Novel texture synthesis methods and their application to image prediction and image inpainting

Mehmet Turkan

N^o d'ordre : 4410 ANNÉE 2011

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 : Informatique

École doctorale Matisse

présentée par

Mehmet TÜRKAN

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

Nouvelles méthodes de synthèse de texture ; application à la prédiction et à l'inpainting d'images

Thèse soutenue à Rennes le 19 décembre 2011

devant le jury composé de :

Béatrice PESQUET-POPESCU
Professeur, Télécom ParisTech / *Présidente*
A. Enis ÇETİN
Professeur, Université de Bilkent / *Rapporteur*
Peter SCHELKENS
Professeur, Université Libre de Bruxelles / *Rapporteur*
Jean-Jacques FUCHS
Professeur, Université de Rennes 1 / *Examinateur*
Patrick PÉREZ
Chercheur, Technicolor R&D / *Examinateur*
Christine GUILLEMOT
Directrice de recherche, INRIA / *Directrice de thèse*

Victory is for those who can say "Victory is mine."
Success is for those who can begin saying "I will succeed"
and say "I have succeeded" in the end.

Mustafa Kemal Atatürk

To my family

Acknowledgments

I would like to thank Prof. Béatrice Pesquet-Popescu, Prof. Ahmet Enis Çetin, Prof. Peter Schelkens, Prof. Jean-Jacques Fuchs, and Dr. Patrick Pérez for accepting to be members of the Ph.D. jury and for taking time to read and review this manuscript. Their suggestions have been taken into account and have significantly improved the final version.

I would like to express my deep gratitude to my supervisor, Dr. Christine Guillemot, first for accepting me as a Ph.D. student, for making available all the resources needed to carry out the work in this thesis, and for opening the doors to the various interesting topics treated in this work. Her experience on research and also patience in supervising me made this thesis more valuable.

I would like to thank Prof. Ahmet Enis Çetin for supporting me constantly in all steps of my academic progress. Besides being my former supervisor, special thanks to him for always giving positive and helpful advices.

Special thanks to Dr. Olivier Le Meur and Dr. Aline Roumy for being always helpful and for enlightening me with their ambition and enthusiasm on research.

Special thanks also go to Dr. Kamil Adiloğlu and Katja Adiloğlu, and also Dr. Mehmet Ali Avcı and Florence Avcı for being next to me in the past three years.

Many thanks to Simon Bos, Julien Fayolle, Marco Bevilacqua, and Josselin Gautier for being real friends and for helping me in solving any kind of problems without questioning or expecting any return.

I would also thank my former and current labmates in TEMICS group:
Dr. Jonathan Taquet, Vincent Jantet, Dr. Ana Charpentier, Dr. Thomas Colleu, Dr. Simon Malinowski, Dr. Gagan Rath, Dr. Denis Kubasov, Dr. Velotiaray Toto-Zarasoa, Dr. Raul Martinez Noriega, Dr. Joaquin Zepeda Salvatierra.

Also friends in other groups: Gylfi Thór Gudmundsson, Dr. Bogdan Ludusan.

Last but not the least, I would like to thank my family for supporting me spiritually throughout my life.

Contents

List of Figures

List of Tables

Glossary

(G)PCA	(Generalized) Principle Component Analysis
ATM	Average Template Matching
AVC	Advanced Video Coding
BCD	Block–Coordinate Descent
BP	Basis Pursuit
CMP	Complementary Matching Pursuit
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
HEVC	High Efficiency Video Coding
i.i.d.	Independent and Identically Distributed
ICA	Independent Component Analysis
KLT	Karhunen–Loeve Transform
LARS	Least Angle Regression
LLE	Locally Linear Embedding
LSE	Least Square Estimation
MDS	Multidimensional Scaling
MOD	Method of Optimal Directions
MP	Matching Pursuit
MRF	Markov Random Field
MSE	Mean Square Error
NLM	Non–local Means
NMF	Non–negative Matrix Factorization
OCMP	Orthogonal Complementary Matching Pursuit
OFD	On–the–Fly Dictionary
OMP	Orthogonal Matching Pursuit
OOCMP	Optimized Orthogonal Complementary Matching Pursuit
OOMP	Optimized Orthogonal Matching Pursuit
PDE	Partial Differential Equation
PSNR	Peak Signal–to–Noise Ratio
RD	Rate Distortion
SAE	Sum of Absolute Error
SOT	Sparse Orthonormal Transforms
SP	Sparse Prediction
SSE	Sum of Square Error
STFT	Short Time Fourier Transform
StOMP	Stagewise Orthogonal Matching Pursuit
SVD	Singular Value Decomposition
TM	Template Matching

Chapter 1

Résumé en Français

1.1 Introduction

L'analyse et la synthèse de texture sont des domaines de recherches importants pour de nombreuses applications dans la communauté traitement d'image (*computer graphics*), vision par ordinateur (*computer vision*), imagerie numérique et traitement vidéo. En ce qui concerne le *computer graphics*, la synthèse de texture est utilisée pour reproduire en détails l'apparence d'une surface en plaquant une texture synthétisée réaliste. Généralement, le plaquage d'une texture sur une surface convexe s'effectue en déformant cette texture ou en synthétisant une autre texture adaptée à la surface. En vision par ordinateur, le but de l'analyse et la synthèse de texture est d'abord de comprendre, de modéliser puis de traiter la texture. Concernant le domaine du traitement d'images et de la vidéo, la synthèse de texture est une méthode pour générer des textures. Plus précisément, connaissant un petit échantillon de texture, l'idée est synthétiser une texture de très grande taille réaliste présentant des caractéristiques similaires à l'échantillon et sans artefact visible.

Avec l'émergence de nouvelles techniques de calcul rapides et efficaces, les algorithmes d'analyse et de synthèse de texture peuvent être appliqués à de nombreuses applications dans le domaine du traitement d'images et de la vidéo telles que l'édition d'images et vidéo (suppression d'objet dans une scène, dissimulation d'erreurs, etc), l'amélioration (applications liées à la super-résolution) et le débruitage d'images, pour la compression d'images et de vidéo (prédiction intra et inter images).

Dans ce manuscrit, nous nous intéressons essentiellement aux méthodes de synthèse de texture dans un contexte de prédiction intra image et dans un contexte d'inpainting pour de l'édition d'images (suppression d'objet) et de l'amélioration d'images.

1.1.1 Prédiction et compression d'images de vidéo

Depuis le théorème de Shannon [1], le domaine de la compression de données reste un domaine de recherche très actif dans lequel de nombreuses avancées ont été effectuées. Le but de la compression est de réduire la redondance spatiale et temporelle de données afin d'être capable de transmettre ou d'archiver de grandes quantités d'information. Bien qu'il existe aujourd'hui de nombreux outils mathématiques (allant de simples heuristiques à des approches sophistiquées) offrant des solutions rapides et efficaces de compression, le besoin de compression reste fondamental du fait de la demande croissante de transmission et d'archivage de données.

Deux types de compression sont possibles: l'une dite sans perte et l'autre avec perte. La compression sans perte permet de préserver l'intégrité des données compressées, ainsi il est possible de retrouver les données exactes. Ce type de compression est utilisée dans de nombreuses applications telles que l'imagerie médicale, le format d'archivage ZIP et même comme étape de codage dans un schéma de compression image et vidéo avec pertes. Contrairement au codage sans perte, la compression avec pertes ne permet pas de retrouver les données originales. Elle fournit seulement une approximation des données à compresser au profit d'un meilleur taux de compression. Dans ce type de compression, les informations jugées non nécessaires sont simplement supprimées de façon irréversible. La plupart des applications de compression cherchent à compresser les données avec pertes en optimisant le compromis débit-distorsion.

Un schéma de codage typique de compression avec pertes est composé d'un ensemble d'outils mathématiques suivi d'une transformation des données à compresser en un train binaire. La première étape d'un schéma de compression cherche à décorréler les données à coder. Cette étape peut être perçue comment un traitement réduisant la redondance dans le signal. Les techniques utilisées pour décorréler les données sont des *transformations* telles que la transformée en cosinus discrète. Les transformées projettent les données dans un autre domaine de représentation afin d'obtenir une représentation compacte de l'information. En effet, l'énergie des données transformées est souvent concentrée sur un nombre limité de coefficients. Par ailleurs, le fait de transformer les données permet d'identifier plus aisément les informations visuellement importantes des informations moins importantes. Le procédé suivant la transformée consiste à éliminer les informations les moins importantes préservant la qualité des données visuellement importantes. Il s'agit de l'étape de *quantification* des données. Les données transformées sont donc quantifiées afin d'être représentées sur moins de bits. Cette perte d'information est irréversible. Finalement, les données transformées quantifiées sont converties en train binaire et transmises au décodeur.

Dans cette étude, nous nous intéressons à une méthode de réduction d'information appelée *codage prédictif*. Le codage prédictif est utilisé pour estimer la valeur d'un pixel à partir de son contexte, c'est-à-dire en utilisant les pixels déjà reconstruits du voisinage causal. L'estimation des valeurs des pixels inconnus (pixels d'un bloc à encoder par exemple) s'effectue à partir des pixels voisins connus, et une erreur de prédiction est mesurée en calculant la différence (pixel par pixel) entre les pixels estimés et les pixels originaux. Cette erreur de prédiction est ensuite transformée, encodée et transmise. L'utilisation d'un codage prédictif permet de réduire significativement la quantité d'information à transmettre. Bien évidement, la réduction d'information dépend de la qualité de l'opérateur de prédiction, c'est-à-dire de la méthode de synthèse ou de prédiction. Cette étape de prédiction diffère peu de la synthèse de texture. La différence concerne essentiellement l'utilisation d'une erreur de prédiction permettant de compenser les défauts de la prédiction.

Le codage prédictif a été très étudié dans la littérature; les opérateurs de prédiction sont nombreux et plus ou moins complexes allant d'une simple interpolation (ou extrapolation) à des techniques bien plus complexes. A titre d'exemple, neuf modes de prédiction intra (pour des blocs de tailles 4×4) sont disponibles dans la norme de codage H.264/AVC. Ces prédicteurs propagent suivant une direction donnée les pixels précédemment décodés. L'encodeur détermine le meilleur mode en fonction d'un critère débit-distorsion. Pour la prédiction intra H.264/AVC, les coefficients des filtres d'interpolation pour chaque direction de prédiction sont précalculés et permettent d'obtenir de bonnes performances sur des textures simples. Cependant, lorsque les textures sont plus complexes, la prédiction est moins efficace du fait que les coefficients des filtres d'interpolation sont fixes.

Une alternative à ce type de méthode pour la prédiction est basée sur les champs aléatoires de Markov, aussi appelés *template matching*. Un *template* est supposé être composé de pixels connus localisés dans un voisinage proche du bloc à prédire. La recherche de la meilleure correspondance entre le *template* et les textures situées dans un large voisinage causal permet de prédire par une simple opération de "copier-coller" les pixels à encoder. Le procédé de *template matching* peut être considéré comme une extension de synthèse de texture basé sur l'*exemple*. La recherche de la meilleure correspondance texture se fait dans un voisinage causal du bloc à prédire dans la même image au lieu d'utiliser un petit échantillon de texture.

Une extension du *template matching* consiste à combiner linéairement plusieurs candidats. L'idée sous-jacente repose sur le théorème de la limite centrale stipulant que pour des échantillons linérairement indépendants, la moyenne de plusieurs candidats tend vers une distribution Gaussienne caractérisée par une variance inversement proportionnelle au nombre de candidats utilisés. Ainsi, statistiquement, l'erreur de prédiction peut être inférieure à celle obtenue avec un *template matching* simple. L'approche la plus simple pour combiner les candidats est de leur affecter une pondération uniforme. Cette méthode est appelée *template matching moyen*. Plus récemment, une approche appelée moyennes non locales (*non-local means*) a été proposée pour effectuer de la synthèse de texture. De façon analogue au *template matching moyen*, l'approche des moyennes non locales combine linéairement les différents candidats. Cependant, les poids de la combinaison sont dépendants de la similarité entre les pixels du *template* et ceux des candidats se traduisant par une importance plus forte donnée aux candidats proches du *template*.

Les approches brièvement décrites ci-dessus, combinant plusieurs candidats, sont plus robustes et donnent globalement de meilleurs résultats que les approches utilisant un seul candidat. Cependant, ces approches ne cherchent pas à minimiser l'erreur d'approximation du *template*. Elles se basent plutôt sur des heuristiques de calcul afin de déterminer des coefficients de pondération pertinents. L'idée principale de ce mémoire de thèse est d'utiliser des techniques d'optimisations formulant le problème comme un problème aux moindres carrés avec différents types de contraintes. Les coefficients de pondération des candidats sont alors le résultat de l'optimisation. La motivation sous-jacente peut être exprimée de la façon suivante: "*Une bonne approximation* *des pixels connus du template doit permettre également d'avoir une bonne approximation des pixels inconnus de bloc à prédire.*" De cette façon, nous croyons que l'on peut améliorer la qualité de la prédiction et les performances globales de compression avec certaines techniques d'optimisation supplémentaires telles que la sélection du nombre de candidats de texture à être utilisés pour la prédiction, ou l'adaptation des modes de prédictions supplémentaires, et ainsi de suite.

Pour cela, dans le chapitre 3, nous plaçons le problème de prédiction dans un contexte de prédiction parcimonieuse, c'est-à-dire que l'optimisation sera réalisée avec des contraintes de parcimonie. Le terme parcimonie convient particulièrement bien au problème posé puisqu'une sélection de plusieurs candidats parmi un grand nombre de candidats potentiels est effectuée. Ensuite dans le chapitre 4, nous utilisons des méthodes d'apprentissage (pour des représentations parcimonieuses de données) pour définir un dictionnaire adapté au problème de prédiction. La version simplifiée de la méthode proposée peut être considérée comme un problème aux moindres carrés sans contrainte. Finalement, dans le chapitre 5, le problème de prédiction est placé dans un contexte de *neigbor embedding* en adaptant deux méthodes de réduction de dimensionnalité appelé "*locally linear embedding*" et "factorisation de matrice non négative". Ces deux méthodes sont deux problèmes aux moindres carrés avec deux contraintes différentes, l'une de non négativité et l'autre une contrainte sur la somme des coefficients qui doit valoir 1.

