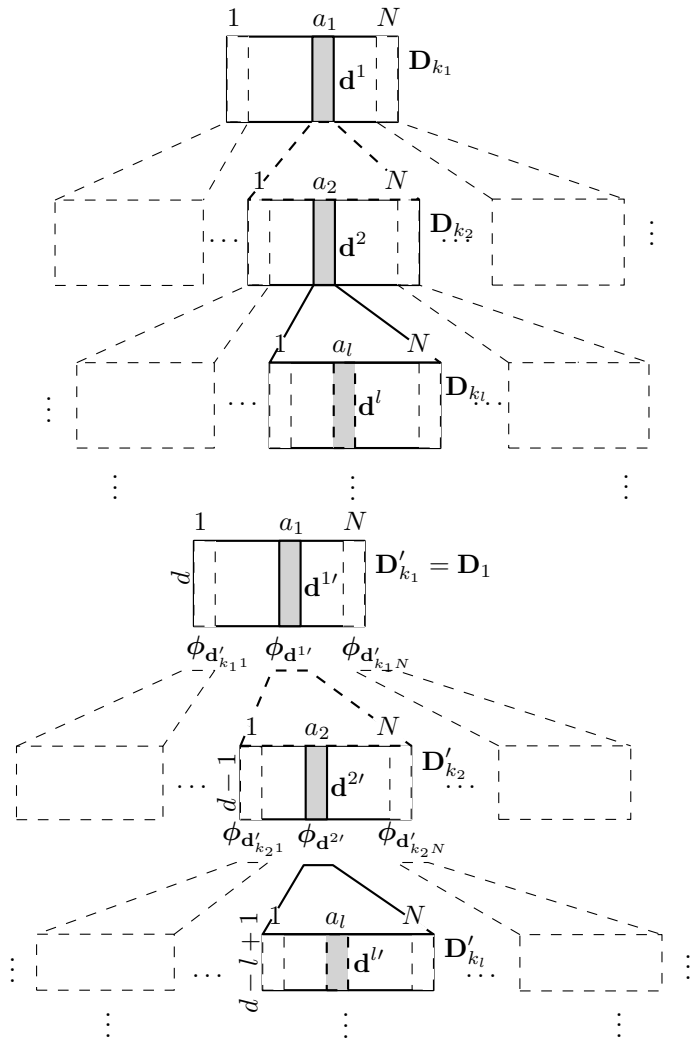

Figure 5.3: *Top:* The TSITD tree-structure. The root node $k_1 = 1$ is indicated, as well as two descendant nodes k_2, k_3; atoms $\mathbf{d}^j \in \mathbf{D}_{k_j}, j = 1, 2, 3$ denote a corresponding sequence of descendant atoms.
Bottom: The equivalent rTSITD representation. The path reduction matrices $\mathbf{\Phi}_i$ (not illustrated) of a given $\mathbf{D}'_{k_i}$ can be obtained by cumulatively left-multiplying the $\phi_{\mathbf{d}^{j\prime}}, j = i-1, \ldots, 1$, backwards along the path $\mathcal{A}_i$.

The third observation is that the training sets of $\mathbf{D}_{k_1}, \ldots, \mathbf{D}_{k_l}$ span nested subspaces:

$$\operatorname{span}(\{\mathbf{r}_t^j\}_{t \in \mathcal{T}_{k_{j+1}}}) \subseteq \operatorname{span}(\{\mathbf{r}_t^{j-1}\}_{t \in \mathcal{T}_{k_j}}). \tag{5.23}$$

This follows since

$$\mathbf{r}_t^j = \mathbf{r}_t^{j-1} - \left((\mathbf{r}_t^{j-1})^\mathsf{T} \mathbf{d}^j\right) \mathbf{d}^j \tag{5.24}$$

and, for $t \in \mathcal{T}_j$, both $\mathbf{r}_t^{j-1}$ and (from (5.22)) $\mathbf{d}^j \in \mathbf{D}_{k_j}$ are contained in $\operatorname{span}(\{\mathbf{r}_t^{j-1}\}_{t \in \mathcal{T}_{k_j}})$,

thus establishing (5.23).

The theorem follows since, from (5.22) and (5.23), each candidate $\mathbf{D}_{k_j}$ is contained inside a training subspace that is nested inside the training subspace containing the parent candidate $\mathbf{D}_{k_{j-1}}$. Hence, $\mathbf{D}_{k_j}$ retains orthogonality to all ancestor atoms $\mathbf{d}^{j-1}, \mathbf{d}^{j-2}, \ldots, \mathbf{d}^1$ since, from (5.21), these are orthogonal to training subspaces that encompass $\mathbf{D}_{k_j}$. The previous discussion establishes that any given atom is orthogonal to all its ancestors, and this can only be the case if all $\mathbf{S}^l$ is orthogonal, thus establishing the theorem.

□

An important consequence of the orthogonality of TSITD selected atom matrices is that all MP schemes (*i.e.,* MP, OMP, and OOMP) will produce the same result. This follows from the fact that, for orthogonal $\mathbf{S}^l$, the pseudo-inverse $(\mathbf{S}^l)^+$ in (5.7) will be given by the transpose $(\mathbf{S}^l)^\mathsf{T}$. Hence the large redundancy of the TSITD tree structure can be exploited using the more powerful OOMP algorithm while incurring the lower complexity of the MP algorithm.

An important consequence of Theorem 5.4.1 concerns the maximum number of layers (denoted C) of the TSITD structure:

Corollary 5.4.1 (Maximum number of layers C)**.** *Any TSITD satisfying* (5.20) *will have $C \leq d$ layers, where d is the dimensionality of the input signal space, $\mathbf{y} \in \mathbb{R}^d$.*

This corollary follows from the observation that, given an orthogonal $\mathbf{S}^d$ (of size $d \times d$), the approximation $\hat{\mathbf{y}}^d = \mathbf{S}^d\mathbf{\Gamma}^d$ at the output of the d-th layer will be exact. Hence any remaining layers after the d-th will serve no useful purpose. Note that one can have $C < d$ depending on the manner in which the sparsities of L_t of the training vectors are selected (*eg.,* when using fixed $L_t < d$ for all $\mathbf{y}_t$).

In Fig. 5.4 we present an example of TSITD candidate dictionaries along three different branches of the tree-structure. Note that the three different branches correspond to three different atoms of the root dictionary and that each branch inherits structural characteristics from its parent atom: The three chosen root atoms contain smooth edges along different directions (diagonal, anti-diagonal and vertical) and their children dictionaries respectivly contain atoms mostly displaying ringing parallel to the parent's edge. This inherited structure is more evident in the second layer but is still observable in the third layer. This characteristic is expected given the candidate selection law used for TSITD.

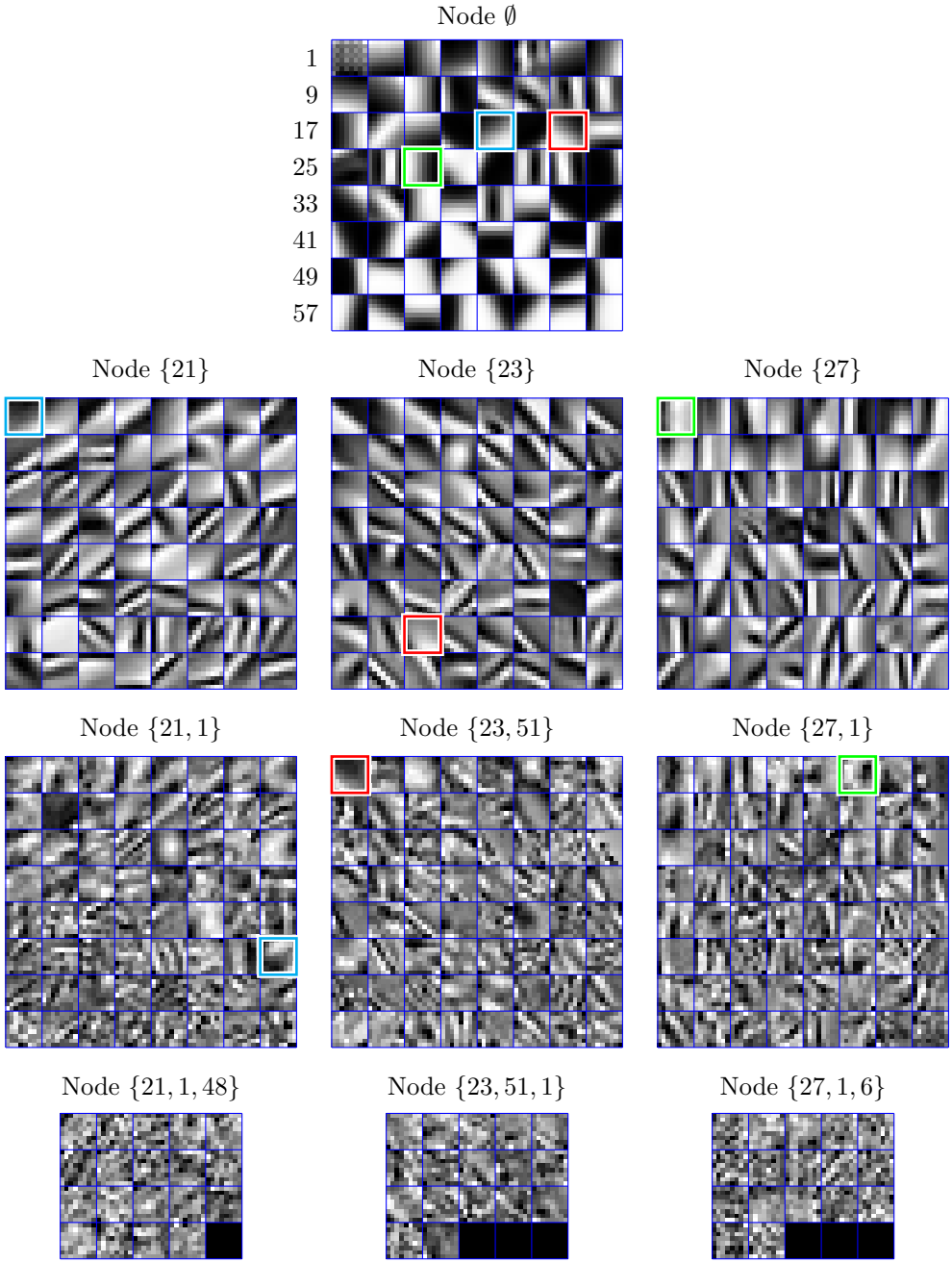

Figure 5.4: An example of TSITD candidate dictionaries along three different branches of the tree-structure. Each candidate $\mathbf{D}_k$, $k \in \mathcal{L}_l$ is labeled using the indices $\{a_1, \ldots, a_{l-1}\}$ of its ancestor atoms $\mathbf{d}^j = \mathbf{d}_{k_j a_j}$, $k_j \in \mathcal{L}_j$. Atom-indices increase along the rows, as indicated for the root dictionary . The atoms selected from each dictionary are indicated by a heavy frame. All candidate dictionaries consist of a nominal number of 64 atoms.

5.5 Reduced Tree-Structured Iteration-Tuned Dictionary (rTSITD)

The TSITD structure presented in the previous section exposes a very rich redundancy with a computational complexity that is nonetheless comparable or even lower than that of fixed-dictionary representations using MP. The scheme nonetheless suffers from a storage footprint that can be prohibitively large. As we will see, the *reduced Tree-Structured Iteration-Tuned Dictionary* (rTSITD) that we now present takes advantage of the orthogonality of the selected-atoms matrices (*cf.* Theorem 5.4.1) to address this shortcoming.

We begin the present section by showing how Theorem 5.4.1 implies that we can represent each TSITD candidate dictionary in a space of reduced dimensionality. An rTSITD will hence make use of these *reduced candidate dictionaries* to obtain a more compact yet equivalent representation of TSITD. Under the right configurations, we will see that rTSITD enjoys a slightly smaller storage footprint and reduced training and decomposition complexity relative to TSITD.

5.5.1 Reduced candidate dictionaries and their path reduction matrices

Consider a TSITD candidate dictionary $\mathbf{D}_k, k \in \mathcal{L}_l$ having a sequence of ancestor atoms $\mathbf{d}^j \in \mathbf{D}_{k_j}, j = 1, \ldots, l-1$, so that $k_j \in \mathcal{L}_j$. All selected-atoms matrices using atoms $\mathbf{d} \in \mathbf{D}_k$ will have the form $\mathbf{S}^l = \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^{l-1} & \mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{S}^{l-1} & | & \mathbf{d}^l \end{bmatrix}$. From Theorem 5.4.1, we know that all atoms $\mathbf{d} \in \mathbf{D}_{k_l}$ must be orthogonal to all their (shared) ancestor atoms in $\mathbf{S}^{i-1}$. Hence we can write the following consequence of Theorem 5.4.1:

Corollary 5.5.1 (Reduced candidate dictionaries)**.** *Assume the TSITD candidate* $\mathbf{D}_k$ *is found in the* l*-th layer and has ancestor atoms* $\mathbf{S}^{l-1} = \begin{bmatrix} \mathbf{d}^1 & \ldots & \mathbf{d}^{l-1} \end{bmatrix}$*. From Theorem 5.4.1, it follows that*

$$(\mathbf{S}^{l-1})^T \mathbf{D}_k = \mathbf{0}. \tag{5.25}$$

Hence each $\mathbf{D}_k$ *is contained in a subspace of dimensionality* $d - l + 1$ *and thus, given an adequate rotation matrix, one can obtain an equivalent reduced form*

$$\mathbf{D}'_k \in \mathbb{R}^{(d-l+1)\times N}. \tag{5.26}$$

To obtain the reduced representation $\mathbf{D}'_k$ promised by Corollary 5.5.1 for some candidate $\mathbf{D}_k$ of the l-th layer, we consider some arbitrary orthonormal basis

$\mathbf{\Phi}_k \in \mathbb{R}^{d\times(d-l+1)}$ (chosen by convention) of the left null-space of the ancestor atoms $\mathbf{S}^{l-1}$ of $\mathbf{D}_k$,

$$(\mathbf{\Phi}_k)^\mathsf{T}\,\mathbf{\Phi}_k = \mathbf{I}, \tag{5.27a}$$
$$(\mathbf{\Phi}_k)^\mathsf{T}\,\mathbf{S}^{l-1} = \mathbf{0}, \tag{5.27b}$$

where, by convention, we let $\mathbf{\Phi}_1 = \mathbf{I}$ (recall that $k = 1$ denotes the root node). The matrix $\mathbf{\Phi}_k$ is the required rotation matrix stipulated by Corollary 5.5.1 and thus the desired reduced representation of $\mathbf{D}_k$ follows as

$$\mathbf{D}'_k = (\mathbf{\Phi}_k)^\mathsf{T}\,\mathbf{D}_k. \tag{5.28}$$

We refer to $\mathbf{\Phi}_k$ as the *path reduction matrix* since it defines the mapping between the original signal space of the root node and the reduced space containing $\text{span}(\mathbf{D}_k)$.

5.5.2 Branch reduction matrices

The transpose of the path reduction matrix $\mathbf{\Phi}_k$ of $\mathbf{D}_k$, $k \in \mathcal{L}_l$, reduces the dimensionality of the signal-space atoms $\mathbf{d} \in \mathbf{D}_k$ from d to $d-l+1$ in a single operation. We will find it useful to instead carry out this reduction of dimensionality one dimension at a time. Later we will see that doing so will allow rTSITD structures to enjoy a reduced storage footprint and decomposition complexity relative to TSITD.

To this end, we consider the reduced representation $\mathbf{d}' \in \mathbf{D}'_k$ of an atom from node $k \in \mathcal{L}_l$. The *branch reduction matrix* $\boldsymbol{\phi}_{\mathbf{d}'} \in \mathbb{R}^{(d-l+1)\times(d-l)}$ of $\mathbf{d}'$ is any orthogonal matrix $\boldsymbol{\phi}_{\mathbf{d}'}$ (again chosen by convention) that spans the left null-space of $\mathbf{d}'$,

$$(\boldsymbol{\phi}_{\mathbf{d}'})^\mathsf{T}\,\boldsymbol{\phi}_{\mathbf{d}'} = \mathbf{I}, \tag{5.29a}$$
$$(\boldsymbol{\phi}_{\mathbf{d}'})^\mathsf{T}\,\mathbf{d}' = \mathbf{0}. \tag{5.29b}$$

Using the branch reduction matrices thus defined one can factor the path reduction matrix $\mathbf{\Phi}_k$ of a node with index $k \in \mathcal{L}_l$ as follows:

$$\mathbf{\Phi}_k = \boldsymbol{\phi}_{\mathbf{d}^{1\prime}} \cdot \ldots \cdot \boldsymbol{\phi}_{\mathbf{d}^{(l-1)\prime}}, \tag{5.30}$$

where $\mathbf{d}^{1\prime}, \ldots, \mathbf{d}^{l-1\prime}$ are the reduced representations of the ancestor atoms $\mathbf{d}^1, \ldots, \mathbf{d}^{l-1}$ of node k. Recall that $\mathbf{\Phi}_k$ is orthogonal to all $l-1$ of its ancestor atoms in $\mathbf{S}^{l-1}$. Each $\boldsymbol{\phi}_{\mathbf{d}^{j\prime}}$ in (5.30) enforces orthogonality of $\mathbf{\Phi}_k$ to one of its ancestors $\mathbf{d}^j$.

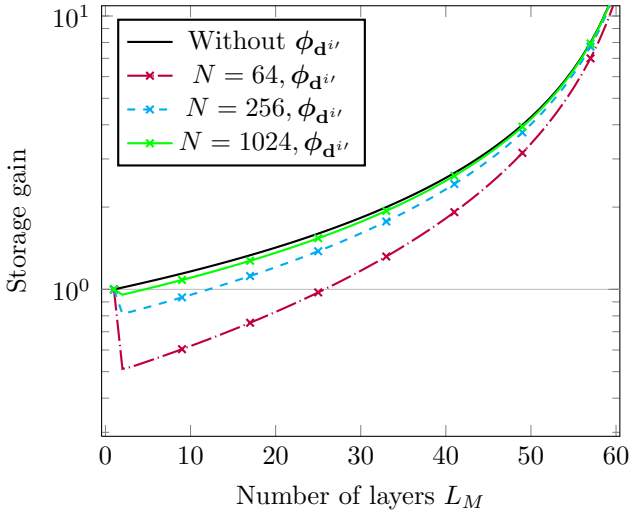

Figure 5.5: Storage gain of rTISTD over TSITD. Storage is measured in terms of the total number of matrix entries for the entire structure consisting of C layers.

5.5.3 The rTSITD tree structure

on the bottom of Fig. 5.3 we illustrate the rTSITD-based representation of the TSITD tree-structure; the original TSITD tree-structure is found on the top of the same figure. The first layer is left unchanged with $\mathbf{D}_1 = \mathbf{D}'_1$. Starting with the second layer, each TSITD candidate dictionary $\mathbf{D}_k$ has been substituted by its reduced version $\mathbf{D}'_k$ and the branch reduction matrix $\boldsymbol{\phi}_{\mathbf{d}^{(l-1)\prime}}$ of its parent atom. The signal-space version of the rTSITD tree could be reconstructed by means of the path reduction matrix $\boldsymbol{\Phi}_k$. Although these are not illustrated, they can be obtained from the the branch reduction matrices using (5.30).

To illustrate the resulting storage gain of the rTSITD structure, we consider (r)TSITDs having N atoms per candidate dictionary and count the cumulative number of matrix entries in all dictionaries (*eg.*, $(d-l+1)\cdot N$ for $\mathbf{D}'_k \in \mathbb{R}^{(d-l+1)\times N}$, $k \in \mathcal{L}_l$) for all C layers of the tree structure. For the case of rTSITD, we also need to include the parent branch reduction matrices of all dictionaries for layers $1 < i \leq C$. Due to the storage overhead represented by the branch reduction matrices, rTSITD enjoys a storage gain only for tree-structures with a sufficiently large number of layers C. In order to avoid this storage penalty for low C, it is nonetheless possible to use a hybrid TSITD / rTSITD structure consisting of signal-space (TSITD) candidates in the first few layers having nodes with indices $k \in \mathcal{L}_l, l < q$ and reduced-space (rTSITD) representations in the remaining layers $\mathcal{L}_l, l \geq q$. Path reduction matrices $\boldsymbol{\Phi}_k, \forall k \in \mathcal{L}_q$ can be used to accomplish the transition between signal-space and reduced-space layers, while the subsequent,

reduced-space layers (those with nodes indexed by $k \in \mathcal{L}_l, l > q$) of the hybrid structure would again use branch reduction matrices $\boldsymbol{\phi}_{\mathbf{d}'}, \forall \mathbf{d} \in \mathbf{D}_k$.

It is further possible to enjoy a storage gain for all C layers using only the rTSITD structure since, in fact, it is not necessary in general to store the branch reduction matrices $\boldsymbol{\phi}_{\mathbf{d}'}$: Given some algorithm (chosen by convention, *eg.,* Gram-Schmidt) that can be used to construct the same orthonormal basis given the first member of the basis, one can build the branch reduction matrices $\boldsymbol{\phi}_{\mathbf{d}'}$ on-the-fly by applying said algorithm to the corresponding reduced atom $\mathbf{d}' \in \mathbf{D}'_k$. As illustrated in Fig. 5.5, the resulting rTSITD storage gain is uniform for all tree-structure dimensions C. Note, however, that this approach suffers from an increased computational complexity since the matrices $\boldsymbol{\phi}_{\mathbf{d}'}$ need to be computed on-the-fly.

5.5.4 Signal decomposition using rTSITD

One can use the rTSITD structure illustrated on the bottom of Fig. 5.3 to implement signal decomposition and reconstruction directly in the reduced spaces (as opposed to recurring to regeneration of the signal-space TSITD structure). This will allow the rTSITD structure to enjoy a reduced storage footprint and computational complexity when considering large tree structures.

In order to demonstrate first how to decompose signals using the rTSITD structure, we recall that the MP signal decomposition rules in (5.3) rely on the residual vectors $\mathbf{r}^i$ at the output of the i-th iteration,

$$\mathbf{r}^i = \mathbf{r}^{i-1} - \gamma^i \cdot \mathbf{d}^i, \tag{5.31}$$

where $\mathbf{d}^i \in \mathbf{D}_{k_i}$ and $k_1, k_2, \ldots, L$, with $k_j \in \mathcal{L}_j$, denotes a sequence of descendant node indices such that $k_1 = 1$ is the root node and, for $i > 2$, $\mathbf{D}_{k_i} = child(\mathbf{d}^{i-1})$. The orthogonality of selected atoms (Theorem 5.4.1) implies that $\mathbf{r}^i$ computed as above is contained in the orthogonal complement of the atoms $\mathbf{S}^i$ chosen from the descendant nodes $k_1, \ldots, k_i$. This orthogonal complement space is spanned by $\boldsymbol{\Phi}_{k_{i+1}}$ (*cf.* (5.27b)), and thus we can represent $\mathbf{r}^i$ exactly in the reduced space $\mathbb{R}^{d-i}$ as follows:[3]

$$\mathbf{r}^{i\prime} \triangleq (\boldsymbol{\Phi}_{k_{i+1}})^\top \mathbf{r}^i. \tag{5.32}$$

Using (5.31) and (5.30), the above expression produces the following method for

[3]For $i = C$, where C is the index of the last TSITD layer, this expression depends on $\boldsymbol{\Phi}_{k_{C+1}}$. For consistency, we can assume the existence of a layer $C + 1$ containing empty nodes. Note, from (5.27) and the related discussion, that $\boldsymbol{\Phi}_{k_{C+1}}$ depends only on the ancestor atoms of a node, which are all found in layers $1, \ldots, C$.

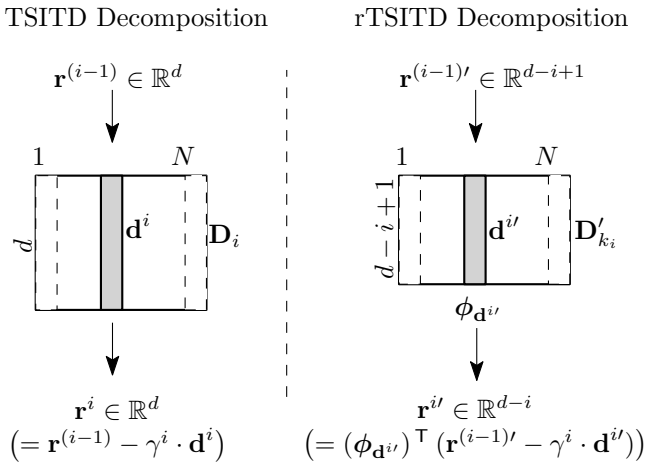

Figure 5.6: Illustration of the TSITD decomposition process at the i-th layer when using TSITD and rTSITD.

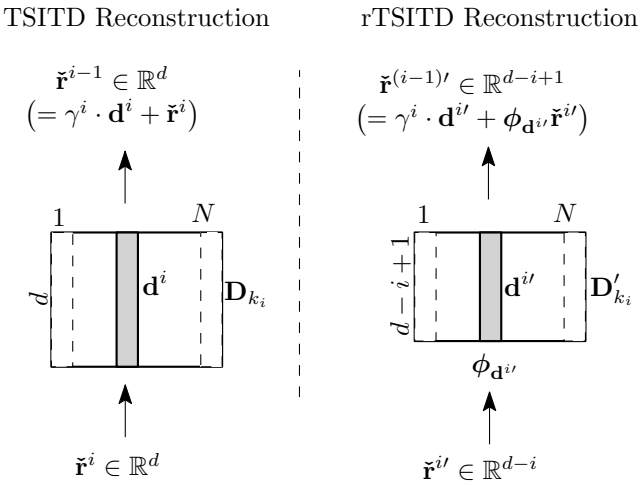

Figure 5.7: Illustration of the TSITD reconstruction process at the i-th layer when using TSITD and rTSITD.

signal decomposition using the rTSITD structure:

$$\begin{aligned}\mathbf{r}^{i\prime} &= (\phi_{\mathbf{d}^{i\prime}})^\mathsf{T} (\mathbf{\Phi}_{k_i})^\mathsf{T} \left(\mathbf{r}^{i-1} - \gamma^i \cdot \mathbf{d}^i\right) \\ &= (\phi_{\mathbf{d}^{i\prime}})^\mathsf{T} \left(\mathbf{r}^{(i-1)\prime} - \gamma^i \cdot \mathbf{d}^{i\prime}\right). \end{aligned} \tag{5.33}$$

From the orthogonality of path reduction matrices, it follows from (5.32) that

$$|\mathbf{r}^i| = |\mathbf{r}^{i\prime}|. \tag{5.34}$$

Hence, the γ_i and $\mathbf{d}^{i\prime}$ chosen to minimize $|\mathbf{r}^{i\prime}|$ will also minimize $|\mathbf{r}^i|$. The atom selection and decomposition can thus be carried out in the reduced-spaces of the

rTSITD structure by applying the MP rules directly in the corresponding rTSITD layer:

$$\mathbf{d}^{i\prime} = \underset{\mathbf{d}' \in \mathbf{D}'_{k_i}}{\operatorname{argmax}} \left| (\mathbf{d}')^{\mathsf{T}} \mathbf{r}^{(i-1)\prime} \right|, \tag{5.35a}$$

$$\gamma_i = (\mathbf{d}^{i\prime})^{\mathsf{T}} \cdot \mathbf{r}^{(i-1)\prime}. \tag{5.35b}$$

The resulting atom/coefficient pair produces an approximation $\gamma^i \cdot \mathbf{d}^{i\prime}$ of the input residual $\mathbf{r}^{(i-1)\prime}$. The dimensionality of the resulting approximation error vector is then reduced using the branch reduction matrix of the selected atom according to (5.33), thus producing the reduced residual $\mathbf{r}^{i\prime}$ at the input of the following layer. Given the property (5.34), it follows that the RMSE-based stopping criterion (5.8) can be applied directly to the reduced residual $\mathbf{r}^{i\prime}$:

$$L = \underset{i \in \{1,2,\ldots\}}{\operatorname{argmin}} \, i \text{ s.t. } |\mathbf{r}^{i\prime}|^2 \leq d \cdot \varepsilon^2, \tag{5.36}$$

We illustrate the original TSITD decomposition process and the equivalent rTSITD approach in the left-hand and right-hand sides of Fig. 5.6, respectively.

5.5.5 Signal reconstruction using rTSITD

To illustrate how the signal reconstruction process can also be carried out using the rTSITD structure, we first consider the following strategy for the equivalent operation in signal-space using TSITD (left-hand side of Fig. 5.7): We traverse the TSITD tree backwards from the last node k_L visited in the decomposition down to the root $k_1 = 1$, accumulating the contributions of each atom $\mathbf{d}^j \in \mathbf{D}_{k_j}$ and coefficient γj previously selected along the way. Let the vector $\check{\mathbf{r}}^{i-1}$ denote the result after accumulating the contributions from layer L up to the i-th layer:

$$\check{\mathbf{r}}^{i-1} \triangleq \gamma^i \cdot \mathbf{d}^i + \ldots + \gamma^L \cdot \mathbf{d}^L \tag{5.37}$$

$$= \gamma^i \cdot \mathbf{d}^i + \check{\mathbf{r}}^i, \tag{5.38}$$

where, by convention, we let $\check{\mathbf{r}}^L = \mathbf{0}$ and we note that $\check{\mathbf{r}}^0 = \hat{\mathbf{y}}^L$ produces the desired reconstruction.

The same accumulation strategy can be implemented using the rTSITD structure and we demonstrate this by considering the reduced accumulator

$$\check{\mathbf{r}}^{(i-1)\prime} \triangleq (\mathbf{\Phi}_{k_i})^{\mathsf{T}} \check{\mathbf{r}}^{i-1} \qquad (\in \mathbb{R}^{d-i+1}) \tag{5.39a}$$

$$= (\mathbf{\Phi}_{k_i})^{\mathsf{T}} \left(\gamma^i \cdot \mathbf{d}^i + \check{\mathbf{r}}^i \right), \tag{5.39b}$$

where the second form follows from (5.38). If we now (*i*) write the signal-space atom $\mathbf{d}^i$ in terms of its reduced-space version according to (5.28) and (*ii*) express

Operation	Complexity
$\mathbf{a}_n + \mathbf{b}_n$	n
$\mathbf{a}_n^\mathsf{T}\,\mathbf{b}_n$	$2n-1$
$\mathbf{A}_{m\times n}\mathbf{B}_{n\times k}$	$m(2n-1)k$
$\mathbf{A}_{m\times m}^{-1}\mathbf{a}_m$	$(1/3)m^3 + 2m^2$

Table 5.2: Complexities of common algebraic operations [Boyd 2004] in terms of floating point operations. The subscripts denote the dimensions of the vectors or matrices involved.

$\check{\mathbf{r}}^i$ in terms of its reduced version using (5.39a), we arrive at the following form of (5.39b):

$$\check{\mathbf{r}}^{(i-1)\prime} = (\mathbf{\Phi}_{k_i})^\mathsf{T}\left(\gamma^i \cdot \left(\mathbf{\Phi}_{k_i}\mathbf{d}^{i\prime}\right) + \mathbf{\Phi}_{k_{i+1}}\check{\mathbf{r}}^{i\prime}\right) \tag{5.40}$$

$$= \gamma^i \cdot \mathbf{d}^{i\prime} + \boldsymbol{\phi}_{\mathbf{d}^{i\prime}}\check{\mathbf{r}}^{i\prime}. \tag{5.41}$$

For completeness we note that, in substituting $\check{\mathbf{r}}^i = \mathbf{\Phi}_{k_{i+1}}\check{\mathbf{r}}^{i\prime} = \mathbf{\Phi}_{k_{i+1}}(\mathbf{\Phi}_{k_{i+1}})^\mathsf{T}\check{\mathbf{r}}^i$ in (5.40) we have apparently neglected the component of $\check{\mathbf{r}}^i$ that exists in the left null-space of $\mathbf{\Phi}_{k_i}$. Yet this is not a problem since, from (5.37) and (5.27b), we know that $\check{\mathbf{r}}^i \in \text{span}^\perp\left(\mathbf{S}^i\right) = \text{span}\left(\mathbf{\Phi}_{k_{i+1}}\right)$.

Expression (5.41) specifies a method to carry out signal reconstruction using the rTSITD structure directly. We illustrate the approach on the right-hand side of Fig. 5.7: As for the TSITD strategy first described, the tree is traversed backwards from the last node k_L visited during the decomposition, but this time the branch reduction matrices $\boldsymbol{\phi}_{\mathbf{d}^{i\prime}}$ (where $\mathbf{d}^{i\prime} \in \mathbf{D}'_{k_i}$ is the atom selected in the i-th iteration of (5.35)) are used to re-map the reduced accumulator vectors $\check{\mathbf{r}}^{i\prime}$ back to reduced spaces of successively higher dimensionality. The process continues until the root node, where the reconstructed signal will be given by $\check{\mathbf{r}}^{0\prime} = \check{\mathbf{r}}^0 = \hat{\mathbf{y}}^L$.

5.5.6 Complexity

The rTSITD-based decomposition and reconstruction schemes outlined on the right-hand side of Fig. 5.6 and Fig. 5.7 can be used to reduce the decomposition and reconstruction complexities of their TSITD counterparts. To demonstrate this, we use the number of floating-point operations as our complexity measure. Boyd [Boyd 2004] derives this complexity measure for basic algebraic operations and we reproduce them in Table 5.2. We will thus compute (r)TSITD complexity using the values in this table summed over the corresponding operations.