1.1.2 Inpainting d'images fixes

L'inpainting d'images (aussi connu sous le nom d'interpolation d'images ou de vidéo) est une application des méthodes de synthèse de texture utilisée pour remplacer (ou reproduire) les zones perdues ou corrompues d'une image (ce sont principalement des régions ou des défauts de faibles tailles). Les techniques d'inpainting sont utilisées dans de nombreuses applications et plus particulièrement pour les applications d'édition d'images (suppression d'objets, dissimulation d'erreurs, amélioration, etc).

Pour les algorithmes d'inpainting basés sur l'exemple, les valeurs inconnues de la zone à reconstruire sont obtenues en cherchant dans l'image source un candidat pour lequel la similarité entre pixels connus est maximale. Comme précédemment, ce procédé repose sur une approche *template matching*. Nous faisons remarquer que les extensions du *template matching* décrites précédemment sont également utilisées en inpainting. En effet, il y a une forte similarité entre les problèmes d'inpainting et de prédiction. Les méthodes d'inpainting utilisent donc les approches *template matching* et également des techniques plus récentes basées sur des approximations parcimonieuses. Cependant, il existe une importante différence puisque, pour l'inpainting, les échantillons pouvant être utilisés ne sont pas contraints spatialement et peuvent ne pas appartenir au voisinage causal. Les zones inconnues peuvent donc être remplies en parcourant l'image de gauche à droite et de haut en bas ou non. L'inpainting basé sur l'exemple définit une priorité de remplissage de façon à propager d'abord les structures importantes de l'image. En prédiction, cela n'est pas possible puisque l'ordre de traitement est en "raster scan". Les candidats possibles pour la prédiction appartiennent donc forcément au voisinage causal du bloc à prédire. La priorité, telle que définie pour les algorithmes d'inpainting, ne peut être utilisée facilement en prédiction. D'un autre côté, pour l'inpainting, il n'y a pas de notion d'erreur de prédiction et de codage. De ce fait, les erreurs de remplissage sont critiques d'autant plus qu'elles risquent de se propager.

Dans le chapitre 6, nous proposons un algorithme d'inpainting basé sur l'exemple utilisant les techniques de *neighbor embedding*. Le problème d'inpainting est formulé comme un problème aux moindres carrées avec différents types de contraintes, de façon analogue aux problèmes de prédiction décrits dans le chapitre 5. Par ailleurs, le calcul de la priorité a été revisité notamment en y incluant une information de densité de contours de la zone à remplir.

1.2 Contributions et Organisation du Manuscrit

1.2.1 Chapitre 2: synthèse de texture et prédiction d'images

Dans le chapitre 2, nous introduisons les définitions et notations relative aux méthodes de synthèse de texture basées sur l'exemple. Après avoir défini brièvement les termes *texture* et *synthèse de texture*, un état de l'art des méthodes de synthèse de texture basées sur l'exemple est présenté. Il inclut les méthodes basées pixels ainsi que les méthodes basées "patch", lesquelles sont largement influencées par les méthodes basées sur les champs aléatoires de Markov. On effectue dans ce chapitre un lien entre le problème de synthèse de texture et la prédiction d'images dans un contexte de compression en considérant d'abord la prédiction intra de la norme de compression H.264/AVC et ensuite les approches basées *template matching*. Une présentation brève de la prédiction intra H.264/AVC et de ses extensions avec les méthodes de synthèse de texture est effectuée. Finalement, les motivations de cette thèse ainsi que les contributions sont résumées.

1.2.2 Chapitre 3: prédiction d'images utilisant des représentations parcimonieuses

Dans le chapitre 3, nous commençons par présenter l'utilisation des représentations parcimonieuses pour un problème de prédiction d'images. Les algorithmes d'approximation parcimonieuse conviennent bien au problème de prédiction puisque seulement quelques candidates de texture ("patchs") extraits de l'image sont utilisés. Afin de calculer les coefficients de pondération des différents candidats retenus, un problème aux moindres carrés est formulé avec une contrainte de parcimonie sur l'approximation du *template*. Supposons que les pixels du *template* soient rangés dans un vecteur colonne $\mathbf{b}_c$ et sachant que la matrice $\mathbf{A}_c$, on essaie de résoudre

$$\arg\min_{\mathbf{x}} \|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\|_2^2 \quad \text{sous contrainte} \quad \|\mathbf{x}\|_0 \leqslant K \tag{1.1}$$

avec $\mathbf{x}$ le vecteur la représentation parcimonieuse de $\mathbf{b}_c$ et $\|\mathbf{x}\|_0$ la norme ℓ_0 de $\mathbf{x}$, c'est-à-dire le nombre de valeurs non nulles de $\mathbf{x}$. K est la valeur maximale de valeurs non nulle autorisée.

Les algorithmes heuristiques gloutons (communément appelé en anglais "heuristic greedy algorithm") tels que les algorithmes de *matching pursuit* et *basis pursuit* ont été développés pour trouver une solution, qui n'est pas systématiquement optimale mais présentant une erreur d'approximation acceptable. Nous proposons d'utiliser un algorithme *matching pursuit* pour approximer le *template*. Cette approximation fournit un vecteur parcimonieux $\mathbf{x}$ contenant les coefficients de pondération associés aux patchs candidats. Notre seconde proposition est de remplacer les dictionnaires classiques (tels que la transformée en cosinus discrète (TCD) ou la transformée de Fourier discrète (TFD)) avec un nouveau dictionnaire adapté localement: les atomes du dictionnaire sont construits à partir de la texture locale sélectionnée dans une fenêtre de recherche causale. On appellera $\mathbf{A}$ le *dictionnaire* qui est supposé être composé de deux sous-matrices $\mathbf{A}_c$ et $\mathbf{A}_t$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_c \\ \mathbf{A}_t \end{bmatrix} \tag{1.2}$$

avec $\mathbf{A}_c$ le sous-dictionnaire correspondant aux pixels du *template* et $\mathbf{A}_t$ un sous-dictionnaire spatial correspondant aux pixels du bloc inconnu à prédire. Cette procédure par conséquent peut être assimilée à une extension du *template matching* dans un contexte parcimonieux. De plus, nous proposons une optimisation itérative pour déterminer les vecteurs parcimonieux avec différents voisinages d'approximation afin de rendre l'approche plus flexible et plus adaptée aux irrégularités locales d'une image. Pour calculer le vecteur parcimonieux optimal $\mathbf{x}_{opt}$, le nombre d'itération ainsi que le support d'approximation doivent être encodés et transmis au décodeur. Finalement, la prédiction des pixels inconnus peut être réalisée en multipliant le vecteur parcimonieux optimal (contenant les coefficients de pondération) avec le sous-dictionnaire $\mathbf{A}_t$.

Une évaluation de la qualité de prédiction et des courbes débit-distorsion a été menée avec des dictionnaires localement adaptés. Les expérimentations montrent que la solution proposée offre une meilleure efficacité de codage comparativement à l'utilisation d'un dictionnaire TCD redondant. L'utilisation de différents supports choisis localement permet également d'améliorer les performances. En outre, des études comparatives impliquant une version simplifiée de cette méthode et un *template matching*, ainsi qu'une comparaison à la méthode *template matching moyen* et aux méthodes de moyennes non-locales ont été menées. Les avantages et inconvénients de la méthode de prédiction intra proposée sont finalement discutés. De possibles améliorations et des perspectives concluent ce chapitre.

1.2.3 Chapitre 4: apprentissage de dictionnaire pour la prédiction d'images

Dans le chapitre 4, nous plaçons le problème de prédiction dans un contexte d'apprentissage de dictionnaires en effectuant l'apprentissage des deux sous-dictionnaires, $\mathbf{A}_c$ et $\mathbf{A}_t$. En d'autres termes, un schéma conventionnel d'apprentissage de dictionnaire (pour des représentations parcimonieuses) composé de ses deux étapes classiques, *codage parcimonieux* et *mise à jour du dictionnaire*, travaillant sur deux ensembles d'apprentissage distincts notés $\mathbf{T}_c$ and $\mathbf{T}_t$, est étudié. La première contribution est d'utiliser les patchs de texture dans un voisinage causal (une fenêtre de recherche) du bloc à prédire comme échantillons d'apprentissage. Cette approche se différencie de l'approche présentée dans le chapitre 3 puisque, dans l'approche précédente, les patchs de texture étaient directement utilisés comme les éléments (atomes) du dictionnaire. La justification sous-jacente est de développer une méthode simple d'apprentissage de dictionnaire au fil de l'eau, qui apprendra ces deux sous-dictionnaries connexes en offrant une prédiction des moindres carrés optimisée des pixels du bloc à prédire. Pour atteindre cet objectif, l'apprentissage du sous-dictionnaire $\mathbf{A}_c$ (utilisant l'ensemble des échantillons $\mathbf{T}_c$) est effectué conduisant à une approximation parcimonieuse des échantillons connus du *template*. Par ailleurs, l'autre sous-dictionnaire $\mathbf{A}_t$ a été optimisé (utilisant l'ensemble des échantillons $\mathbf{T}_t$) au moindre carré (avec deux méthodes différentes), pour être certain que le vecteur d'approximation parcimonieux obtenu pour l'approximation du *template* donnera effectivement une bonne approximation du bloc à prédire. C'est le point important de la méthode que nous proposons et qui à notre connaissance est une contribution nouvelle au regard des techniques d'apprentissage classique de dictionnaires.

Supposons que le sous-dictionnaire $\mathbf{A}_c$ ait été appris en utilisant l'ensemble d'apprentissage $\mathbf{T}_c$, alors l'ensemble des vecteurs d'approximation parcimonieuse (regroupé dans la matrice $\mathbf{Y}$) de $\mathbf{T}_c$ a été utilisé pour optimiser le sous-dictionnaire $\mathbf{A}_t$ (en ayant utilisé les échantillons d'apprentissage $\mathbf{T}_t$) via une approche aux moindres carrés

$$\arg\min_{\mathbf{A}_t} \|\mathbf{T}_t - \mathbf{A}_t\mathbf{Y}\|_F^2 \tag{1.3}$$

en utilisant deux méthodes différentes, l'une basée sur la solution exacte aux moindres carrés et l'autre basée sur *block-coordinate descent*. Finalement, la prédiction des valeurs des pixels inconnus est obtenue en calculant la représentation parcimonieuse du vecteur $\mathbf{x}$ du *template* $\mathbf{b}_c$ à partir du sous-dictionnaire $\mathbf{A}_c$ appris, et en multipliant ce vecteur (contenant les coefficients de pondération optimaux) par le sous-dictionnaire optimisé $\mathbf{A}_t$. Nous avons également proposé une sélection optimisée des différents voisinages, comme dans le chapitre 3, de façon à s'adapter aux irrégularités locales des images.

Suite à l'analyse de l'impact de la contrainte de parcimonie sur la qualité de la prédiction, une approche simplifiée de la méthode proposée est décrite. Par ailleurs, une méthode basée sur une classification des patchs a également été développée. Des études comparatives sur la qualité de la prédiction et sur le compromis débit-distorsion montrent la pertinence de l'approche développée comparativement à des méthodes de prédiction parcimonieuse et des prédictions intra de type H.264/AVC. La classification des patchs donne également des performances débit-distorsion comparables avec des besoins de calculs moindres.

1.2.4 Chapitre 5: prédiction d'images en utilisant les méthodes de neighbor embedding

Dans le chapitre 5, le problème de la prédiction est abordé en utilisant des méthodes appelées *neighbor embedding.* Nous proposons de nouveau deux autres formulations du problème faisant toujours référence aux moindres carrés pour l'approximation du *template* $\mathbf{b}_c$ et cela en utilisant deux méthodes issues des techniques de réduction de dimensionnalités: *locally linear embedding* (LLE) et factorisation de matrice non négative (FMN). L'idée principale est d'explorer de nouveau comment combiner linéairement les différents candidats issus d'un voisinage causal afin d'approximer les valeurs connues du *template* et ainsi d'utiliser les mêmes facteurs de pondération pour prédire les valeurs des pixels inconnues.

Pour les approches LLE et FMN, une contrainte de parcimonie a été adapté via l'utilisation d'un nombre limité d'atomes du dictionnaire. Le dictionnaire $\mathbf{A}$ est construit avec des patchs de texture présents dans la fenêtre de recherche, d'une façon similaire à ce qui est utilisé dans le chapitre 3. Egalement, le dictionnaire est composé de deux sous-dictionnaire $\mathbf{A}_c$ et $\mathbf{A}_t$. Une méthode itérative est décrite en utilisant les $k, k = 1...K$, plus proches patchs voisins du *template.* Une optimisation itérative avec différents supports (voisinages d'approximation) est proposée de façon similaire aux approches parcimonieuses utilisées dans un contexte de prédiction d'images. Le meilleur nombre de patchs utilisés et le support sélectionné nécessitent d'être encodés et transmis au décodeur comme information complémentaire.

Dans l'approche FMN, les pondérations du vecteur d'approximation sont forcément non négatives afin d'approximer un patch à valeurs positives. Puisque les valeurs dans le domaine spatial sont toutes non négatives, le signal reconstruit sera également non négatif et situé sur le même sous-espace que ses plus proches voisins. Cette contrainte d'optimisation peut être formulée de la façon suivante

$$\arg\min_{\mathbf{x}} \left[\frac{1}{2}\left\|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\right\|_2^2\right] \quad \text{sous contrainte} \quad \mathbf{x} \geqslant \mathbf{0} \tag{1.4}$$

où $\mathbf{x}$ est un vecteur parcimonieux contenant les facteurs de pondération associés aux k patchs de texture sélectionnés dans $\mathbf{A}_c$. Soulignons ici que la matrice $\mathbf{A}_c$ est supposée constante et qu'elle n'est pas adaptée contrairement à l'approche classique FMN. Seuls les coefficients du vecteur $\mathbf{x}$ sont ajustés jusqu'à convergence ou en fonction d'un nombre maximum d'itérations.

Il y a également une contrainte sur les coefficients calculés avec l'approche LLE: leur somme doit être unitaire, de façon à ce que les patches reconstruits se situent dans le même sous-espace défini par ses plus proches voisins. Ainsi, on peut formuler le problème de la façon suivante

$$\arg\min_{\mathbf{x}} \|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\|_2^2 \quad \text{sous contrainte} \quad \mathbf{1}^{\mathrm{T}}\mathbf{x} = 1, \tag{1.5}$$

avec $\mathbf{x}$ le vecteur parcimonieux contenant des valeurs non nulles associées aux k plus proches voisins sélectionnés.

Une analyse détaillée de l'impact des contraintes de parcimonies (nombre de candidats utilisés), du bruit de quantification, de la non-négativité et de la contrainte sur la somme des coefficients a été réalisée en fonction de la qualité de prédiction et des performances débit-distorsion. Les résultats expérimentaux indiquent que les méthodes proposées donnent une meilleure qualité de prédiction ainsi qu'une meilleure efficacité de codage comparativement au *template matching* simple, au *template matching moyen*, aux approches de moyennes non-locales, aux approches de prédiction parcimonieuses et aux modes de prédiction H.264/AVC intra.

1.2.5 Chapitre 6: inpainting d'images fixes en utilisant les approches neighbor embedding

Dans le chapitre 6, nous avons élargi notre étude sur les méthodes *neighbor embedding* aux méthodes d'inpainting basées sur l'exemple en posant le problème d'inpainting comme un problème aux moindres carrés, de façon similaire à ce qui est proposé au chapitre 5 pour la prédiction d'images. Nous introduisons tout d'abord une nouvelle méthode de calcul de la priorité de remplissage, en utilisant les informations de *contours*. Associée aux classiques mesures de confiance ("confidence term") et de données ("data term"), la nouvelle méthode de calcul de la priorité prend en compte les informations de contours afin de favoriser la propagation des patchs contenant de fortes structures.

Les pixels inconnus du patch à remplir sont estimés à partir d'une combinaison linéaire des K plus proches voisins. Ces voisins sont déterminés à partir de l'image source et des pixels connus dans le *template* du patch à remplir. Une méthode pour choisir le nombre K de voisins à utiliser est décrite.

Les méthodes d'inpainting proposées ont été évaluées pour deux types d'applications: la suppression d'objets dans un contexte d'édition et le remplissage de zones manquantes dans un contexte de dissimulation d'erreurs. Les expérimentations montrent la pertinence des méthodes. Elles permettent en effet de remplir les zones de façon naturelle avec moins d'artefacts visuels comparativement aux méthodes d'inpainting basées sur l'exemple utilisant un *template matching* simple et moyen ou des approches basées sur moyennes non-locales. Les résultats des méthodes proposées ont également été comparés à d'autres algorithmes de l'état de l'art (basées sur l'exemple ou sur des équations aux dérivées partielles).

1.2.6 Chapitre 7: conclusion et perspectives

Dans le chapitre 7, nous effectuons une conclusion du travail réalisé en résumant les idées principales étudiées et en donnant des perspectives sur la prédiction d'images et sur les problèmes d'inpainting.

1.3 Conclusion

La prédiction et l'inpainting d'images peuvent être abordés comme deux applications différentes de synthèse de texture avec des contraintes spécifiques telles que l'ordre de traitement des informations. Ces deux problèmes sont très importants pour de nombreuses applications traitant de l'image ou de la vidéo. Les motivations de cette thèse portent sur l'amélioration de techniques d'extrapolation aux moindres carrés. Elles sont appliquées à la prédiction d'images et à l'inpainting. Dans l'état de l'art, la synthèse de texture basée sur l'exemple utilisent principalement des techniques basées sur le *template matching* et sur des heuristiques. Nos contributions principales portent sur l'amélioration du *template matching* simple. Cependant, elles s'inscrivent dans un contexte d'optimisation bien formalisé sous différentes contraintes. Nous pensons que les méthodes et idées proposées dans cette thèse ouvrent de nouvelles perspectives pour les applications des méthodes de synthèse de texture telles que la compression via la prédiction d'images et l'inpainting d'images.

Chapter 2

Texture Synthesis and Image Prediction

Texture synthesis algorithms have recently found wide range of application areas in digital signal processing such as image and video editing (e.g., object removal, missing region recovery, error concealment, post production of movies, etc.), enhancement applications (e.g., denoising, restoration, super-resolution, etc.), and also compression (i.e., intra image and inter frame prediction).

Existing texture synthesis methods can be grouped into two main categories. The fist group of methods relies on either the use of higher order partial differential equations or variational approaches. Ideas presented in this type basically try to propagate, *via diffusion*, both geometric and photometric information that hits the border of the region to be synthesized (predicted or filled-in). These methods are known to work relatively well for synthesizing small regions in an image. However, they tend to introduce some blur for large regions, and the results are generally dependent on the input parameters. The second group of texture synthesis concerns exemplar-based methods. This type of methods sample texture patches (from an other image, or from the image itself) in order to synthesize new textures. Because of their simplicity and efficiency in synthesizing textures, exemplar-based methods have been used in a large variety of applications, hence in this study we focus particularly on these methods and their adaptations to different image processing problems.

In this chapter, we first review the fundamental concepts and methodologies of the *exemplar-based texture synthesis* problem. The aim of this first chapter is to construct

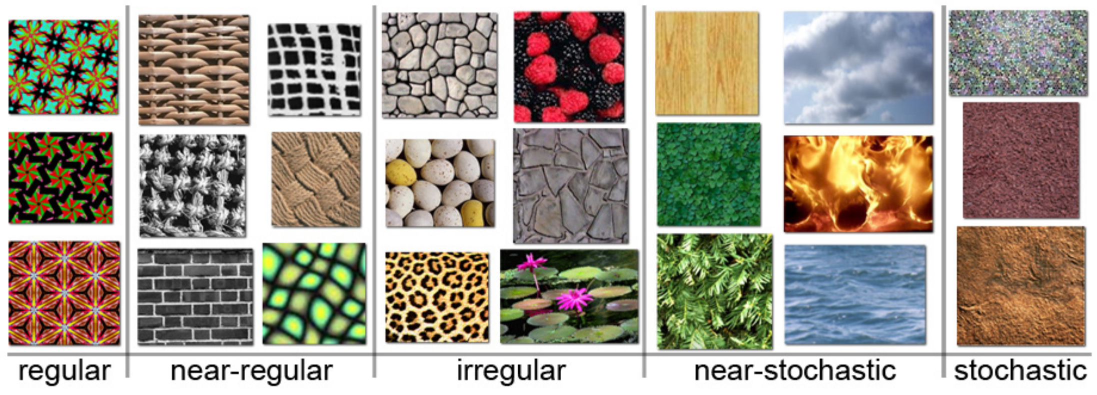


Figure 2.1: Spectrum of texture patterns - Examples ranging from stochastic to regular patterns. This image is taken from http://en.wikipedia.org/wiki/Texture_synthesis.

a basis which leads to a better understanding of the main objective of this manuscript: image prediction (or *predictive coding*). We then describe the image prediction problem with its state-of-the-art applications to intra *image compression*. Finally we conclude this chapter by drawing a summary of the basic contributions of this study.

2.1 Exemplar-based Texture Synthesis

2.1.1 What is texture?

In common definition, the word "texture" is usually referred to as "surface structure". In computer graphics, a texture is a synthetic or digital image which is mapped onto a 3-D object surface by *texture mapping* [2, 3] to achieve a more realistic appearance. In image processing and computer vision communities, textures are usually referred to as visual or tactile surfaces composed of repeating patterns [4]. In this manuscript, we focus on the latter definition of textures as images composed of repeated structures since natural images contain large varieties of textural patterns at different scales.

Textures can be classified into several groups, e.g., stochastic, irregular, regular, or a mixture of them, depending on the *randomness* over their repeating patterns. Regular textures usually contain regular tilings of elements organized into strong periodic patterns. On the other side, stochastic textures contain less noticeable elements and relatively random patterns, i.e., generally look like noise. Stochastic textures can be characterized with a stochastic process which generates the underlying texture. Fig 2.1

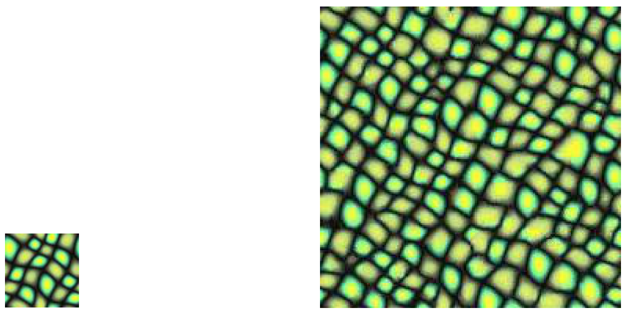


Figure 2.2: Texture synthesis - Given (left) an input sample texture, the objective is to produce (right) a new arbitrary size texture that looks like the input.

shows some visual examples of textures along a spectrum from stochastic to regular ones. For more information on the classification of texture patterns, please refer to [5].

2.1.2 What is texture synthesis?

In terms of image processing definition, texture synthesis is a way of creating textures. Formally, given a small texture sample, i.e., *exemplar*, the aim is to synthesize an arbitrarily large texture which visually appears to be generated by the same underlying process, i.e., the new texture should be perceived as identical to the exemplar in terms of textural characteristics and should not contain any artifacts (please see Fig 2.2).

The texture synthesis process can be divided into two major components as *analysis* and *synthesis* [4]. The analysis part mainly consists of modeling the texture generating process of the given exemplar, and the synthesis step contains the development of an efficient texture generation (sampling) procedure in order to construct the new texture from the given analysis model. Both the analysis and the synthesis components play an important role on a successful texture synthesis process.

2.1.3 Exemplar-based texture synthesis methods

Before going into details of exemplar-based texture synthesis methods, let us start by providing two important key definitions:

- A texture image is regarded as *local* if (a) each pixel of the image can be predicted from a small set of spatially neighboring pixels independently from the rest of the image, and (b) this characterization is the same for all pixels.

- A texture image is regarded as *stationary* if, given a movable proper size window, the observed portion of the image in this small moving window always appears to be similar to the viewer.

Although there exists a large variety of texture synthesis algorithms developed by various researchers, the most successful application so far is Markov Random Field (MRF) [6, 7]. MRF models texture patterns as a realization of a *local* and *stationary* random process [4]. In terms of MRF, a texture synthesis problem can be defined as follows: *Given an input texture, an output texture can successfully be synthesized by the assumption that, for each output pixel, its spatial neighborhood is similar to at least one neighborhood at the input texture.* The similarity between input and output neighbors guarantees a perceptual similarity between input and output texture images as well.

There are several MRF methodologies relying on probability sampling, e.g., [8, 9, 10], which are often complex algorithms with heavy computational requirements. Instead, here we focus particularly on relatively simple MRF influenced methods which are not only efficient but also easy to understand and use.

2.1.3.1 Pixel-based texture synthesis

Pixel-based texture synthesis methods synthesize the new texture pixel by pixel. At each step of the algorithm, the value of an unknown pixel p is determined according to its local neighboring pixels $N(p)$. The neighborhood $N(p)$ of a pixel p is generally modeled as a square window centered around that pixel. The size of the window is a parameter, which can be tuned by the user, that specifies a measure on the randomness over the repeating patterns of the texture. When the selected neighborhood is too small, the output texture would be too random. On the other hand, when the neighborhood is too large, the output might become a regular pattern or contain meaningless results.

A very simple and efficient non-parametric method based on the MRF model has been introduced by Efros and Leung [11]. Their method "grows" texture pixel by pixel, in layers outward from an initial seed, e.g., a 3×3 patch, which has been randomly taken from a portion of the input exemplar. To generate a new pixel p, all previously synthesized pixels in the neighborhood $N(p)$ are used as the context, and they are searched in the input texture to find the most similar candidates. In order to emphasize local structure, the pixels in the neighborhood are weighted with a two-dimensional

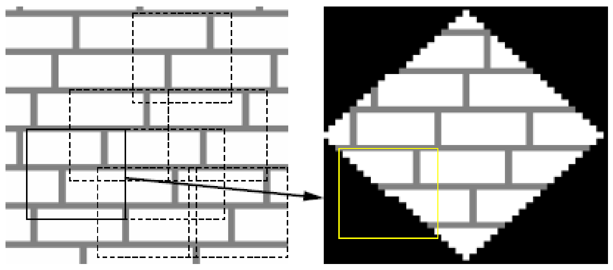


Figure 2.3: Texture synthesis by non-parametric sampling - Algorithm overview. Given (left) a sample texture image, (right) a new image is being synthesized one pixel at a time. To synthesize a pixel, the algorithm first finds all neighborhoods in the sample image (boxes on the left) that are similar to the pixel's neighborhood (box on the right) and then randomly chooses one neighborhood and takes its center to be the newly synthesized pixel. This image is taken from [11].

Gaussian kernel, i.e., in order to give large weighting values for nearby pixels and small weights for far away pixels. The algorithm finally chooses one random neighborhood among the most similar candidates and takes its center as the newly synthesized pixel. This process is repeated until the entire output texture has been synthesized. A brief overview of this algorithm has been shown in Fig. 2.3.

The method in [11] is indeed very simple to implement and works well for a variety of textures. However it suffers from the exhaustive search of variable size neighborhoods, and the spiral like ordering of synthesis makes it slow. Wei and Levoy [12] have addressed these drawbacks and proposed a faster pixel-based algorithm with fixed size neighborhoods and a raster scan ordering through a deterministic synthesis process. The basic idea of this method is illustrated in Fig. 2.4 and it proceeds as follows. After initializing the output texture with random values (i.e., randomly copying pixels from the input texture), each pixel is synthesized one by one in a scanline order. For each output pixel p, a best matching spatial neighborhood is searched in the input texture, and the corresponding pixel from the input is then copied to the output. This method differs from [11] in the sense of the use of a fixed size neighborhood (which leads to an acceleration in the search process by various methods [13, 14]), a raster scan ordering, and a completely deterministic sampling rather than a random sampling.