In the subsequent discussion, we first compare rTSITD complexity to that of TSITD and then to that of fixed-dictionary schemes using the common OMP

algorithm. For simplicity, we will assume that all (r)TSITD candidates and the reference fixed-dictionary have the same number of atoms N.

5.5.6.1 rTSITD complexity gain over TSITD

Consider the TSITD atom / coefficient selection rules in (5.14): From Table 5.2, the corresponding decomposition complexity at iteration i is $(2d-1)N$, where we have assumed that the coefficient is stored during atom selection and hence does not need to be computed during coefficient selection. For all the layers except the last, we further need to compute the output residual using

$$\mathbf{r}^i = \mathbf{r}^{i-1} - \gamma^i \cdot \mathbf{d}^i. \tag{5.42}$$

and this has an added cost of $2 \cdot d$. Hence, the TSITD decomposition operation at layer i incurs the following complexity:

$$C^D_{\text{TSITD}} = \begin{cases} (2d-1)N + 2d & i < L, \\ (2d-1)N & i = L. \end{cases} \tag{5.43}$$

From (5.35), the corresponding rTSITD complexity follows by substituting d by $(d-i+1)$. However, for all layers except the last, we further need to reduce the output residual through left-multiplication by $(\boldsymbol{\phi}_{\mathbf{d}^{i\prime}})^{\mathsf{T}}$ according to (5.33). The resulting rTSITD decomposition complexity at the i-th layer is

$$C^D_{\text{rTSITD}} = \begin{cases} (2d-2i+1)N + 2(d-i+1) \\ \quad + \ (d-i)(2d-2i+1) & i < L, \\ (2d-2i+1)N & i = L. \end{cases} \tag{5.44}$$

Considering next the reconstruction complexity associated to the i-th layer, this can be expressed for TSITD using (5.38) as follows:

$$C^R_{\text{TSITD}} = \begin{cases} 2d & i < L, \\ d & i = L. \end{cases} \tag{5.45}$$

where, for $i = L$ there is no need to add $\check{\mathbf{r}}^L = \mathbf{0}$. Again we obtain the corresponding rTSITD expression from the TSITD version by (i) using $d-i+1$ in place of d and (ii) accounting for the dimensionality increase operation done by left-multiplying by the branch reduction matrix $\boldsymbol{\phi}_{\mathbf{d}^{i\prime}}$ of the layer's chosen atom as indicated in (5.41). The resulting reconstruction complexity at the i-th layer is thus:

$$C^R_{\text{rTSITD}} = \begin{cases} 2(d-i+1)+ \\ (d-i+1)(2d-2i-1) & i < L, \\ (d-i+1) & i = L. \end{cases} \tag{5.46}$$

In the top of Fig. 5.8 we compare rTSITD to TSITD by plotting (see the curves with the $\times$ markers) the gain in cumulative complexity for decompositions and reconstructions involving L layers given by

$$\frac{\sum_{i=1}^{L}\left(C^{D}_{\text{TSITD}}(i)+C^{R}_{\text{TSITD}}(i)\right)}{\sum_{i=1}^{L}\left(C^{D}_{\text{rTSITD}}(i)+C^{R}_{\text{rTSITD}}(i)\right)}. \tag{5.47}$$

The three different curves appearing in the figure correspond to three different candidate dictionary sizes $N = 64, 256$, and 1024 (with signal dimension $d = 64$). Given the complexity overhead related to the dimensionality reduction operation, a reduction in complexity is attained only for sufficiently large sparsity ($L \geq 14$ and $L \geq 30$, for $N = 1024$ and 256, respectively). For too low N (*eg.,* $N = 64$), rTSITD does not attain a complexity gain at all. To address these shortcomings, we again consider the hybrid TSITD / rTSITD structure discussed in Section 5.5.3: up to a certain layer (layers 13, 29 and 46, for $N = 1024$, 256 and 64) the hybrid structure will rely on the signal-space representations of the TSITD structure. In the subsequent transition layer (respectively, layers $q = 14$, $q = 30$ and $q = 47$), the input residues are reduced using the path reduction matrices $\mathbf{\Phi}_k, \forall k \in \mathcal{L}_q$, thus enabling the usage of rTSITD in the following layers. The top of Fig. 5.8 also displays the complexities corresponding to this hybrid structure (labeled *hTSITD* in the figure), thus illustrating how it matches or outperforms TSITD's complexity uniformly over all layers. Note that the rTSITD and hTSITD curves for a given N define a region of attainable complexities corresponding to a tradeoff between storage footprint and decomposition / reconstruction complexity. The operating curve along this region is determined by the position of the hTSITD transition layer along the hybrid tree-structure.

5.5.6.2 Comparison to fixed-dictionary schemes using OMP

The OMP fixed-dictionary complexity expressions are similar to the TSITD complexities in (5.43) and (5.45). The only difference is that the decomposition expression (5.43) needs to also include the increased cost of building the approximation $\hat{\mathbf{y}}^i$ required to obtain the OMP residue $\mathbf{r}^i = \mathbf{y} - \hat{\mathbf{y}}^i$ at the output of the i-th ($i < L$) iteration:

$$\hat{\mathbf{y}}^i = \overbrace{\mathbf{S}^i \cdot}^{d(2i-1)} \underbrace{\overbrace{\left((\mathbf{S}^i)^{\mathsf{T}}\mathbf{S}^i\right)^{-1}}^{i^2(2d-1)} \cdot \overbrace{(\mathbf{S}^i)^{\mathsf{T}}\mathbf{y}}^{i(2d-1)}}_{(1/3)i^3+2i^2}. \tag{5.48}$$

The complexities of the sub-operations involved in building $\hat{\mathbf{y}}^i$ are indicated in the above expression, and the OMP fixed-dictionary decomposition complexity follows by adding the sum of all these terms to (5.43) for $i < L$. In the bottom of

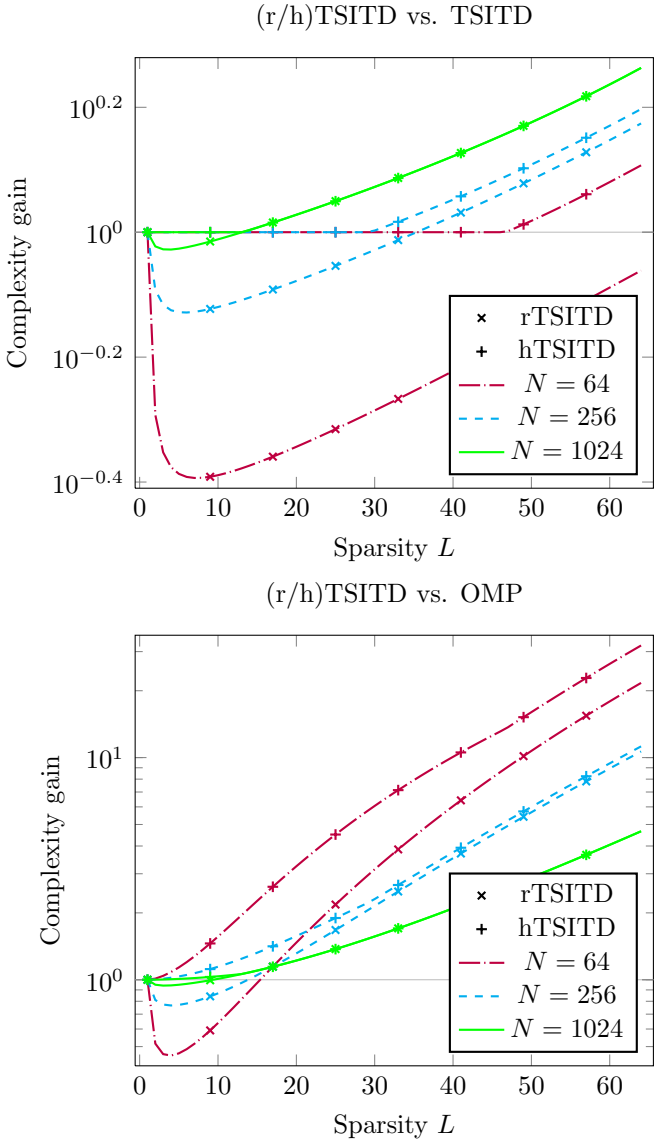

Figure 5.8: Complexity gain of the rTSITD structure and the hybrid rTSITD / TSITD (hTSITD) structure over *(top)* the TSITD structure as given by (5.47) and *(bottom)* fixed-dictionary OMP schemes as given by modifying (5.47) using (5.48).

Fig. 5.8 we thus illustrate rTSITD's complexity gain over fixed-dictionary OMP decompositions, again providing curves for hTSITD and rTSITD for several values of N. As shown, the hTSITD setup enjoys a complexity advantage uniformly over all L; the rTSITD setup enjoys an advantage for sufficiently high L.

5.5.6.3 Comparison to (separable SD) fixed-dictionary schemes of larger size using OMP

In the results section of this chapter we will see that, in fact, (r/h)TSITD candidate dictionaries having less atoms than a reference fixed dictionary can render a signal class more compressible. Thus in the top of Fig. 5.9 we evaluate the complexity gain of (r/h)TSITD structures with candidates having $N < 256$ atoms over OMP schemes using fixed-dictionaries with 256 atoms. As illustrated in the figure, the (r/h)TSITD structures with lower N can offer important performance gains of up to 10× in the lowest sparsities and up to 100× in the highest sparsities.

In the bottom of Fig. 5.9 we show that (r/h)TSITD also enjoys similar complexity gain when compared against OMP schemes using a Sparse Dictionary (SD) with $N = 256$ atoms. Such dictionaries (*cf.* Section 2.3.4.1) have a structure $\mathbf{D} = \mathbf{BA}$ such that the analysis coefficients

$$\mathbf{D}^\mathsf{T}\mathbf{r}^i = \mathbf{A}^\mathsf{T}\mathbf{B}^\mathsf{T}\mathbf{r}^i \tag{5.49}$$

can be computed efficiently. For example, let us assume that $\mathbf{B} \in \mathbb{R}^{d\times N}$ is separable and effectively applies the same transform $\mathbf{B}_0 \in \mathbb{R}^{\sqrt{d}\times\sqrt{N}}$ along the rows and columns of the residual image block $\mathbf{r}^i_{\sqrt{d}\times\sqrt{d}}$ ($\mathbf{r}^i$ is just the vertical concatenation of the columns of $\mathbf{r}^i_{\sqrt{d}\times\sqrt{d}}$). The projection operation $\mathbf{B}^\mathsf{T}\mathbf{r}^i$ can hence be calculated using $\mathbf{B}_0^\mathsf{T}\,\mathbf{r}^i_{\sqrt{d}\times\sqrt{d}}\mathbf{B}_0$, and this incurs a complexity of

$$(2\sqrt{d}-1)(\sqrt{Nd}+N). \tag{5.50}$$

The complexity of the subsequent multiplication by $\mathbf{A}^\mathsf{T}$ (where $\mathbf{A}$ is sparse with L_a non-zero coefficients per column) can then be lower bounded by the complexity of multiplying a full matrix of size $N \times L_a$ by a vector of size L_a (a complexity of $N(2L_a - 1)$). Hence the total complexity when projecting a signal unto the columns of a separable SD structure is

$$(2\sqrt{d}-1)(\sqrt{Nd}+N) + N(2L_a - 1). \tag{5.51}$$

This expression can be used in place of $2N(d-1)$ in (5.43) to obtain the SD decomposition complexity; the SD reconstruction complexity will again be given by (5.45) (if we assume that the atoms of $\mathbf{B}$ are stored in full form and do not need to be reconstructed from $\mathbf{B}_0$).

5.5.6.4 Dictionary training

Dictionary training complexity can also benefit from the reduced representations: Using the branch reduction matrices $\boldsymbol{\phi}_{\mathbf{d}^{i'}}$ (as per (5.33)) to obtain the reduced

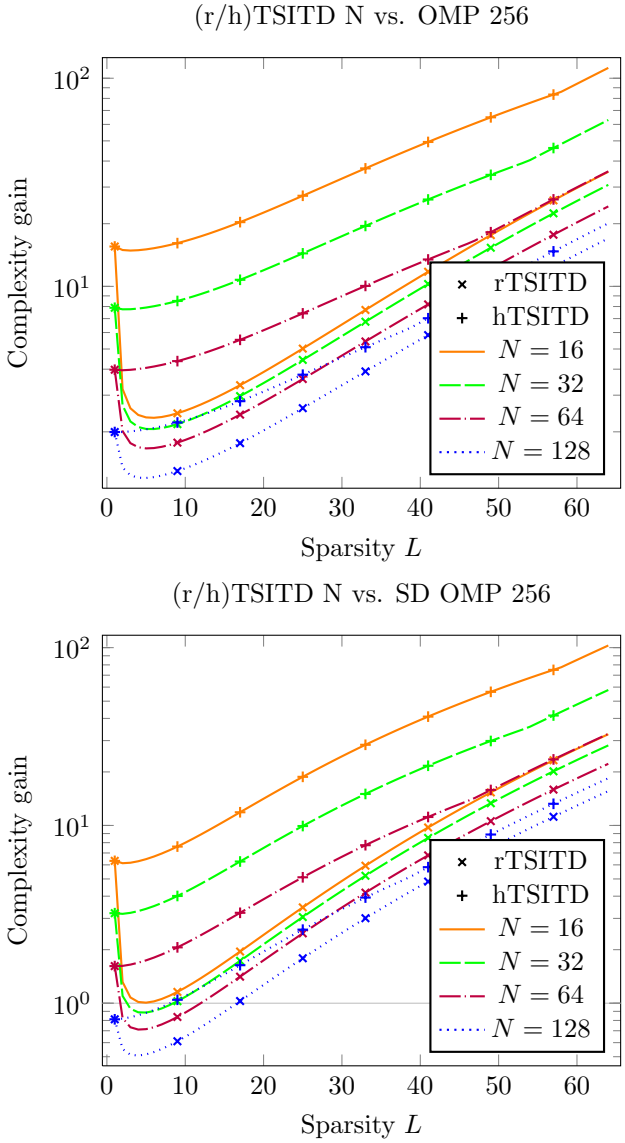

Figure 5.9: Complexity gain (5.47) of the rTSITD structure and the hybrid rTSITD / TSITD (hTSITD) structure over *(top)* OMP schemes with general fixed dictionaries of 256 atoms and *(bottom)* OMP schemes using SD fixed dictionaries of 256 atoms; the SD scheme further uses an atom sparsity of $L_a = 15$.

residues $\mathbf{r}_t^{(i-1)\prime}$ of the training vectors $\{\mathbf{y}_t\}_{t=1}^T$ at the input of dictionaries of successive layers, one can train $\mathbf{D}'_k$ directly by applying the candidate construction algorithm in Fig. 5.1 to the reduced training sets as follows:

$$\mathbf{D}'_k = \textsc{Candi}\left(\{\mathbf{r}_t^{(i-1)\prime}\}_{t\in\mathcal{T}_k}\right), \tag{5.52}$$

where $\mathcal{T}_k$ is defined in (5.18) (equivalent definitions can be obtained using $\mathbf{D}'_k = child(\mathbf{d}_t^{l-1\prime})$). The complexity benefit will follow from carrying the iterative clustering of the candidate construction algorithm in spaces of decreasing dimensionality.

5.5.7 Practicality for large tree-structures

The rTSITD dictionary we presented in this section allowed us to benefit from the rich redundancy and low complexity of the TSITD structure while at the same time mitigating, to some extent, the related TSITD storage overhead. The resulting reduction in storage overhead is nonetheless too small to render the scheme practical when considering candidate dictionaries with more than a few atoms or tree-structures with more than a few layers. The TSITD storage footprint grows very quickly with the layer index l, and the rTSITD version reduces this footprint only by a small amount (*cf.* Fig. 5.5). This shortcoming will be addressed by the ITD structure introduced in the coming section.

5.6 The Iteration-Tuned and Aligned Dictionary (ITAD)

The rTSITD structure discussed up to now (and its signal-space version TSITD) benefits from several important properties including a large redundancy with tractable decomposition and reconstruction complexity, orthogonality of its selected-atoms matrix and the resulting equivalence of all three algorithms of the matching pursuit family. These properties are manifestations of the tree-structuring and the ensuing fact that TSITD residues at the input of dictionaries $\mathbf{D}_k$, $k \in \mathcal{L}_l$, exist in subspaces of reduced dimensionality $d - l + 1$. Yet the fact that the number of (r)TSITD candidate dictionaries per layer will grow rapidly with the layer index i makes the structure difficult to store and train. As we will now show, the *Iteration-Tuned and Aligned Dictionary* (ITAD) that we introduce in this section succeeds in retaining the dimensionality reduction property of (r)TSITD with a storage footprint that is nonetheless drastically smaller.

5.6.1 Geometrical motivation for the ITAD alignment operation

In order to arrive at the ITAD structure, let us begin by revisiting the original ITD structure introduced in Chapter 4 and discussed in Section 5.3. Recall that each ITD layer l consists of a single candidate $\mathbf{D}_l$ and, as a result, ITD enjoys a storage footprint that grows only linearly with the layer index l. Yet, unlike (r)TSITD,

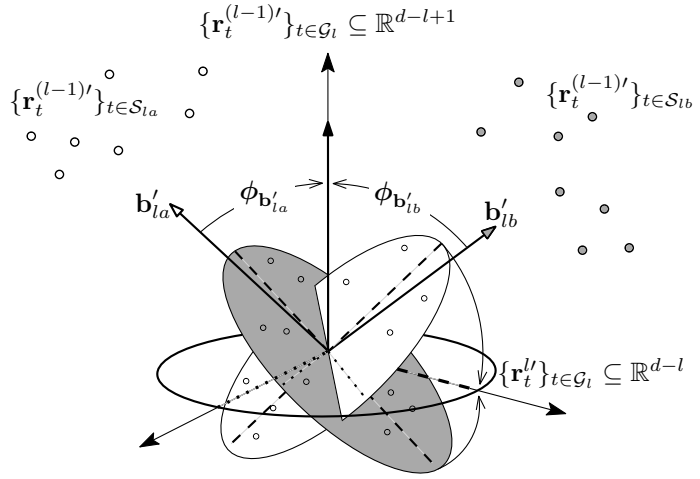

Figure 5.10: The ITAD alignment operation: The residual classes of each atom are rotated so that their null-spaces as well as their principal bases are aligned. Un-aligned (respectively, aligned) residual classes are denoted by the oblique (horizontal) ellipses. The first and second principal components of all three ellipses are denoted by the dashed and dotted lines.

ITD does not enjoy a reduction of dimensionality across layers l: Considering one ITD candidate $\mathbf{D}_l$, we observe that the class of residue vectors $\mathbf{r}^l$ produced by the l-th layer spans the entire input signal space $\mathbb{R}^d$. However, the class of residue vectors $\mathbf{r}^l$ produced by an ITD atom $\mathbf{d} \in \mathbf{D}_l$ does enjoy a reduction in dimensionality since it is contained in the left null-space of $\mathbf{d}$. This suggests *aligning* the residual classes of atoms at the output of each ITD layer by rotating them so that the union of the resulting rotated classes enjoys the reduced dimensionality. To retain the reduced storage of the ITD structure, this union of aligned classes can then be used to train a single candidate dictionary in the subsequent layer. Hence there is some interest in ensuring that the alignment operation not only serves to reduce the dimensionality of the residual sets but that it further better structures them for eventual representation with a single dictionary. Thus the chosen alignment matrix will also ensure that the aligned residual classes share a common principal basis that will further be the principal basis of the union of aligned residual classes.

In Fig. 5.10 we illustrate the ITAD alignment process above described: Let us assume that we have, at the input of the ITAD structure, the set of vectors $\{\mathbf{y}_t\}_{t=1}^T$ that were used to train the structure. Let

$$\mathcal{I}_1 = \{t = 1, \ldots, T : |\mathbf{y}_t|^2 \geq d\kappa^2\} \text{ and} \tag{5.53a}$$

$$\mathcal{I}_l = \{t \in \mathcal{I}_{l-1} : |\mathbf{r}_t^{l-1}|^2 \geq d\kappa^2\}, l > 1. \tag{5.53b}$$

The set of residues at the input of the l-th ITAD layer will be $\{\mathbf{r}_t^{(l-1)\prime}\}_{t\in\mathcal{I}_l}$. Dictionary matrix $\mathbf{B}'_l = (\mathbf{b}'_{la})_{a=1}^N$ will denote the l-th ITAD dictionary, which is called the *prototype dictionary* of the l-th layer. Let $\mathcal{S}_{la}$, $a = 1, \ldots, N$ denote a partition of $\mathcal{I}_l$ such that residues $\{\mathbf{r}_t^{(l-1)\prime}\}_{t\in\mathcal{S}_{la}}$ are those that are assigned to the a-th atom from layer l, *i.e.,*

$$\mathcal{S}_{la} = \left\{ t \in \mathcal{I}_l : \underset{\mathbf{b}'\in\mathbf{B}'_l}{\operatorname{argmax}} |(\mathbf{r}_t^{(l-1)\prime})^{\mathsf{T}}\, \mathbf{b}'| = \mathbf{b}'_{la} \right\}. \tag{5.54}$$

The large white and gray dots in $\mathbb{R}^{d-l+1}$ represent two such sets of residues $\{\mathbf{r}_t^{(l-1)\prime}\}_{t\in\mathcal{S}_{lj}}, j = a, b$ for two atoms $\mathbf{b}'_{lj} \in \mathbf{B}'_l$. The atom selected for each set is incorporated into the representation of the signals, producing the sets

$$\left\{ \mathbf{r}_t^{(l-1)\prime} - \left((\mathbf{r}_t^{(l-1)\prime})^{\mathsf{T}}\, \mathbf{b}'_{lj} \right) \mathbf{b}'_{lj} \right\}_{t\in\mathcal{S}j} \tag{5.55}$$

represented by the white and gray dots in the subspaces denoted by ellipses orthogonal to the atoms $\mathbf{b}'_{lj}$. At this point, the ITAD alignment process takes place: each subspace is rotated so that it coincides with the horizontal plane representing $\mathbb{R}^{d-l}$ using the rotation matrices denoted $\boldsymbol{\phi}_{\mathbf{b}'_{lj}}$. The union of all residues from all rotated subspaces is denoted $\{\mathbf{r}_t^{l\prime}\}_{t\in\mathcal{I}_l}$. Pruning this set according to (5.53b) produces $\{\mathbf{r}_t^{l\prime}\}_{t\in\mathcal{I}_{l+1}}$ (not denoted), which is the input residual set of layer $l+1$. The rotation matrices $\boldsymbol{\phi}_{\mathbf{b}'_{lj}}$ are selected so that the sets $\{\mathbf{r}_t^{l\prime}\}_{t\in\mathcal{S}_{lj}}$ also share the same principal component directions. In the figure, the first and second principal components of the unaligned, projected residues (*i.e.,* those in the white and gray ellipses, given by (5.55)) and those of the aligned residues $\{\mathbf{r}_t^{l\prime}\}_{t\in\mathcal{S}_{lj}}$ (not illustrated but contained inside the clear ellipse) are denoted by dashed and dotted lines.

Note that in our discussion we have assumed that the input vectors $\{\mathbf{y}_t\}_{t=1}^T$ are the training vectors used to build the ITAD structure. For a generic input set, the reduction in dimensionality illustrated in Fig. 5.10 will nonetheless hold, although the rotation matrices $\boldsymbol{\phi}_{\mathbf{b}'_{lj}}$ (which are learned, as we will see) are not guaranteed to align the principal components of the sets $\{\mathbf{r}_t^{l\prime}\}_{t\in\mathcal{S}_{lj}}$. We can however assume that the training set is a representative sampling of the target signal class, in which case we can likewise assume alignment of principal components.

5.6.2 The ITAD structure as a particular case of (r)TSITD

The ITAD structure above described is illustrated in Fig. 5.11: Each layer $l = 1, \ldots, C$ of the ITAD structure contains a single prototype dictionary matrix $\mathbf{B}'_l \in \mathbb{R}^{(d-l+1)\times N}$. The output residues of each of the atoms of $\mathbf{B}'_l = (\mathbf{b}'_{la})_{a=1}^N$ are aligned / reduced by means of the corresponding alignment matrix $\boldsymbol{\phi}_{\mathbf{b}'_{la}}$. Note that the

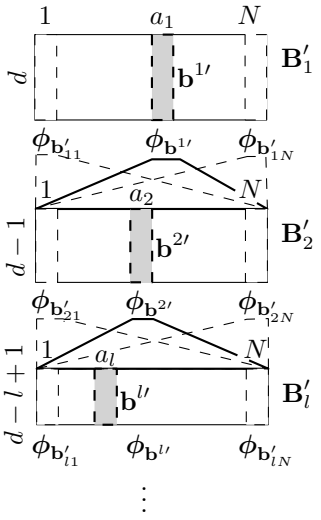

Figure 5.11: The ITAD structure. An equivalent rTSITD diagram for this structure can be obtained from the right-hand side of Fig. 5.3 by letting $\mathbf{D}'_{k_i} = \mathbf{D}'_i$ and accordingly substituting each $\boldsymbol{\phi}_{\mathbf{d}^{i\prime}}$ by the corresponding $\boldsymbol{\phi}_i$.

figure resembles the rTSITD structure on the bottom of Fig. 5.3. Indeed the ITAD structure of Fig. 5.11 is just a particular instance of the general rTSITD structure in Fig. 5.3 where the tree-nodes of any given layer l are folded unto themselves because they all contain the same reduced candidate dictionary $\mathbf{D}'_l$. Note that the unfolded version of the ITAD structure in Fig. 5.11 can be obtained from the bottom of Fig. 5.3 by (i) substituting general rTSITD dictionaries $\mathbf{D}'_k$, $k \in \mathcal{L}_l$, by the corresponding ITAD prototype dictionary $\mathbf{B}'_l$,

$$\mathbf{D}'_k = \mathbf{B}'_l \quad \forall k \in \mathcal{L}_l, \tag{5.56}$$

and (ii) substituting general rTSITD branch reduction matrices by the corresponding ITAD alignment matrices,

$$\boldsymbol{\phi}_{\mathbf{d}'_{ka}} = \boldsymbol{\phi}_{\mathbf{b}'_{la}} \quad \forall k \in \mathcal{L}_l, \forall a = 1, \ldots, N. \tag{5.57}$$

Thus, the ITAD structure will enjoy all the rTSITD properties and applications discussed in previous sections. In particular, the ITAD signal decomposition and reconstruction complexity will be that of rTSITD, and the hybrid TSITD/rTSITD structure discussed in Section 5.5.3 can be implemented using ITAD in place of the rTSITD component. Furthermore, since rTSITD is just an equivalent representation of an underlying TSITD structure, one can also obtain a TSITD representation of ITAD corresponding to the top illustration in Fig. 5.3 (this will be useful to visually inspect the ITAD dictionaries in signal space).

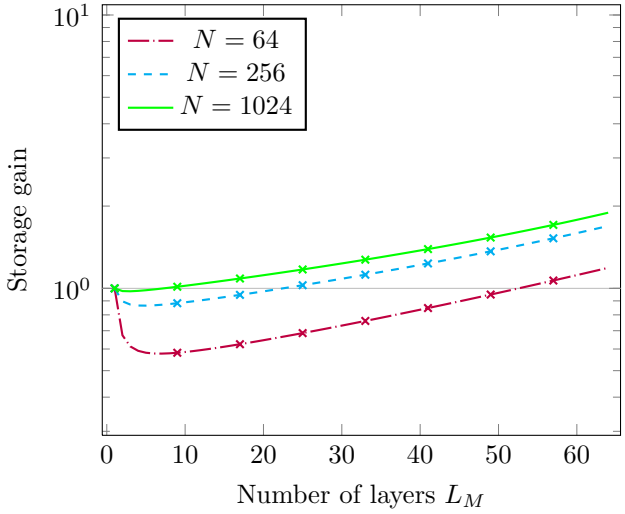

Figure 5.12: Storage gain of ITAD over ITD. Storage is measured in terms of the total number of matrix entries for the entire structure consisting of C layers.

The fact that ITAD requires storage of a single prototype dictionary per layer represents an important reduction in storage footprint relative to general (r)TSITD structures. Note, however, that this comes at the expense of requiring the storage of the branch reduction matrices: the matrices $\boldsymbol{\phi}_{\mathbf{b}'_{kl}}$ will be obtained as part of the training process and can no longer be built given the related reduced atom and an orthogonalization algorithm chosen by convention (*cf.* Section 5.5.3). Yet even with this consideration, the fact that the ITAD storage footprint grows linearly with the layer index l results in a drastic storage advantage relative to (r)TSITD and for all layers l. Even more, for a sufficiently large number of layers C, the ITAD storage footprint will even be smaller than that of ITD. We illustrate this point in Fig. 5.12, where we show the gain in storage footprint of ITAD relative to ITD. Note that, in the figure, N denotes the total number of atoms per ITAD prototype $\mathbf{B}'_l$ as well as the total number of atomes per ITD layer dictionary $\mathbf{D}_l$.

5.6.3 Training the ITAD structure

We now propose a training algorithm to construct the ITAD structure described above.

5.6.3.1 The prototype dictionary

Similarly to the case of rTSITD reduced candidate dictionaries in (5.52), the ITAD prototype dictionaries will be constructed using the candidate construction

algorithm in Fig. 5.1 applied to the reduced residuals. However, the training set will consist of all the (*aligned*) reduced residual vectors at the output of the previous layer $l-1$, subject to a minimum energy constrain as per (5.53b):

$$\mathbf{B}'_i = \textsc{Candi}\left(\{\mathbf{r}_t^{l-1\prime}\}_{t\in\mathcal{I}_l}\}\right). \tag{5.58}$$

This approach is justified given (*i*) the ITAD constraint that a single prototype dictionary exist per layer and (*ii*) the assumption that a single atom / coefficient pair will be chosen from each ITAD layer (as is done for ITDs in general).

5.6.3.2 The alignment / reduction matrices

Regarding the ITAD alignment matrices, recall first that general rTSITD branch reduction matrices are defined in (5.29) as being any orthonormal basis of the left null-space of the corresponding reduced atom. Indeed, an infinity of possible branch reduction matrices will satisfy this constraint. The ITAD scheme uses this liberty of selection to force the aligned residual classes $\{\mathbf{r}_t^{l\prime}\}_{t\in\mathcal{S}_{la}}, a=1,\ldots,N$ to share a common principal basis that is further the principal basis of the union of aligned classes. Letting $\text{LSV}(\cdot)$ return the left-singular vectors ordered by decreasing singular value, the previous statement can be expressed as follows:

$$\mathbf{I} = \text{LSV}\left((\mathbf{r}_t^{l\prime})_{t\in\mathcal{S}_{la}}\right) = \text{LSV}\left((\mathbf{r}_t^{l\prime})_{t\in\mathcal{G}_l}\right), \forall a\in\{1,\ldots,N\}, \tag{5.59}$$

where $\mathbf{I}$ is the identity matrix. The right-most equality follows because the principal basis of a dataset partitioned into subsets sharing the same principal basis is given by that same basis. Regarding the left-most equality, the uniquely defined alignment matrices $\boldsymbol{\phi}_{\mathbf{b}'_{la}}$ for this task correspond to the ordered left-singular vectors (excluding the first) of the residual vectors used to train $\mathbf{b}'_{la}$ (equal to the first left-singular vector). We can express this formally by assuming that the training process in (5.58) has finished and hence the set of vectors used to train atom $\mathbf{b}'_{la}$ is given by $(\mathbf{r}_t^{l-1\prime})_{t\in S_{la}}$. Atom $\mathbf{b}'_{la}$ and its alignment matrix $\boldsymbol{\phi}_{\mathbf{b}'_{la}}$ are obtained from the left-singular vectors of these residues as follows:

$$\left[\mathbf{b}'_{la} \quad | \quad \boldsymbol{\phi}_{\mathbf{b}'_{la}}\right] = \text{LSV}\left((\mathbf{r}_t^{l-1\prime})_{t\in S_{la}}\right). \tag{5.60}$$

Notice that the required $\text{LSV}(\cdot)$ operation is carried out in line 13 of Fig. 5.1 as part of the $\textsc{Candi}(\cdot)$ algorithm, and hence the ITAD structure can be trained using this same algorithm.

5.6.3.3 Example of an ITAD structure

In Fig. 5.13 we carry out a visual inspection of the l-th ITAD prototype dictionary $\mathbf{B}'_l$ (for $l=1,\ldots,4$) by plotting its *mean atom energy vector* given by

$$\bar{\mathbf{b}}'_l = \frac{1}{N}\left(\mathbf{B}'_l \circ \mathbf{B}'_l\right)\mathbf{1}, \tag{5.61}$$

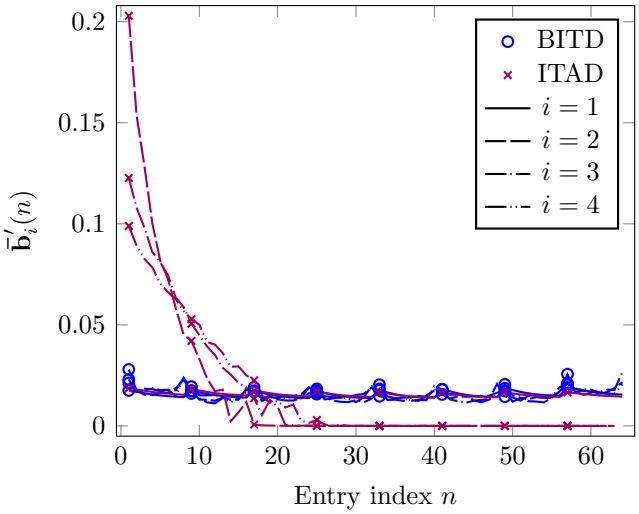

Figure 5.13: Plot of the mean atom energy vector $\bar{\mathbf{b}}'_i$ of the dictionary $\mathbf{D}'_i$ (for $i = 1, \ldots, 4$) as defined in (5.61) for ITAD (the corresponding ITD expression follows from (5.61) by substituting the reduced dictionary $\mathbf{D}'_i$ by the signal-space ITD dictionary $\mathbf{D}_i$).

where the operator $\circ$ carries out the entry-wise product of two matrices and $\mathbf{1}$ is the all-ones vector. As a reference, we plot the corresponding measure for the first four layers of a ITD structure built using the same training set. Note that, for $l \geq 2$ the ITAD reduced atoms are similar to the first elementary vector $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix}^{\mathsf{T}}$ which, according to (5.59), is the first principal vector of the layer's training set $\{\mathbf{r}_t^{(l-1)\prime}\}_{t \in \mathcal{G}_l}$ or of any of its atom-input constituent subsets $\{\mathbf{r}_t^{(l-1)\prime}\}_{t \in \mathcal{S}_{la}}, a = 1, \ldots, N$. This is reasonable since the prototype dictionaries are used to produce single-atom approximations of a layer's input residuals and the first principal vector of a dataset by definition provides the optimal single-vector approximation basis [Klema 1980]. The ITD atoms, on the other hand, display comparable energy in all positions, suggesting that they are less structured than their ITAD counterparts. Note that the first ITAD layer also displays comparable energy in all positions. This is expected since the first set of training residues $\{\mathbf{y}_t\}_{t \in \mathcal{L}_1}$ does not consist of aligned subsets.

In Fig. 5.14 we present an example of the corresponding TSITD (signal-space) representation of the candidate dictionaries of the ITAD structure along three different branches of the same first four layers shown in Fig. 5.13. Note that the ITAD alignment operation results in signal-space candidate dictionaries that display organization and structure: Layers with a higher index l consist of dictionaries with higher frequency content. Atoms of successive layers further inherit the structural constitution and orientation of their ancestor atoms.

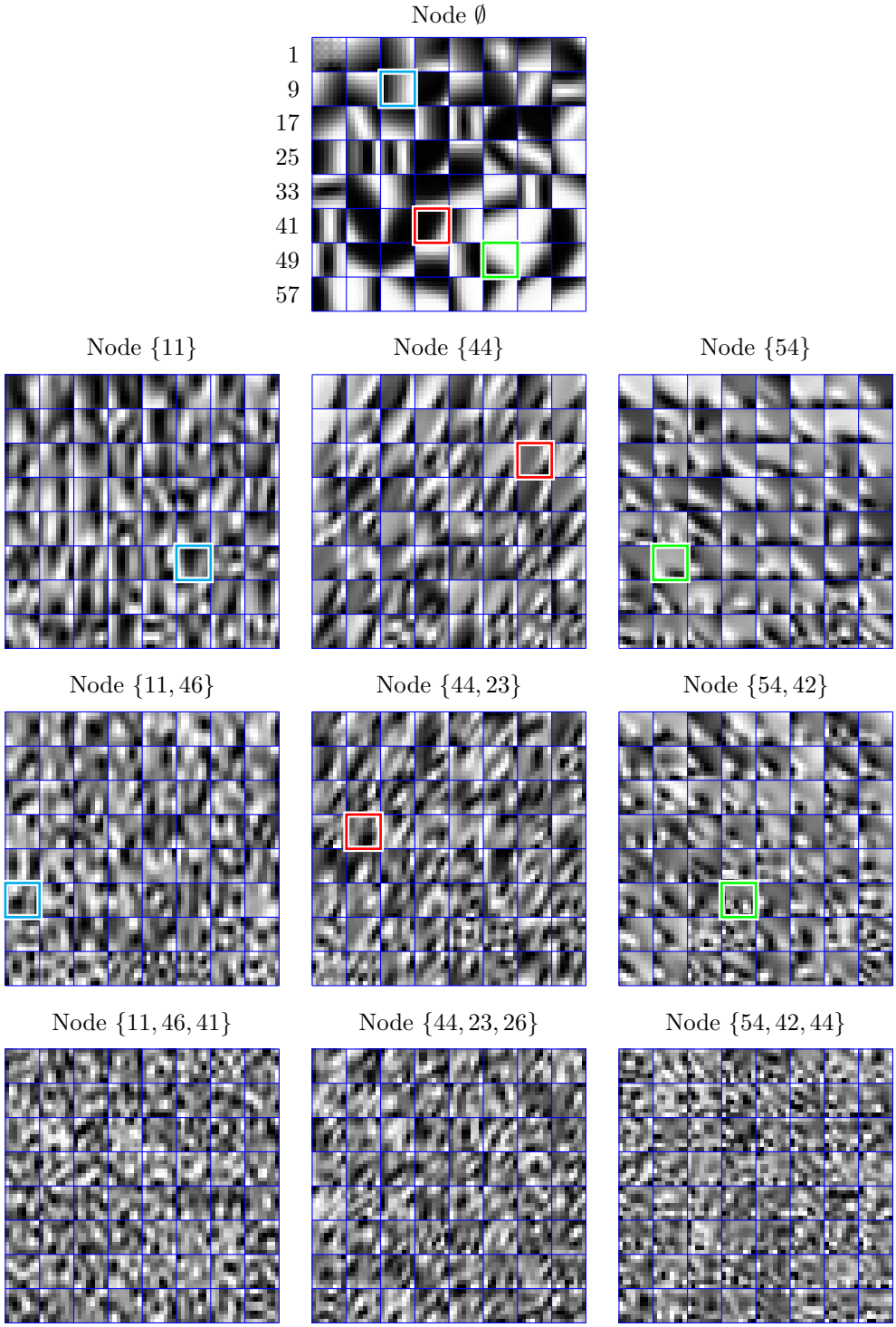

Figure 5.14: An example of ITAD candidate dictionaries (TSITD signal-space representation) along three different branches of the tree-structure. Each candidate $\mathbf{D}_k$, $k \in \mathcal{L}_l$ is labeled using the indices $\{a_1, \ldots, a_{l-1}\}$ of its ancestor atoms $\mathbf{d}^j = \mathbf{d}_{k_j a_j}$, $k_j \in \mathcal{L}_j$. Atom-indices increase along the rows, as indicated for the root dictionary. The atoms selected from each dictionary are indicated by a heavy frame. All candidate dictionaries consist of 64 atoms.

Dictionary	Parameters
K-SVD [Aharon 2006b]	$\kappa = 8$, 40 iterations, DCT initialization
Sparse Dictionary (SD) [Rubinstein 2010a]	$\kappa = 14$, $L_a = 15$, 40 iterations, random initialization, DCT base dictionary of same size
Online Learned Dictionary (ONLD) [Mairal 2010a]	$\kappa = 10$, 1000 seconds

Table 5.3: Parameters used for reference trained dictionaries in Fig. 5.16 and Fig. 5.17.

5.7 Results

In this section we evaluate the proposed TSITD and ITAD dictionaries by comparing them against (*i*) the ITD scheme of Chapter 4; (*ii*) the complete DCT dictionary used in the JPEG standard; (*iii*) its overcomplete version (widely used for initialization of learning schemes); and the state-of-the-art (*iv*) K-SVD dictionary [Aharon 2006b], Online Learned Dictionary (ONLD) [Mairal 2010a] and (*v*) Sparse Dictionary (SD) [Rubinstein 2010a].

In a first experiment, we evaluate the PSNR of the sparse approximations obtained with the various dictionaries as a function of sparsity. Better sparse representations are likely to result in better performance in various real-life applications [Guleryuz 2006, Mallat 2008]. The second and third experiments thus evaluate the performance of the various dictionaries in the context of image compression and image denoising. For the denoising experiment, we only compare against K-SVD , although comparisons against various other schemes (including one based on the overcomplete DCT) can be found in [Elad 2006b] and subsequent publications. See [Chatterjee 2010] for a survey and interesting exploration of the theoretical limits of image denoising.

5.7.1 Summary of results

We show experimentally that, because of the large size of the TSITD tree structure, certain TSITD configurations suffer from over-fitting to the training data, subsequently affecting TSITD performance when evaluated on a test set different from the training set. Our tests show that ITAD does not suffer from over-fitting. For the denoising experiment, the training and test data are the same, and thus TSITD over-fitting is not an issue. Thus in that experiment we observe that ITAD generally outperforms ITD and under-performs when compared to TSITD, corresponding respectively to an increased redundancy relative to ITD and a reduced

redundancy relative to TSITD.

When compared against the (overcomplete) DCT and the state-of-the-art K-SVD , ONLD and SD dictionaries (experiments 1 and 2), our scheme will prove to offer significant advantages in terms of performance / complexity tradeoff.

5.7.2 Dictionary training

The number of atoms in each ITAD prototype dictionary is specified using a nominal value N. The actual number of atoms N_k of each $\mathbf{D}'_k$ is selected to ensure that at least 5 training samples exist per candidate dictionary atom using

$$N_k = \min(N, \left\lfloor \frac{|\mathcal{T}_k|}{5} \right\rfloor), \tag{5.62}$$

were $|\mathcal{T}_k|$ is the size of the residual training set used to train $\mathbf{D}_k$ (*cf.* (5.18)).

In all experiments, training signals (respectively, test signals) will be decomposed using an RMSE threshold (*cf.* (5.8)) denoted κ (ε). We use ITD-MP for all ITD schemes and OMP for all fixed-dictionary schemes. For ITDs in general (including ITAD), the training threshold κ serves to prune the training set at the input of successive layers since training residuals satisfying $|\mathbf{r}_{\mathcal{A}_i}^{i-1}| < d \cdot \kappa^2$ are implicitly removed from the training set.

Training of K-SVD , ONLD and SD dictionaries is done using the software made publicly available by the authors [Aharon 2006a, Mairal 2010b, Rubinstein 2010c]. The parameters used for each scheme are specified in Table 5.3. We used extensive experimentation to select these parameters following the procedure detailed in Appendix 5.A. Note that this procedure favors the reference dictionaries over ITAD, as it consist of using the dictionary (learned over the training set) that yields the best curve computed over the testing set (the ITAD learning scheme had knowledge only of the training set). For experiments 1 and 2, ITAD further required no parameter selection, as we set $\kappa = 0$ and $C = d/2$ for these first two experiments.

5.7.3 Datasets

Throughout our experiments we use image blocks of size 8×8 to form signal vectors $\mathbf{y}$. For the first two experiments (sparsity vs. PSNR and compression) we will use a dataset that is very similar to the one used in the K-SVD paper [Aharon 2006b]. It consists of frontal pose face images of 545 different subjects from the FERET dataset [Phillips 2000]: the first 445 images comprise the training set and the remaining 100 images comprise the test set. The size of each image is 768×512,

Figure 5.15: Sample images from the FERET dataset used for PSNR vs. sparsity and compression experiments.

for a total of 2.7×10^6 training vectors and 6.1×10^5 test vectors. We provide some sample images in Fig. 5.15. This is the same dataset used in Chapter 4.

For the denoising experiment we will work with commonly used images (*peppers256* and *Barbara*).

5.7.4 Experiment 1: PSNR vs. sparsity

Reference Dictionary	N	ITAD PSNR Gain (dB)	ITAD Complexity Gain (×)
DCT-256	32	0.47	1.45
K-SVD	64	0.30	3.96
SD	32	0.24	3.20
ONLD	32	0.58	7.76

Table 5.4: ITAD (N atoms) compressibility and complexity gains over reference dictionaries (256 atoms). The value of N used in the comparisons (column N) corresponds to the lowest N resulting in an ITAD PSNR advantage across all sparsities shown in Fig. 5.16. The values of columns *PSNR Gain* and *Complexity Gain* are measured using $L = 2.5$ and assuming an hTSITD implementation of ITAD. Complexity gains are given in Fig. 5.9 top for K-SVD / ONLD and bottom for SD. Complexities for the overcomplete DCT follow from (5.50) and the related discussion.

In the top of Fig. 5.16 we compare the approximation error produced by ITAD, ITD and TSITD as a function of the sparsity of the representation. Given the fact that ITAD is just a constrained instance of (r)TSITD, one would expect TSITD to outperform ITAD. One would further expect the greater redundancy exposed

by the ITAD / (r)TSITD tree-structures to result in improved performance over ITD. The ITAD behavior relative to ITD is indeed as expected: ITAD outperforms ITD for all dictionary sizes. However, ITAD also outperforms TSITD for $N \geq 64$. TSITD becomes over-fitted to the training data for sufficiently high N given the exponential growth of the TSITD layers. ITAD, on the other hand, does not suffer from an exponential growth of its layers and thus does not suffer from over-fitting. Hence ITAD can outperform TSITD with comparable decomposition / reconstruction complexity (*cf. rTSITD* curves in the top of Fig. 5.8) and a storage footprint comparable (or even smaller) to that of ITD (*cf.* Fig. 5.12).

In the middle and bottom of Fig. 5.16 we also compare the PSNR vs. sparsity performance of ITAD versus the complete and overcomplete DCT dictionary and the state-of-the-art learned dictionaries K-SVD , ONLD and SD. The results reported for the reference learned dictionaries correspond to the best curve obtained when varying the training parameters over a reasonable range (*cf.* Appendix 5.A). Note that ITAD structures having $N < 256$ atoms per candidate outperform all reference fixed-dictionary schemes of 256 atoms. This gain in compressibility is hence attained at a reduced decomposition / reconstruction complexity. In Table 5.4 we summarize these results by displaying the PSNR and complexity gains over the reference schemes for select N using a sparsity of $L = 2.5$.

We note that the results in Fig. 5.16 indicate that K-SVD can outperform SD in terms of compressibility, which seems to contradict [Rubinstein 2010a], where it is stated that the restricted structure of the SD scheme makes it less predisposed to become over-fitted to the training data. We believe that the optimal number of free optimization variables (note that for K-SVD this is $d \cdot N$) is inversely proportional to the redundancy of the data class. The data class used in [Rubinstein 2010a] is indeed very redundant, as it consists of CT images of a fixed body part. These images are more similar across different subjects than their corresponding face images taken in uncontrolled conditions (as is the case for the FERET dataset). Furthermore, as observed in [Rubinstein 2010a], the size of the training set also influences the relative performance of the two schemes, and indeed we use much larger training sets (2.7×10^6 as opposed to at most 8×10^4).

5.7.5 Experiment 2: Image compression

We now evaluate the performance of ITAD in the context of image compression by plotting analytically derived (*cf.* Appendix 5.B) rate-distortion curves. Note that schemes based on overcomplete dictionaries have recently been used to construct image codecs that perform comparably [Sezer 2008] or even significantly better [Bryt 2008] than the state-of-the-art JPEG2000 [Adams 2005] algorithm.

In the top of Fig. 5.17 we plot the rate-distortion curves for ITAD, TSITD

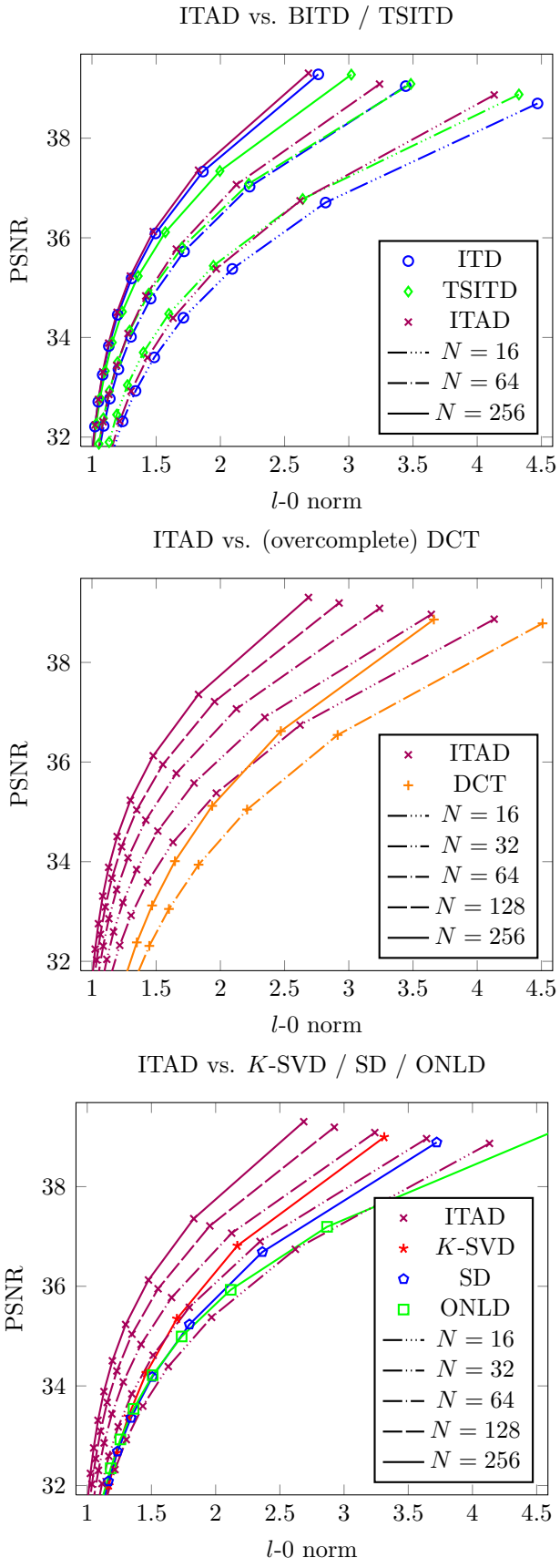

Figure 5.16: PSNR vs. sparsity. Top: ITAD vs. ITD and TSITD; bottom: ITAD vs. K-SVD and the (over)complete DCT. N denotes the number of atoms per layer (ITD), per candidate dictionary (TSITD), per prototype dictionary (ITAD) and per fixed-dictionary (K-SVD , ONLD and SD), resulting in comparable complexities for all schemes.

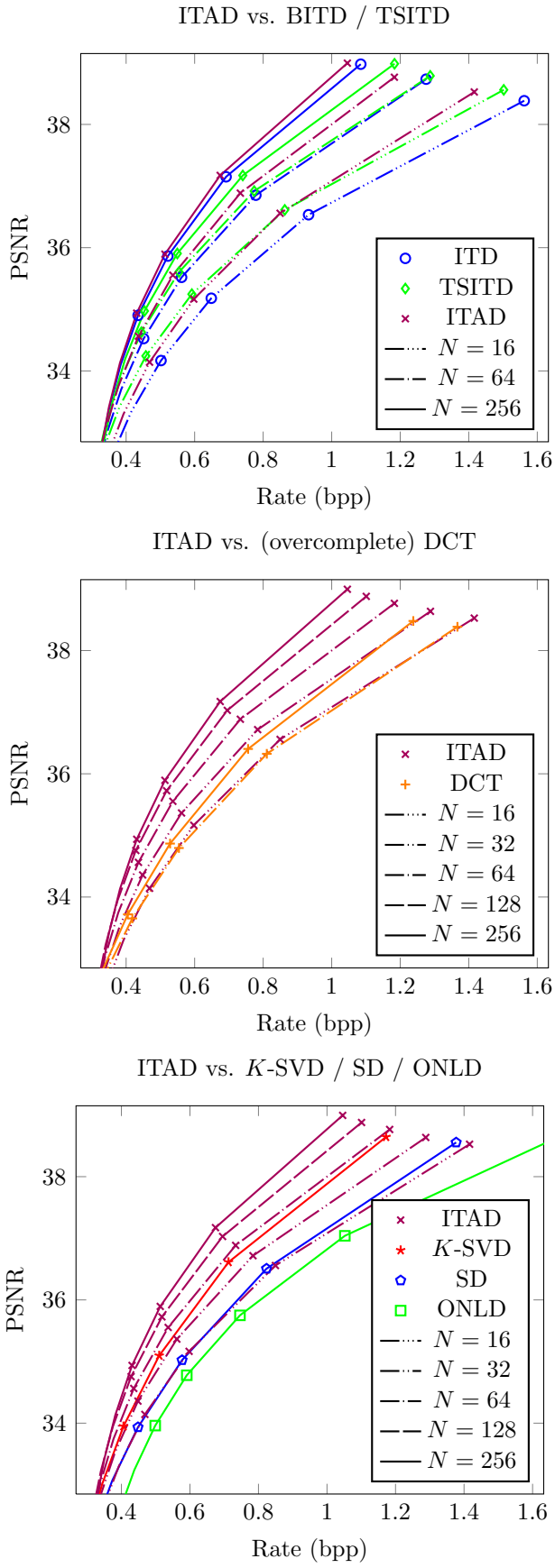

Figure 5.17: Rate-distortion curves: *(top)* ITAD vs. ITD / TSITD; *(bottom)* ITAD vs. DCT / K-SVD. N denotes the number of atoms per layer (ITD), per candidate dictionary (TSITD), per prototype dictionary (ITAD) and per fixed-dictionary (K-SVD , ONLD and SD), resulting in comparable complexities for all schemes.

Reference Dictionary	ITAD N	ITAD PSNR Gain (dB)	ITAD Complexity Gain (×)
DCT-256	32	0.09	1.45
K-SVD	64	0.12	3.96
SD	32	0.38	3.20
ONLD	16	0.26	14.90

Table 5.5: ITAD (N atoms) rate-distortion and complexity gains over reference dictionaries (256 atoms). The value of N used in the comparisons (column N) corresponds to the lowest N resulting in an ITAD PSNR advantage across all rates shown in Fig. 5.17. The *PSNR Gain* is measured at 1 bpp and the *Complexity Gain* at $L = 2.5$ and assuming an hTSITD implementation of ITAD. Complexity gains are given in Fig. 5.9 top for K-SVD / ONLD and bottom for SD. Complexities for the overcomplete DCT follow from (5.50) and the related discussion.

and ITD. The curves display a trend similar to that observed in Fig. 5.16: TSITD performance drops with increasing N relative to that of ITD due to over-fitting and again ITD outperforms TSITD for $N > 64$. ITAD, on the other hand, does not suffer from over-fitting and outperforms ITD uniformly. At a rate of 1 bpp ITAD outperforms both ITD and TSITD by at least 0.25 dB for $N = 64$ and by at least 0.18 dB for $N = 256$. This performance benefit comes at a complexity that is comparable to that of either TSITD or ITD (or even lower, if the hTSITD configuration of ITAD were used instead, *cf.* Section 5.5.6.1) and a storage footprint below that of ITD.

In the middle and bottom of Fig. 5.17 we compare ITAD to the (overcomplete) DCT and K-SVD , SD and ONLD. The curves show that, for a fixed N, ITAD offers a significant advantage relative to the other schemes. At 1 bpp, the difference is 0.9 dB relative to the best reference curve (K-SVD for $N = 256$). This performance improvement over K-SVD comes with the added benefit of reduced complexity (*cf.* bottom of Fig. 5.8). Since the other reference dictionaries (except ONLD) are designed with complexity in mind, in Table 5.5 we show that ITAD is able to outperform them in terms of rate-distortion (PSNR) and at the same time enjoy a lower complexity.

5.7.6 Experiment 3: Denoising

We now test the denoising performance of the proposed structured dictionaries in additive white Gaussian noise. We use the overcomplete-dictionary image de-

	ε/σ	κ/σ	$\lambda \cdot \sigma$
Fig. 5.18	Top	Middle	Bottom
BITD	1.15	1.15	30
TSITD	1.15	1.30	1

Table 5.6: Denoising parameters selected corresponding roughly to the maxima of the curves in Fig. 5.18.

noising method introduced by Elad *et al.* in [Elad 2006b]: the method consists of first obtaining 64 sparse approximations of each image pixel (one per each 8×8 block containing the pixel). The denoised pixel estimate is then obtained from the sum of these estimates and a weighted version of the input noisy image. The weight λ applied to the noisy image is chosen as a function of the noise variance σ (assumed known). The denoising algorithm is thus defined by λ and the training and decomposition RMSE thresholds κ and ε. To select these three parameters, we use image *peppers256* and a noise variance $\sigma = 10$. In Fig. 5.18 we illustrate the resulting curves used to select the three parameters for BITD and TSITD; the chosen parameters are indicated in Table 5.6. When choosing the first parameter ε (top of Fig. 5.18) we use $\kappa = \varepsilon$ and the best λ (chosen from a finite set). When choosing κ (middle of Fig. 5.18), we use the previously selected ε and the best λ. Both ε and κ are fixed to the previously chosen values when choosing λ (bottom of Fig. 5.18). Note that we only select parameters using $\sigma = 10$ and thus can expect improved performance if the parameters were adapted to each noise level.

In Fig. 5.19 we plot the objective denoising results for three images: *peppers256*, *house* and *Barbara*. The PSNR difference displayed is relative to the K-SVD results ($N = 256$) reported in [Elad 2006b] and uses the same experimental setup, including averaging the PSNR of the denoised image over 5 noise realizations. We note in particular that improvements over K-SVD are possible even for BITD dictionaries with $N = 64$. TSITD further improves upon BITD for 29 out of the 36 plotted points (corresponding to 3 images, 6 noise levels, and 2 values of N). For $N = 256$, TSITD outperforms BITD for 15 out of the 18 points plotted since both training and test data sets correspond to the denoised image and thus over-fitting is not an issue for large N (unlike Fig. 5.16 and fig:rxdxITAD).

In Fig. 5.20 we also present subjective results of the BITD and TSITD denoised versions of *barbara* for $\sigma = 10$. Note how the dictionary used succeeds in retaining edge details and cloth patterns.

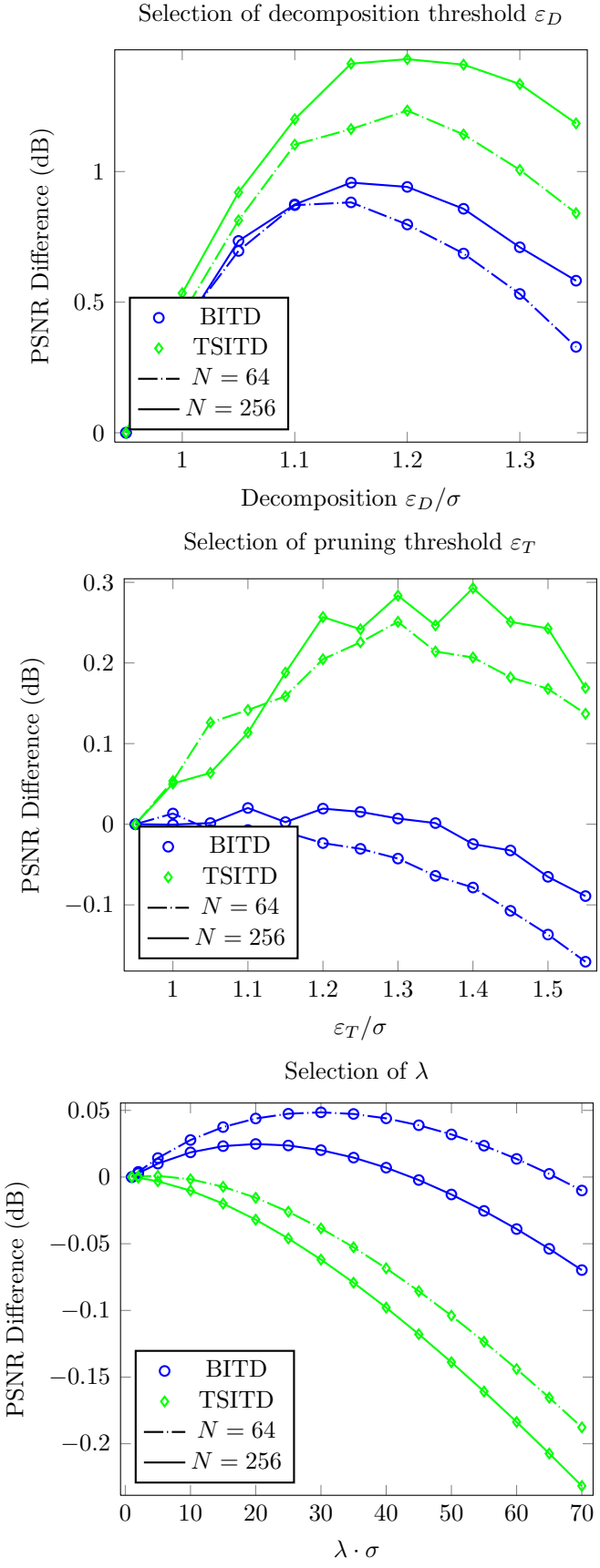

Figure 5.18: Selection of denoising parameters using image *peppers256* and $\sigma = 10$. The selected parameters are specified in Table 5.6. The PSNR difference is relative to the left-most point of each curve.

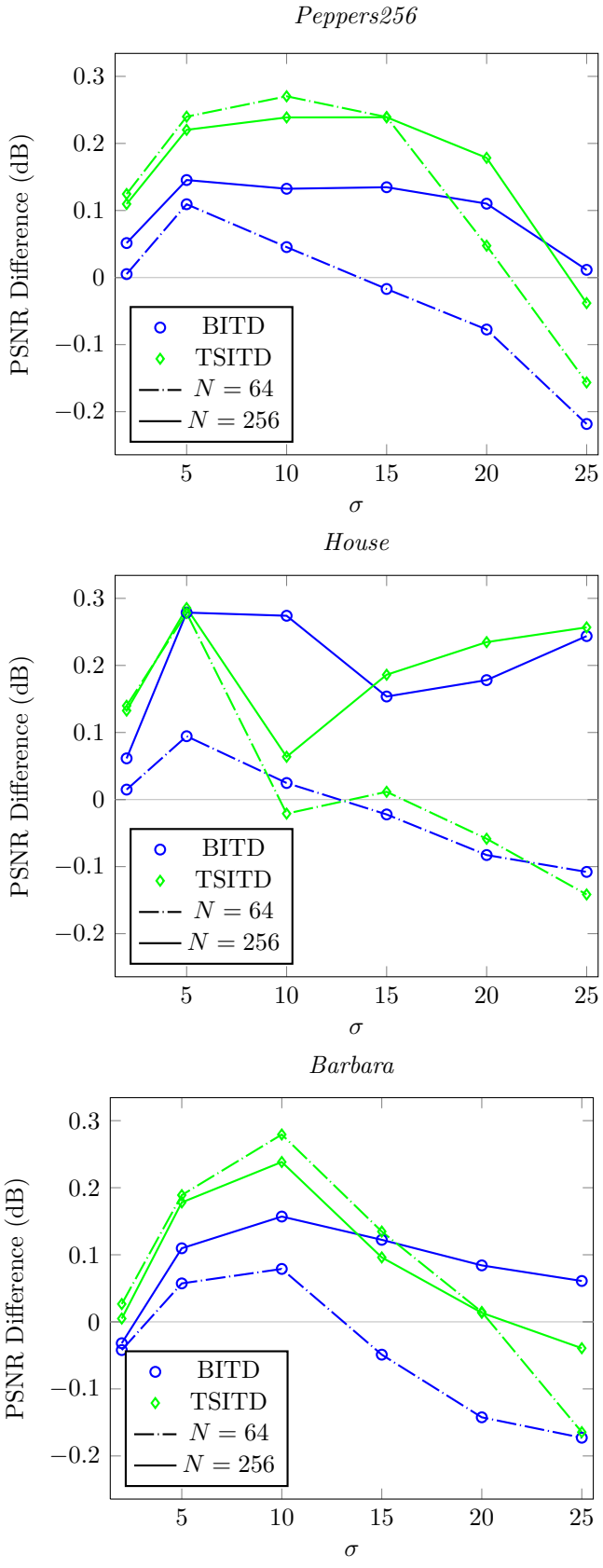

Figure 5.19: PSNR difference (dB) relative to the results reported in [Elad 2006b] for various test images.

Figure 5.20: Qualitative denoising results for image *barbara*. From top-left, clockwise: original, noisy, TSITD denoised, BITD denoised.

5.7.6.1 Comparison against ITAD

In Fig. 5.21 we plot the objective denoising results for ITAD with $N = 256$ computed on images *peppers256* and *Barbara*. We repeat the corresponding plots for both ITD and TSITD from Fig. 5.19. The PSNR difference displayed is again relative to the K-SVD results reported in [Elad 2006b] and uses the same experimental setup. Note that TSITD outperforms ITD for 9 out of the 12 points plotted since both training and test data sets correspond to the denoised image and thus over-fitting is not an issue for large N (unlike Fig. 5.16 and Fig. 5.17). Thus, for the images displayed, TSITD performs best in general, followed by ITAD, and finally by ITD.