Ashikhmin [15] has later extended the algorithm in [12] by considering input texture pixels (as candidates) with neighborhoods which are similar to shifted versions of the current neighborhood $N(p)$ in the output texture.

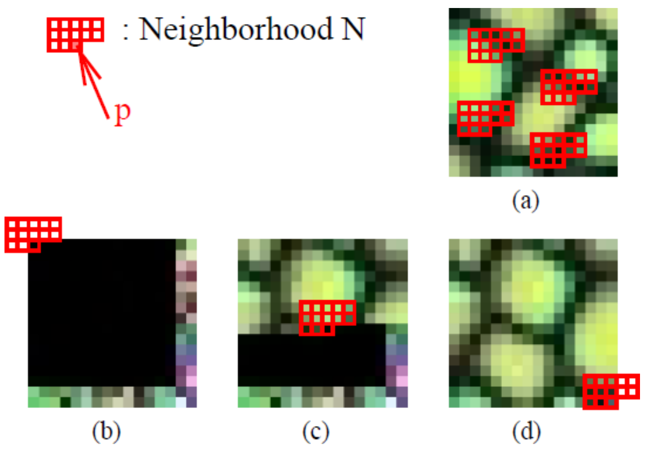


Figure 2.4: Fast texture synthesis - (a) The input texture and (b)-(d) different synthesis stages of the output texture. Pixels in the output image are assigned in a raster scan ordering. The value of each output pixel p is determined by comparing its spatial neighborhood $N(p)$ with all neighborhoods in the input texture. The input pixel with the most similar neighborhood is assigned to the corresponding output pixel. Neighborhoods crossing the output image boundaries are handled toroidally. Although the output image starts as a random noise, only the last few rows and columns of the noise are actually used. For clarity, the unused noise pixels are shown as black. This illustration is taken from [12].

2.1.3.2 Patch-based texture synthesis

Synthesizing texture using patches rather than pixels can improve the speed as well as the quality of pixel-based methods. Patch-based methods can therefore be regarded as an extension of pixel-based synthesis where, instead of copying pixels, one copies patches. Similar to pixel-based methods, the patch to be copied is selected according to its neighborhood in order to ensure the consistency with already synthesized pixels. Fig. 2.5 shows the main idea of patch-based texture synthesis in comparison to pixel-based methods.

Since the output texture is synthesized with patches rather than pixels taken from the input exemplar, one can expect a quality improvement at the output at least in the patch level. However, a patch is larger than a pixel, and usually it overlaps with the already synthesized portion of the output texture. Here the problem arises with the utilized synthesis technique –that copies the selected patches to the output– which should be capable of handling the conflicting texture regions in order to keep the continuity of patterns through patch boundaries.

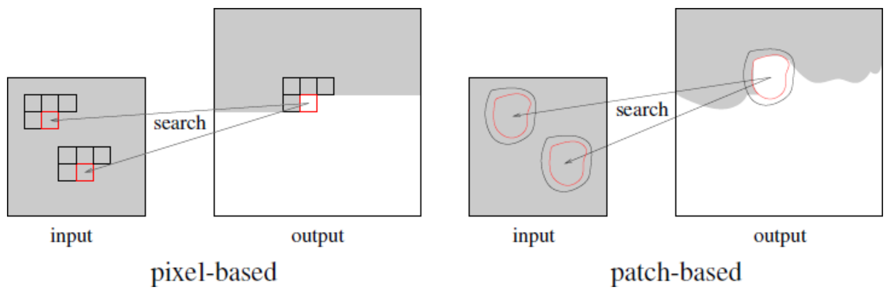


Figure 2.5: Comparison of pixel-based and patch-based texture synthesis - The gray region in the output indicates already synthesized portion. This figure is taken from [4].

A simple synthesis procedure can be just overwriting the new patches over existing regions as proposed in [16], or simply blending the overlapped areas [17]. Even though these simple methods are fast and memory efficient, they may introduce visible seams or blurry artifacts to the final output (see Fig. 2.6). Efros and Freeman [18] proposed instead a synthesis process, which is called *image quilting*, by stitching the image patches optimally (through a minimum cost path) using dynamic programming. This idea has then been further extended and improved by using a *graph-cut* technique in Kwatra *et al.* [19]. Fig. 2.6 illustrates the results of different synthesis approaches.

There are more synthesis approaches in the literature, such as the *Chaos Mosaic* [20] relying on tilings of input textures, or an optimization method [21] as a mixture of pixel and patch based schemes, as well as the multiresolution based methods [22, 23, 24, 25].

2.1.4 Inverse texture synthesis

The inverse texture synthesis problem has been addressed in [26]. Contrary to conventional forward synthesis, the method here runs an optimization in the opposite direction such that *it automatically produces a small texture compaction that best summarizes the given (original) arbitrarily large and globally variant texture*. The original texture can then be reconstructed, or new textures can be synthesized, using this small compaction. Some examples of compaction and resynthesis textures are shown in Fig. 2.7.

The texture compaction, which is the output of inverse texture synthesis, needs less storage due to the reduced size, hence could be used for texture compression purposes. Moreover, this method can easily be adapted for compressing texture regions in natural images by taking advantage of the spatial repetition nature of textures.

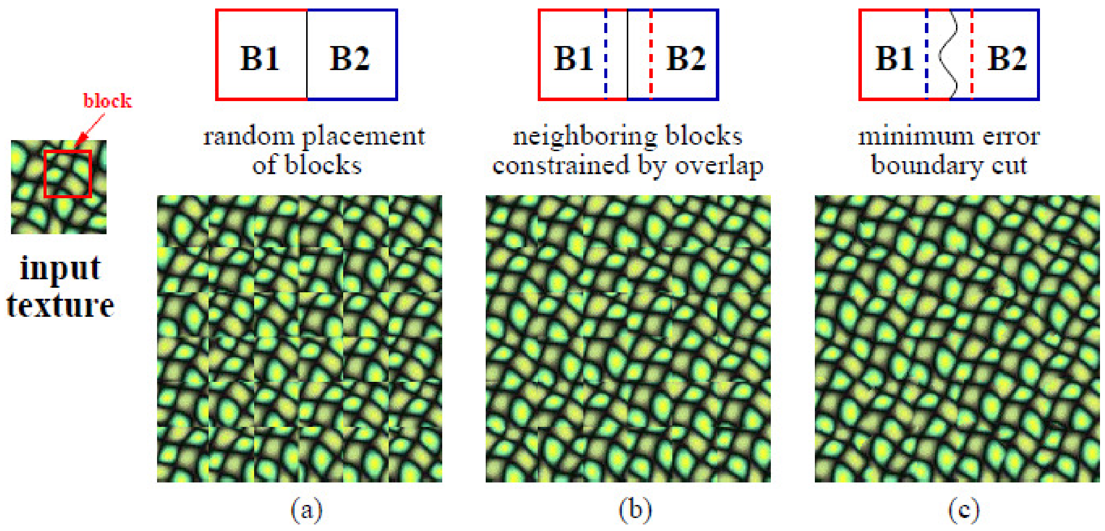

B1
B2
block
input texture
random placement of blocks
neighboring blocks constrained by overlap
minimum error boundary cut
(a)
(b)
(c)

Figure 2.6: Methods for handling adjacent patches during synthesis - Square patches (blocks) from the input texture are patched together to synthesize a new texture: (a) blocks are chosen randomly (similar to [16]), (b) blocks overlap and each new block is chosen so as to "agree" with its neighbors in the region of overlap (similar to [17]), (c) to reduce the blockiness, the boundary between blocks is computed as a minimum cost path through the error surface at the overlap [18, 19]. This figure is taken from [18].

2.2 Image Prediction

Most of the natural images are often composed of various textural regions which are generally separated by object contours or edges. Thus an image can be regarded as a combination of several types of textures interacting among themselves, and with the other structures and objects in the image. Fig. 2.8 shows an example of a natural image which has different local textural characteristics.

Observing the fact that natural images exhibit *local* textural characteristics, one can rely on MRF models for local image processing. Therefore, under the assumption of local stationarity, the texture synthesis methodologies described above are also applicable to natural images. For example, image prediction and image inpainting problems can be formularized in this context, and very efficient solutions can be obtained by extending already proposed texture synthesis methods.

In this section, we mainly focus on closed-loop intra prediction which is a key component of image and video compression algorithms. The term "intra prediction" refers to the fact that the prediction technique is performed using only the information

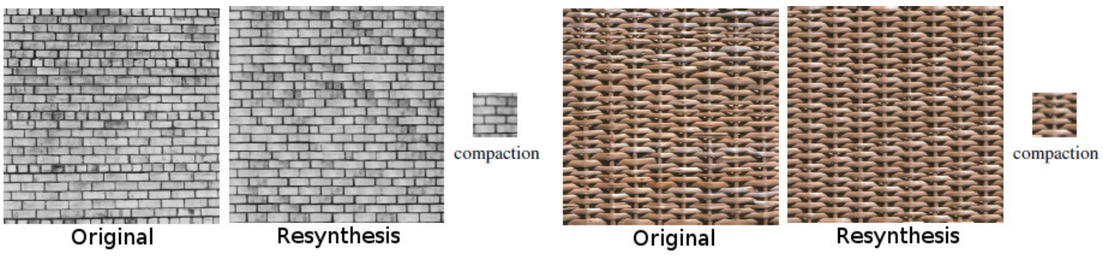


Figure 2.7: Inverse texture synthesis - Examples of two stationary textures with inverse synthesis and resynthesis. Images are taken from [26].

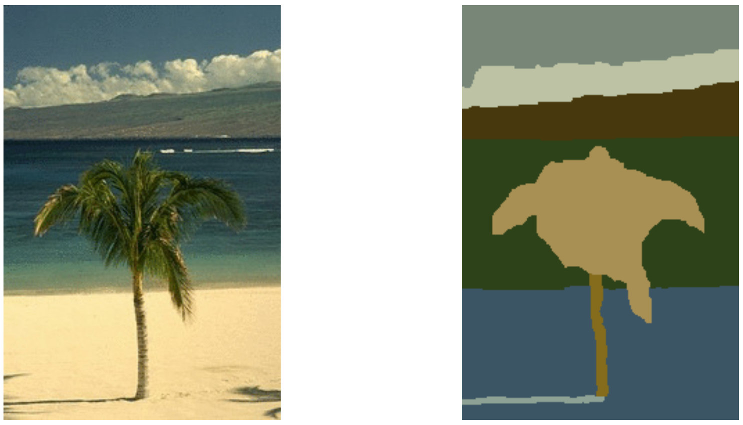


Figure 2.8: An example natural image which is composed of several textural regions - (Left) The "beach" image, and (right) the texture segmented image. Different textural regions have been shown with different colors.

that is contained within an image or an intra frame (I-frame) in a video sequence. The underlying basic idea is to first predict or synthesis a patch in the image (or in the intra frame), and then encode the prediction residue signal, instead of the patch itself, in order to minimize the encoded information. Similar to patch-based texture synthesis, most of the image prediction algorithms operate on square patches, i.e., blocks of size $n \times n$, in a raster scan ordering. The blocks usually do not overlap so that residue signals are transformed, quantized, and entropy encoded disjointly. The reconstructed block is finally obtained by adding the quantized residue to the prediction. A general block diagram of block-based intra image compression has been shown in Fig. 2.9.

We next briefly review state-of-the-art spatial image prediction methods including the directional modes in H.264/AVC intra coding, and the template matching method with its extensions.

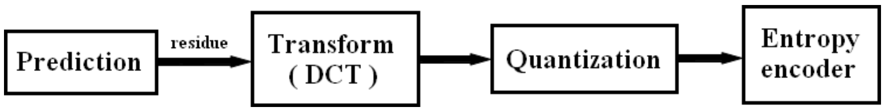


Figure 2.9: Block diagram of block-based intra image compression - The compression process starts with prediction. After predicting a block, the residue signal is transformed, quantized, and entropy encoded.

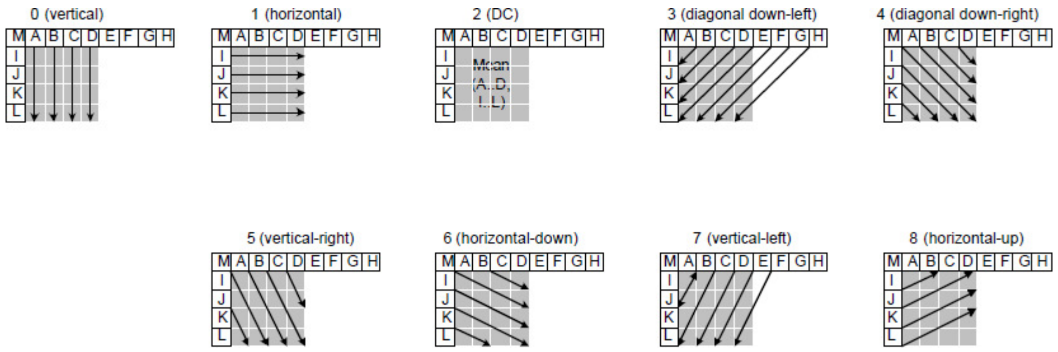


Figure 2.10: Nine prediction modes of H.264/AVC Intra–4×4 type - Unknown pixels are predicted using pixels A-M. The prediction is done by simply propagating (or interpolating) the pixel values along the specified direction.

2.2.1 H.264/AVC intra prediction

In H.264/AVC, there are mainly two intra prediction types called Intra–16×16 and Intra–4×4 [27]. There is also Intra–8×8 type in H.264/AVC FRExt (Fidelity Range Extension) [28], however most H.264/AVC profiles do not support this prediction type.

The Intra–4×4 type supports nine modes as shown in Fig. 2.10. Each 4×4 block is predicted from prior encoded pixels from spatially neighboring blocks. In addition to the so-called "DC" mode which consists in predicting the entire 4×4 block from the mean of neighboring pixels, eight directional prediction modes are specified (Fig. 2.10). The prediction is done by simply propagating (or interpolating) the pixel values along the specified direction. A visual illustration of prediction of a 4×4 block with nine modes has been shown in Fig. 2.11. In the experiments reported in this manuscript, Intra–4×4 has been used as defined in the standard (please refer to [29]).