5.8 Conclusion

We introduced a new Iteration-Tuned Dictionary (ITD), the Iteration-Tuned and Aligned Dictionary (ITAD), that succeeds in retaining the large redundancy of the Tree-Structured Iteration-Tuned Dictionary with a drastically smaller storage footprint. ITAD accomplishes this task by means of an alignment matrix that aligns the residues of the atoms of each layer. The alignment operators are trained

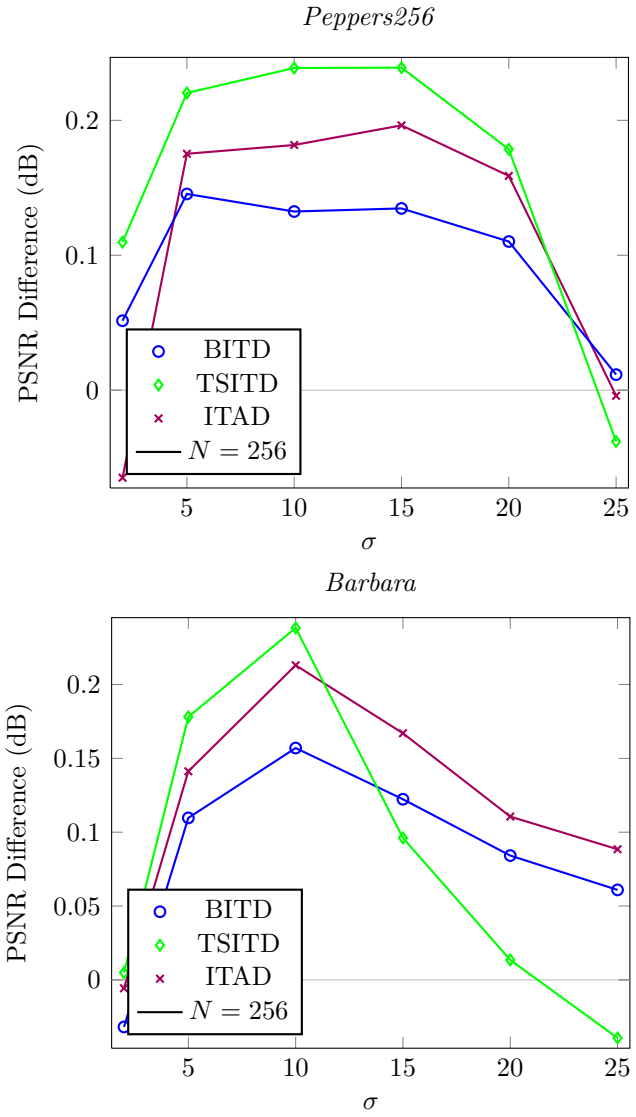

Figure 5.21: PSNR difference (dB) relative to the results reported in [Elad 2006b] for various test images.

to produce aligned residual sets that share a common principal basis further shared by the union of aligned classes. We compared the resulting structure to previous ITDs as well as to other well established dictionary structures and showed that ITAD indeed offers an advantage in terms of sparse approximation capability, as well as in two practical scenarios, compression and denoising.

Appendix 5.A Parameter selection for reference dictionaries

In this appendix we provide the experimental curves used to select the parameters specified in Table 5.3 for the reference trained dictionaries: the K-SVD dictionary of [Aharon 2006b], the Online Learned Dictionary (ONLD) of [Mairal 2010a], and the Sparse Dictionary (SD) of [Rubinstein 2010a]. The training methods of all three reference dictionaries require a decomposition RMSE value κ (*cf.* (5.8)). The SD scheme further requires an atom sparsity L_a. Note that the ITAD curves did not require any parameter tuning (κ was set to 0 for all ITAD setups in Fig. 5.16 and Fig. 5.17).

The dictionaries are trained using the facial image training dataset described at the beginning of Section 5.7, but the parameter selection curves presented in this appendix are obtained using the corresponding facial image testing dataset. Note that this favors the performance of the reference methods over ITAD, as ITAD has no knowledge of the testing set. All training is carried out using the code made available by the authors [Aharon 2006a, Mairal 2010b, Rubinstein 2010c].

We present PSNR versus sparsity curves throughout, but we note that the corresponding rate-distortion curves displayed similar relative curve positions as a function of the parameter values. In all figures, the curve corresponding to the selected parameter is the only solid (not dash-dotted) curve.

5.A.1 Selection of K-SVD parameter κ [Aharon 2006b]

We run 40 iterations of the K-SVD training algorithm. This number is greater than the number of iterations used in [Aharon 2006b, Bryt 2008, Elad 2006b]. Fig. 5.22 shows the results of experiments used to select $\kappa = 8$.

5.A.2 Selection of SD parameters L_a and κ [Rubinstein 2010a]

We use an SD dictionary of the form $\mathbf{D} = \mathbf{BA} \in \mathbb{R}^{64\times256}$ with a square sparsity matrix $\mathbf{A}$ having L_a non-zero coefficients per row and a separable overcomplete DCT base dictionary $\mathbf{B} \in \mathbb{R}^{64\times256}$. This setup is similar to that used in [Rubinstein 2010a] (they use 100 atoms). We selected $L_a = 15$ using the curves illustrated in Fig. 5.23 and $\kappa = 14$ using the curves in Fig. 5.17. We run 40 iterations of the training method, which is greater than the number used in [Rubinstein 2010a].

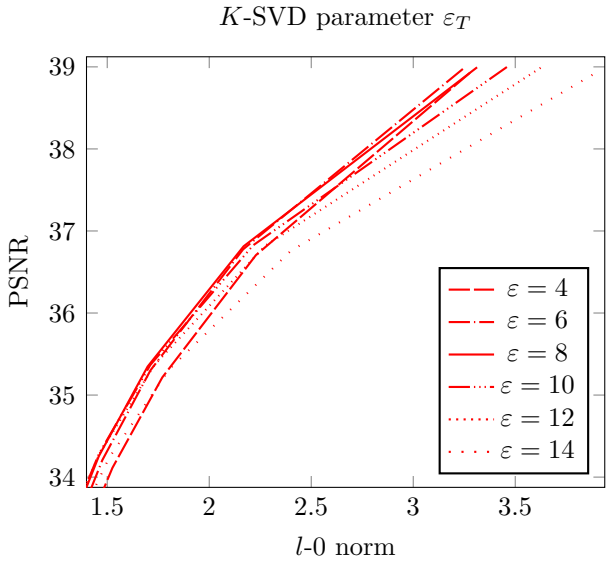

Figure 5.22: Selection of K-SVD training RMSE threshold κ (*cf.* (5.8)). From top to bottom at an l-0 of 2.2, the curves correspond to values of $\kappa = 8, 6, 10, 4, 12, 14$. The selected value used in the experiments of Fig. 5.16 and Fig. 5.17 is $\kappa = 8$.

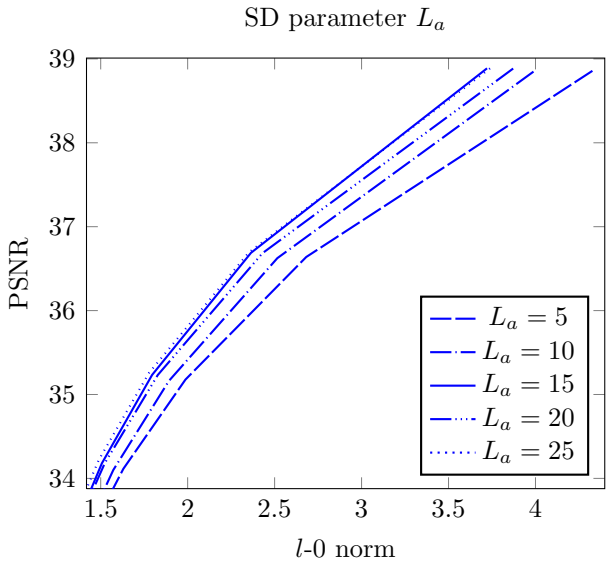

Figure 5.23: Selection of SD atom sparsity L_a. From top to bottom at an l-0 of 3.8, the curves correspond to values of $L_a = 15, 25, 20, 10, 5$. The selected value used in the experiments of Fig. 5.16 and Fig. 5.17 is $L_a = 15$.

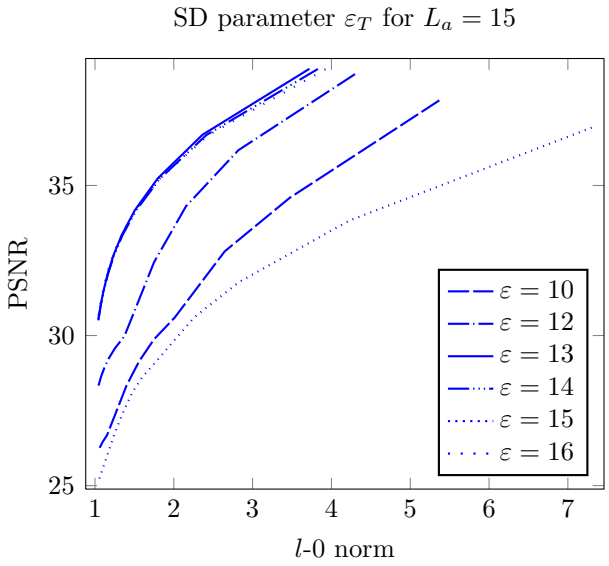

Figure 5.24: Selection of SD training RMSE threshold κ (*cf.* (5.8)) using $L_a = 15$. From top to bottom at an l-0 of 3, the curves correspond to values of $\kappa = 14, 15, 10, 13, 17, 16$. The selected value used in the experiments of Fig. 5.16 and Fig. 5.17 is $\kappa = 14$.

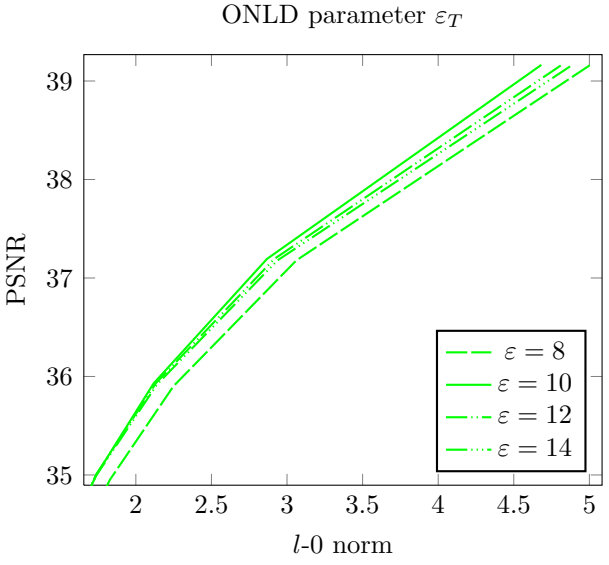

Figure 5.25: Selection of ONLD training RMSE threshold κ (*cf.* (5.8)). From top to bottom at an l-0 of 3, the curves correspond to values of $\kappa = 10, 12, 14, 8$. The selected value used in the experiments of Fig. 5.16 and Fig. 5.17 is $\kappa = 10$.

5.A.3 Selection of ONLD parameter κ [Mairal 2010a]

We used $\kappa = 10$ for the ONLD method $\mathbf{D} \in \mathbb{R}^{64\times256}$ using the curves illustrated in Fig. 5.25. We ran the method for 1000 seconds as this value is shown experimentally to be sufficient for convergence in various curves of [Mairal 2010a] for comparable training sets. Furthermore, we used a dedicated 16 core CPU (the program is multi-threaded and used all cores).

Appendix 5.B Rate-distortion analysis for codecs based on overcomplete dictionaries

Consider a general sparse representation of a signal vector $\mathbf{y} \in \mathbb{R}^d$ defined by a set of L atom-index / coefficient pairs (a, γ). The indices a specify the columns of the overcomplete dictionary used in the representation and we group these selected columns to form the selected-atoms matrix $\mathbf{S}$. We group the corresponding coefficients likewise to form the vector $\mathbf{\Gamma}$. Without quantization of coefficients, we can thus write

$$\mathbf{y} = \mathbf{S}\mathbf{\Gamma} + \mathbf{r}, \tag{5.63}$$

where $\mathbf{r}$ is the approximation residual vector. The number of (a, γ) pairs L is chosen to satisfy an RMSE threshold ε:

$$L \text{ s.t. } \sqrt{|\mathbf{r}|^2/d} \leq \varepsilon. \tag{5.64}$$

An expression for the rate and distortion of codecs using overcomplete dictionaries needs to take into account (*i*) the lossy coding of the coefficients, (*ii*) the fact that the non-orthogonality of the dictionary means that the distortion in transform space is not equal to the distortion in signal space, and (*iii*) the encoding of the selected atom indices.

To lossy coding of the coefficients will be accounted for using an additive quantization noise model $\mathbf{q}$. From (5.63), the resulting distortion X will be given by:

$$\begin{aligned} X &= \mathrm{E}\left(|\mathbf{y} - \mathbf{S}(\mathbf{\Gamma} + \mathbf{q})|^2\right) = \mathrm{E}\left(|\mathbf{r} - \mathbf{S}\mathbf{q}|^2\right) \\ &= \mathrm{E}\left(|\mathbf{r}|^2\right) - 2\mathrm{E}\left(\mathbf{r}^\mathsf{T}\mathbf{S}\mathbf{q}\right) + \mathrm{E}\left(\mathbf{q}^\mathsf{T}\mathbf{S}^\mathsf{T}\mathbf{S}\mathbf{q}\right). \end{aligned} \tag{5.65}$$

We assume that $\mathbf{q}$ is uncorrelated to $\mathbf{r}^\mathsf{T}\mathbf{S}$ and thus we can discard the second term in the above expression. Note that this assumption is not necessary for decomposition schemes (*eg.,* OMP [Pati 1993], OOMP [Rebollo-Neira 2002] and

BP [Chen 2001]) that enforce orthogonality between the residual vector $\mathbf{r}$ and the selected atoms in $\mathbf{S}$.

Regarding the third term, we note that $\mathbf{S}^\mathsf{T}\mathbf{S}$ contains unit valued diagonal entries; we let $\mathbf{S}^\mathsf{T}\mathbf{S} = \mathbf{I} + \mathbf{G}$ for some symmetric matrix $\mathbf{G}$ having zeros along its diagonal and with SVD $\mathbf{U_G}\mathbf{\Delta_G}\mathbf{U_G}^\mathsf{T}$. (Note that $\mathbf{G} = \mathbf{0}$ for dictionaries that produce orthogonal selected atom matrices, *eg.,* orthogonal transforms and unions of orthogonal bases [Sezer 2008]).

Letting $\mathbf{q_U} = \mathbf{U_G}^\mathsf{T}\mathbf{q}$, we can write

$$\mathrm{E}\left(\mathbf{q}^\mathsf{T}\mathbf{S}^\mathsf{T}\mathbf{S}\mathbf{q}\right) = \mathrm{E}\left(|\mathbf{q}|^2\right) + \mathrm{E}\left(\mathbf{q_U}^\mathsf{T}\mathbf{\Delta_G}\mathbf{q_U}\right) \tag{5.66}$$

$$= \mathrm{E}\left(|\mathbf{q}|^2\right) + \mathrm{E}\left(\sum_k \mathbf{q_U}(k)^2\mathbf{\Delta_G}(k,k)\right). \tag{5.67}$$

Note that the term $\mathbf{q_U}(k)^2$ is strictly positive with probability one. Furthermore, for non-orthogonal $\mathbf{S}$, $\mathbf{\Delta_G}(k,k)$ is strictly positive for all $1 \leq k \leq \mathrm{rank}(\mathbf{S})$. Hence this second term represents a non-orthogonality penalty.

5.B.1 Distortion as a function of quantization noise variance

In analyzing the compression potential of an overcomplete dictionary, we wish to avoid delving into the quantizer design, and for this reason it would be convenient to summarize vector $\mathbf{q}$ in (5.67) into a single quantizer parameter (*i.e.,* the quantization noise variance). In order to do so, we note first that the expectations in (5.67) involve quantities $\mathbf{q} \in \mathbb{R}^L$, $\mathbf{q_U} \in \mathbb{R}^L$ and $\mathbf{\Delta_G} \in \mathbb{R}^{L\times L}$, of dimensionality that depends on the random variable L. In order to simplify the analysis, we hence assume that, when conditioned on L, the entries of $\mathbf{q}$ (*i*) have variance σ_q^2 and (*ii*) are uncorrelated:

$$\mathrm{E}\left(\mathbf{q}\mathbf{q}^\mathsf{T} \;\middle|\; L\right) = \sigma_q^2\mathbf{I}. \tag{5.68}$$

We thus proceed by first expressing the left-hand term of (5.67) as a function of the quantization noise variance σ_q^2 as follows:

$$\mathrm{E}\left(|\mathbf{q}|^2\right) = \sum_{L\in\mathbb{Z}}\left(\int_{\mathbb{R}^L}|\mathbf{q}|^2 p(\mathbf{q}|L)\;\mathrm{d}\mathbf{q}\right)p(L) \tag{5.69}$$

$$= \sum_{L\in\mathbb{Z}}\sigma_q^2 L p(L) \tag{5.70}$$

$$= \sigma_q^2\mathrm{E}\left(L\right). \tag{5.71}$$

Likewise, we can express the right-hand term of (5.67) as a function of σ_q^2 as follows:

$$\begin{aligned}
&\mathrm{E}\left(\sum_k \mathbf{q_U}(k)^2 \mathbf{\Delta_G}(k,k)\right) \\
&\quad = \sum_{L\in\mathbb{Z}} \Bigg[\int_{\mathbb{R}^{L\times L}} \int_{\mathbb{R}^L} \sum_{k=1}^{L} \mathbf{q_U}(k)^2 \mathbf{\Delta_G}(k,k) \cdot \\
&\qquad p(\mathbf{q_U}|\mathbf{G},L) p(\mathbf{G}|L)\ \mathrm{d}\mathbf{q_U}\ \mathrm{d}\mathbf{G} \Bigg] p(L) \qquad (5.72) \\
&\quad = \sum_{L\in\mathbb{Z}} \Bigg[\int_{\mathbb{R}^{L\times L}} \sum_{k=1}^{L} \mathbf{\Delta_G}(k,k) \\
&\qquad \left(\int_{\mathbb{R}^L} |\mathbf{q_U}(k)|^2 p(\mathbf{q_U}|\mathbf{G},L)\ \mathrm{d}\mathbf{q_U} \right) \\
&\qquad p(\mathbf{G}|L)\ \mathrm{d}\mathbf{G}\ \mathrm{d}L \Bigg] p(L). \qquad (5.73)
\end{aligned}$$

The inner-most integral in this last expression evaluates to σ_q^2 as follows from the fact that the entries of $\mathbf{q_U}$ are uncorrelated and white when conditioned on $\mathbf{G}$ and L:

$$\begin{aligned}
\mathrm{E}\left(\mathbf{q_U}\mathbf{q_U}^\mathsf{T} \mid \mathbf{G}, L\right) &= \mathrm{E}\left(\mathbf{U_G}^\mathsf{T} \mathbf{q}\mathbf{q}^\mathsf{T} \mathbf{U_G} \mid \mathbf{G}, L\right) && (5.74)\\
&= \mathbf{U_G}^\mathsf{T} \mathrm{E}\left(\mathbf{q}\mathbf{q}^\mathsf{T} \mid L\right) \mathbf{U_G} && (5.75)\\
&= \sigma_q^2 \mathbf{I}. && (5.76)
\end{aligned}$$

Using this, expression (5.73) thus reduces to

$$\mathrm{E}\left(\sum_k \mathbf{q_U}(k)^2 \mathbf{\Delta_G}(k,k)\right) = \sigma_q^2 \mathrm{E}\left(\sum_{k=1}^{L} \mathbf{\Delta_G}(k)\right). \qquad (5.77)$$

Using this last expression along with (5.71), we arrive at last at the following expression for the distortion in (5.67) as a function of σ_q^2:

$$X = \mathrm{E}\left(|\mathbf{r}|^2\right) + \sigma_q^2 \cdot \left(\mathrm{E}\left(L\right) + \mathrm{E}\left(\sum_k \mathbf{\Delta_G}(k,k)\right)\right). \qquad (5.78)$$

5.B.2 Estimate of the rate

To obtain an expression for the rate, we assume that the coded stream consists of the number of atoms L followed by the L atom index / coefficient pairs (a, γ). If

we treat the coefficients γ as a random source of continuous-valued symbols, the coding rate at a given distortion σ_q^2 is known to be [Mallat 2008]

$$\mathcal{H}_d(\gamma) - \frac{1}{2}\log_2(12\sigma_q^2),$$

where $\mathcal{H}_d(\cdot)$ denotes the differential entropy. Letting $\mathcal{H}(\cdot)$ denote the entropy for finite alphabets, we can write an estimate of the rate in bits per patch for a sparse coded patch of size $\sqrt{d} \times \sqrt{d}$ as follows:

$$R = \mathcal{H}(L) + \mathrm{E}(L) \cdot \left(\mathcal{H}(a) + \mathcal{H}_d(\gamma) - \frac{1}{2}\log_2(12\sigma_q^2)\right). \tag{5.79}$$

We note that, unlike ITD frameworks, the order of the atom-index / coefficient pairs is irrelevant for traditional fixed-dictionary representations where the atom indices $a_1, \ldots, a_L$ all reference the same dictionary $\mathbf{D}$. Thus for these fixed-dictionary cases, instead of $\mathcal{H}(a)$ in (5.79), we consider also the entropy of the difference $(a_i - a_{i-1})$ after sorting the a_i such that $0 = a_0 < a_1 < \ldots < a_L$. The value for the atom-index entropy used in (5.79) will be the lowest of the two possibilities.

Both X and R in (5.78) and (5.79) require the quantization noise power σ_q^2. To obtain an estimate of this value, we note that there are two different ways to increase the rate of a coded patch:

1. by adding more (a, γ) pairs to the coded stream while keeping the quantizer fixed (*i.e.,* fixed σ_q^2) or

2. by increasing the quantization rate (*i.e.,* decreasing σ_q^2) for the same set of (a, γ) pairs.

We wish to obtain σ_q^2 values corresponding to the boundary of the rate-distortion region defined by (5.78) and (5.79). To do so, we note that, at the rate-distortion boundary, the change in distortion resulting from either method of rate control has to be equal (otherwise the distortion can be reduced for fixed rate by transferring bits between the two rate control methods). We let $(\Delta R)_k$ and $(\Delta X)_k$ (for $k = 1, 2$) denote, respectively, the change in rate under the k-th method of rate control and the corresponding change in distortion. The previously stated rate-distortion boundary constraint is thus:

$$\frac{(\Delta X)_1}{(\Delta R)_1} = \frac{(\Delta X)_2}{(\Delta R)_2}. \tag{5.80}$$

We will solve this equality for σ_q^2 to obtain the optimal σ_q^2 as a function of the decomposition threshold ε. We proceed by first computing all the entropies and

expectations appearing in (5.78) and (5.79) as a function of ε. The change in distortion and change in rate under the first method of rate control (fixed σ_q^2) will then be:

$$(\Delta X)_1 = \Delta\left(\mathrm{E}\left(|\mathbf{r}|^2\right)\right) + \sigma_q^2 \cdot \Delta\left(\mathrm{E}(L) + \sum_k \mathrm{E}\left(\boldsymbol{\Delta}_{\mathbf{G}}(k,k)\right)\right), \tag{5.81}$$

$$(\Delta R)_1 = \Delta\left(\mathcal{H}(L) + \mathrm{E}(L) \cdot \left(\mathcal{H}(a) + \mathcal{H}_d(\gamma)\right)\right) - \frac{1}{2}\Delta\left(\mathrm{E}(L)\right) \cdot \log_2(12\sigma_q^2). \tag{5.82}$$

To obtain the corresponding quantities under the second method of rate control, we consider adding on average one more quantization bit to each of the y coefficients. This will have the effect of halving the quantizer bin size, effectively reducing the quantization noise power from σ_q^2 to $\sigma_q^2/4$ (a difference of $(3/4)\sigma_q^2$):

$$(\Delta X)_2 = \frac{3}{4}\sigma_q^2 \cdot \left(\mathrm{E}(L) + \sum_k \mathrm{E}\left(\boldsymbol{\Delta}_{\mathbf{G}}(k,k)\right)\right), \tag{5.83}$$

$$(\Delta R)_2 = L_y. \tag{5.84}$$

Chapter 6

Image Compression Using the Iteration-Tuned and Aligned Dictionary

6.1 Introduction

Recent research effort has been dedicated to learning dictionaries which would thus be adapted to a signal class for the purpose of image compression. By using a learned dictionary, the image encoder can benefit from the ensuing greater compressibility of the considered signal class. An example of this approach is embodied in the facial image codec based on the K-SVD dictionary introduced by Bryt and Elad [Bryt 2008]. Their approach nonetheless employs a piecewise-affine warping of the face that ensures that the various facial features coincide with those of a prespecified face template. Each block of the face template (corresponding roughly to a facial feature such as the nose) thus defines a class of signals that is then represented with a corresponding K-SVD dictionary. It is important to note that the compressibility of the image blocks in that approach relies, to a large extent, not on the K-SVD dictionary but rather on the affine warping procedure. This warping procedure in turn increases the codec complexity and is further sensitive to image variations encountered in practical scenarios (*eg.,* in lighting conditions, pose and particularities of the subject).

Another example of an image compression system based on trained overcomplete dictionaries is that developed by Sezer *et al.* [Sezer 2008]. Their dictionary structure consists of a concatenation of orthogonal bases. A single one of this basis is selected to encode any given image block of fixed size. This approach has the advantage that it reduces the atom-index coding overhead, yet it also reduces the effective size of the dictionary.

In this chapter we introduce a new image codec based on the Iteration-Tuned and Aligned Dictionary (ITAD) of Chapter 5. The ITAD structure is a recently introduced learned structured dictionary that has been shown to outperform other learned overcomplete dictionaries. ITAD is a particular case of the more general Iteration-Tuned Dictionary (ITD) framework consisting of a dictionary structure

adapted to the iterative nature of greedy pursuit algorithms such as those of the matching pursuit family [Mallat 1993, Pati 1993, Rebollo-Neira 2002]. The proposed codec uses the ITAD transform to encode the mean-removed image blocks, while the block-mean is encoded using a common DPCM-based arrangement. The ITAD transform coefficients are encoded using a simple uniform quantizer / entropy encoder combination, while the atom indices are encoded using a fixed-length code. We further introduce a new global method for jointly selecting the sparsity of the image blocks based on a rate-distortion criterion. The proposed ITAD codec is shown to outperform the JPEG and JPEG2000 compression standards in quantitative and qualitative evaluations.

We begin in the following section (Section 6.2) by first reviewing the ITAD structure that is at the core of our codec. Then, in the subsequent section (Section 6.3), we introduce the codec structure and, in particular, develop a global (image-wide) rate-distortion criterion that allows us to better select the sparsity of each of the component image blocks. We then carry out both quantitative and qualitative evaluations of our codec (Section 6.4), comparing it against both JPEG and JPEG2000 standards. Finally, we provide some concluding remarks in the conclusion section (Section 6.5).

6.2 Notation

For ease of analysis, throughout our discussion we we will use the TSITD representation of ITAD. Hence the structure is assumed to consist of K candidate dictionaries $\mathbf{D}_k \in \mathbb{R}^{N\times d}$ organized into nodes k of a tree-structure. The l-th level of the structure consists of nodes $k \in \mathcal{L}_l \subset \{1,\dots,K\}$. As before, we let $k_1 = 1$ denote the root node, the sequence $k_1, k_2, \dots, k_l$ denote a sequence of descendant nodes, and $\mathbf{d}^j \in \mathbf{D}_{k_j}$ a corresponding sequence of descendant atoms. Hence we can arrange the atoms selected up to the i-th MP iteration into the *selected-atoms matrix*

$$\mathbf{S}^i = \begin{bmatrix}\mathbf{d}^1 & \dots & \mathbf{d}^i\end{bmatrix} \tag{6.1}$$

and, along with the corresponding MP coefficients vector $\mathbf{\Gamma}^i = \begin{bmatrix}\gamma_1 & \dots & \gamma_i\end{bmatrix}^\mathsf{T}$, we can write the i-th approximation of an input signal $\mathbf{y}$ as

$$\hat{\mathbf{y}}^i = \mathbf{S}^i\mathbf{\Gamma}^i. \tag{6.2}$$

The choice of atoms and coefficients calculation can be expressed in signal space as follows:

$$\mathbf{d}^i = \underset{\mathbf{d}\in\mathbf{D}_i}{\operatorname{argmax}} \left|(\mathbf{d})^\mathsf{T} \cdot \mathbf{r}^{i-1}\right|, \tag{6.3a}$$

$$\gamma_i = (\mathbf{d}^i)^\mathsf{T}\mathbf{r}^{i-1}. \tag{6.3b}$$

The approximation residue $\mathbf{r}^i$ will satisfy

$$\mathbf{y} = \hat{\mathbf{y}}^i + \mathbf{r}^i. \tag{6.4}$$

In our subsequent discussion we will also take advantage of the property

$$(\mathbf{S}^{i-1})^{\mathsf{T}}\,\mathbf{D}_i = \mathbf{0} \tag{6.5}$$

outlined in Corollary 5.5.1, Pg. 92.

Recall (Section 5.2, Pg. 80) that the notation $\mathbf{d}^j$ is shorthand notation for $\mathbf{d}_{k_j a_j}$, *i.e.,* $\mathbf{d}^j$ is an atom that is column a_j of dictionary $\mathbf{D}_{k_j}$. Note that, given the TSITD tree-structured atom selection constrain, the node index k_j is uniquely specified by the sequence of ancestor atoms:

$$k_j \iff \{a_1, \ldots, a_j\}. \tag{6.6}$$

Our strategy will be to code the atom indices, and thus we use

$$a_j = index(\mathbf{d}^j) \tag{6.7}$$

to denote the atom index of a given atom.

We will employ a block-coding strategy wherein an image is split into B non-overlapping blocks. When necessary, we will hence use a subscript b notation to differentiate quantities belonging to different blocks of the image. Hence $\mathbf{d}_b^i$, a_{ib} and γ_{bi} will denote, respectively, the i-th selected atom for block b, its atom index, and the corresponding coefficient.

6.3 The proposed image codec

We now present the proposed ITAD image codec that is the main contribution of this chapter. The encoder uses the ITAD transform to compress the mean-removed component of image blocks taken over a regular grid; we refer to the mean of a block and a block's mean-removed version as its *DC* and *AC* components, respectively. In order to determine the sparsity of each individual block, we derive a global sparsity criterion that accounts for the global image rate and distortion.

6.3.1 Block slicer and AC / DC splitter

The block diagram in Fig. 6.1 illustrates the major components of our proposed image encoder. The first step of the process consists of slicing a given input image $\mathbf{J}$ into non-overlapping $\sqrt{d} \times \sqrt{d}$ blocks using a regular grid. Vector $\mathbf{z}_b \in \mathbb{R}^d$

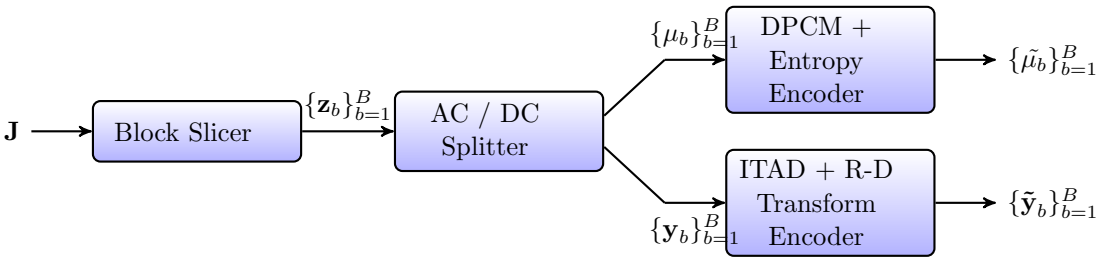

Figure 6.1: The ITAD-based image codec: The input image $\mathbf{J}$ is split into disjoint blocks $\mathbf{z}$. The DC and AC components μ and $\mathbf{y}$ of each block $\mathbf{z}$ are then encoded separately. The operation of the bottom-right AC encoding block *ITAD / R-D Transform Coding* is described in Fig. 6.2 and Fig. 6.3.

denotes the vectorized version of one of these blocks, and thus the input image $\mathbf{J}$ is represented at the output of the block-slicer by an ordered set these vectors:

$$\mathbf{J} \iff \{\mathbf{z}_b\}_{b=1}^B. \tag{6.8}$$

As illustrated in Fig. 6.1 each block $\mathbf{z}_b$ is subsequently split into a DC and an AC component, and the resulting AC and DC streams are encoded separately. We let μ_b denote the mean component of each block, given by

$$\mu_b = \frac{1}{d}\sum_{j=1}^{d} \mathbf{z}_b[j]; \tag{6.9}$$

$\mathbf{y}_b$ will denote the AC (mean-removed) version obtained by subtracting μ_b from each entry of $\mathbf{z}_b$.

6.3.2 DPCM coding of DC components

The DC components are encoded using an approach similar to that of various block-based codecs including the JPEG image coder [Wallace 1991]. The approach exploits the spatial correlation of the DC coefficients by means of a Differential Pulse Code Modulation (DPCM) applied to the DC stream $\{\mu_b\}_{b=1}^B$. By convention, we order the DC stream using row-wise rastering and reversal of order from line to line to better exploit the spatial correlation of the DC coefficients. The DPCM symbols are subsequently encoded using an entropy encoder. The resulting coded version of the DC values is denoted $\tilde{\mu}_b$.

Note that, following removal of the DC components, all AC blocks $\mathbf{y}_b$ are contained in the orthogonal complement of the all-ones vector $\mathbf{1}$. The fact that AC and DC components are orthogonal means that the distortion at the output

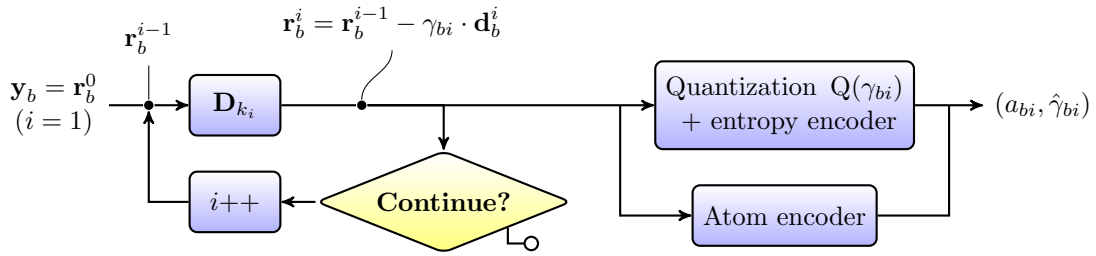

Figure 6.2: The ITAD Block Coder (BC): Sparse decomposition at the i-th ITD-MP iteration and subsequent encoding of the atom-index / coefficient pair.

of the codec is the sum of the distortions of the two components. For this reason, we can disregard the DC component in the subsequent discussion on AC encoding and instead focus on minimizing the AC distortion.

6.3.3 ITAD-based transform coding of AC components

The remaining AC component blocks $\mathbf{y}_b$ are compressed using an encoder based on the ITAD transform. The proposed encoder distributes bits to the various AC blocks $\mathbf{y}_b$ using a rate / distortion criterion that we will derive shortly. Throughout the discussion we assume that the ITAD structure has been trained using a large number of example AC blocks taken from a set of training images according to the algorithm described in Section 5.6.3.

6.3.3.1 ITAD sparse decomposition of AC blocks

In the left-hand side of Fig. 6.2 we illustrate the ITAD atom / coefficient selection process. For ease of analysis, we have used the signal-space representation of the residue vectors $\mathbf{r}_b^i$ and of the ITAD candidate dictionaries $\mathbf{D}_{k_i}$. The sparse decomposition of the signal blocks $\mathbf{y}_b$ proceeds iteratively, with each iteration of the loop selecting one atom $\mathbf{d}_b^i$ with index a_{bi} and a corresponding coefficient γ_{bi} from the i-th ITAD layer and accordingly producing the i-th residual vector $\mathbf{r}_b^i$.

6.3.3.2 Quantization of the coefficients

The coefficients γ_{bi} selected using the ITAD structure need to be quantized in order to produce a compact representation of each $\mathbf{y}_b$. We use $\hat{\gamma}_{bi}$ to denote the quantized version of γ_{bi} and, accordingly, we let $\tilde{\mathbf{\Gamma}}_b^i = \begin{bmatrix} \hat{\gamma}_{b1} & \dots & \hat{\gamma}_{bi} \end{bmatrix}$ denote the quantized version of the coefficients vector $\mathbf{\Gamma}_b^i$.

The coefficient encoding strategy consists of one uniform scalar quantizer common to all layers i of the ITAD structure:

$$\hat{\gamma}_{bi} = \mathrm{Q}(\gamma_{bi}). \tag{6.10}$$

The quantized symbols $\hat{\gamma}_{bi}$ are then encoded using an entropy encoder unique to each layer i.

Previous work [Goyal 1997] on quantization of coefficients from overcomplete transforms has considered adding the quantization step in the atom selection process illustrated in the left-hand side loop of Fig. 6.2 by substituting $\hat{\gamma}_{bi}$ in place of $\hat{\gamma}_{bi}$ when building the residue $\mathbf{r}^i$ at the input of the following iteration. For the case of general overcomplete transforms, this approach produces better sparse representations under quantization because the latter iterations of the decomposition process consider the non-orthogonality between residue and selected atom(s) resulting from the finite precision of the quantized projection coefficients $\hat{\gamma}_i$.

However, as in the case of general orthogonal transforms, the orthogonality of the ITAD selected-atoms matrices implies that ITAD sparse decompositions do not require this extra consideration at the encoder. To illustrate this, we consider the operation at the i-th decomposition iteration in Fig. 6.2 (for simplicity we drop the block index $b0$: Note first that at the d-th decomposition iteration (where d is the signal dimension) the approximation $\hat{\mathbf{y}}^d$ of $\mathbf{y}$ is exact,

$$\mathbf{y} = \mathbf{S}^d\mathbf{\Gamma}^d. \tag{6.11}$$

Then for any $i < d$ we can expand (6.4) as

$$\mathbf{y} = \hat{\mathbf{y}}^i + \mathbf{r}^i \tag{6.12a}$$

$$= \mathbf{S}^i\mathbf{\Gamma}^i + \bar{\mathbf{S}}^{i+1}\bar{\mathbf{\Gamma}}^{i+1} \tag{6.12b}$$

where $\bar{\mathbf{S}}^{i+1} = \begin{bmatrix}\mathbf{d}^{i+1} & \dots & \mathbf{d}^d\end{bmatrix}$ and $\bar{\mathbf{\Gamma}}^{i+1} = \begin{bmatrix}\gamma_{i+1} & \dots & \gamma_d\end{bmatrix}^\mathsf{T}$ contain the atoms and coefficients corresponding to layers $(i+1), \dots, d$.

We denote the i-th approximation (6.2) under coefficient quantization as

$$\tilde{\mathbf{y}}^i = \mathbf{S}^i\tilde{\mathbf{\Gamma}}^i. \tag{6.13}$$

Accordingly, the i-th residual vector $(1 \leq i < d)$ under the influence of quantization can be expressed using (6.4) as

$$\tilde{\mathbf{r}}^i = \mathbf{y} - \tilde{\mathbf{y}}^i \tag{6.14}$$

$$= \mathbf{S}^d\mathbf{\Gamma}^d - \mathbf{S}^i\tilde{\mathbf{\Gamma}}^i \tag{6.15}$$

$$= \mathbf{S}^i(\mathbf{\Gamma}^i - \tilde{\mathbf{\Gamma}}^i) + \bar{\mathbf{S}}^{i+1}\bar{\mathbf{\Gamma}}^{i+1} \tag{6.16}$$

We further define the quantization error vector $\mathbf{q}^i \triangleq (\mathbf{\Gamma}^i - \tilde{\mathbf{\Gamma}}^i)$ and, using $\mathbf{r}^i = \bar{\mathbf{S}}^{i+1}\bar{\mathbf{\Gamma}}^{i+1}$ from (6.12), we re-write (6.16) as

$$\tilde{\mathbf{r}}^i = \mathbf{S}^i\mathbf{q}^i + \mathbf{r}^i. \tag{6.17}$$

Hence we can express the inner-product (for all atoms) appearing in the ITAD MP decomposition operation (6.3) as follows

$$(\tilde{\mathbf{r}}^{i-1})^\mathsf{T}\,\mathbf{D}_i = \left(\mathbf{S}^{i-1}\mathbf{q}^{i-1} + \mathbf{r}^{i-1}\right)^\mathsf{T}\,\mathbf{D}_i \tag{6.18}$$

$$= (\mathbf{r}^{i-1})^\mathsf{T}\,\mathbf{D}_i, \tag{6.19}$$

where we drop the term $(\mathbf{q}^{i-1})^\mathsf{T}\,(\mathbf{S}^{i-1})^\mathsf{T}\,\mathbf{D}_i$ in the second equality following (6.5). Thus we have shown that using either $\tilde{\mathbf{r}}^i$ or $\mathbf{r}^i$ in the ITAD decomposition loop in Fig. 6.2 produces the same result.

6.3.3.3 Atom encoding

The decoder needs to know which atoms $\mathbf{d}_b^i$ have been chosen at the encoder, and thus the encoder needs to transmit the index a_{bi} of each atom selected. The order of these atom indices is important, since, following (6.6), the ordered set $\{a_{b1}, \ldots a_{L_b}\}$ (L_b denotes the number of atoms chosen for block b) fully specifies the atoms chosen across all L_b layers of the ITAD structure.

We use a fixed length code to encode each atom index a_{bi}, since we have observed that there is only a small gain resulting from using an entropy code for the atom indices. Assuming that the i-th ITAD dictionary contains N atoms, each atom index thus incurs a rate penalty of

$$\mathrm{R}\,(a_i) = \log_2(N). \tag{6.20}$$

At the output of the encoder, each image block $\mathbf{y}$ will be represented by an ordered set of atom-index / quantized coefficient pairs which we denote as

$$\mathcal{Y}_b^L = \{(a_{bi}, \hat{\gamma}_{bi})\}_{i=1}^{L_b}. \tag{6.21}$$

6.3.4 Global rate-distortion criterion for block sparsity selection

The encoder needs to select the number of atom-index/coefficient pairs L_b used to represent each block (*i.e.,* the block sparsity), and we now address this issue. The sparsity selection problem can be expressed formally as follows:

$$\underset{L_1,\ldots,L_B \in \{1,\ldots,d\}}{\operatorname{argmin}} \sum_{b=1}^{B} |\mathbf{y}_b - \tilde{\mathbf{y}}_b^{L_b}|^2 \text{ s.t. } \sum_{b=1}^{B} \mathrm{R}\left(\mathcal{Y}_b^{L_b}\right) \leq \Psi, \tag{6.22}$$

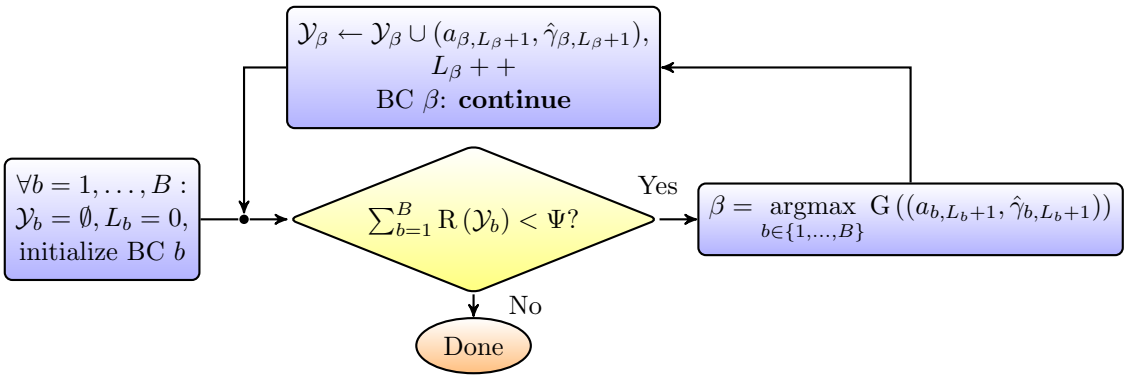

Figure 6.3: Rate-distortion transform coding using ITAD. Each of the image blocks $\mathbf{y}_b$ is assumed to have a corresponding Block Coder (BC) (illustrated in Fig. 6.2).

where $\tilde{\mathbf{y}}_b^{L_b}$ denotes the signal vector reconstructed using L_b quantized coefficients $\hat{\gamma}_{bi}$, Ψ is the allocated image rate and the $\mathrm{R}(\cdot)$ operator denotes coding rate.

The above stated problem is difficult to solve exactly and would likely require an intractable combinatorial approach. Hence we use the following strategy to approximate the solution: We build the reconstructed image by first initializing the approximations of all blocks to zero,

$$\tilde{\mathbf{y}}_b^0 = \mathbf{0},\ \mathcal{Y}_b = \emptyset, L_b = 0, \forall b. \tag{6.23}$$

We then select one image block β at a time and improve its approximation $\tilde{\mathbf{y}}_\beta^{L_\beta}$ by adding a single atom / coefficient pair to its representation $\mathcal{Y}_\beta$, repeating the block selection and improvement process as long as the rate constraint in (6.22) has not been crossed. The block β chosen for improvement will be the one offering the largest reduction in approximation error versus rate increase. Letting, $(a_{b,i}, \hat{\gamma}_{b,i})$ denote the i-th atom-index / quantized coefficient pair of block number b, we can express this as

$$\beta = \underset{b}{\operatorname{argmax}} \frac{|\mathbf{y}_b - \tilde{\mathbf{y}}_b^{L_b}|^2 - |\mathbf{y}_b - \tilde{\mathbf{y}}_b^{L_b+1}|^2}{\mathrm{R}\left((a_{b,L_b+1}, \hat{\gamma}_{b,L_b+1})\right)}, \tag{6.24}$$

where the denominator contains the rate of the atom-index / quantized coefficient pair.

We now simplify the numerator of (6.24) with the help of the (orthogonal) selected-atoms matrix $\mathbf{S}^i$ (*cf.* (6.1)) and the coefficients vector $\mathbf{\Gamma}^i = \begin{bmatrix} \gamma_1 & \ldots & \gamma_i \end{bmatrix}^\mathsf{T}$. At layer $i = d$ (d the input signal dimension), $\mathbf{S}^d$ is square and hence block $\mathbf{y}$ (we drop the block index b for notational convenience) is given exactly by:

$$\mathbf{y} = \mathbf{S}^d\mathbf{\Gamma}^d = \left[\mathbf{S}^L \mid \bar{\mathbf{S}}^{L+1}\right] \left[(\mathbf{\Gamma}^L)^\mathsf{T} \mid (\bar{\mathbf{\Gamma}}^{L+1})^\mathsf{T}\right]^\mathsf{T} \tag{6.25}$$

where $\bar{\mathbf{S}}^{L+1}$ contains the atoms from layers $(L+1), \ldots, d$ and $\bar{\mathbf{\Gamma}}^{L+1}$ the corresponding coefficients. Using (6.25) and $\tilde{\mathbf{y}}^L = \mathbf{S}^L \tilde{\mathbf{\Gamma}}^L$, we write

$$\begin{aligned} |\mathbf{y} - \tilde{\mathbf{y}}^L|^2 &= \left| \left[\mathbf{S}^L \mid \bar{\mathbf{S}}^{L+1} \right] \left[(\mathbf{\Gamma}^L)^\mathsf{T} \mid (\bar{\mathbf{\Gamma}}^{L+1})^\mathsf{T} \right]^\mathsf{T} - \mathbf{S}^L \tilde{\mathbf{\Gamma}}^L \right|^2 \\ &= \left| \left[\mathbf{S}^L \mid \bar{\mathbf{S}}^{L+1} \right] \left[(\mathbf{\Gamma}^L - \tilde{\mathbf{\Gamma}}^L)^\mathsf{T} \mid (\bar{\mathbf{\Gamma}}^{L+1})^\mathsf{T} \right]^\mathsf{T} \right|^2 \qquad (6.26) \\ &= \left| \mathbf{\Gamma}^L - \tilde{\mathbf{\Gamma}}^L \right|^2 + \left| (\bar{\mathbf{\Gamma}}^{L+1})^\mathsf{T} \right|^2, \qquad (6.27) \end{aligned}$$

where we used the orthogonality of $\mathbf{S}^d = \left[\mathbf{S}^L \mid \bar{\mathbf{S}}^{L+1} \right]$ to go from (6.26) to (6.27). When subtracting two expressions of the form (6.27) for sparsities L_b and $L_b + 1$, as done in the numerator of (6.24), only a single squared coefficient will remain from each of the two squared norm terms in (6.27):

$$|\mathbf{y}_b - \tilde{\mathbf{y}}_b^{L_b}|^2 - |\mathbf{y}_b - \tilde{\mathbf{y}}_b^{L_b+1}|^2 = (\hat{\gamma}_{b,L_b+1} - \gamma_{b,L_b+1})^2 + \gamma_{b,L_b+1}^2. \qquad (6.28)$$

This last result can be used directly in place of the numerator in (6.24) to define the block selection expression:

$$G((a_{b,L_b+1}, \gamma_{b,L_b+1})) = \frac{(\hat{\gamma}_{b,L_b+1} - \gamma_{b,L_b+1})^2 + \gamma_{b,L_b+1}^2}{\mathrm{R}\left((a_{b,L_b+1}, \hat{\gamma}_{b,L_b+1})\right)}, \qquad (6.29)$$

On the left-hand column of Fig. 6.4 we illustrate the performance of the proposed rate-distortion based global sparsity criterion above described by plotting, from top to bottom, (*i*) a reconstructed image and its (*ii*) atom-distribution map and (*iii*) RMSE per-block map.On the right-hand column of the same figure we show the same three graphics obtained when using a common sparsity-selection approach based on an RMSE threshold:

$$\underset{L}{\operatorname{argmin}}\, L \text{ s.t. } |\mathbf{y} - \hat{\mathbf{y}}^L|^2 \leq d \cdot \varepsilon^2. \qquad (6.30)$$

Note that the proposed sparsity criterion distributes atoms more uniformly than the scheme based on (6.30), while the resulting RMSE per block of the proposed scheme is less uniform. For the same coding rate (0.5 bpp), the proposed scheme offers an advantage of 0.63 dB.

6.3.5 Bit-stream format

In Fig. 6.5 we propose a simple bit-stream format for the ITAD codec discussed above. The structuring of the bit-stream is carried out using a one-bit end-of-block (`EOB`) flag. Thus we include this flag when calculating the rate of an $(a_{bi}, \hat{\gamma}_{bi})$ pair to compute the block selection criterion in (6.29),

$$\mathrm{R}\left((a_{bi}, \hat{\gamma}_{bi})\right) = \mathrm{R}\left(a_{bi}\right) + \mathrm{R}\left(\hat{\gamma}_{bi}\right) + \mathrm{R}\left(\texttt{EOB}\right), \qquad (6.31)$$

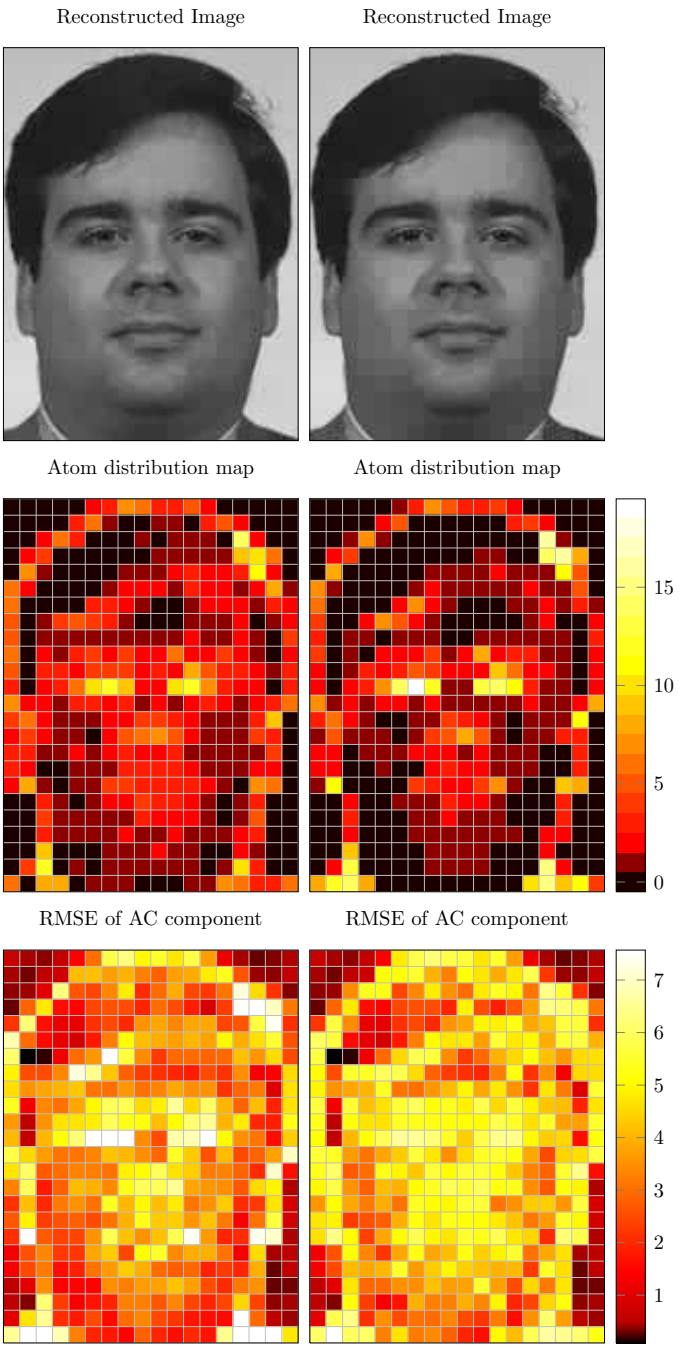

Figure 6.4: An example of *(top)* the reconstructed image *(middle)* the encoder distribution of atoms-per-block and *(bottom)* the RMSE of the AC component of each block when using sparsity-selection criteria based on *(left)* a global rate-distortion scheme as illustrated in Fig. 6.3 or *(right)* a constant RMSE threshold ε as in (6.30). Both setups correspond to 0.5 bpp; the resulting PSNR and mean block-sparsity are: *(left)* 36.40 dB and 2.07 atoms; *(right)* 35.77 dB and 1.92 atoms. The original image used is shown in the second row of Fig. 6.9.

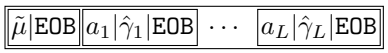

Figure 6.5: The bit-stream used to represent the set $\mathcal{Y}$ of $(a_i, \hat{\gamma}_i)$ pairs defining the approximation $\tilde{\mathbf{y}}$ of an image block and the DPCM coded DC component $\tilde{\mu}$.

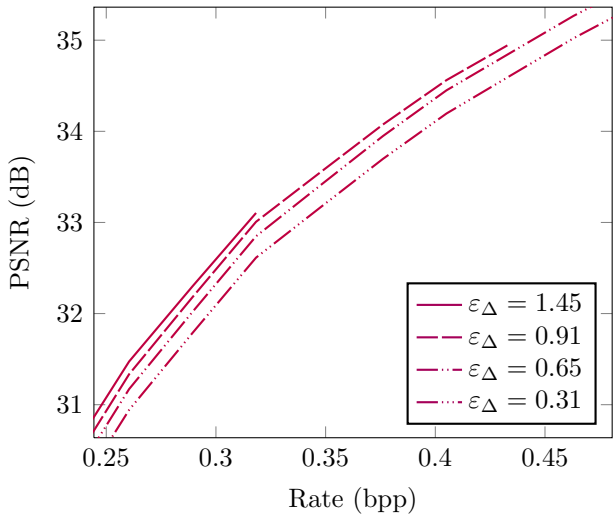

Figure 6.6: Experimental rate-distortion curves for the ITAD-codec as a function of the quantization step. The values displayed are averaged over all 100 test images.

where $\mathrm{R}(a_{bi})$ is given in (6.20), $\mathrm{R}(\hat{\gamma}_{bi})$ is the length in bits of the codeword representing $\hat{\gamma}_{bi}$ (we assume non-adaptive codebooks that are pre-computed and fixed), and $\mathrm{R}(\texttt{EOB}) = 1$. The corresponding rate for the AC component of an image block b is given by $\mathrm{R}(\mathcal{Y}) = \sum_{i=1}^{L_b} \mathrm{R}((a_{bi}, \hat{\gamma}_{bi}))$.

6.4 Results

In the current section we evaluate the proposed image codec in compression of the class of facial images, comparing it against the state-of-the art in image compression. Our evaluations show that our codec can offer an advantage that can be as high as several dBs for certain coding rates.

6.4.1 Experimental setup

We use an image dataset consisting of frontal pose images of 764 different subjects: 664 of these images are used to train the image codec while the remaining (mutually exclusive) 100 images are used as a test set. The images are high-resolution uncompressed images taken from the FERET image dataset [Phillips 2000], man-

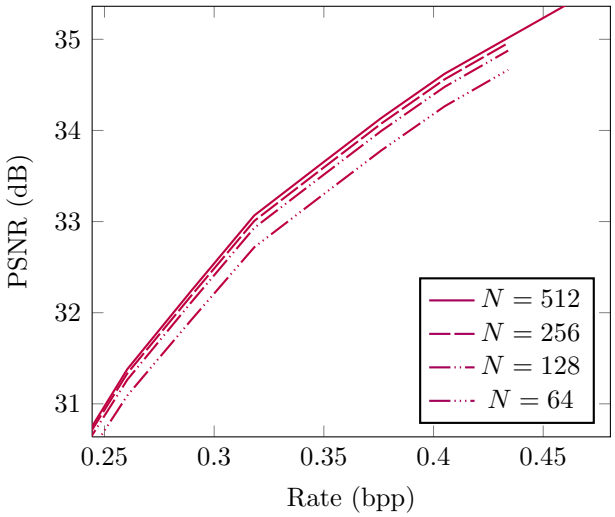

Figure 6.7: Experimental rate-distortion curves for the ITAD-codec as a function of the ITAD prototype dictionary size N (constant for all layers). The values displayed are averaged over all 100 test images.

ually cropped to focus on the face and re-sized to a uniform size of 192×144 pixels.

As a comparison reference we use the state-of-the-art JPEG2000 [Adams 2005] image encoder and its widely used predecessor, the JPEG encoder.

6.4.2 ITAD codec construction

To construct the ITAD codec above described, we begin by first extracting non-overlapping image blocks $\mathbf{z}$ from all 664 training images using a regular grid. To test the influence of the block size on the results, we build codecs using three different block sizes: $8 \times 8, 12 \times 12$ and 16×16. This produces, respectively, training sets containing $2.9 \times 10^5, 1.27 \times 10^5$ and 0.71×10^5 vectors.

For each of these training sets, we first extract all the means μ and quantize them using a uniform quantizer with unit quantization step and dynamic range between 0 and 255. The resulting streams $\{\tilde{\mu}_b\}_{b=1}^{B}$ are used to form DPCM symbols subsequently used to train the DC entropy encoder (*cf.* Fig. 6.1). The mean-removed version $\mathbf{y}$ of the training set is then used to construct the ITAD transform required for AC encoding.

For simplicity, the AC coefficients γ_{bi} are quantized using a single uniform quantizer that is common across all layers. The AC quantization step Δ is defined

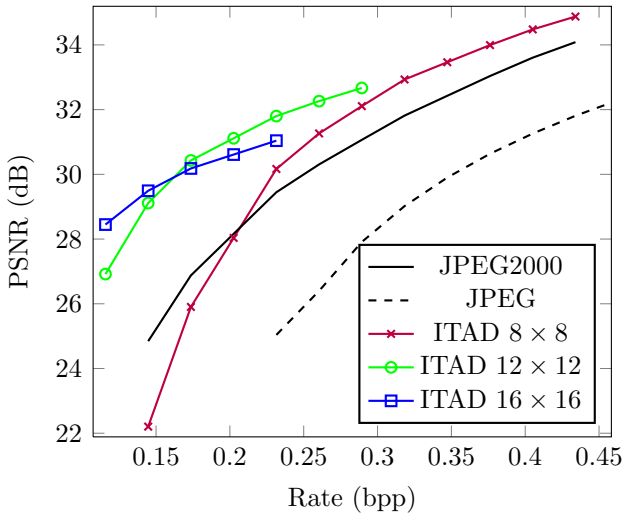

Figure 6.8: Experimental rate-distortion curves for the ITAD-codec using various block sizes versus JPEG2000 and JPEG. The values displayed are averaged over all 100 test images.

in terms of the per-pixel RMSE

$$\varepsilon_\Delta = \left(\frac{1}{b^2} \int_{-\Delta/2}^{\Delta/2} x^2 p(x)\, \mathrm{d}x \right)^{\frac{1}{2}}$$

resulting when the quantization error is uniformly distributed (*i.e.*, $p(x) = 1/\Delta$):

$$\Delta = \varepsilon_\Delta \cdot b \cdot \sqrt{12}, \tag{6.32}$$

where the block is assumed to be $b \times b$. While we use an AC quantizer that is common to all layers, the subsequent entropy encoder is layer dependent. We use the same set of per-layer encoders i for all test images; each encoder i is trained using the quantization symbols $\hat{\gamma}_{bi}$ of the training set at the corresponding layer i.

We use Huffman codes [Huffman 1952] for entropy encoding of both AC and DC symbols.

6.4.3 Quantitative experiments

We carry out three different quantitative experiments: In the first two experiments we evaluate, respectively, the influence of (*i*) the quantization step and of (*ii*) the number of prototype dictionary atoms N (kept constant for all ITAD layers). In the third experiment we compare our image codec to the state-of-the-art JPEG2000 image encoder and its predecessor the JPEG image encoder. For all three experiments we plot the average PSNR as a function of the coding rate,

where the average is taken over all 100 test images. From Fig. 6.5 and (6.31) the total rate for a given image is given by

$$\sum_{b=1}^{B} \left(\mathrm{R}\left(\tilde{\mu}_b\right) + \mathrm{R}\left(\texttt{EOB}\right) + \sum_{i=1}^{L_b} \mathrm{R}\left((a_{bi}, \hat{\gamma}_{bi})\right) \right). \tag{6.33}$$

The rate plotted is the corresponding value expressed in bits-per-pixel (bpp).

In the Fig. 6.6 we carry out the first experiment evaluating the quantization step-size (expressed in terms of the per-pixel RMSE ε_Δ). We use an ITAD structure having $N = 128$ atoms per layer. Note that the curves for ε_Δ values of 1.45 and 0.91 appear truncated in the figure. The reason for this is that it is not possible to achieve all rates for step sizes that are too large since, for sufficiently large i, all coefficients γ_{bi} are quantized to zero. This problem can be addressed by decreasing the quantizer step size along with increasing layer index, but we do not pursue this approach in the present work.

In Fig. 6.7 we carry out the second experiment which evaluates performance of the ITAD codec as a function of the total number of atoms N in all layers. Note that larger dictionaries result in improved performance even though the atom index coding penalty $\log_2(N)$ increases with N. The increased sparsity of the representation indeed succeeds in overcoming the related atom-index penalty.

In Fig. 6.8 we compare our proposed ITAD codec against the JPEG and JPEG2000 image encoders. The three different curves shown for the ITAD codec correspond to three different block sizes ($8 \times 8, 12 \times 12$ and 16×16 where, respectively, we use $\varepsilon_\Delta = 0.91$, 0.8 and 0.72). Note that different ITAD codecs are capable of outperforming the two reference codecs at all plotted rates by a wide margin. The 8×8 ITAD codec, for example, outperforms JPEG2000 for all rates above 0.23 bpp by at least 0.5 dB. At 0.4 bpp, the ITAD Codec gain is 0.9 dB. The ITAD codecs based on the two larger block sizes offer gains of several dB for lower bit-rates. For example, at 0.3 bpp, the 12×12 codec offers a gain of 1.5 dB.

6.4.4 Qualitative experiments

We now carry out a qualitative comparison of the ITAD codec and the JPEG2000 and JPEG codecs on four images chosen from our test set. The results are illustrated in Fig. 6.9 and Fig. 6.10. As indicated in the figures, each of the four columns of either figure corresponds, respectively, to (*i*) the original image, (*ii*) the JPEG2000 decoded image, (*iii*) the JPEG decoded image and (*iv*) the ITAD decoded image. The difference between the two figures is in the nominal rate used for all encoders: In Fig. 6.9 we use a nominal rate of 0.3 bpp and in Fig. 6.10 we use a nominal rate of 0.4 bpp. Using Fig. 6.8, we choose an ITAD block size of 12×12 for the 0.3 bpp rate and of 8×8 for the 0.4 bpp.

JPEG2000		JPEG		ITAD	
bpp	dB	bpp	dB	bpp	dB
0.31	30.73	0.32	28.93	0.31	33.99
0.31	31.46	0.33	29.15	0.31	33.90
0.32	35.58	0.32	31.20	0.32	36.59
0.31	31.71	0.32	29.78	0.31	32.91

Table 6.1: Rate (bpp) and distortion (dB) for the images in Fig. 6.9.

JPEG2000		JPEG		ITAD	
bpp	dB	bpp	dB	bpp	dB
0.40	33.26	0.40	31.09	0.40	36.49
0.40	33.48	0.40	31.18	0.40	35.10
0.39	37.33	0.40	33.75	0.40	38.45
0.39	32.99	0.40	31.35	0.39	33.79

Table 6.2: Rate (bpp) and distortion (dB) for the images in Fig. 6.10.

From the illustrations, one can observe that the images at the output of the ITAD codec indeed display an improved visual quality relative to either of the reference codecs. The JPEG encoder suffers from a very pronounced blocking artifact, particularly in the low-rate figure. The JPEG2000 images, on the other hand, suffer from blurring of the facial features. This is especially noticeable (in all images) around the eyes and nose, which are a lot sharper in the ITAD decoded images. For completeness, we provide the exact rates and PSNRs for both qualitative comparisons in Table 6.1 and Table 6.2, with each row of the table corresponding, respectively, to a row of Fig. 6.9 and Fig. 6.10.

6.5 Conclusion

In this chapter we have shown how the superior sparse approximation capability of the ITAD dictionary can be leveraged to construct an image codec capable of outperforming state-of-the art algorithms such as JPEG2000. The codec selects the sparsity of the various blocks using a rate-distortion criterion. Coding of the ITAD coefficients is then carried out using a standard quantizer / entropy encoder

combination. The evaluations we carried out show that our proposed codec can outperform the state of the art by at least 0.5 dB over a large range of rates. We further showed that the reconstructed image is more clear and better preserves details.