The Intra–16×16 type supports only four prediction modes including vertical (i.e., extrapolation from upper samples), horizontal (i.e., extrapolation from left samples), DC (i.e., mean of upper and left samples), and plane (i.e., with a linear function which

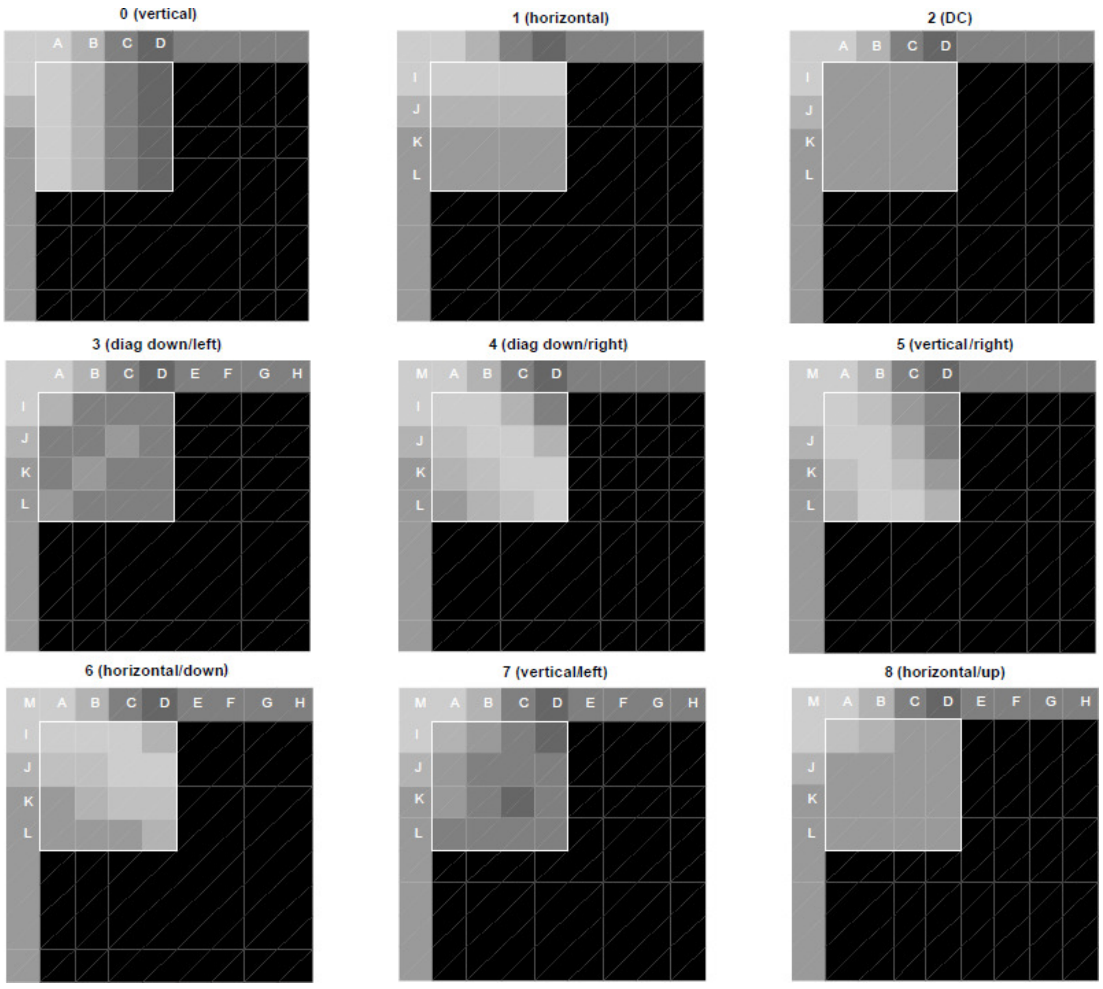


Figure 2.11: A visual illustration of H.264/AVC Intra–4×4 prediction - Nine prediction modes calculated for a 4×4 block.

uses upper and left samples) modes. This type is actually suitable for smooth image regions, hence chroma samples prediction has been designed very similar to it. Fig. 2.12 illustrates four prediction modes of Intra–16×16.

In H.264/AVC intra prediction, a *macroblock* of size 16×16 pixels is divided into sixteen 4×4 blocks. The encoder uses a tool which is called Lagrangian rate-distortion optimization (RDO) to select the best prediction type (either Intra–4×4 or Intra–16×16) and also the best mode (out of nine modes in Intra–4×4 or of four modes in Intra–16×16). Different prediction modes can be selected for each of the sixteen 4×4 blocks in a macroblock. The selected prediction type and the mode(s) information have

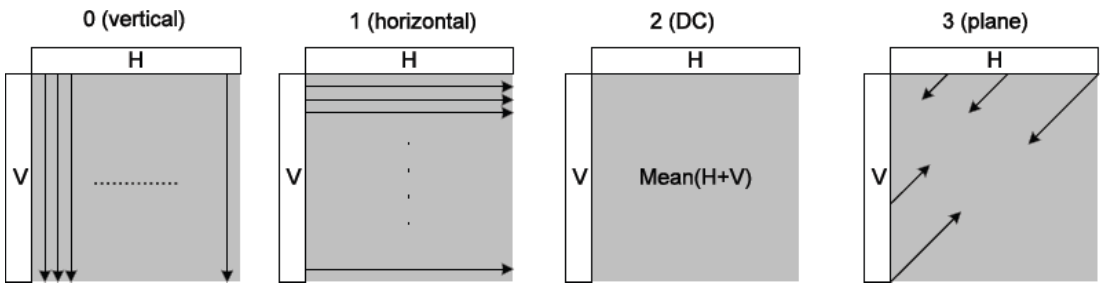


Figure 2.12: Four prediction modes of H.264/AVC Intra–16×16 type - Unknown pixels are predicted using pixels in H and V.

to be then transmitted to the decoder (in addition to residue signals).

The H.264/AVC intra prediction approach (as briefly described above) is suitable in the presence of contours when the directional mode chosen corresponds to the orientation of the contour. However, it fails in more complex textured areas. Thus there is still research going on for intra prediction in the existing H.264/AVC codec to achieve better performance in compression.

Wien [30] proposed a concept for variable-block size transform coding in H.264/AVC. The modes defined for the Intra–4×4 prediction type are extended to block sizes similar to ones used for inter prediction [27], i.e., this is done by partitioning a macroblock into 8×8, 8×4, and 4×8 pixels of blocks in addition to 4×4 blocks. In this way, the gap between Intra–4×4 and Intra–16×16 has been filled. Experimental observations show an average bit-rate savings of about 8% and PSNR gains of 0.49 dB for intra coding.

A similar approach based on geometry-adaptive block partitioning for intra prediction has been presented in [31]. In this method, geometric partitions of blocks (of size 8×8 or 16×16 pixels) are calculated by means of an implicit parametric linear model which classifies pixels within blocks to one of the partitions, or to a line boundary. Although adaptive block partitioning is very costly to apply, the experimental results show significantly improved rate-distortion performance (up to 11.19% in the average) when compared to standard H.264/AVC intra coding.

An interesting approach based on block oriented transforms has been introduced in [32]. The idea of this study is to perform pre-defined rotations (with simple circular shifts rather than classical rotation schemes) on the residue blocks in order to straighten them out towards horizontal and vertical axes, i.e., in order to be able to transform residue blocks efficiently with discrete cosine transform (DCT). Here the encoder needs to transmit also the rotation information to the decoder. This method can be seen as a post-processing step after prediction, leading up to 1 dB gain compared to the standard intra prediction modes of H.264/AVC.

There are various other research papers in the literature, e.g., [33, 34, 35, 36, 37, 38], which propose extensions as well as novel methodologies aiming at improving currently available H.264/AVC intra prediction and coding. As a final remark, a successor to H.264/AVC, High Efficiency Video Coding (HEVC) is currently under development. In HEVC, there will be up to 34 intra prediction directions with quadtree partitioned blocks of size from 4×4 to 32×32 samples.

2.2.2 Template matching based intra prediction

An alternative method based on *template matching* has been widely considered for intra image prediction [35, 39, 40, 41, 42]. A so-called *template* is formed by previously encoded and decoded pixels in a close neighborhood of the unknown block to be predicted. The best match between the template and the candidate texture patch neighborhood (of the same shape as template), within a causal search window, allows finding the predictor of the unknown block (please see Chapter 3 for more information). This method can indeed be regarded as an extension of patch-based texture synthesis, where patches to be copied are selected according to their neighborhood (template). In contrary to texture synthesis methods, the candidate patch is searched in a causal search window in the reconstructed image instead of using an exemplar image.

In [35], a prediction scheme has been proposed by replacing the H.264/AVC (Intra–4×4) DC mode with template matching. This simple replacement results in an overall performance gain of 0.1-0.4 dB in intra coding. In a similar spirit, another template matching based algorithm has been described in [39]. In this approach, the block to be predicted of size 4×4 is further divided into four 2×2 subblocks. Template matching prediction is then conducted for each subblock accordingly. The four best match candidate subblocks finally constitute the prediction of the block to be predicted. This method has been included into H.264/AVC Intra–4×4 as an additional prediction mode leading to more than 11% bit-rate savings. It has later been improved in [40] by averaging multiple template matching predictors, also including larger and directional templates, as a result of more than 15% coding efficiency in H.264/AVC.

There are various extensions of template matching based image prediction in the literature, e.g., a priority-based approach as proposed in [42], and an adaptive illumination compansation based method [41]. Note also that there are other applications to image inpainting [43, 44] and inter frame prediction [45].

2.3 Motivation

Exemplar-based texture synthesis methods usually copy a single patch (or a pixel) from the input exemplar to the output texture. The template matching algorithm operates as an extension to exemplar-based texture synthesis for the image prediction problem. The experimental results show a general improvement in H.264/AVC intra prediction with template matching integration. This fact motivates us to give more attention to template matching based methods for image prediction.

We consider here weighted linear combinations of several image patches, taken from a search window in the image, instead of using a single "best" patch as most MRF based methods. The simplest way of combining several patches is to assign uniform weights for each patch. This method is the same with average of multiple template matching predictors proposed in [40]. Recently, non-local means based approaches [46, 47, 48, 49, 50] have also been proposed for different texture synthesis problems. These methods which combine several patches have been proven to be more robust in estimating the unknown values and producing better synthesis results. However these approaches do not search to minimize an approximation error on the template signal. They are rather heuristic methods to calculate weighting coefficients.

The main idea of the work in this manuscript is to use some optimization techniques to formulate the prediction problem as a least-squares texture synthesis problem possibly with different types of constraints, hence calculating the weighting coefficients with an optimization for approximating the template information. The underlying motivation can be thought of as: "*A good approximation of the known pixel values in the template would lead also a good approximation of the unknown values in the block to be predicted.*" In this sense, we believe that one can improve the overall prediction and coding performance with some additional optimization techniques such as on selecting the number of patches to be used for prediction, or selecting and/or adapting additional prediction modes (i.e., different shapes of templates), and so on.

2.4 Conclusion

In this chapter, we reviewed underlying basic principles of exemplar-based texture synthesis methods. We considered applications which build a bridge between the texture synthesis problem and the image prediction problem. The rest of this manuscript consists of detailed descriptions of our contributions for image prediction, as well as an extension to image inpainting, together with an extensive literature survey of the conventional algorithms in different research frameworks, in which we have placed our image prediction and inpainting problems.

Briefly, in Chapter 3 the image prediction problem has been formularized in sparse representations framework, hence a least-squares optimization is run for approximating the template signal under some sparsity constraints for selecting image patches to be used and for calculating the corresponding weighting coefficients. In Chapter 4, the prediction problem has been put into a dictionary learning framework by optimizing the prediction of the unknown block of pixels in a least-squares sense through training image patches which have been collected from a search window in the image. In Chapter 5, we have placed the image prediction problem into a neighbor embedding framework by adapting two different dimensionality reduction methods: locally linear embedding and non-negative matrix factorization. These neighbor embedding methods provide also two least-squares problems with different constraints, one with non-negativity and the other with sum-to-one constraints, on the weighting coefficients for approximating the template signal. Finally in Chapter 6, the proposed neighbor embedding methods in Chapter 5 have been extended to an image inpainting application by formulating the inpainting problem also as a least-squares optimization.

The contributions of this thesis could be seen as extensions to template matching based methods, however rather than using heuristic approaches, the problems here are well formularized in several optimization frameworks with different constraints. We believe that the algorithms proposed in this study will open new doors in texture synthesis applications by offering new directions for image prediction (i.e., predictive coding), and also for image inpainting.

Chapter 3

Image Prediction based on Sparse Representations

One of the features of well-known transforms, such as the Fourier transform and the wavelet transform, is to reveal certain structures of a signal and to represent these structures in a compact, in other words *sparse*, manner. Sparse representations have therefore become a very active research topic as they provide high performance in a large diversity of signal processing applications.

In this chapter[1], we first explain the basic principles of sparse representations by formulating the underlying optimization problem and then present various greedy (pursuit) algorithms used for solving this problem. Next we introduce the concept of "dictionaries" generally used in sparse representations. We then describe how sparse representations can be adapted and applied to the image prediction problem by defining **dynamic** and **locally adaptive** dictionaries. Finally we conclude this chapter by presenting the experimental results obtained for the proposed framework for image prediction, which we refer to as **sparse prediction (SP)**.

3.1 Sparse Representations

Sparse representations have become an important topic with numerous applications in signal and image processing, e.g., image denoising [54, 55], texture modeling [56],

[1]The content of this chapter is related to our publications in [51, 52]. [52] is the recipient of the Huawei Best Student Paper Award in 2010 IEEE Int. Conf. on Image Process. (ICIP'10), Sept. 2010. A related joint publication, which is not included into this chapter, is available in [53].

image restoration [57, 58], image compression [59, 60], and more [61, 62, 63]. Sparse representations consist in representing most or all information contained in a signal with a linear combination of a small number of elements or *atoms* adequately chosen from an overcomplete or redundant basis or *dictionary.* Formally, such a dictionary is a collection of atoms whose number is much larger than the dimension of the signal space, i.e., than the number of components of the vector representing the signal. Any signal admits then an infinite number of representations and the sparsest such representation happens to have interesting properties for a number of image processing tasks.