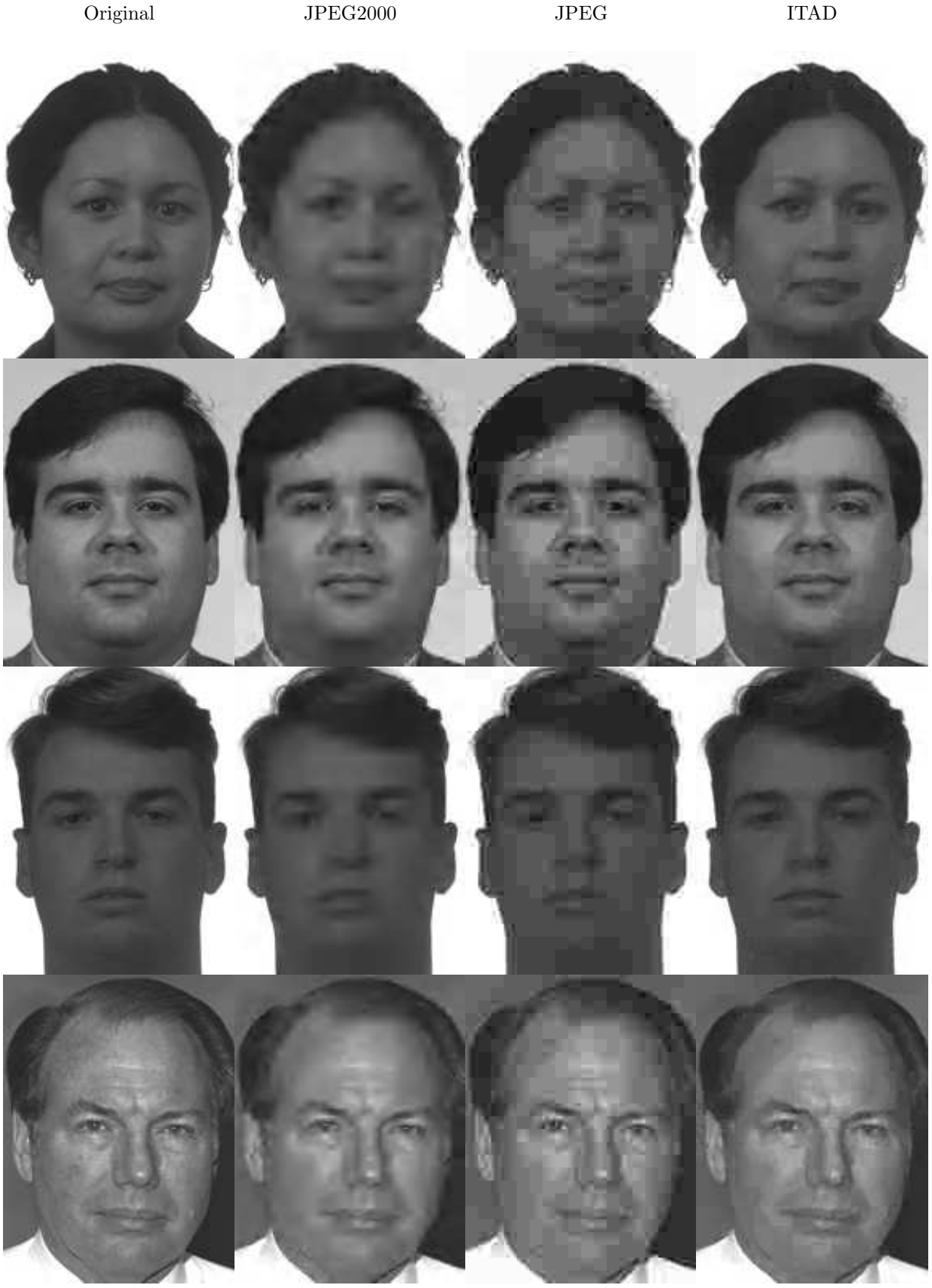

Figure 6.9: Qualitative evaluation of the ITAD codec at a nominal rate of 0.3 bpp. The exact rate and distortion for the rows of the last three columns are given in the respective cells of Table 6.1.

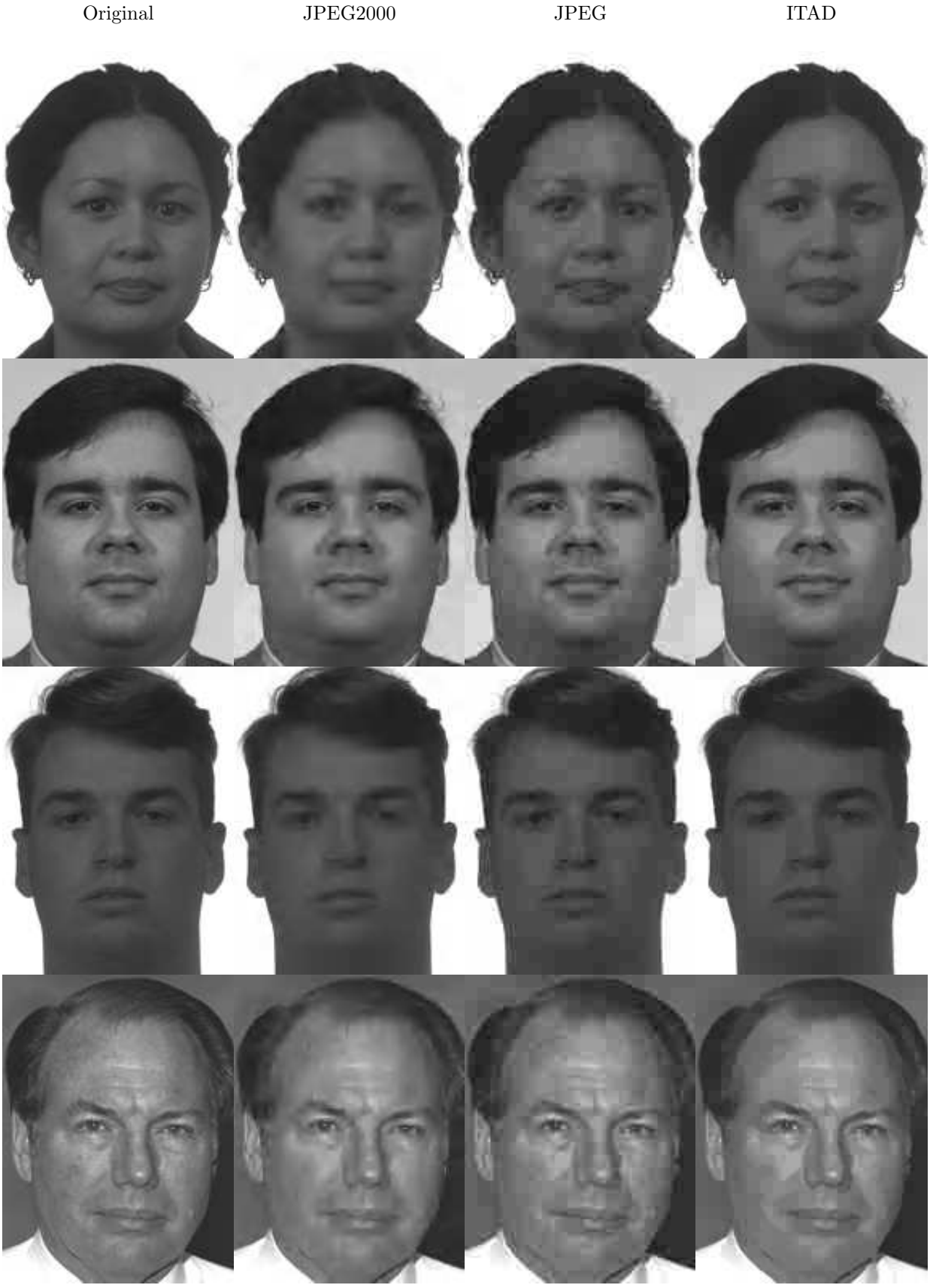

Figure 6.10: Qualitative evaluation of the ITAD codec at a nominal rate of 0.4 bpp. The exact rate and distortion for the rows of the last three columns are given in the respective cells of Table 6.2.

Chapter 7

Approximate Nearest Neighbors Using Sparse Representations

7.1 Introduction

Local descriptors computed on affine normalized image regions have proven successful in various computer vision applications under realistic perturbations such as partial occlusion or background changes [Jia 2008]. The local image description process consists of first selecting local regions from an image (the *affine covariant* regions) and then geometrically normalizing them to thus achieve some level of invariance to camera perspective changes. Different descriptors $\mathbf{y}$ have been developed to describe the resulting normalized image regions. At query time, a nearest neighbors (NN) search is carried out between the query descriptor $\mathbf{v}$ and the descriptor vectors $\mathbf{y}_b$ computed on the database images. Yet since local descriptors are high-dimensional, they are subject to the *curse of dimensionality* [Jia 2008], meaning that the NN search complexity is very high.

Approximate Nearest-Neighbor (ANN) searches based on various sparse representation schemes have been recently proposed to address the high computational complexity in local descriptor query systems [Zepeda 2009, Philbin 2008, Jégou 2008, Sivic 2003]. Given some sparse representation $\mathbf{x}_b$ of each $\mathbf{y}_b$, the search index will be the sparse matrix with columns $\mathbf{x}_b$. This *sparse-matrix index* is stored in compact row-major format by grouping all non-zero coefficients of any given row (and their column indices) to form a contiguous memory bin. This results in an implicit complexity-reducing pruning mechanism when using similarity measures based on the inner-product $\mathbf{q}^\mathsf{T}\mathbf{x}_b$, as only the bins corresponding to the non-zero positions of $\mathbf{q}$ need to be processed.

The work carried out in the present chapter uses sparse representation algorithms that are based on overcomplete dictionaries $\mathbf{D}$.[1] As discussed previously (*cf.* Chapter 2), the resulting sparse representation $\mathbf{x}$ of $\mathbf{y}$ satisfies $\mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{r}$, where $\mathbf{r}$ is the approximation error. The selection of $\mathbf{x}$ is done to minimize $|\mathbf{r}|$ (*i.e.*, the distortion) under a constraint on the number of non-zero positions of $\mathbf{x}$

[1] For ease of analysis, in this chapter we will consider fixed-dictionary schemes instead of the iteration-tuned dictionary structures introduced in previous chapters.

(a measure of rate). While this approach makes sense in various applications (*eg.,* compression and denoising), it poses a problem for the ANN search application: Since the geometrical normalization of an image's affine covariant regions is not perfect, the normalization errors will result in representations $\mathbf{x}$ that have an unstable *support* (the positions of non-zero coefficients). This instability adversely impacts the similarity score between realizations (*eg.,* in different images) of the same region and therefore affects the performance of the ANN search task.

In this chapter, we address the problem of instabilities in the support of $\mathbf{x}$ by first modelling these instabilities and then constructing, from $\mathbf{x}$, a new vector, called the *reduced vector*, that mitigates the effects of these instabilities in the ANN task. The construction of the *reduced vector* is formulated as a minimization of the reference distance approximation error subject to a sparsity constraint as, for sparse matrix indices, sparsity is related to both memory and computational complexity. Computational complexity is further minimized for a uniform distribution of non-zero positions in $\mathbf{x}$, which is in turn favored by a uniform distribution of $\mathbf{y}$ on the unit sphere. Thus we further introduce a data conditioning method which succeeds in approximately preserving the relative position of data points $\mathbf{y}$ on the unit sphere while making their distribution more uniform.

One possible application of our proposed method is that of enhancing the performance/complexity tradeoff of bag-of-features (BOF) indices [Sivic 2003] (*cf.* Section 3.3.1). BOFs make use of vector quantization (VQ), which is a specific case of the more general sparse representation framework. Thus VQ again uses a rate-distortion criterion that does not favor a stable support of the resulting $\mathbf{x}$, an issue addressed by our proposed method.

A second possible application involves using $\mathbf{x}$ the case when the normalized image regions are used directly as descriptors $\mathbf{y}$. In that scenario, the descriptors $\mathbf{y}$ can be exploited as side information to design a compact image / local descriptors package: Transmitting the descriptors $\mathbf{x}$ thus obtained yields an initial image estimate at no extra rate penalty. Since the receiver requires $\mathbf{x}$ rather than $\mathbf{y}$ for querying or indexing, including $\mathbf{x}$ in the transmitted package further exempts the receiver from descriptor extraction and processing.

The rest of this chapter is organized as follows: In Section 7.2 we first describe the system setup considered in this chapter. We then present the proposed reduced vector construction strategy in Section 7.3. Our algorithm makes use of an adaptive sparse correlation matrix, and we explain how to obtain it in Section 7.4. Our proposed data conditioning method is then presented in Section 7.5, and evaluated along with our main approach in Section 7.6. We provide concluding remarks in the last section.

7.2 Background

Let us assume that we are given a d-dimensional query signal $\mathbf{v}$ and a set of d-dimensional vectors $\{\mathbf{y}_b\}_{b=1}^{B}$ representing the database. The Nearest-Neighbor (NN) search process can be defined as producing a ranking of the database vectors according to a descending order of some similarity measure $d(\mathbf{v}, \mathbf{y}_b)$. Practical difficulties will arise when carrying out the NN search task in large databases and particularly those consisting of large dimensional vectors (*i.e.,* $d \geq 30$). It is well known [Jia 2008] that, for large dimensional signals, one is better off carrying out an exhaustive brute-force search than trying to otherwise order the database (*eg.,* as is done for 1-D signals by simply sorting the data) to thus decrease the computational burden. This bottleneck, known as the *curse of dimensionality* has led researchers to instead consider the Approximate Nearest-Neighbor (ANN) search scheme, where the resulting ranking of the database vectors is approximately corrct, preferably more lower-ranked database vectors. As an aside we point out that for content based search schemes, the descriptor vectors used are only an approximation of the described content (*eg.,* visual characteristics of an image patch), and thus exact or approximate rankings are both adequate.

In this work, we will be concerned particularly by ANN searches based on the normalized inner-product or *correlation* similarity measure

$$\langle \mathbf{v}, \mathbf{y}_b \rangle = \frac{1}{|\mathbf{v}| \cdot |\mathbf{y}_b|} \mathbf{v}^{\mathsf{T}} . \mathbf{y}_b \tag{7.1a}$$

Indeed a more common approach instead uses the Euclidean distance $|\mathbf{v} - \mathbf{y}_b|$. We use (7.1) nonethless as it lends itself more easily to the analysis we carry out. In practice, using correlations or Euclidean distances will not necessarily alter the results. We illustrate this experimentally for the case when signal vectors $\mathbf{q}$ and $\mathbf{y}_b$ consist of image patches. This is also the case for the widely used SIFT descriptor [Lowe 2004] given the particular normalization scheme used in that description algorithm.

7.2.1 Sparse representations

In our work we will be concerned with carrying out approximate searches based on sparse representations of the query vector $\mathbf{v}$ and database vectors $\mathbf{y}_b$. A sparse representation $\mathbf{x} \in \mathbb{R}^N$ of a signal vector $\mathbf{y}$ is based on an overcomplete dictionary matrix $\mathbf{D} \in \mathbb{R}^{d \times N}$. The representation $\mathbf{x}$ aims to produce the smallest approximation error based on a linear combination of only L columns of $\mathbf{D}$:

$$\underset{\mathbf{x} \in \mathbb{R}^N}{\operatorname{argmin}} |\mathbf{y} - \mathbf{D}\mathbf{x}| \text{ s.t. } |\mathbf{x}|_0 \leq L, \tag{7.2}$$

where the l-0 norm $|\cdot|_0$ counts the number of non-zero coefficients of a vector. We will solve the above problem using a relaxation of the constraint wherein the l-0 norm is substituted by the l-1 norm. Doing so allows the problem to be cast into a more familiar linear programming problem, solved using existing mathematical tools [Boyd 2004]. The resulting algorithm is known as Basis Pursuit (BP) [Chen 2001].

7.2.2 The sparse-matrix index

Let $\mathbf{x}_b$ denote the spare representations of the database vectors $\mathbf{y}_b$ and $\mathbf{q}$ denote the sparse representations of the query vector $\mathbf{v}$. We organize the sparse representations of the database vectors into a matrix $\mathbf{X} = (\mathbf{x}_b)_{b=1}^{B}$ with columns $\mathbf{x}_b$. To enjoy complexity and memory benefits, the sparse-matrix index $\mathbf{X}$ is stored in row-major format consisting of N memory bins $j = 1, \ldots, N$ such that the j-th bin contains the non-zero coefficients and column identifiers of row $\mathbf{X}[j, *]$, *i.e.,*

$$\{(\mathbf{X}[j, b], b) : b \in \{1, \ldots, B\} \wedge \mathbf{X}[j, b] \neq 0\}. \tag{7.3}$$

Using this index, one could attempt an approximation of $\langle \mathbf{v}, \mathbf{y}_b \rangle$ using either of

$$d_1(\mathbf{q}, \mathbf{x}_b) = \mathbf{q}^\mathsf{T} \mathbf{x}_b \quad \text{or} \tag{7.4a}$$

$$d_2(\mathbf{q}, \mathbf{x}_b) = \frac{\mathbf{q}^\mathsf{T} \mathbf{x}_b}{|\mathbf{q}||\mathbf{x}_b|}. \tag{7.4b}$$

This approach or variants thereof has in fact been used in several contextual image search algorithms [Sivic 2003, Nister 2006, Philbin 2008, Jégou 2008, Zepeda 2009] due to the low computational and memory complexity it enjoys: If we assume that $\mathbf{q}$ consists of L non-zero coefficients and that each bin (7.3) of the sparse-matrix index consists of $\alpha B/N$ non-zero coefficients (this is the case when $\mathbf{x}_b$ has uniformly distributed support, *cf.* Section 7.B) the query complexity incurred by (7.4) involves

$$L\frac{\alpha \cdot B}{N} \tag{7.5}$$

multiplications. This calculation will be efficient for high N (large dictionaries) and comparably low sparsities L (*i.e.,* for compressible data). Note that the values α, B and N do not depend on the query vector $\mathbf{q}$, and thus the query complexity incurred by $\mathbf{q}$ can be summarized by its l-0 norm L. Note further that the query operation (7.4) will need to access L row-bins from the sparse matrix index, each stored contiguously in memory. Hence L is also an adequate measure of memory complexity.

7.3 Formulating Sparse Support Selection as an Optimization Problem

The aim of the work presented in this chapter is to develop a similarity measure that enjoys the complexity and memory benefits of the approach in (7.4), while at the same time more closely approximating the ground truth distance in (7.1). Thus our approach is based on products between sparse vectors, but instead of using the sparse approximation vectors $\mathbf{q}$ and $\mathbf{x}_b$, we derive a new sparse vector that we call the *reduced vector*. As we will see, inner-products between reduced vectors can better approximate the ground-truth distance.

In this section we explain the method used to build the proposed *reduced vector* $\hat{\mathbf{q}}$ from the sparse representation $\mathbf{q}$ of the query vector. Note that the same process described below can be applied to instead construct the reduced vectors $\hat{\mathbf{x}}_b$ of the database signals $\mathbf{x}_b$ by setting $\mathbf{q} = \mathbf{x}_b$.

7.3.1 The reduced vector

Let us assume that the signal vectors $\mathbf{v}$ and $\mathbf{y}_b$ are unit norm and that their sparse approximations produce negligible residual error, *i.e.,*

$$\mathbf{v} \simeq \mathbf{D}\mathbf{q} \text{ and } \mathbf{y}_b \simeq \mathbf{D}\mathbf{x}_b. \tag{7.6a}$$

Using this assumptions, we can expand the ground-truth distance in (7.1) as follows:

$$\begin{aligned} \langle \mathbf{v}, \mathbf{y}_b \rangle &= \mathbf{q}^\mathsf{T} \mathbf{D}^\mathsf{T} \mathbf{D}\mathbf{x}_b && (7.7)\\ &= \mathbf{q}^\mathsf{T} \mathbf{C}_\mathbf{D} \mathbf{x}_b && (7.8)\\ &= (\mathbf{C}_\mathbf{D}\mathbf{q})^\mathsf{T} \mathbf{x}_b && (7.9) \end{aligned}$$

where $\mathbf{C}_\mathbf{D} = \mathbf{D}^\mathsf{T}\mathbf{D}$ is the Gram matrix of dictionary $\mathbf{D}$. Since computational and memory complexity are related to sparsity, we will build the *reduced vector* $\hat{\mathbf{q}}$ by retaining from vector $\mathbf{C}_\mathbf{D}\mathbf{q}$ only $L = |\mathbf{q}|_0$ coefficients at select positions $p \in \mathcal{P}$; $\hat{\mathbf{q}}$ will be zero elsewhere. We select the positions $\mathcal{P}$ that minimize the distance approximation error. Let the notation $\mathbf{a}_{|\mathcal{S}}$ denote the vector with entries

$$\mathbf{a}_{|\mathcal{S}}[j] = \begin{cases} \mathbf{a}[j] & \text{if } j \in \mathcal{S}, \\ 0 & \text{if } j \notin \mathcal{S}, \end{cases} \tag{7.10}$$

and let $\mathcal{C}$ denote all possible selections of L positions from

$$\mathcal{C} = \{\{p_1, \ldots, p_l\} : p_j \in \{1, \ldots, N\} \text{ and } \forall j \neq i, p_j \neq p_i\}.$$

The selection of the positions $\mathcal{P}$ can be expressed as follows:

$$\mathcal{P} = \underset{\mathcal{P}' \in \mathcal{C}}{\operatorname{argmin}} \left| \left(\mathbf{C_D q} - (\mathbf{C_D q})_{|\mathcal{P}'} \right)^{\mathsf{T}} \mathbf{x}_b \right|^2 \text{ s.t. } |\mathcal{P}'| = L \tag{7.11}$$

Using the $\mathcal{P}$ thus chosen, the reduced vector $\hat{\mathbf{q}}$ is given by

$$\hat{\mathbf{q}} \triangleq (\mathbf{C_D q})_{|\mathcal{P}}. \tag{7.12}$$

In searching for a solution to (7.11) we will find it more convenient to write expression (7.11) in terms of the set of complementary positions $\bar{\mathcal{P}}' = \{p = 1, \ldots, N : p \notin \mathcal{P}'\}$ as follows

$$\underset{\mathcal{P} \in \mathcal{C}}{\operatorname{argmin}} \, |(\mathbf{C_D q})_{\bar{\mathcal{P}}}^{\mathsf{T}} \mathbf{x}_b|^2 \text{ s.t. } |\mathcal{P}| = L, \tag{7.13}$$

where we have used the fact that $\mathbf{C_D q} - (\mathbf{C_D q})_{\mathcal{P}} = (\mathbf{C_D q})_{\bar{\mathcal{P}}}$.

7.3.2 Relation to reference sparse distances

Recall that the assumption of compressibility of the sparse vector allows us to write $\mathbf{v} \simeq \mathbf{C_D q}$; we can express the reduced vector $\hat{\mathbf{q}}$ in a similar manner as follows:

$$\hat{\mathbf{q}} = \mathbf{C}_{\mathbf{D}|\mathcal{P}} \mathbf{q},$$

where matrix $\mathbf{C}_{\mathbf{D}|\mathcal{P}}$ has the same size as $\mathbf{C_D}$ and rows given by

$$\mathbf{C}_{\mathbf{D}|\mathcal{P}}[j, *] = \begin{cases} \mathbf{C_D}[j, *] & \text{if } j \in \mathcal{P} \\ \mathbf{0}^{\mathsf{T}} & \text{if } j \notin \mathcal{P}. \end{cases} \tag{7.14}$$

Using this notation, the approximation of the ground-truth similarity measure $\mathbf{q}^{\mathsf{T}} \mathbf{C} \mathbf{x}_b$ based on $\hat{\mathbf{q}}$ can be written as

$$\hat{\mathbf{q}}^{\mathsf{T}} \mathbf{x}_b = \mathbf{q}^{\mathsf{T}} \, \mathbf{C}_{\mathbf{D}|\mathcal{P}} \, \mathbf{x}_b. \tag{7.15}$$

Thus the approximate distance based on $\hat{\mathbf{q}}$ is seen to have a general form related to that of the reference system $\mathbf{q}^{\mathsf{T}} \mathbf{x}_b$ in (7.4a): both can be generalized by

$$\mathbf{q}^{\mathsf{T}} \, \mathbf{C}' \, \mathbf{x}_b$$

where $\mathbf{q}^{\mathsf{T}} \mathbf{x}_b$ neglects the cross-atom correlations with $\mathbf{C}' = \mathbf{I}$, while (7.15) considers to some extent these cross-atom correlations with $\mathbf{C}' = \mathbf{C}_{\mathbf{D}|\mathcal{P}}$. Hence we expect and indeed observe in the results section that the proposed distances consistently outperform $\mathbf{q}^{\mathsf{T}} \mathbf{x}_b$.

7.3.3 Exact solution for certain $\mathbf{x}_b$

In searching a solution to (7.13) we will first consider an approach wherein $\mathcal{P}$ is set equal to the sparse support of $\mathbf{q}$. This approach has the interesting property that the approximation $\hat{\mathbf{q}}^{\mathsf{T}}\mathbf{x}_b$ will in fact equal $\langle \mathbf{v}, \mathbf{y}_b \rangle$ for all $\mathbf{y}_b = \mathbf{D}\mathbf{x}_b$ having $\mathbf{x}_b$ with support contained in $\mathcal{P}$. We formalize this in the following proposition.

Proposition 7.3.1. *Let $\mathcal{P}^o$ be given by the sparse support of $\mathbf{q}$, i.e.,*

$$\mathcal{P}^o = \{p = 1, \ldots, N : \mathbf{q}[p] \neq 0\}, (7.18) \tag{7.16}$$

and let $\mathcal{B} = \{b = 1, \ldots, B : \forall p \notin \mathcal{P}^o, \mathbf{x}_b[p] = 0\}$. Then it follows that

$$\forall b \in \mathcal{B}, \hat{\mathbf{q}}^{\mathsf{T}}\mathbf{x}_b = \langle \mathbf{v}, \mathbf{y}_b \rangle. \tag{7.17}$$

For latter reference, we will let

$$\hat{\mathbf{q}}^o = (\mathbf{C}_\mathbf{D}\mathbf{q})_{|\mathcal{P}^o}. \tag{7.18}$$

denote the reduced vector built using the above position selection method.

7.3.4 Minimizing upper bound

The position selection scheme specified in (7.18) disregards the original problem formulation (7.13). We present now an approach that consists of substituting the cost function $|(\mathbf{C}_\mathbf{D}\mathbf{q})_{\bar{\mathcal{P}}}^{\mathsf{T}}\mathbf{x}_b|^2$ by the upper bound $|(\mathbf{C}_\mathbf{D}\mathbf{q})_{\bar{\mathcal{P}}}|^2|\mathbf{x}_b|^2$. Noting that the term $|\mathbf{x}_b|^2$ is constant with $\mathcal{P}$ and can hence be dropped, we write the resulting cost function as

$$|(\mathbf{C}_\mathbf{D}\mathbf{q})_{\bar{\mathcal{P}}}|^2 = \sum_{k \notin \mathcal{P}'} |(\mathbf{C}_\mathbf{D}\mathbf{q})[k]|^2,$$

exposing the optimal $\mathcal{P}'$ as that excluding the L coefficients of $\mathbf{C}_\mathbf{D}\mathbf{q}$ with the strongest magnitude:

$$\mathcal{P}^u = \{p = 1, \ldots, N : \forall k \notin \mathcal{P}^u,\ |(\mathbf{C}_\mathbf{D}\mathbf{q})[p]| \geq |(\mathbf{C}_\mathbf{D}\mathbf{q})[k]|\}\,. \tag{7.19}$$

We will denote the reduced vector using the above method of selection of non-zero positions using a superscript u:

$$\hat{\mathbf{q}}^u = (\mathbf{C}_\mathbf{D}\mathbf{q})_{|\mathcal{P}^u} \tag{7.20}$$

7.3.5 Probabilistic approach

The solutions obtained in (7.18) and (7.19) disregard the index vectors $\mathbf{x}_b$ appearing in the original problem (7.13). We would prefer to build *reduced vectors* $\hat{\mathbf{q}}$ better suited to finding nearest neighbors from within a particular database $\{\mathbf{x}_b\}_{b=1}^B$. To this end, let us treat the database vectors as random vectors with realizations given by $\{\mathbf{x}_b\}_{b=1}^B$. Taking the expectation over $\mathbf{x}_b$ of the target function (7.13) produces

$$\underset{\mathcal{P}'}{\operatorname{argmin}} \, (\mathbf{C_D q})_{\bar{\mathcal{P}}}^{\mathsf{T}} \, \mathbf{B} (\mathbf{C_D q})_{\bar{\mathcal{P}}} \text{ s.t. } |\mathcal{P}'| = L, \tag{7.21}$$

with $\mathbf{B} = \mathrm{E}\left[\mathbf{x}_b \mathbf{x}_b^{\mathsf{T}}\right]$. Unlike (7.18) and (7.19), a solution to (7.21) is not straightforward and could in general require a combinatorial approach wherein all $\binom{L}{N}$ possible values for $\mathcal{P}$ are tried.

7.3.5.1 Hybrid upper bound using first singular vector

We thus consider a simplification of (7.21) that aims to enjoy the simplicity in computing $\mathcal{P}$ of (7.19) while considering at the same time the statistics of the database vectors. To this end we substitute the cost function $|(\mathbf{C_D q})_{\bar{\mathcal{P}}}|^2$ used to obtain (7.19) by $|\mathrm{diag}(\mathbf{v}_1)(\mathbf{C_D q})_{\bar{\mathcal{P}}}|^2$, where $\mathrm{diag}(\mathbf{v}_1)$ is the diagonal matrix with $\mathbf{v}_1$ along its main diagonal. The resulting set of selected positions is likewise simple to obtain and follows directly from the discussion related to (7.19):

$$\mathcal{P}^1 = \left\{ p = 1, \ldots, N : \forall k \notin \mathcal{P}^1, \; |\mathbf{v}_1[p](\mathbf{C_D q})_{\bar{\mathcal{P}}}[p]|^2 \geq |\mathbf{v}_1[k](\mathbf{C_D q})_{\bar{\mathcal{P}}}[k]|^2 \right\}, \tag{7.22}$$

where, accordingly,

$$\hat{\mathbf{q}}^1 = (\mathbf{C_D q})_{|\mathcal{P}^1}. \tag{7.23}$$

The vector $\mathbf{v}_1$ needs to be a model for vectors $\mathbf{x}_b$ obtained from signal vectors $\mathbf{y}_b$ that are similar (*i.e.,* under (7.1)) to the query $\mathbf{v}$, as we are searching for nearest neighbors. We note in particular that nulls $\mathbf{v}_1[k]$ will force the corresponding positions k to be excluded from $\mathcal{P}^1$. We propose here using $\mathbf{v}_1$ corresponding to the first singular vector of $\mathbf{B}$. Later we will show how to build an estimate of $\mathbf{B}$ that adapts to $\mathbf{q}$ and thus yields a $\mathbf{v}_1$ that is a good model of correct responses $\mathbf{x}_b$.

7.3.5.2 Approximate solution

The solution in (7.23) does not address the formulation (7.21) directly. Thus, we now present an iterative approximation of (7.21) that consists in selecting, at iteration l, the single position p_l yielding the greatest decrease in magnitude of the cost function when all previous positions

$$\mathcal{P}_{l-1} = \{p_1, \ldots, p_{l-1}\}$$

are nulled out. The selected position p_l can be expressed as follows, where we let $\mathcal{P}_0$ denote the empty set:

$$p_l = \underset{p \in \{1,\ldots,N\}/\mathcal{P}_{l-1}}{\operatorname{argmin}} |(\mathbf{C_D q})_{|\bar{\mathcal{Q}}}^{\mathsf{T}} \mathbf{B}| \text{ where } \mathcal{Q} = \mathcal{P}_{l-1} \cup p \tag{7.24}$$

Accordingly, we denote the reduced vector obtained with the positions $\mathcal{P}_L$ at the L-th iteration as

$$\hat{\mathbf{q}}^h = (\mathbf{C_D q})_{|\mathcal{P}_L}. \tag{7.25}$$

7.4 Construction of $\mathbf{C}_B$

The sparse correlation matrix $\mathbf{B}$ is needed in (7.23) and (7.25) to build the *reduced vectors* $\hat{\mathbf{q}}^1$ and $\hat{\mathbf{q}}^h$ and we now consider its construction. We will use an unbiased estimate of $\mathbf{B}$, where the realizations used to build the estimate are chosen as a function of the query signal vector $\mathbf{v}$ or its sparse representation $\mathbf{q}$ in one of two manners specified shortly. This adaptive approach is meant to result in better distance approximations $\hat{\mathbf{q}}^{\mathsf{T}} \mathbf{x}_b$ for stored vectors $\mathbf{y}_b = \mathbf{D}\mathbf{x}_b$ that are more similar (*i.e.,* under (7.1)) to $\mathbf{v}$. The estimate of $\mathbf{B}$ thus built can be seen as a model for the instabilities in the sparse support $\mathcal{P}$ of $\mathbf{q}$.

We propose two methods to obtain the select the realizations $\mathbf{x}_b$ s.t. $b \in \mathcal{N}$, (*i*) a two-stage method and (*ii*) an episode method.

The two-stage approach consists of first carrying out an initialization query using a sparse-matrix index $\mathbf{X} = (\mathbf{x}_b)_{b=1}^B$ and a query vector $\hat{\mathbf{q}}^o$ built as in (7.19). What is important in the choise of $\hat{\mathbf{q}}^o$ is that it does not require $\mathcal{B}$, and hence either $\mathbf{q}$ or $\hat{\mathbf{q}}^u$ could likewise be used in this first stage. The indices b of the first several realizations $\mathbf{x}_b$ thus obtained will define a set $\mathcal{N}$, and the estimate of $\mathbf{B}$ can then be built using

$$\frac{1}{|\mathcal{N}| - 1} \sum_{b \in \mathcal{N}} \mathbf{x}_b \mathbf{x}_b^t. \tag{7.26}$$

The second method of obtaining $\mathbf{B}$ relies on an assumed prior distribution of the data vectors $\mathbf{v}$ that can be used to obtain a set of similar vectors $\{\mathbf{v}_e\}_{e=1}^E$ referred to as the *episode* of $\mathbf{v}$. The related sparse decompositions $\mathbf{q}_e$ of the episode vectors $\mathbf{v}_e$ provide the realizations used to build the unbiased estimate of $\mathbf{B}$ as follows:

$$\frac{1}{E - 1} \sum_{e=1}^{E} \mathbf{q}_e \mathbf{q}_e^t. \tag{7.27}$$

An example of a possible assumed prior distribution would be the case where the $\mathbf{y}_q$ are noisy measurements; the episode could then be obtained by applying multiple realizations of the noise model to $\mathbf{y}_q$.