3.1.1 Problem formulation

The objective of sparse representations is to find a sparse approximation of a given input signal $\mathbf{b}$. In theory, given $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ and $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$ with $\mathrm{N} \leqslant \mathrm{M}$ and $\mathbf{A}$ is of full rank, one seeks the solution of

$$\mathbf{b} = \mathbf{A}\mathbf{x} \;\; \text{subject to} \;\; \min \|\mathbf{x}\|_0 \tag{3.1}$$

where the vector $\mathbf{x} \in \mathbb{R}^{\mathrm{M}}$ denotes the sparse representation of the signal $\mathbf{b}$ and $\|\mathbf{x}\|_0$ is the ℓ_0–norm of $\mathbf{x}$, i.e., the number of non-zero components in $\mathbf{x}$. The matrix $\mathbf{A}$ is known as the *dictionary* and its columns $\mathbf{a}_m, m = 1...\mathrm{M}$, are the *atoms*, they are assumed to be normalized in the ℓ_2–norm, i.e., $\|\mathbf{a}_m\|_2 = 1 \; \forall m$.

There are infinitely many solutions $\mathbf{x}$ to $\mathbf{b} = \mathbf{A}\mathbf{x}$ and the problem is to find the sparsest, i.e., the one for which $\mathbf{x}$ has the fewest non-zero components. In practice, depending on the application, one actually seeks an approximate and thus even sparser solution which satisfies

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_p \leqslant \rho \;\; \text{subject to} \;\; \min \|\mathbf{x}\|_0 \tag{3.2}$$

for some $\rho \geqslant 0$ characterizing an admissible approximation error, or

$$\|\mathbf{x}\|_0 \leqslant K \;\; \text{subject to} \;\; \min \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_p \tag{3.3}$$

for some integer $K > 0$ representing the maximum number of allowed non-zero components in $\mathbf{x}$. The norm p is usually 2, but could be 1 or ∞ as well.[1]

[1]Given a vector $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$, the p–norm is defined as $\|\mathbf{b}\|_p = \left(\sum_{n=1}^{\mathrm{N}} |b_n|^p\right)^{1/p}$ for $p \geqslant 1$, and $\|\mathbf{b}\|_\infty = \max\left(|b_1|, ..., |b_{\mathrm{N}}|\right)$.

Except for the exhaustive combinatorial approach, there is no known method to find the exact solution under general conditions on the dictionary **A**. Searching for this sparsest representation is hence unfeasible, and the problems defined in (3.1)–(3.3) are computationally intractable [64].

3.1.2 Sparse decomposition algorithms

A wide variety of pursuit algorithms [65, 66, 67, 68, 69, 70, 71] have been introduced as heuristic greedy methods which aim at finding approximate solutions to the above problems with tractable complexity.

The matching pursuit (MP) [66] algorithm yields an approximation error which decreases with each iteration. However, it is sub-optimal. At any iteration, the newly obtained residual signal is orthogonal only to the immediately selected atom, but it may not be orthogonal to all the atoms selected at the previous iterations. As a result, some atoms selected at an earlier iteration may get selected again. This causes slow convergence. The orthogonal matching pursuit (OMP) [67] algorithm removes this drawback by updating the weighting coefficients of all previously selected atoms so that the newly derived residual signal is orthogonal to not only the immediately selected atom but also to all the atoms selected at previous iterations. As a consequence, once an atom is selected, it is never selected again in subsequent iterations. In OMP, the approximation error at any iteration is minimized only for the immediately selected atom, however, this error may not be the minimum over all previously selected atoms. The optimized OMP (OOMP) [68] algorithm has then been introduced to improve on this sub-optimality in OMP by performing a more computationally demanding search.

The difficulty in solving the problems in (3.1)–(3.3) exactly lies in the ℓ_0–norm minimization. An easy way to get an approximate solution to this problem is to replace the ℓ_0–norm by the ℓ_1–norm. The advantage of using the ℓ_1–norm is that these problems are easily transformed into linear programs whose solutions are straightforward. This approach is known in the literature as basis pursuit (BP) [65]. Clearly, the two problems are different, however solving the ℓ_1 minimization problem results in an approximate solution for the ℓ_0 minimization problem. In BP, the atoms are selected simultaneously instead of choosing them one by one, but at the expense of increased complexity.

Recently, complementary matching pursuit (CMP) [71] algorithms, e.g., orthogonal CMP (OCMP) and optimized OCMP (OOCMP), have been introduced into the literature aiming at improving currently available pursuit algorithms.

3.1.2.1 Matching pursuit (MP)

The MP algorithm offers a sub-optimal solution via an iterative algorithm. It generates a sequence of M dimensional vectors $\mathbf{x}_k$ having an increasing number of non-zero components in the following way. Initially $\mathbf{x}_0 = \mathbf{0}$ and an initial residual vector $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{b}$ is computed. At the k^{th} iteration, the algorithm selects the basis function (atom) $\mathbf{a}_{m_k}$ having the highest correlation with the current residual vector $\mathbf{r}_{k-1} = \mathbf{b} - \mathbf{A}\mathbf{x}_{k-1}$, that is, such that

$$m_k = \arg\max_m \left|\mathbf{a}_m^{\mathrm{T}}\mathbf{r}_{k-1}\right|, \quad m = 1...\mathrm{M}. \tag{3.4}$$

The weighting coefficient α_{m_k} of this new atom $\mathbf{a}_{m_k}$ is then calculated as

$$\alpha_{m_k} = \mathbf{a}_{m_k}^{\mathrm{T}}\mathbf{r}_{k-1} \tag{3.5}$$

to minimize the energy of the new residual vector which thus becomes equal to

$$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{m_k}\mathbf{a}_{m_k}. \tag{3.6}$$

The new optimal weight α_{m_k} is introduced into $\mathbf{x}_{k-1}$ to yield $\mathbf{x}_k$. Note that the same atom may be chosen several times by MP. In this case, the value of the coefficient α_{m_k} is added to the previous one. The algorithm proceeds until a stopping criterion ($\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|_2^2 \leqslant \rho$ or $\|\mathbf{x}_k\|_0 > K$) is satisfied, where ρ and K are the parameters which control the sparseness of the representation.

3.1.2.2 Orthogonal matching pursuit (OMP)

OMP removes the drawback of MP by updating the weighting coefficients of all previously selected atoms so that the newly derived residual signal is orthogonal to both immediately selected atom and all the atoms selected at previous iterations. As a result, once an atom is selected, it is never selected again.

Like in MP, initially $\mathbf{x}_0 = \mathbf{0}$ and an initial residual vector $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{b}$ is computed. At the k^{th} iteration, the algorithm first identifies the atom $\mathbf{a}_{m_k}$ having the maximum correlation with the current residual vector $\mathbf{r}_{k-1} = \mathbf{b} - \mathbf{A}\mathbf{x}_{k-1}$. Let $\mathbf{A}^{k-1}$ denote the matrix containing all the atoms selected in the previous iterations, and $\mathbf{A}^k$ be defined as $\mathbf{A}^k = \mathbf{A}^{k-1} \cup \{\mathbf{a}_{m_k}\}$. One then projects the signal $\mathbf{b}$ onto the subspace spanned by the columns of $\mathbf{A}^k$, i.e., one solves

$$\min_{\boldsymbol{\alpha}_k} \left\| \mathbf{b} - \mathbf{A}^k \boldsymbol{\alpha}_k \right\|_2^2, \tag{3.7}$$

and the optimum solution vector $\boldsymbol{\alpha}_k$ is given as

$$\boldsymbol{\alpha}_k = \left(\mathbf{A}^{k^{\mathrm{T}}} \mathbf{A}^k \right)^{-1} \mathbf{A}^{k^{\mathrm{T}}} \mathbf{b} = \mathbf{A}^{k^{+}} \mathbf{b} \tag{3.8}$$

where $\boldsymbol{\alpha}_k = [\alpha_{m_1} \ ... \ \alpha_{m_k}]^{\mathrm{T}}$ contains the weighting coefficients of all the atoms selected up to the k^{th} iteration, and $\mathbf{A}^{k^{+}}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}^k$.

The optimal weights calculated in $\boldsymbol{\alpha}_k$ are introduced into $\mathbf{x}_{k-1}$ to yield $\mathbf{x}_k$ by updating (i.e., replacing) the coefficients with the previous ones. The new residual vector can then be calculated as $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$. Notice that here $\mathbf{x}_k$ is the sparse vector of coefficients at the k^{th} iteration and all the coefficients assigned to the selected atoms are recomputed at each iteration, unlike in the case of MP.

An alternative approach, stagewise OMP (StOMP) [69] has later been introduced. At any iteration (stage) s, instead of simply selecting the maximum correlated atom with the residual vector $\mathbf{r}_{s-1}$, StOMP selects a set of atoms in which all the correlation values are above a specified threshold. It then solves the least-squares problem as in OMP in order to calculate the optimum weights, and calculates the new residual signal $\mathbf{r}_s$. Please see [69] for more information on StOMP and its thresholding strategy.

3.1.2.3 Optimized orthogonal matching pursuit (OOMP)

Although OMP achieves the best approximation of a given signal $\mathbf{b}$ that one can obtain by using the selected atoms, it is not optimum since the atoms are selected with the MP criterion. MP minimizes the energy of the residual signal at the current iteration but it does not guarantee that the energy of the residual signal of the OMP decomposition is minimized.

The OOMP algorithm has been introduced in order to achieve the optimality on selecting the dictionary atom, at each iteration k, which minimizes the residual signal energy of the orthogonal projection approximation of the signal $\mathbf{b}$. At the k^{th} iteration, OOMP aims at selecting the optimum atom $\mathbf{a}_{m_k}$ by minimizing the energy of a residual signal, that is, such that

$$\mathbf{r}_k = \mathbf{b} - P_k\mathbf{b} \tag{3.9}$$

where the operator P_k is the orthogonal projector onto the subspace spanned by the columns of $\mathbf{A}^k = \mathbf{A}^{k-1} \cup \{\mathbf{a}_{m_k}\}$. The optimum atom $\mathbf{a}_{m_k}$ minimizing $\|\mathbf{r}_k\|_2^2$ can be selected by maximizing the value of the functionals e_m as given by [68]

$$e_m = \frac{b_m}{d_m} = \frac{\left|\mathbf{a}_m^{\mathrm{T}}\mathbf{r}_{k-1}\right|^2}{1 - \mathbf{a}_m^{\mathrm{T}}\left(P_{k-1}\mathbf{a}_m\right)}, \quad m = 1...\mathrm{M}, \tag{3.10}$$

where the selection is restricted to $b_m > 0$, i.e., excluding the atoms with $b_m = 0$ which have a linear dependence with the previously selected atoms,

$$m_k = \left\{ m : \arg\max_m \left(e_m\right), b_m > 0 \right\}, \tag{3.11}$$

in order to guarantee that all the selected atoms in $\mathbf{A}^k$ are linearly independent.

3.1.2.4 Basis pursuit (BP)

BP is also known as the *Lasso* [72] with a similar problem formulation. It is yet another approach to obtain a simplified and approximate solution for the aforementioned problems which consists in replacing the ℓ_0–norm constraint by an ℓ_1–norm constraint,

$$\mathbf{b} = \mathbf{A}\mathbf{x} \;\; \text{subject to} \;\; \min \|\mathbf{x}\|_1. \tag{3.12}$$

The formulation in (3.12) can easily be transformed into a linear program and thus solved with standard linear programming routines. BP can also be thought of as a least-squares problem with an ℓ_1–norm regularizer. It can be reformulated in a Lagrangian form as

$$\min_{\mathbf{x}} \left[\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \bar{\lambda}\|\mathbf{x}\|_1 \right] \tag{3.13}$$

where $\bar{\lambda}$ is a regularization parameter.

3.1.3 Sparse representation dictionaries

One crucial question in sparse representations is the choice of the dictionary $\mathbf{A}$. One can use a variety of pre-defined sets of functions (or waveforms), e.g., wavelets [73], curvelets [74], contourlets [75], shearlets [76], bandelets [77], wedgelets [78]. However, both the sparsity and the quality of the representation depend on how well the used dictionary is adapted to the data at hand. The problem of dictionary learning, or even simply finding adaptive ways to construct/select relevant dictionaries, for sparse representations —that goes beyond the concatenation of a few off-the-shelf bases (like DCT, discrete Fourier transform (DFT), wavelets, or the others)— has therefore become a key issue for further progress in this area.

Note here that when we talk about a "dictionary", we refer to an **overcomplete** or **redundant** set of functions (atoms) stored in the columns of a full rank matrix $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$, $\mathrm{N} \leqslant \mathrm{M}$, where $\mathrm{M} = \eta\mathrm{N}$ and $\eta \geqslant 1$ being defined as the *redundancy factor* of the dictionary.

3.1.3.1 Static dictionaries

A simple and fast way to obtain a sparse representation dictionary is to choose a pre-specified transform matrix. This is indeed the case for overcomplete DFT, DCT, wavelets, and so on. The success of such a dictionary depends on how well the set of analysed signals can be represented sparsely in that domain. Generally, these kinds of dictionaries are particularly well suited for representing periodic texture patterns. However, they fail for more complex and non-periodic structures. We will consider a dictionary as *static*, if its atoms are generated by means of a pre-defined set of *analytic* functions.

DFT dictionary

DFT decomposes a discrete time-domain function into components of different frequencies, i.e., harmonics. It is known to be efficient in representing uniformly smooth signals and patterns. However, its strong smoothing property makes it difficult to represent complex structures and discontinuities.

Formally, given a one-dimensional signal $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$, the DFT coefficients vector $\mathbf{B} \in \mathbb{R}^{\mathrm{M}}$, $\mathrm{N} \leqslant \mathrm{M}$, can be obtained by the following formula

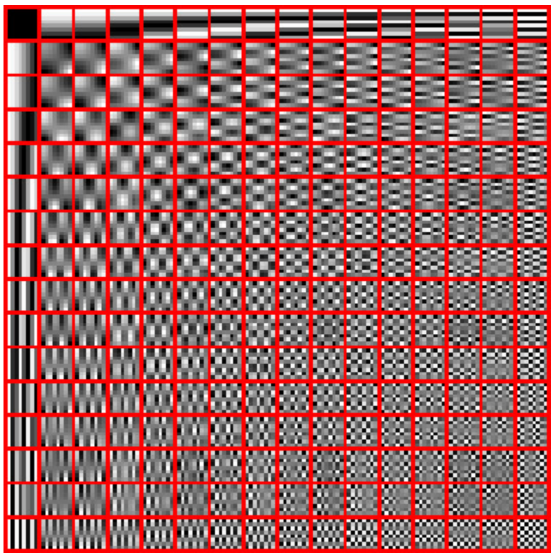


Figure 3.1: An overcomplete real-DFT dictionary - The dictionary is constructed with atom size 8×8 and redundancy factor $\eta = 4$.

$$B_m = \sum_{n=0}^{\mathrm{N}-1} b_n \boldsymbol{\phi}_{m,n} \tag{3.14}$$

where $\boldsymbol{\phi}_{m,n} = \exp\left(-j\frac{2\pi}{\mathrm{M}}mn\right)$ for $m = 0...\mathrm{M}-1$ and $n = 0...\mathrm{N}-1$.[1] A one-dimensional overcomplete DFT dictionary can be constructed using the basis functions $\boldsymbol{\phi}_{m,n}$.

Now we turn our attention to two-dimensional signals where the atoms can be generated by an extended function

$$\hat{\boldsymbol{\phi}}_{m_1,m_2,n_1,n_2} = \boldsymbol{\phi}_{m_1,n_1}\boldsymbol{\phi}_{m_2,n_2} \tag{3.15}$$

for $m_1 = 0...\mathrm{M}_1 - 1$, $m_2 = 0...\mathrm{M}_2 - 1$, $n_1 = 0...\mathrm{N}_1 - 1$, and $n_2 = 0...\mathrm{N}_2 - 1$. In this case, the columns of the dictionary $\mathbf{A}$ correspond to the vectorized and then ℓ_2–normalized forms of the generated atoms.

Note here that DFT signals are complex-valued. A general procedure consist in using only the real-part of the dictionary. An overcomplete real-DFT dictionary with atom size 8×8, i.e., $\mathrm{N} = 64$, and $\eta = 4$, i.e., $\mathrm{M} = 256$, is shown in Fig. 3.1.

[1]When dealing with a sequence, let us say of length N, it is assumed that the individual elements of the sequence are indexed in the range [1...N], however, here because of the definition of DFT, without loss of generality, the elements are indexed in the range $[0...\mathrm{N}-1]$.

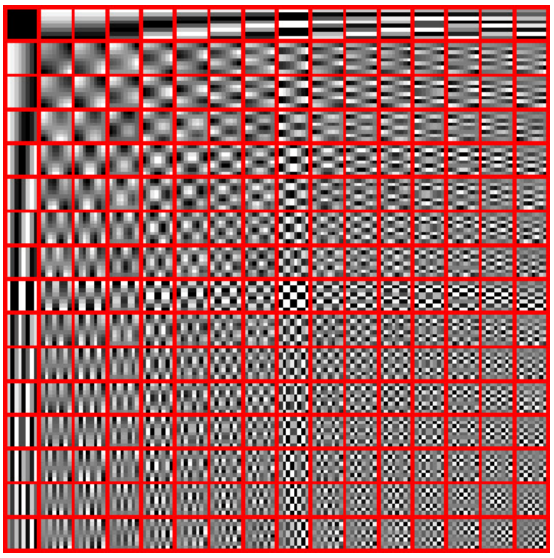


Figure 3.2: An overcomplete DCT dictionary - The dictionary is constructed with atom size 8×8 and redundancy factor $\eta = 4$.

DCT dictionary

DCT is also a Fourier related transformation (i.e., similar to DFT) which expresses a discrete time-domain function in terms of a sum of "cosine" basis functions oscillating at different frequencies. It assumes an anti-symmetric extension of the signal resulting in a more efficient signal representation when compared to DFT. Furthermore, DCT has a great advantage of producing real-valued coefficients which makes it to be preferred in many practical applications.

In theory, given a one-dimensional signal $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$, the DCT coefficients vector $\mathbf{B} \in \mathbb{R}^{\mathrm{M}}$, $\mathrm{N} \leqslant \mathrm{M}$, can be obtained by

$$B_m = \sum_{n=0}^{\mathrm{N}-1} b_n \boldsymbol{\varphi}_{m,n} \tag{3.16}$$

where $\boldsymbol{\varphi}_{m,n} = \cos\left[\frac{\pi}{\mathrm{M}}\left(n + \frac{1}{2}\right) m\right]$ for $m = 0...\mathrm{M}-1$ and $n = 0...\mathrm{N}-1$.

Here again two-dimensional atoms can be generated by an extended function

$$\hat{\boldsymbol{\varphi}}_{m_1,m_2,n_1,n_2} = \boldsymbol{\varphi}_{m_1,n_1}\boldsymbol{\varphi}_{m_2,n_2} \tag{3.17}$$

where $m_1 = 0...\mathrm{M}_1 - 1$, $m_2 = 0...\mathrm{M}_2 - 1$, $n_1 = 0...\mathrm{N}_1 - 1$, and $n_2 = 0...\mathrm{N}_2 - 1$. The columns $\mathbf{a}_m$ of the dictionary $\mathbf{A}$ correspond to the vector forms of the generated atoms where $\|\mathbf{a}_m\|_2 = 1 \ \forall m$. An overcomplete DCT dictionary with atom size 8×8, i.e., $\mathrm{N} = 64$, and $\eta = 4$, i.e., $\mathrm{M} = 256$, is shown in Fig. 3.2.

Time-frequency (space-frequency) dictionaries

A better *localization* is highly required to achieve sparsity [79]. Atoms with fixed supports, e.g., DFT and DCT, have less flexible representations which can not adapt to possible irregularities. In order to achieve better sparsity, atoms with concentrated supports on local signal characteristics should be considered. In this context, the Short Time Fourier Transform (STFT) [80], which is basically a localized Fourier transform revealing the time-frequency relation of the signal, has been extended and generalized leading to a well-known time-frequency decomposition called the *Gabor* transform [81, 82, 83].

A single window function $g(t)$ can be scaled, translated, and modulated in order to generate a family of time-frequency atoms [66]. For a given scale $s > 0$, frequency modulating ξ, and translation τ, we denote the parameter set $\gamma = \{s, \xi, \tau\}$ and define the function $g_\gamma(t)$ as

$$g_\gamma(t) = \kappa(\gamma) g\left(\frac{t-\tau}{s}\right) e^{j\xi t} \tag{3.18}$$

where $\kappa(\gamma)$ is the normalization factor of $g_\gamma(t)$ such that $\|g_\gamma(t)\|_2 = 1$. If the function $g(t)$ is even (which is generally the case and typically a Gaussian), $g_\gamma(t)$ has the energy mostly concentrated in a neighborhood of τ in time domain with size propotional to s, and of ξ in frequency domain with size propotional to $1/s$.

Multi-resolution dictionaries

Most of the natural signals, and also images, show meaningful structural properties over different resolutions, hence can be analysed efficiently in a multi-resolution framework such as *wavelet analysis* [73]. A multi-resolution wavelet basis (dictionary atoms) can be constructed from two localized functions referred to as the *scaling function* (which is a low frequency signal) and the *mother wavelet* (which is a high frequency signal) [84, 85]. The scaling function covers the wavelet spectrum and approximates the signal coarsely with its translations, whereas the mother wavelet $\psi(t)$ approximates the signal details with its various dilations and translations, i.e., $\psi\left(\frac{t-\tau}{s}\right)$.

3.1.3.2 Adaptive dictionaries

The *analytic* dictionary construction approaches as defined above fundamentally correspond to the pre-defined implicit description of highly structural dictionaries. Although these approaches have a fast mathematical description, hence simplicity in numerical implementation, the choice of the dictionary requires a strong pre-knowledge of the structure of the data for an efficient signal representation. In practice, for natural image signals, it has already been proven that learning the dictionary from a set of training samples leads to much better representation results since the learned dictionary has been well-adapted and fine-tuned to the given set of training data or a particular image. Thus, various researchers have developed various dictionary learning schemes in order to provide adapted dictionaries for the data considered. These dictionaries are usually learned with ℓ_0–norm (or ℓ_1–norm) constraints so that the data can be represented efficiently by sparse decomposition algorithms.

The popular dictionary learning algorithms include Method of Optimal Directions (MOD) [86], unions of orthobases [87, 88], K–SVD [89], and (Generalized) Principle Component Analysis ((G)PCA) [90, 91], and more. These learning methods are often conducted as an offline pre-processing step resulting in a significant improvement in signal representation when compared to off-the-shelf (i.e., static) bases or dictionaries. Chapter 4 is devoted to the basic principles of these dictionary learning methods and especially to their adaptation to the image prediction problem.

3.2 Image Prediction based on Sparse Representations

3.2.1 Background work

The first sparse spatial image prediction approach with static dictionaries formed by DCT and/or real-DFT basis functions has been proposed by Martin *et al.* [92] in which image prediction is regarded as a problem of signal extension (extrapolation) from noisy data taken from a causal neighborhood. The sparse signal approximation method is run with a set of masked (DCT and/or real-DFT) basis functions, the masked samples correspond to the spatial location of the pixels to be predicted. The basic principle of this approach is to first search for a linear combination of masked basis functions which best approximates known pixel values in the causal neighborhood, and keep the

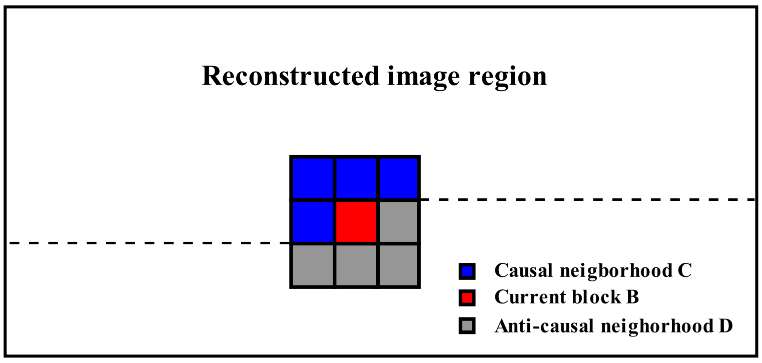


Figure 3.3: Sparse image prediction method proposed by Martin *et al.* - C is the causal neighborhood and D is the anti-causal neighborhood of the current block to be predicted B of size $n \times n$.

same linear combination of basis functions to extrapolate the unknown sample values in the block to be predicted. The stopping criterion (which is the energy of the residual signal) is computed on the known causal neighborhood. To compute it on the causal neighborhood would lead to a residue of small energy, however, this residue might take potentially large values in the block to be predicted. The optimum number of used atoms (basis functions) is selected in order to minimize a given criterion (i.e., the energy of the residual signal) on the block to be predicted, and then it is transmitted to the decoder. The decoder similarly runs the algorithm with the masked basis functions by taking the previously decoded neighborhood as the known support. The optimum number of atoms selected by the encoder can thus be used by the decoder as a stopping criterion.

The algorithm

Let S denote a region in the image containing the block B of $n \times n$ pixels to be predicted, and its causal and anti-causal neighborhoods C and D respectively as shown in Fig. 3.3. The entire region S contains nine blocks and hence of size $3n \times 3n$ pixels of DCT or real-DFT dictionary basis functions have been generated with $\eta = 1$, i.e., a complete dictionary $\mathbf{A}$ of size $9n^2 \times 9n^2$. However, the dictionary $\mathbf{A}$ can also be formed by a concatenation of DCT and real-DFT basis functions, or even can be extended to include some other basis functions such as Gabor or wavelets.

A column vector $\mathbf{b}$ of dimension $9n^2$ has been formed with the pixel values of the entire region S. Note here that there are known values (4 blocks of the causal neighborhood C) and unknown values (the area containing the block B and the anti-causal neighborhood D) in the region S. Only the known pixel values in the vector $\mathbf{b}$ are kept in a compacted vector $\mathbf{b}_c$ of size $4n^2$. Similarly, the dictionary $\mathbf{A}$ is modified by masking its rows corresponding to the spatial location of the pixels which are not in the causal neighborhood C. A compacted dictionary matrix $\mathbf{A}_c$ of size $4n^2 \times 9n^2$ is obtained. The sparse representations algorithm, i.e., MP or global matched filter (GMF) [93], then proceeds by solving a constrained approximate minimization as

$$\arg\min_{\mathbf{x}} \|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\|_2^2 \quad \text{subject to} \quad \min_{\mathbf{x}} \|\mathbf{b}_t - \mathbf{A}_t\mathbf{x}\|_2^2 \quad \text{and} \quad \|\mathbf{x}\|_0 \leqslant K \tag{3.19}$$

where the vector $\mathbf{b}_t$ of size n^2 represents the original pixel values of the block to be predicted B, $\mathbf{A}_t$ is the corresponding spatial dictionary obtained by masking the rows of $\mathbf{A}$ with respect to the spatial location of the pixels which are not in B, and K being the maximum number of allowed iterations (or similarly the maximum number of allowed non-zero components in $\mathbf{x}$).

3.2.2 Locally adaptive dictionaries

The dictionaries formed by DCT and/or real-DFT waveforms are particularly well suited for predicting periodic texture patches. However, the prediction fails for more complex and non-periodic structures with discontinuities. Instead of considering dictionaries formed by fixed and pre-defined waveforms, the approach here consists in searching for a solution to the constrained sparse approximation (3.19) of the vector $\mathbf{b}_c$ with **dynamic** and **locally adaptive** dictionaries which are formed by atoms derived from possibly smoothed and normalized **texture patches** present in a larger search window W in a causal vicinity of the block to be predicted (see Fig. 3.4). The principle of the approach is thus to first search for a linear combination of *image patches* (where the luminance values are stacked as columns in the dictionary matrix $\mathbf{A}$) which best approximates the causal neighborhood C, and then keep the same linear combination of co-located pixel values to extrapolate the unknown sample values in the block to be predicted B.