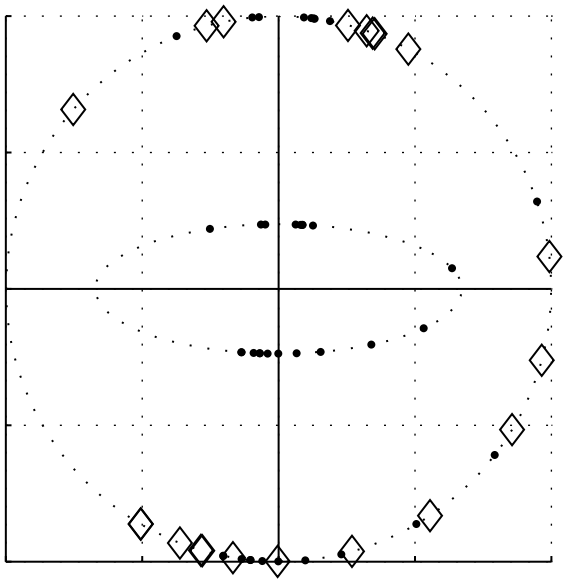

Figure 7.1: Illustration of data conditioning process in (7.30). The unconditioned data $\mathbf{y}$ is denoted by the dots on the unit circle. Following anisotropic scaling (*i.e.,* $\mathbf{D}_y^{-1/2}\mathbf{U}_y\mathbf{y}$) , these data-points are found on the ellipse. Following re-normalization (*i.e.,* $\mathbf{D}_y^{-1/2}\mathbf{U}_y\mathbf{y}/|\mathbf{D}_y^{-1/2}\mathbf{U}_y\mathbf{y}|$) the position of these points is illustrated by the diamond shaped markers.

7.5 Data conditioning

We now present a data conditioning method that better adapts the vectors $\mathbf{y}$ to the calculation of approximate distance using sparse-matrix indices built from the reduced vectors described above. The method aims to achieve two properties for the conditioned data:

1. A more uniform distribution over the unit sphere and
2. preservation of the relative positions of the original data vectors.

The motivation for this data-conditioning is twofold: Firstly, a uniform distribution is desirable because it maximizes the average minimum pairwise correlation $\min_{\mathbf{a}\neq b}\langle \mathbf{y}_a, \mathbf{y}_b\rangle$ of the database vectors, thus allowing for a larger distance approximation error without affecting data ranking. Secondly, a uniform distribution better distributes the coefficients of all $\mathbf{x}_b$ amongst sparse-matrix index row bins. Query complexity is related to bin size and thus reducing the largest bin size is a good complexity reduction strategy. As we show in Appendix 7.B query complexity is in fact minimized for a uniform distribution of the sparse support.

Let $\mathbf{y}$ denote an arbitrary signal vector (*eg.,* a database signal vector $\mathbf{y}_b$ or the query signal vector $\mathbf{v}$). The two desired properties of the data conditioning scheme

(uniform distribution and distance preservation) can be expressed as follows:

$$\mathbf{y}^c \sim \begin{cases} 1/V_1 & \text{if } |\mathbf{y}^c| = 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \tag{7.28a}$$

$$\langle \mathbf{y}, \mathbf{y}_1 \rangle \leq \langle \mathbf{y}, \mathbf{y}_2 \rangle \iff \langle \mathbf{y}^c, \mathbf{y}_1^c \rangle \leq \langle \mathbf{y}^c, \mathbf{y}_2^c \rangle \tag{7.28b}$$

where V_1 denotes the surface area of the unit sphere.

The approach we use to approximate the above conditions (7.28) relies on the singular value decomposition of the data correlation matrix

$$\mathbf{Y} = \mathrm{E}\left[\mathbf{y}\,\mathbf{y}^\mathsf{T}\right] = \mathbf{U}_\mathbf{Y}^\mathsf{T}\,\mathbf{D}_\mathbf{Y}\,\mathbf{U}_\mathbf{Y}, \tag{7.29}$$

where we note that, in practice, the expectation above is computed experimentally using the database vectors $\{\mathbf{y}_b\}_{b=1}^B$. Using the above, the conditioned data vectors are given by the following expression:

$$\mathbf{y}^c = \frac{\mathbf{D}_\mathbf{Y}^{-1/2}\mathbf{U}_\mathbf{Y}\mathbf{y}}{|\mathbf{D}_\mathbf{Y}^{-1/2}\mathbf{U}_\mathbf{Y}\mathbf{y}|}. \tag{7.30}$$

The proposed approach is justified by noting that, since the $\mathbf{y}$ are normalized, the singular vectors in $\mathbf{U}_\mathbf{Y}$ are determined by concentrations of points on the unit sphere, with singular values measuring the corresponding point density. Hence the vectors $\mathbf{D}_\mathbf{Y}^{-1/2}\mathbf{U}_\mathbf{Y}\mathbf{y}$ have undergone anisotropic scaling along principal directions in a manner inversely proportional to the point density; re-normalizing subsequently spreads high density concentrations, thus addressing condition (7.28a). Regarding the second condition (7.28b), we consider the 2-D case (illustrated in Fig. 7.1), where it is easy to see that anisotropic scaling and re-normalization preserves the relative neighbors of a given point along *either of two* angular directions.

7.6 Results

We construct query and index vectors $\mathbf{v}$ and $\mathbf{y}_b$ by extracting up to 150 MSER regions [Matas 2002] from images of the *Holidays* dataset [Jégou 2008]. This results in $60,909$ MSER regions of arbitrary size that are affine normalized to size 11×11 and vectorized to obtain the query vectors $\mathbf{v} \in \mathbb{R}^{121}$. To obtain the database vectors $\mathbf{y}_b$, we need to model residual normalization errors as they cause instabilities in the sparse support of the sparse representations $\mathbf{q}$ and $\mathbf{x}$. We do this by applying random affine transformations of small magnitude to the original normalized query patches $\mathbf{v}$. As illustrated in Fig. 7.2, these transformations are defined as mapping the equilateral triangle with vertices $\mathbf{a}_i$ on the unit circle to one with vertices $\mathbf{a}_i + \mathbf{n}_i$, where $\mathbf{n}_i$ is a random vector taken uniformly over the

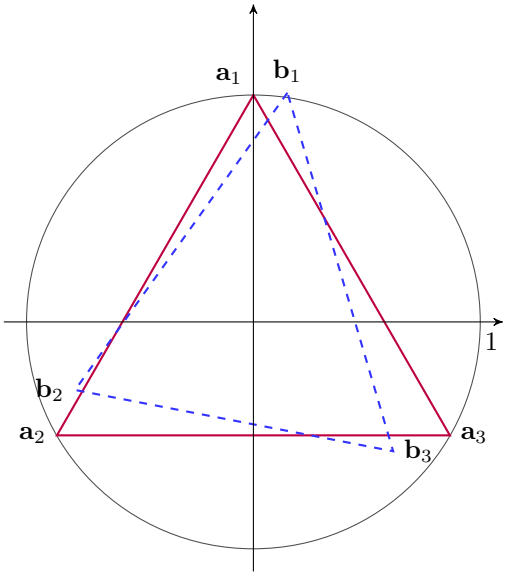

Figure 7.2: The affine normalization errors are modeled using the transformation that maps the triangle with vertices $\mathbf{a}_i$ on the unit sphere, to that with vertices $\mathbf{b}_i = \mathbf{a}_i + \mathbf{n}_i$, where the $\mathbf{n}_i$ are independent and uniform over the disk of radius 0.3.

circle of radius 0.3. For each query signal $\mathbf{v}$, we build 10 such realizations (see Fig. 7.3 for a visual example), for a total of $609,090$ database vectors $\mathbf{y}_b$. Since we employ inner-product distances, both $\mathbf{v}$ and $\mathbf{y}_b$ are re-centered using the mean vector calculated on the database set $\{\mathbf{y}_b\}_{b=1}^{B}$. Given the query and database sets thus constructed, we further build the conditioned vectors $\mathbf{v}^c$ and $\mathbf{y}_b^c$ using (7.30). All sparse decompositions $\mathbf{q}$ / $\mathbf{x}_b$ are built using basis pursuit and the DCT dictionary of size 121×1024 [Chen 2001].

We use $|\mathcal{N}| = 200$ when building the *reduced vector* based on two-stage method, denoted $\hat{\mathbf{q}}^i$ to differentiate it from other $\hat{\mathbf{q}}$ built using the episode method; episodes are obtained assuming a prior distribution given by the residual affine normalization model described above and illustrated in Fig. 7.2.

System performance will be measured by recall, *i.e.,* the number of correct $\mathbf{y}_b$ (up to 10) retrieved for a given $\mathbf{v}$. Complexity will be measured by the total number of non-zero coefficients in all index bins accessed by the query. This measure neglects pre-processing overhead (*eg.,* sparse decomposition of $\mathbf{y}_q$ and query-side *reduced vector* construction) yet this is valid since *1*) query complexity is much larger than pre-processing overhead for sufficiently large databases; *2*) the application (discussed in the introduction) involving a compact image/local descriptors package provides pre-computed sparse representations; and *3*) for compressible signals, $\hat{\mathbf{q}}$ (*cf.* (7.13)) can be obtained directly from $\mathbf{v}$ with no need of $\mathbf{x}_q$ since $\hat{\mathbf{q}} = \mathbf{D}^{\mathsf{T}}\mathbf{D}\mathbf{x}_q \simeq \mathbf{D}^{\mathsf{T}}\mathbf{v}$. We will compare our proposed systems relative to the

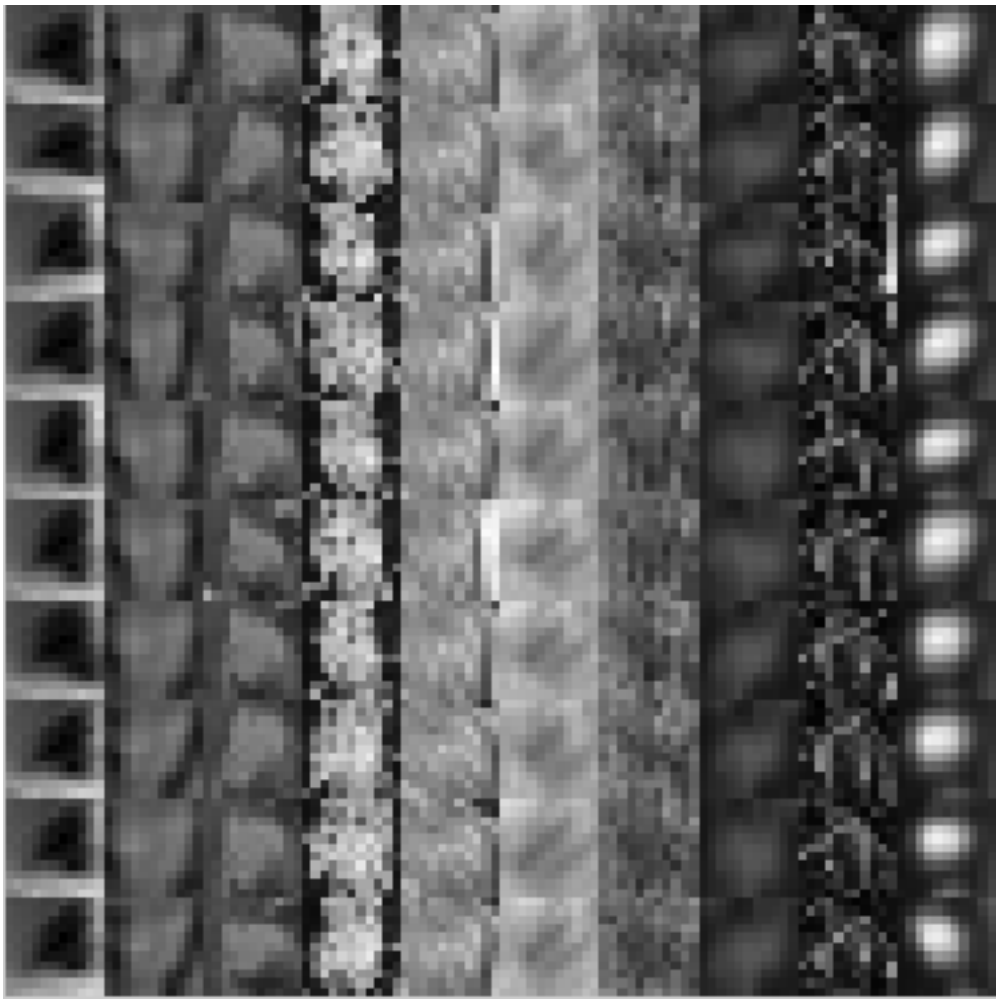

Figure 7.3: An example of several distorted image patches obtained with the model in Fig. 7.2.

reference distances (7.4)

7.6.1 Evaluation of data conditioning

In Fig. 7.4 we evaluate how well (7.30) satisfies the uniform distribution condition (7.28a) by comparing the distribution of projections of $\mathbf{y}_b$ or $\mathbf{y}_b^c$ unto their respective first two principal vectors $\mathbf{u}_1$ and $\mathbf{u}_2$. Projections along the first two principal vectors deviated the most from the projection of the ideal distribution, given by $\frac{1}{\alpha}(\sqrt{1-p^2})^{119}$ (we derive this equation in Appendix 7.A) with projection $p \in [-1, 1]$ and α a normalization constant. It is evident from the graph that the conditioning scheme succeeds in better distributing the data.

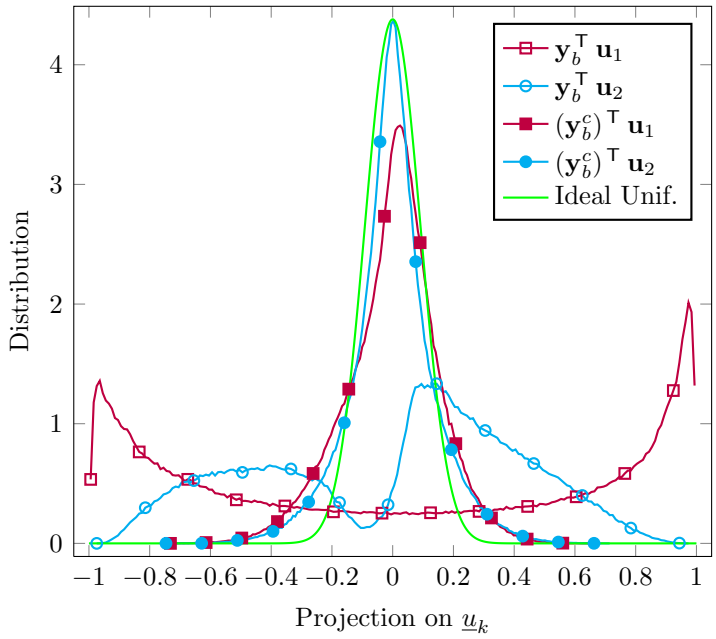

Figure 7.4: Distribution of projections of $\mathbf{y}_b$ and $\mathbf{y}_b^c$ on their first two principal vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ versus ideal distribution (uniform on the unit sphere).

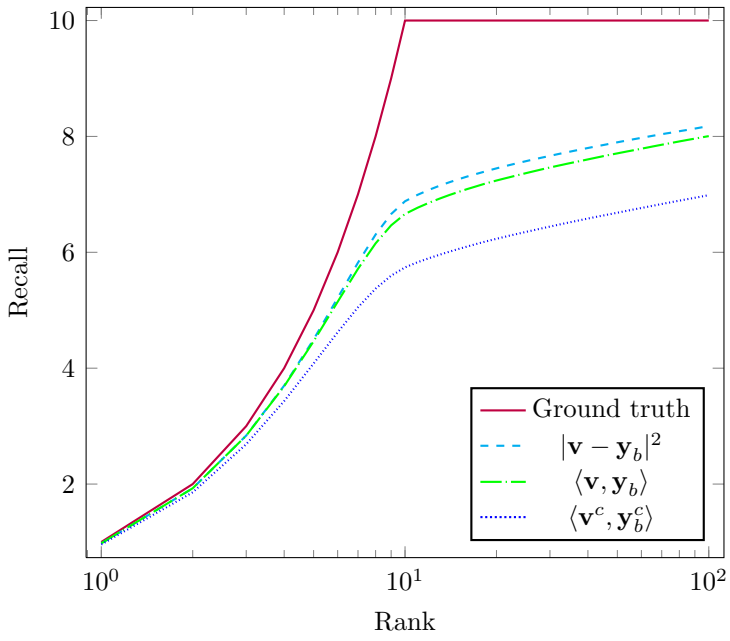

Figure 7.5: Recall of systems based on various distances: $\langle \mathbf{v}, \mathbf{y}_b \rangle$ and $\langle \mathbf{v}^c, \mathbf{y}_b^c \rangle$ using normalized data vectors and l-2 distance using un-normalized data vectors. The solid line is the ground truth response.

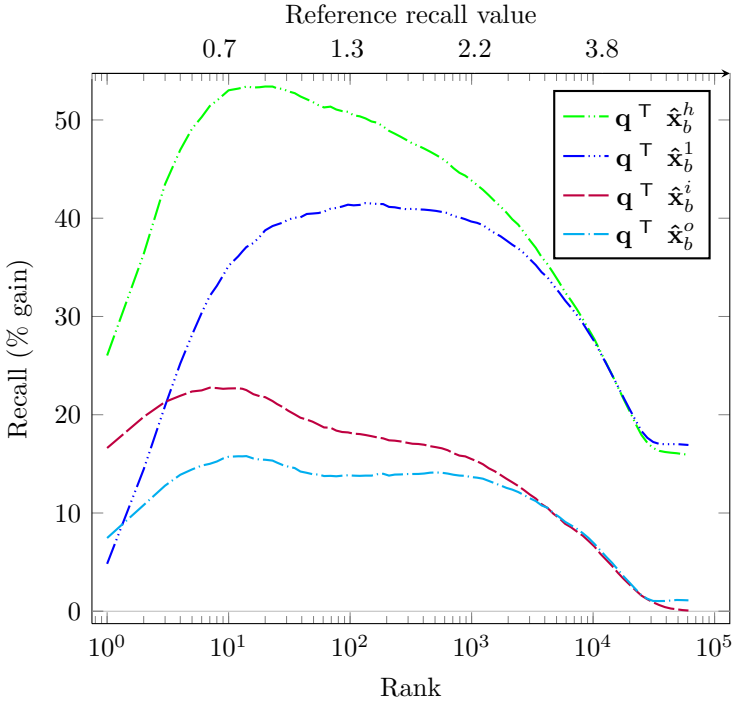

Figure 7.6: Percent gain of $\hat{\mathbf{x}}_b^h$ in (7.25), $\hat{\mathbf{x}}_b^1$ in (7.23), $\hat{\mathbf{x}}_b^o$ in (7.18) and the two-stage $\hat{\mathbf{x}}_b^i$. The reference recall values (indicated in the top abscissa labels) are the best recall per rank of either system corresponding to the reference distances (7.4).

To evaluate the distance preserving condition (7.28b) we compare average recall produced by a ranking based on $\langle \mathbf{v}, \mathbf{y}_b \rangle$, one based on $\langle \mathbf{q}^c, \mathbf{y}_b^c \rangle$ and the ground truth recall given from the database construction process. Results are shown in figure Fig. 7.5, along with the recall under the Euclidean distance (with un-normalized $\mathbf{v}$ and $\mathbf{y}_b$) used elsewhere [Mikolajczyk 2005b]. Data conditioning degrades the response relative to the unconditioned data, but the response is still of sufficient quality to yield an advantage for $\mathbf{y}^c$ in a subsequent test.

7.6.2 Improvement over reference systems

We compare the various methods used to construct index-side *reduced vectors* $\hat{\mathbf{x}}_b$ from conditioned data vectors $\mathbf{y}_b^c$ in Fig. 7.6. As a reference we take the maximum recall (at each rank) obtained under either reference system in (7.4) and plot the percent gain obtained with reduced vectors over this maximum reference. Note that all systems based on reduced vectors display a performance advantage.

In Fig. 7.7, $\mathbf{q}^{\mathsf{T}}\,\hat{\mathbf{x}}_b^h$ (the best system of Fig. 7.6) and its query-side version $\hat{\mathbf{q}}^{h\,\mathsf{T}}\,\mathbf{x}_b$ are compared against the reference systems (7.4). Both conditioned and uncondi-

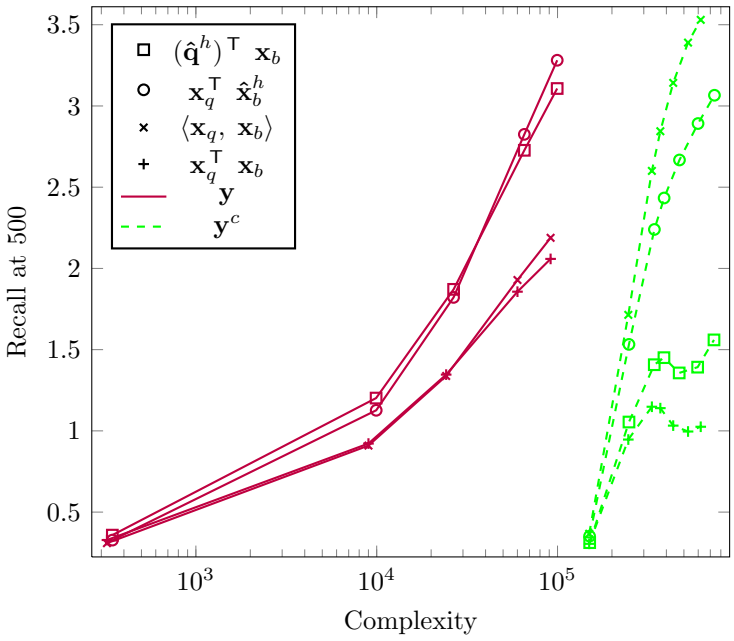

Figure 7.7: Recall at rank 500 versus query complexity when using the proposed distances $\hat{\mathbf{q}}^{h\,\mathsf{T}}\,\mathbf{x}_b$ and $\mathbf{x}_q^\mathsf{T}\,\hat{\mathbf{x}}_b^h$ based respectively on query-side and index-side *reduced vectors* (built as in (7.25)) and the reference distances in (7.4). Solid (respectively dashed) lines are obtained using the conditioned (unconditioned) datasets $\mathbf{y}$ ($\mathbf{y}^c$).

tioned data are used to obtain the sparse representations $\mathbf{x}_b$ and $\mathbf{q}$. We first note that the unconditioned data results in a wide spread in performance between both reference systems. The systems based on reduced vectors consistently outperform the related reference system (7.4a), yet the reference system (7.4b) outperforms all.

The unconditioned dataset $\mathbf{y}^c$ results in comparable performance for both reference systems, indicating a more uniform balancing of the sparse-matrix index bins. The systems based on $\hat{\mathbf{q}}^h\mathbf{x}_b$ or $\mathbf{x}_q^\mathsf{T}\,\hat{\mathbf{x}}_b^h$ improve upon (7.4a) and thus also (7.4b). At a complexity of 10^5, $\mathbf{x}_q^\mathsf{T}\,\hat{\mathbf{x}}_b^h$ performs at a recall of 3.3 that is 50% better than that of the best reference system, 2.2. Our data conditioning scheme offers a clear advantage: for a fixed recall, the reduced-vector systems built using the conditioned data display a complexity improvement of 0.5 to 3 orders of magnitude relative to any system built using the unconditioned data.

7.7 Conclusion

We introduced a method that succeeded in improving the performance / complexity tradeoff when approximating the normalized inner-product distance between

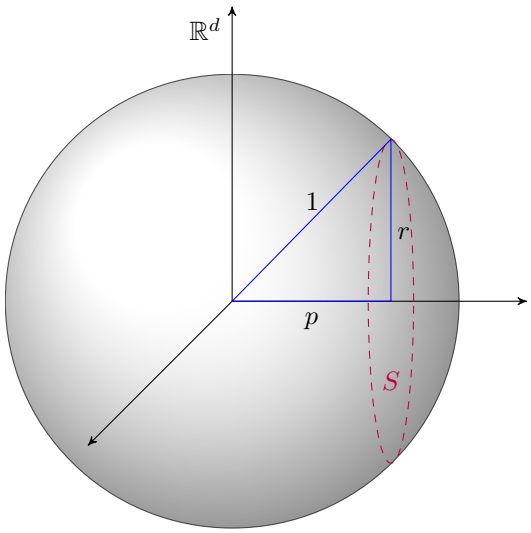

Figure 7.8: The distribution at p of points on the unit hyper-sphere of dimension d projected unto the horizontal axis is proportional to the surface area of the hyper-sphere of dimension $d-1$ obtained by slicing $\mathbb{R}^d$ with a plane orthogonal to the horizontal axis and intersecting it at p.

compressible signals relative to approximations that use the sparse representations directly. The proposed approach included a method for data conditioning that succeeds in approximately preserving the relative positions of the original data points while making their distribution more uniform over the unit sphere. The conditioning method is verified experimentally, and its application to the distance approximation scheme proposed yields complexity improvements as high as 3 orders of magnitude for the same performance.

Appendix 7.A Distribution of the projection of data uniformly distributed on the unit hyper-sphere

We now derive the distribution of data uniformly distributed on the unit hyper-sphere in $\mathbb{R}^d$ when projected unto an arbitrary line. Given the symmetry of the problem, the orientation of the line is arbitrary, and thus we use the line along the horizontal axis (see Fig. 7.8). From the figure it is evident that the projection p will be in $[-1, 1]$. The density of the projections at a value p will be proportional to the surface area of the hyper-sphere of dimension $d-1$ obtained by slicing $\mathbb{R}^d$ with a plane perpendicular to the horizontal axis and intersecting it at p. We let S denote this lower-dimensional hyper-sphere in the figure, and note that it has a

radius given by

$$r = \sqrt{1 - p^2}. \tag{7.31}$$

Hence the surface are of this lower-dimensional hyper-sphere at a given p (and accordingly the distribution of projections at p) will be given by

$$\alpha(\sqrt{1 - p^2})^{d-2}, \tag{7.32}$$

where α is an adequate normalization constant.

Appendix 7.B Optimal distribution of sparse support

We now show that the query complexity of a sparse-matrix index is minimized when the support of the sparse representation (either χ or $\mathbf{x}$) has constant mean across all vector entries.

Without loss of generality, we assume that we are dealing with sparse vectors $\mathbf{x}$. We wish to obtain the distribution of the sparse support of $\mathbf{x}$ that minimizes the mean complexity of the inner-product $\mathbf{x}_q^\mathsf{T}\,\mathbf{x}_b$, where complexity is measures by the total number of multiplications (between non-zero coefficients) carried out. Letting vector $\boldsymbol{\omega} \in \{0, 1\}^N$ denote the sparse support of $\mathbf{x} \in \mathbb{R}^N$, this complexity is given by

$$\mathrm{E}\left(\boldsymbol{\omega}_q^\mathsf{T}\,\boldsymbol{\omega}_b\right) = \boldsymbol{\mu}^\mathsf{T}\,\boldsymbol{\mu}, \tag{7.33}$$

where $\boldsymbol{\mu}$ denotes the mean of $\boldsymbol{\omega}$ and we have assumed that the supports of two different sparse vectors are uncorrelated (entry-wise). We wish to minimize the complexity (7.33) subject to a constraint

$$\mathrm{E}\left(\boldsymbol{\omega}^\mathsf{T}\,\mathbf{1}\right) = \boldsymbol{\mu}^\mathsf{T}\,\mathbf{1} = l \tag{7.34}$$

on the mean sparsity of the data, where $\mathbf{1}$ denotes the all-ones vector. We express the Lagrangian of this problem as follows:

$$L(\boldsymbol{\mu}, \lambda) = \boldsymbol{\mu}^\mathsf{T}\,\boldsymbol{\mu} - \lambda(\boldsymbol{\mu}^\mathsf{T}\,\mathbf{1} - l). \tag{7.35}$$

The corresponding conditions for optimality are

$$\frac{\mathrm{d}L(\boldsymbol{\mu}, \lambda)}{\mathrm{d}\lambda} = \boldsymbol{\mu}^\mathsf{T}\,\mathbf{1} - l = 0, \text{ and} \tag{7.36}$$

$$\frac{\mathrm{d}L(\boldsymbol{\mu}, \lambda)}{\mathrm{d}\boldsymbol{\mu}} = 2\boldsymbol{\mu} - \lambda\mathbf{1} = 0. \tag{7.37}$$

$$\tag{7.38}$$

These two expressions can be solved to show that the complexity of the sparse inner-product is minimized when the sparse support has a mean that is constant across all entries given by

$$\boldsymbol{\mu} = \frac{l}{N}\mathbf{1}. \tag{7.39}$$

Chapter 8
Conclusion

In this manuscript we have introduced a new dictionary structure which we call the Iteration-Tuned Dictionary (ITD). ITDs are layered structures containing a set of *candidate* dictionaries in each layer. An ITD-based iterative pursuit decomposition can then be carried out using, at each iteration i, one of the candidates from the i-th layer. We proposed a general ITD framework and three different ITD variants (Chapter 4 and Chapter 5): the Basic Iteration-Tuned Dictionary (BITD), the Tree-Structured Iteration-Tuned Dictionary (TSITD) and the Iteration-Tuned and Aligned Dictionary (ITAD). These structures were shown to outperform various state-of-the-art reference algorithms in their ability to render a dataset compressible, as well as in image denoising and compression applications.

An important question that remains is whether ITDs can produce good results in the many other applications where learned dictionaries have succeeded. Some of these applications not considered herein include source and texture separation, inpainting or prediction, classification, representation of color images, and online processing techniques. Much work has been carried out in these directions and to some extent it is just a question of adapting the existing formulations to the ITD case.

Another promising perspective is that of ITDs that can select multiple atom / coefficient pairs in each layer. The ODSAR algorithm developed in this work no longer provides the optimal candidates for this scenario, yet the many dictionary training methods available in the literature provide suitable alternatives. One simple way of addressing the distribution of the layers in this new setup is to allow selection of multiple atom / coefficient pairs in only the last layer L_M, thus gaining some control on the storage footprint of ITDs. Another important open question is that of the candidate selection law. The one we have used to derive TSITD and its variants offers several advantages, notably its simplicity and low complexity. Extensions of this same approach are possible that retain, nonetheless, these same advantages.

The ITAD image codec presented in Chapter 6 could also be improved in many ways. For example, the quantization scheme used is too simple and should be better adapted to the layered structure of the setup. The block size used by the codec also proved to be an important paramater and should thus be considered more carefully. Since the optimal block size is a function of the rate, one could

instead consider a scheme that employs an adapive block size. One potential approach to achieve this is to use a multi-scale ITD variant following an approach similar to that used for the multi-scale K-SVD variant. Other more promising approaches exist and are currently being tested.

The new approximate vector search method explored in Chapter 7 considered a new application of sparse representations aimed at exploiting the low complexity of computations using sparse data in the computationally demanding nearest-neighbor search application. We successfully addressed problems arising in this scenario related to the instability of the sparse support under residual affine-normalization errors. In developing this new approach we also developed a new data conditioning scheme that succeeds in better distributing data on the unit sphere while preserving the relative angles. The applicability of both of these tools in currently existing search schemes is an effort that we believe is worth pursuing.

Appendix A

Review of Selected Topics

In this appendix we review various selected topics coverins important tools used in work presented in this manuscript.

A.1 Matrix algebra

Let us consider a set of column vectors $\{\mathbf{m}\}$ from $\mathbb{R}^d$ or their equivalent representation as a matrix $\mathbf{M}$ with columns $\mathbf{m}$,

$$\mathbf{M} = \text{cols}(\{\mathbf{m}\}). \tag{A.1}$$

The vectors $\mathbf{m}$ are said to be *linearly independent* if it is impossible to express any of them as a linear combination of the others,

$$\mathbf{m}_i \notin \{\text{cols}(\{\mathbf{m}_j | j \neq i\})\mathbf{a} | \forall \mathbf{a}\}. \tag{A.2}$$

If the $\mathbf{m}$ are not linearly independent, then it is always possible to drop one or more of them to obtain a subset that is linearly independent. The *rank*

$$\text{rank}\,(r) \leq d \tag{A.3}$$

of $\mathbf{M}$ is the size of the largest such subset.

The rank of $\mathbf{M}$ is closely related to its *span*, denoted $\text{span}(\mathbf{M})$ or $\text{span}(\{\mathbf{m}\})$ and comprised of all linear combinations of the vectors $\mathbf{m}$:

$$\text{span}(\mathbf{M}) = \{\mathbf{a} | \mathbf{a} = \mathbf{M}\mathbf{b}\} \subseteq \mathbb{R}^d \tag{A.4}$$

with set equality only if $r = d$. The rank r in fact specifies the *dimensionality* of the subspace $\text{span}(\mathbf{M})$. A dimensionality r strictly less than d implies that it is possible to apply a rotation to all $\mathbf{a} \in \text{span}(\{\mathbf{m}\})$ to obtain an equivalent *reduced representation* $\mathbf{a}' \in \mathbb{R}^r$. This can be done given any r orthonormal basis vectors $\boldsymbol{\rho}$ of $\text{span}(\mathbf{M})$ by means of the rotation matrix $\boldsymbol{\phi} = \text{cols}(\{\boldsymbol{\rho}\}) \in \mathbb{R}^{d \times r}$:

$$\mathbf{a}' = \boldsymbol{\phi}^\mathsf{T}\,\mathbf{a}, \tag{A.5}$$

where the $(\cdot)^\mathsf{T}$ operator denotes the vector or matrix transpose.

The *left-null space* of $\mathbf{M}$ or, equivalently, the orthogonal-complement space of span($\mathbf{M}$), denoted span$^{\perp}(\mathbf{M})$ or span$^{\perp}(\{\mathbf{m}\})$, is defined as the set of all vectors $\mathbf{a}$ orthogonal to all columns of $\mathbf{M}$:

$$\text{span}^{\perp}(\mathbf{M}) = \{\mathbf{a} | \mathbf{a}^{\mathsf{T}}\mathbf{M} = \mathbf{0}\}. \tag{A.6}$$

It follows that the dimensionality of span$^{\perp}(\mathbf{M})$ is $d - r$.

A.2 The Singular Value Decomposition

Given a matrix $\mathbf{M}$ of arbitrary size $m \times n$, its singular value decomposition (SVD) is given by

$$\mathbf{M}_{m\times n} = \mathbf{U}_{m\times r}\mathbf{\Delta}_{r\times r}(\mathbf{V}_{n\times r})^{\mathsf{T}}, \tag{A.7}$$

where r is the rank of $\mathbf{M}$; $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices containing columns known, respectively, as the *left singular vectors* (otherwise known as the *principal vectors*) and *right singular vectors*; and the matrix

$$\mathbf{\Delta} = \text{diag}(\sigma_1, \ldots, \sigma_r)$$

is a diagonal matrix with non-negative diagonal entries σ_l known as the *singular values* [Klema 1980].

Without loss of generality, throught this manuscript it is assumed that singular values are ordered in decreasing order of magnitude:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r. \tag{A.8}$$

Note that this further implies that the left and right singular vectors are ordered accordingly and thus the i-th such vector contained along the i-th column (of either $\mathbf{U}$ or $\mathbf{V}$) corresponds to the singular value σ_i of i-th strongest magnitude.

We summarize some important properties of the singular value decomposition $\mathbf{M} = \mathbf{U}\mathbf{\Delta}\mathbf{V}^{\mathsf{T}}$ below. To simplify the presentation, we assume throughout that the singular values $\sigma_1, \ldots, \sigma_r$ are unique ($\sigma_1 > \sigma_2 > \ldots > \sigma_r$), which is indeed the case for practical data sets.

Property A.2.1 (Uniqueness)**.** *The matrices* $\mathbf{U}$*,* $\mathbf{V}$ *and* $\mathbf{\Delta}$ *in* (A.7) *are uniquely defined.*

Property A.2.2 (Best K-rank approximation)**.** *The first* $K \leq r$ *left-singular vectors produce the optimal,* K*-rank approximation of* $\mathbf{M}$*, i.e.,*

$$\begin{bmatrix}\mathbf{u}_1 & \ldots & \mathbf{u}_K\end{bmatrix}\left(\begin{bmatrix}\mathbf{u}_1 & \ldots & \mathbf{u}_K\end{bmatrix}^{\mathsf{T}}\mathbf{M}\right) = \underset{\mathbf{B}}{\text{argmin}}\,|\mathbf{M} - \mathbf{B}|_F^2, \tag{A.9}$$

where the squared Frobenius norm $|\cdot|_F^2$ *sums the square of all the matrix's elements and* $\mathbf{u}_l$ *denotes the* l*-th column of* $\mathbf{U}$*.*

Property A.2.3 (Principal components)**.** *The left singular vectors* $\mathbf{U}$ *of* $\mathbf{M}$ *are known as the* principal components *of the data set.*

Property A.2.4 (Rotation to trivial basis)**.** *The left singular vectors* $\mathbf{U}$ *define a rotation of* $\mathbf{M}$ *that aligns its principal components with the elementary basis* $\text{cols}(\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}) = \mathbf{I}$ *(where* I *is the identity matrix):*

$$\mathbf{U}^T \mathbf{J} = \mathbf{I}\boldsymbol{\Delta}\mathbf{V}^T. \tag{A.10}$$

Property A.2.5 (Preservation of principal components)**.** *Removing the contribution of a set of principal components* $\{\mathbf{u}_l | l \notin \mathcal{L}\}$ *preserves as principal components the complementary set* $\{\mathbf{u}_l | l \in \mathcal{L}\}$*. Thus the data matrix*

$$\mathbf{M} - \text{cols}(\{\mathbf{u}_l | l \notin \mathcal{L}\})\,\text{cols}(\{\mathbf{u}_l | l \notin \mathcal{L}\})^T \mathbf{M} \tag{A.11}$$

will have $\{\mathbf{u}_l | l \in \mathcal{L}\}$ *as its principal components.*

A.3 Information theory [Mallat 2008, Cover 1991]

A.3.1 Entropy coding

Entropy is an important concept in compression that provides a bound on the rate required to store or transmit a realization of a random data source. Consider a random variable x taking realizations in a finite alphabet of N symbols. Letting p_i denote the probability of occurrence of the i-th such symbol, the entropy of x is defined as

$$\mathcal{H}(x) = -\sum_i p_i \log_2(p_i). \tag{A.12}$$

If $\mathbb{R}(x)$ denotes the number of bits required to transmit x, then it is possible to show that

$$\mathrm{E}(\mathbb{R}(x)) \leq \mathcal{H}(x). \tag{A.13}$$

Various *Variable Length Codes* (VLCs) have been proposed that can come arbitrarily close to the entropy bound (A.13). VLCs encode each symbol i of the finite alphabet of x using a codeword $\mathbf{c}_i = \begin{bmatrix} b_1^i & \ldots & b_N^i i \end{bmatrix}^\mathsf{T}$ of varying bit-length N_i. In order for the decoder to be able to extract codewords from the received bit-stream, it is necessary that the codewords define a *prefix code*, meaning that a given codeword $\mathbf{c}_i$ of length N_i is not equal to the first N_i bits of any of the longer codewords $\mathbf{c}_j, N_j > N_i$:

$$\forall \mathbf{c}_i : \nexists \mathbf{c}_j, N_j > N_i, \text{ s.t. } \mathbf{c}_j = \left[\mathbf{c}_i \;\middle|\; b_{N_i+1}^j \;\ldots\; b_{N_j}^j\right]. \tag{A.14}$$

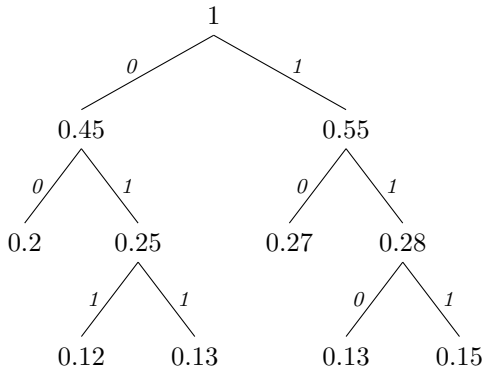

Figure A.1: An example of the Huffman prefix-code tree. The node labels denote the cumulative probabilities of all symbols associated to the descendant leafs. A codeword $\mathbf{c}_i$ is built by traversing the path to the leaf i and concatenating the bits of each branch.

As illustrated in Fig. A.1 prefix codes (A.14) can be organized into a tree-structure with each path i from the root to each i-th leaf defining codeword $\mathbf{c}_i$. It can be shown that there exists a prefix code that yields a mean transmission rate $\mathrm{E}\left(\mathbb{R}(x)\right) \leq \mathcal{H}\left(x\right) + 1$ The Huffman code is an example of one such code. Its construction proceeds from the bottom up by first arbitrarily assigning a symbol probability p_i to each tree-leaf. Then the two leafs with the lowest probabilities p_0 and p_1 are grouped to form one parent node with related probability $p_0 + p_1$, and the process is repeated each time replacing the two grouped children nodes by the new parent. Fig. A.1 illustrates the result for $\{p_i\} = \{0.12, 0.13, 0.13, 0.15, 0.2\}$.

A.3.2 Scalar quantization

In order to transmit or store infinite precission coefficients $\gamma \in \mathcal{G} \subseteq \mathbb{R}$ it is necessary to quantize them into a finite set of possible values. The resulting quantized representation $\hat{\gamma}$ is obtained using the quantization function

$$\mathrm{Q}(\gamma) = l_i \text{ s.t. } \gamma \in \mathcal{G}_i$$

defined by a partition of the range $\mathcal{G}$ of γ into N intervals $\mathcal{G}_i = (g_i, g_{i+1}]$ of length Δ_i and their corresponding reconstruction levels $l_i \in \mathcal{G}_i$. This quantization operation is lossy and incurs an average error

$$\mathrm{E}\left(|\gamma - \hat{\gamma}|^2\right) = \int_{\mathcal{G}} |\gamma - \hat{\gamma}|^2 p(\gamma)\,\mathrm{d}\gamma \tag{A.15}$$

$$= \sum_i \int_{\mathcal{G}_i} |\gamma - l_i|^2 p(\gamma)\,\mathrm{d}\gamma\,. \tag{A.16}$$

Hence the selection of the $\mathcal{G}_i$ and their reconstruction levels l_i needs to be done wisely.

To reduce the burden of the quantizer design it is common to adopt the *high resolution quantization* assumption (see [Mallat 1998] for an analysis of the low bit-rate case) which states that γ is uniformly distributed over each quantization bin $\mathcal{G}_i$:

$$p(\gamma) = p_i/\Delta_i, \tag{A.17}$$

where $p_i = \int_{\mathcal{G}_i} p(\gamma)\,\mathrm{d}\gamma$ is the probability that γ falls in $\mathcal{G}_i$. By substituting (A.17) into (A.16) it follows that the quantization levels l_i minimizing (A.16) are given by

$$l_i = \frac{1}{2} \cdot (g_{i+1} - g_i) \tag{A.18}$$

and the resulting distortion is

$$X = \sum_i \frac{1}{12} \cdot p_i \cdot \Delta_i. \tag{A.19}$$

This result along with the high-resolution assumption (A.17) leads to the following important rate-distortion relationship for high-resolution quantizers:

$$\mathcal{H}\left(\hat{\gamma}\right) \geq \mathcal{H}_d\left(\gamma\right) - \frac{1}{2}\log_2(12 \cdot X^2), \tag{A.20}$$

where the *differential entropy* $\mathcal{H}_d\left(\gamma\right)$ of the continous valued variable γ is defined as

$$\mathcal{H}_d\left(\gamma\right) = -\int_{\mathcal{G}} \gamma \log_2\left(p(\gamma)\right)\,\mathrm{d}\gamma\,. \tag{A.21}$$

The lower bound is attained when the quantizer is uniform, meaning that $\Delta_i = \Delta \forall i$.

Appendix B

Notational Conventions and Acronyms

Notational conventions

Symbol	Definition
General	
$\mathbf{v}, \mathbf{M}, \mathcal{S}$	Typeface denotes, respectively, vectors, matrices and sets
$\mathbf{v}[k], \mathbf{M}[k,l], \mathbf{M}[*,l]$	Respectively, entry k of vector $\mathbf{v}$, entry (k,l) of matrix $\mathbf{M}$, and column l of matrix $\mathbf{M}$
$\mathbf{v}^\mathsf{T}, \mathbf{M}^\mathsf{T}$	Vector and matrix transpose
$\mathbf{M}^+$	The Moore-Penrose pseudo-inverse
$\mathbf{0}, \mathbf{I}, \mathbf{e}_k$	Respectively, the zero vector or matrix, the identity matrix, and the k-th column of $\mathbf{I}$
$\triangleq$	Symbol definition
Norms	
$\lvert\mathcal{S}\rvert$	Cardinality of set $\mathcal{S}$; counts the number of elements
$\lvert\mathbf{v}\rvert$	Euclidean norm of vector $\mathbf{v}$
$\lvert\mathbf{v}\rvert_0$	l-0 norm of vector $\mathbf{v}$; counts non-zero entries
$\lvert\mathbf{M}\rvert_F$	Frobenius norm of matrix $\mathbf{M}$ given by the square root of the sum of squared entries
Matrix constructions	
$\mathrm{diag}(\sigma_1, \ldots, \sigma_N)$	The diagonal matrix with $\sigma_1, \ldots, \sigma_N$ along its main diagonal
$\mathrm{cols}(\{\mathbf{v}_i\})$	For $\mathbf{v}_i \in \mathbb{R}^d \ \forall i$ and $\lvert\{\mathbf{v}_i\}\rvert = N$, the $d \times N$ matrix $\begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_N \end{bmatrix}$

$\text{span}(\{\mathbf{v}_i\})$, $\text{span}(\mathbf{M})$	Column span of $\mathbf{M} = \text{cols}(\{\mathbf{v}_i\})$ (*i.e.,* $\{\mathbf{v} \vert \exists \mathbf{x}$ s.t. $\mathbf{v} = \mathbf{M}\mathbf{x}\}$)
$\text{span}^{\perp}(\{\mathbf{v}_i\})$, $\text{span}^{\perp}(\mathbf{M})$	Orthogonal complement of the column span of $\mathbf{M} = \text{cols}(\{\mathbf{v}_i\})$ (*i.e.,* the *left null space* of $\mathbf{M}$ given by $\{\mathbf{v} \vert \mathbf{v}^{\mathsf{T}}\mathbf{M} = \mathbf{0}\}$)
Sets	
$\mathbb{R}$	The real numbers
$\setminus$	The set difference operator; *eg.,* $\mathcal{S}_1 = \{1, 2, \ldots, 10\}$, $\mathcal{S}_2 = \{3, 4, \ldots, 10\}$, $\mathcal{S}_1 \setminus \mathcal{S}_2 = \{1, 2\}$
Probability theory	
$\mathrm{E}(\mathbf{x})$	Expectation of the random vector $\mathbf{x}$ with distritubtion $p(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$, given by $\int_{\mathbb{R}^d} \mathbf{x} p(\mathbf{x}) \, \mathrm{d}\mathbf{x}$
$\mathcal{H}(x)$	Entropy of a random variable $x \in \{x_i\}$ with probability mass function p_i, given by $-\sum_i p_i \log_2(p_i)$
$\mathcal{H}_d(x)$	Differential Entropy of a continous random variable $x \in \mathbb{R}$ with distribution $\mathbf{x}$, given by $-\int_{\mathcal{G}} x \log_2(p(x)) \, \mathrm{d}x$.

Table B.1: List of notational conventions.

Acronyms

Accronym	Definition
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
ANN	Approximate Nearest Neighbors
MSER	Maximally-Stable Extremal Region
ITD	Iteration-Tuned and Aligned Dictionary
BITD	Basic Iteration-Tuned Dictionary
TSITD	Tree-Structured Iteration-Tuned Dictionary
rTSITD	reduced Tree-Structured Iteration-Tuned Dictionary
SD	Sparse Dictionary, refers to the method presented in [Rubinstein 2010a]
ONLD	Online Learned Dictionary, refers to the method presented in [Mairal 2010a]

Table B.2: List of acronyms.

Appendix C

Article submissions

The work carried out in this thesis has led to various article submissions, some of which are still under review process at the time of final printing. These are the following:

From Chapter 4, [Zepeda 2010d] (accepted) and [Zepeda 2010c] (under review). From Chapter 5, [Zepeda 2010b] (under review). From Chapter 6, [Zepeda 2011] (accepted) and [Zepeda 2010a] (under review). From Chapter 7, [Zepeda 2009] (accepted) and [Zepeda 2010e] (accepted).

Bibliography

[Adams 2005] Michael D. Adams. *The JPEG-2000 Still Image Compression Standard*. Rapport technique, Dept. of Electrical and Computer Engineering, University of Victoria, 2005.

[Aharon 2006a] Michal Aharon, Michael Elad and Alfred Bruckstein. *K-SVD toolbox*, 2006.

[Aharon 2006b] Michal Aharon, Michael Elad and Alfred Bruckstein. *K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation*. IEEE Transactions on Signal Processing, vol. 54, pages 4311–4322, 2006.

[Aharon 2008] Michal Aharon and Michael Elad. *Sparse and redundant modeling of image content using an image-signature-dictionary*. SIAM J. IMAGING SCIENCES, vol. 1, no. 3, pages 228–247, 2008.

[Bay 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool. *SURF: Speeded Up Robust Features*. Computer Vision and Image Understanding, vol. 110, no. 3, pages 346–359, 2008.

[Boyd 2004] Stephen Boyd and Lieven Vandenberghe. Convex optimization. Cambridge University Press, March 2004.

[Bryt 2008] Ori Bryt and Michael Elad. *Compression of facial images using the K-SVD algorithm*. J. Vis. Comun. Image Represent., vol. 19, no. 4, pages 270–282, 2008.

[Chatterjee 2010] Priyam Chatterjee and Peyman Milanfar. *Fundamental limits of image denoising: Are we there yet?* In 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP),, pages 1358–1361, 2010.

[Chen 2001] Scott Shaobing Chen, David L. Donoho and Michael A. Saunders. *Atomic Decomposition by Basis Pursuit*. SIAM Rev., vol. 43, no. 1, pages 129–159, 2001.

[Cotter 1999] S.F. Cotter, R. Adler, R.D. Rao and K. Kreutz-Delgado. *Forward sequential algorithms for best basis selection*. In Vision, Image and Signal Processing, IEE Proceedings -, volume 14, 1999.

[Cover 1991] Thomas M. Cover and Joy A. Thomas. Elements of information theory. Wiley-Interscience, New York, NY, USA, 1991.

[Elad 2005] M. Elad, J.-L. Starck, P. Querre and D.L. Donoho. *Simultaneous Cartoon and Texture Image Inpainting using Morphological Component Analysis (MCA).* Applied and Computational Harmonic Analysis, vol. 19, no. 3, pages 340–358, November 2005.

[Elad 2006a] M. Elad and M. Aharon. *Image denoising via learned dictionaries and sparse representation.* In CVPR, 2006.

[Elad 2006b] M. Elad and M. Aharon. *Image Denoising Via Sparse and Redundant representations over Learned Dictionaries.* IEEE Trans. on Image Processing, vol. 15, no. 12, pages 3736–3745, December 2006.

[Engan 1999] K. Engan, S.O. Aase and J.H. Hakon-Husoy. *Method of optimal directions for frame design.* IEEE International Conference on Acoustics, Speech, and Signal Processing., vol. 5, pages 2443–2446, 1999.

[Goyal 1997] Vivek K Goyal and Martin Vetterli. *Dependent coding in quantized matching pursuit.* In in Proceedings of the SPIE - Visual Communication and Image Processing, pages 2–12, 1997.

[Guleryuz 2006] O.G. Guleryuz. *Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising-part I: theory.* IEEE Transactions on Image Processing, vol. 15, pages 539–554, March 2006.

[Harris 1998] C. Harris and M. Stephens. *A combined corner and edge detector.* In Proceedings of the 4th Alvey Vision Conference, pages 147–151, 1998.

[Huffman 1952] D.A. Huffman. *A Method for the Construction of Minimum-Redundancy Codes.* In Proceedings of the I.R.E., pages 1098–1102, 1952.

[Jégou 2008] Hervé Jégou, Matthijs Douze and Cordelia Schmid. *Hamming embedding and weak geometric consistency for large scale image search.* In ECCV, volume I, pages 304–317, 2008.

[Jenatton 2009] R. Jenatton, J. Mairal, G. Obozinski and F. Bach. *Proximal Methods for Sparse Hierarchical Dictionary Learning.* In Proceedings of the International Conference on Machine Learning (ICML), 2009.

[Jia 2008] Zhen Jia, Laurent Amsaleg and Patrick Gros. *Content-based image retrieval from a large image database.* Pattern Recogn., vol. 41, no. 5, pages 1479–1495, 2008.

[Jost 2006] P. Jost, P. Vandergheynst and P. Frossard. *Tree-Based Pursuit: Algorithm and Properties*. Signal Processing, IEEE Transactions on, vol. 54, no. 12, pages 4685–4697, Dec. 2006.

[Jost 2008] Philippe Jost and Pierre Vandergheynst. *On finding approximate nearest neighbours in a set of compressible signals*. In EUSIPCO'08, 2008.

[Ke 2004] Yan Ke and R. Sukthankar. *PCA-SIFT: a more distinctive representation for local image descriptors*. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–506–II–513 Vol.2, 2004.

[Klema 1980] Virginia C. Klema and Alan J. Laub. *The Singular Value Decomposition: Its Computation and Some Applications*. IEEE Transactions on Automatic Control, vol. AC-25, no. 2, pages 164–176, April 1980.

[Kokiopoulou 2008] Effrosyni Kokiopoulou and Pascal Frossard. *Minimum distance between pattern transformation manifolds: Algorithm and Applications*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.

[Lesage 2005] S. Lesage, R. Gribonval, F. Bimbot and L. Benaroya. *Learning unions of orthonormal bases with thresholded singular value decomposition*. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), volume 5, 2005.

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. Int. J. Comput. Vision, vol. 60, no. 2, pages 91–110, 2004.

[Mairal 2008a] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro and Andrew Zisserman. *Discriminative Learned Dictionaries for Local Image Analysis*. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[Mairal 2008b] Julien Mairal, Michael Elad and Guillermo Sapiro. *Sparse Representation for Color Image Restoration*. IEEE Trans. on Image Processing, vol. 17, no. 1, pages 53–69, January 2008.

[Mairal 2008c] Julien Mairal, Guillermo Sapiro and Michael Elad. *Learning Multiscale Sparse Representations for Image and Video Restoration*. Multiscale Modeling & Simulation, vol. 7, no. 1, pages 214–241, 2008.

[Mairal 2009] J. Mairal, F. Bach, J. Ponce and G. Sapiro. *Online dictionary learning for sparse coding*. In International Conference on Machine Learning (ICML), 2009.

[Mairal 2010a] J. Mairal, F. Bach, J. Ponce and G. Sapiro. *Online Learning for Matrix Factorization and Sparse Coding*. Journal of Machine Learning Research, vol. 11, pages 10–60, 2010.

[Mairal 2010b] Julien Mairal. *SPArse Modeling Software (SPAMS)*, 2010.

[Mallat 1993] S.G. Mallat and Zhifeng Zhang. *Matching pursuits with time-frequency dictionaries*. Signal Processing, IEEE Transactions on, vol. 41, no. 12, pages 3397–3415, Dec 1993.

[Mallat 1998] S. Mallat and F. Falzon. *Analysis of low bit rate image transform coding*. Signal Processing, IEEE Transactions on, vol. 46, no. 4, pages 1027 –1042, apr 1998.

[Mallat 2008] Stephane Mallat. A wavelet tour of signal processing, 3rd ed., third edition: The sparse way. Academic Press, 3 édition, December 2008.

[Matas 2002] J. Matas, O. Chum, M. Urban and T. Pajdla. *Robust wide baseline stereo from maximally stable extremal regions*. In In British Machine Vision Conference, pages 384–393, 2002.

[Mikolajczyk 2005a] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool. *A Comparison of Affine Region Detectors*. IJCV, vol. 65, no. 1-2, pages 43–72, 2005.

[Mikolajczyk 2005b] Krystian Mikolajczyk and Cordelia Schmid. *A Performance Evaluation of Local Descriptors*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 10, pages 1615–1630, 2005.

[Muja 2009] Marius Muja and David G. Lowe. *Fast approximate nearest neighbors with automatic algorithm configuration*. In International Conference on Computer Vision Theory and Applications (VISAPP), 2009.

[Nister 2006] David Nister and Henrik Stewenius. *Scalable recognition with a vocabulary tree*. In CVPR, 2006.

[Olshausen 1996] B. A. Olshausen and B. J. Field. *Natural image statistics and efficient coding*. Network Comp. Neural Syst., vol. 7, no. 2, pages 333–339, 1996.

[Pati 1993] Y. C. Pati, R. Rezaiifar and P. S. Krishnaprasad. *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition*. In A. Singh, editeur, Proc. 27th Asilomar Conference on Signals, Systems and Computers. IEEE Comput. Soc. Press, Los Alamitos, CA, 1993.

[Peyré 2010] Peyré, Gabriel, Fadili, Jalal, Starck and Jean Luc. *Learning the Morphological Diversity*. SIAM Journal on Imaging Sciences, 7 2010.

[Philbin 2008] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman. *Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[Phillips 2000] P. J. Phillips, H. Moon, P. J. Rauss and S. Rizvi. *The FERET evaluation methodology for face recognition algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 10, October 2000.

[Protter 2009] M. Protter and Michael Elad. *Image Sequence Denoising Via Sparse and Redundant Representations*. IEEE Trans. on Image Processing, vol. 18, no. 1, pages 27–36, January 2009.

[Rath 2004] G. Rath and C. Guillemot. *Subspace-Based Error and Erasure Correction with DFT Codes for Wireless Channels*. IEEE Trans on Signal Processing, vol. 52, no. 11, nov 2004.

[Rebollo-Neira 2002] Laura Rebollo-Neira and David Lowe. *Optimized Orthogonal Matching Pursuit Approach*. IEEE Signal Processing Letters, vol. 9, no. 4, pages 137–140, April 2002.

[Ripley 1996] Brian D. Ripley. Pattern recognition and neural networks. Cambridge University Press, January 1996.

[Rubinstein 2010a] Rubinstein, Ron, Zibulevsky, Michael, Elad and Michael. *Double sparsity: learning sparse dictionaries for sparse signal approximation*. Trans. Sig. Proc., vol. 58, no. 3, pages 1553–1564, 2010.

[Rubinstein 2010b] R. Rubinstein, A.M. Bruckstein, and M. Elad. *Dictionaries for Sparse Representation Modeling*. IEEE Proceedings - Special Issue on Applications of Compressive Sensing & Sparse Representation, 2010. (Accepted).

[Rubinstein 2010c] Ron Rubinstein. *Sparse K-SVD toolbox*, 2010.

[Sezer 2008] Osman G. Sezer, Oztan Harmanci and Onur G. Guleryuz. *Sparse Orthonormal Transforms for Image Compression*. In Proc. IEEE Int'l Conf. on Image Proc., 2008.

[Sivic 2003] J. Sivic and A. Zisserman. *Video Google: A Text Retrieval Approach to Object Matching in Videos*. In Proceedings of the ICCV, volume 2, pages 1470–1477, October 2003.

[Starck 2004a] J.-L. Starck, M. Elad and D.L. Donoho. *Image Decomposition via the Combination of Sparse Representations and a Variational Approach*. IEEE Transactions on Image Processing, vol. 14, pages 1570–1582, 2004.

[Starck 2004b] J.-L. Starck, M. Elad and D.L. Donoho. *Redundant multiscale transforms and their application for morphological component analysis*. Advances in Imaging and Electron Physics, vol. 132, 2004.

[Sullivan 2005] Gary J. Sullivan and Thomas Wiegand. *Video Compression - From Concepts to the H.264/AVC Standard*. PROCEEDINGS OF THE IEEE, vol. 93, no. 1, pages 18–31, January 2005.

[Tropp 2004] J. A. Tropp. *Greed is good: algorithmic results for sparse approximation*. Information Theory, IEEE Transactions on, vol. 50, no. 10, pages 2231–2242, 2004.

[Wallace 1991] Gregory K. Wallace. *The JPEG still picture compression standard*. Commun. ACM, vol. 34, no. 4, pages 30–44, 1991.

[Yu 2010] Guoshen Yu, Guillermo Sapiro and Stéphane Mallat. *Solving Inverse Problems with Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity*. IEEE Trans. on Image Processing, 2010. (Submitted).

[Zepeda 2006] Joaquin Zepeda and Fabrice Labeau. *Tandem Filter Bank-DFT Code for Bursty Erasure Correction*. In Proc. IEEE Vehicular Technology Conference, 2006.

[Zepeda 2009] Joaquin Zepeda, Ewa Kijak and Christine Guillemot. *SIFT-Based Local Image Description using Sparse Representations*. In Proceedings of the IEEE International Workshop on MMSP, 2009.

[Zepeda 2010a] Joaquin Zepeda, Christine Guillemot and Ewa Kijak. *Image Compression Using Sparse Representations and the Iteration-Tuned and Aligned Dictionary*. IEEE Journal of Selected Topics in Signal Processing, 2010. (Submitted).

[Zepeda 2010b] Joaquin Zepeda, Christine Guillemot and Ewa Kijak. *The Iteration-Tuned and Aligned Dictionary and its Application in Image Compression and Denoising.* IEEE Transactions on Image Processing, 2010. (Submitted).

[Zepeda 2010c] Joaquin Zepeda, Christine Guillemot and Ewa Kijak. *Iteration-Tuned Dictionaries for Sparse Representations and their Tree-Structured Variant.* IEEE Trans. on Signal Processing, 2010. (Submitted).

[Zepeda 2010d] Joaquin Zepeda, Christine Guillemot and Ewa Kijak. *The Iteration-Tuned Dictionary for Sparse Representations.* In Proc. of the IEEE International Workshop on MMSP, 2010.

[Zepeda 2010e] Joaquin Zepeda, Ewa Kijak and Christine Guillemot. *Approximate nearest neighbors using sparse representations.* In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 2010.

[Zepeda 2011] Joaquin Zepeda, Christine Guillemot and Ewa Kijak. *Image Compression using the Iteration-Tuned and Aligned Dictionary.* In Proceedings of ICASSP, 2011. (Submitted).

[Zezula 2006] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal and Michael Batko. Similarity search: the metric space approach. New York : Springer, 1 édition, 2006.

[Zhao 2009] Peng Zhao, Guilherme Rocha and Bin Yu. *The composite absolute penalties family for grouped and hierarchical variable selection.* Annals of Statistics, vol. 37, no. 6A, pages 3468–3497, 2009.

[Zobel 2006] Justin Zobel and Alistair Moffat. *Inverted files for text search engines.* ACM Comput. Surv., vol. 38, no. 2, page 6, 2006.

Nouvelles méthodes de représentations parcimonieuses ; Application à la compression et l'indexation d'images

Résumé : Une nouvelle structure de dictionnaire adaptés aux décompositions itératives de type poursuite, appelée un *Iteration-Tuned Dictionary* (ITD), est présentée. Les ITDs sont structurés en couche, chaque couche se composant d'un ensemble de *dictionnaires candidats*. Les décompositions itératives basées ITD sont alors réalisées en sélectionnant, à chaque itération i, l'un des dictionnaires de la i-ème couche. Une structure générale des ITDs est proposée, ainsi qu'une variante structurée en arbre appelée *Tree-Structured Iteration-Tuned Dictionary*(TSITD) et une version contrainte de cette dernière, appelée *Iteration-Tuned and Aligned Dictionary* (ITAD). Ces structures sont comparées à plusieurs méthodes de l'état de l'art et évaluées dans des applications de débruitage et de compression d'images. Un codec basé sur le schéma ITAD est également présenté et comparé à JPEG2000 dans des évaluations qualitatives et quantitatives.

Dans le contexte de l'indexation d'images, un nouveau système de recherche approximative des plus proches voisins est également introduit, qui utilise les représentations parcimonieuses pour réduire la complexité de la recherche. La méthode traite l'instabilité dans la sélection des atomes lorsque l'image est soumise à de faibles transformations affines. Un nouveau système de conditionnement des données est également introduit, permettant de mieux distribuer les données sur la sphère unitaire tout en préservant leurs distances angulaires relatives. Il est montré que cette méthode améliore le compromis complexité/performance de la recherche approximative basée décompositions parcimonieuses.

Mots-clés : Apprentissage de dictionnaires, représentations parcimonieuses, Matching Pursuit, réduction de la dimension, décomposition en valeurs singulières, parcimonie structurée, compression d'images, débruitage d'images, recherche d'images, recherche approximative des plus proches voisins, conditionnement de données

Novel Sparse Representation Methods and their Application in Image Indexing and Compression

Abstract: A new dictionary structure is introduced called an Iteration-Tuned Dictionary (ITD). ITDs are layered structures containing a set of *candidate* dictionaries in each layer. ITD-based iterative pursuit decompositions are carried out using, at each iteration i, one of the candidates from the i-th layer. A general ITD framework is proposed as well as a tree-structured variant called the Tree-Structured Iteration-Tuned Dictionary (TSITD) and a constrained tree-structured variant called the Iteration-Tuned and Aligned Dictionary (ITAD). These structures are shown to outperform various state-of-the-art reference algorithms in their ability to approximate a dataset sparsely, and in the applications of image denoising and image compression. The ITAD scheme, in particular, is used to develop an image codec that outperforms JPEG2000.

An approximate vector search method is also introduced which uses sparse representations to carry out low-complexity approximate nearest-neighbor image searches. The approach addresses the related instability of the sparse support when the image patch is subject to weak affine transformations. In developing this new approach, a new data conditioning scheme is introduced that succeeds in better distributing data on the unit sphere while preserving relative angles. It is shown that this new approach improves the complexity/performance tradeoff of approximate searches based on sparse representations.

Keywords: Dictionary learning, sparse representations, pursuit methods, dimensionality reduction, singular-value decomposition, structured sparsity, image compression, image denoising, image search, approximate-nearest neighbors, data conditioning