Notice here that, except for the bordering blocks, every single block B in the image has its own dictionary matrix $\mathbf{A}$ in which the atoms are locally characterized by the

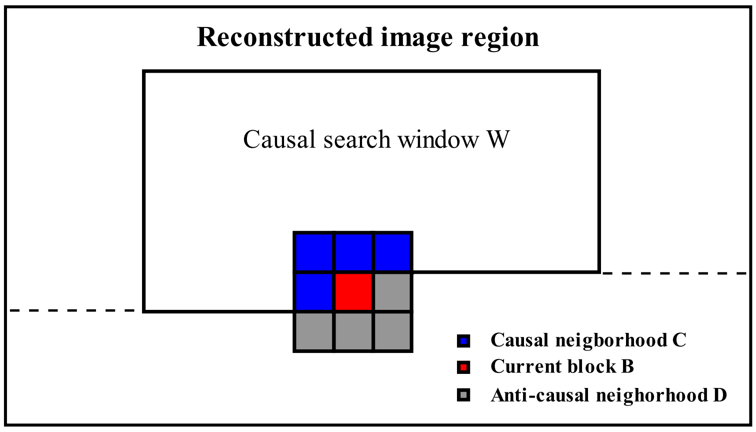


Figure 3.4: Locally adaptive dictionary construction - C is the causal neighborhood and D is the anti-causal neighborhood of the current block to be predicted B of size $n \times n$. W is the search window from which texture patches are taken to construct the dictionary $\mathbf{A}$ to be used for the prediction of B.

spatial location of B and also the size of the search window W. The size of the search window is a parameter which controls the dimension of the dictionary $\mathbf{A}$, hence of the CPU time needed to solve the approximation problem and the quality of the approximation. Different methods can be envisaged for selecting the texture patches, which will be used as atoms, in the search window W. The use of a causal search window guarantees that the decoder can construct exactly the same dictionary.

3.2.3 Dynamic approximation supports

In the method proposed by [92], there is only one approximation support (i.e., the causal neighborhood C) which has been fixed. Limiting the image prediction problem with only one rigid approximation support will obviously lead to less efficient prediction results since natural images usually contain very complex and non-periodic textural and structural areas. Instead of considering only one fixed approximation support (which might not be adaptive to all possible irregularities in the image), the approach here consists in defining a set of pre-defined *dynamic* approximation supports. Although there are limitless ways of obtaining new approximation supports, we have defined seven modes as shown in Fig. 3.5. Here our aim is to cover, as much as possible, the directional structures as well as the textural areas in an image, i.e., Modes 1-2-3 are capable of handling the smooth and periodic textural areas as well as the directional

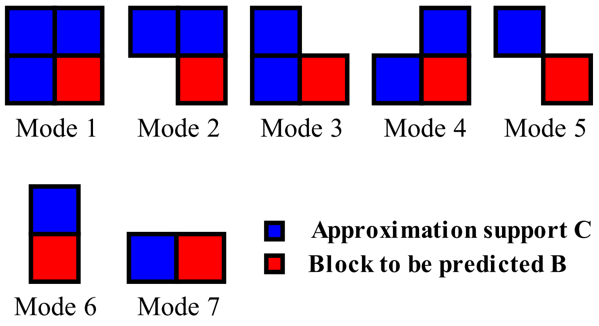


Figure 3.5: Seven possible modes for approximation support selection - The optimum support type is selected according to a given criterion.

structures with low inclinations, whereas Mode 4 and Mode 5 are highly capable of predicting the directional structures with high inclinations, and Mode 6 and Mode 7 can be efficiently used for the vertical and horizontal structures respectively.

The best approximation support is selected according to a given criterion, e.g., minimization of the energy of the residual signal on the block to be predicted B, as in the case of selecting the optimum number of used atoms or basis functions. The selected approximation mode type (which is an integer) then needs to be transmitted to the decoder so that the decoder can use the same approximation support in order to make the same prediction.

3.2.4 Optimum sparse vector selection

A sparse decomposition algorithm, e.g., MP or OMP, stops when the residue energy of the analysed signal is under a pre-specified threshold ρ, or the maximum number of allowed iteration number K is reached. In the prediction point of view, here the set of coefficients selected by the iterative algorithm leads to a good sparse representation of the causal neighborhood C, or equivalently of the vector $\mathbf{b}_c$. However, the algorithm itself does not put any constraint on the values of the block to be predicted B. Therefore, the last iteration may not lead to a best predicted signal, or to a best prediction in a rate-distortion (RD) sense when the prediction method is included in a complete image compression algorithm. This is illustrated in Fig. 3.6 which shows that, while the last iteration of the sparse decomposition algorithm always gives a good representation of the (approximation) support region, the approximation obtained for the unknown pixel values may not be the best. In order to optimize the prediction according to

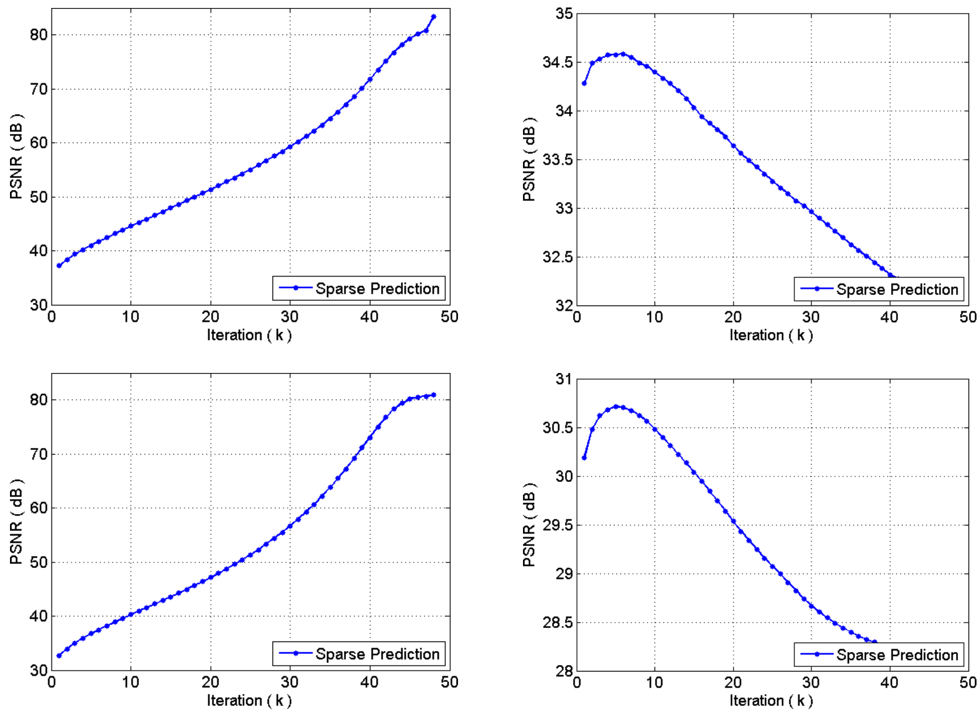


Figure 3.6: Mean prediction PSNR versus sparsity performance curves of (top) Foreman and (bottom) Barbara using sparse prediction - (left) The approximation support, (right) the unknown block. Block size is 4×4 pixels, and the approximation support Mode 1 as illustrated in Fig. 3.5 has been used.

a given criterion, one keeps track of the sparse vectors computed at each iteration of the algorithm and uses the one which satisfies the chosen criterion. Here the method proceeds as follows: the sparse representation algorithm is run until the pre-specified iteration number K is reached, leading to K possible sparse vectors $\mathbf{x}_k, k = 1...K$, one being produced at each iteration of the pursuit algorithm. Then, the optimum vector of coefficients $\mathbf{x}_{k_{opt}}$ is selected according to two different criteria:

- Minimization of the sum of squared error (SSE) on the block to be predicted $\mathbf{b}_t$ in order to observe the impact on the prediction quality, i.e.,

$$k_{opt} = \underset{1 \leqslant k \leqslant K}{\arg\min} \|\mathbf{b}_t - \mathbf{A}_t \mathbf{x}_k\|_2^2 \,. \tag{3.20}$$

- Minimization of an RD cost function of the form $J(\mathbf{D}, \mathbf{R}) = \mathbf{D} + \lambda \mathbf{R}$ when the prediction method is used in a coding scheme in order to observe the impact on the encoding efficiency, i.e.,

$$k_{opt} = \underset{1 \leqslant k \leqslant K}{\arg\min} J_k(\mathbf{D}, \mathbf{R}), \tag{3.21}$$

where $\mathbf{D}$ is the total distortion on the block to be predicted, and $\mathbf{R}$ is the corresponding prediction residue encoding cost (in bits). Here λ represents a Lagrange multiplier which relates the distortion $\mathbf{D}$ and the rate $\mathbf{R}$.

Note here that the optimization is done in two steps, i.e., first for the selection of the optimum iteration number k, and then for the selection of the optimum dynamic approximation support type. The corresponding iteration number and support mode type then need to be transmitted to the decoder. The decoder similarly uses the signaled approximation support type by iterating the pursuit algorithm k_{opt} iterations in order to predict the unknown pixel values in B. The predicted signal $\hat{\mathbf{b}}_t$ is simply calculated by multiplying the dictionary matrix $\mathbf{A}_t$ by $\mathbf{x}_{k_{opt}}$ as

$$\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}_{k_{opt}}. \tag{3.22}$$

Note also that the columns (atoms $\mathbf{a}_{t_m}$) of $\mathbf{A}_t$ are first to be normalized with the norm of corresponding atoms of $\mathbf{A}_c$, i.e., $\mathbf{a}_{t_m} = \mathbf{a}_{t_m} / \|\mathbf{a}_{c_m}\|_2 \ \forall m$, and then the atoms of $\mathbf{A}_c$ are normalized in the ℓ_2–norm, i.e., $\|\mathbf{a}_{c_m}\|_2 = 1 \ \forall m$.

The sparse spatial image prediction algorithm based on sparse representations using OMP is summarized in Table 3.1. Notice again that the dictionary matrices $\mathbf{A}_c$ and $\mathbf{A}_t$ here are updated adaptively (per block) according to the spatial location of the block to be predicted B.

3.3 Image Prediction based on Template Matching

Template matching (TM) based algorithms have been widely considered for texture synthesis problems such as intra image prediction [39, 40, 41, 42], exemplar-based image inpainting [43, 44], and so on. In this section, we give the underlying basic ideas, and also possible extensions, of the TM approach in the image prediction framework.

```
**Inputs:** $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, $K$
**Output:** Predicted values of unknowns $\hat{\mathbf{b}}_t$
**Initialization:** $k = 0$, $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b}_c$, $\mathbf{A}_c^0 = [\,]$
  **repeat** until $k = K$
    $k = k + 1$
    $m_k = \arg\max_m \left|\mathbf{A}_c^{\mathrm{T}} \mathbf{r}_{k-1}\right|$
    $\mathbf{A}_c^k = \mathbf{A}_c^{k-1} \cup \{\mathbf{a}_{c_{m_k}}\}$
    $\boldsymbol{\alpha}_k = \mathbf{A}_c^{k+} \mathbf{b}_c \to \mathbf{x}_k$
    $\mathbf{r}_k = \mathbf{b}_c - \mathbf{A}_c \mathbf{x}_k$
    $\mathbf{p}_k = \mathbf{A}_t \mathbf{x}_k$
  end **repeat**
Select the optimum $k$ minimizing the selected criterion
Set $\hat{\mathbf{b}}_t = \mathbf{p}_{k_{opt}}$
```

Table 3.1: Image prediction algorithm based on sparse representations using OMP - The proposed sparse prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

3.3.1 Simple template matching (TM)

The main idea of TM consists in estimating the unknown (luminance and chrominance) pixel values using image patches in a local search window where all the pixel values are known and available for processing. More precisely, the values of each pixel to be estimated (or predicted) are determined by comparing their spatial neighboring pixels, so-called the *template*, with all candidate neighborhoods in the search window, and assigning the candidate pixel values as a predictor (or synthesizer) for the unknown sample values by minimizing a distance measure between the template and the candidate neighborhood. The distance measures used are usually classical ones such as the sum of absolute error (SAE), SSE, etc. Fig. 3.7 demonstrates a simple illustration of the TM algorithm for block-based intra image prediction application.

The TM approach highly relies on the assumption that the estimated (predicted) texture contains a similar textural and structural characteristics as the template. However, in natural images, there are more complex and textured structures where template matching might not be sufficient in terms of visual and statistical quality, and even it fails completely in some cases. The idea instead is to consider the proposed prediction method based on sparse representations. Notice here that sparse prediction with

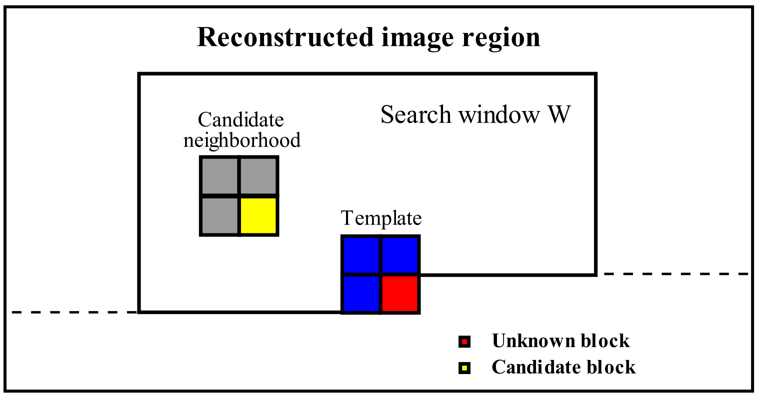


Figure 3.7: Intra image prediction based on template matching - A template is formed by previously encoded pixels in the close neighborhood of the unknown block. The best match between the template and the candidate neighborhood, within the search window W, allows finding the predictor (candidate block) of the unknown block.

locally adaptive dictionaries can be regarded as an extension of the TM method since the texture prediction process is carried out as a weighted linear combination of more than one image patch in the search window W. In contrary to simple TM where texture is generated by only one patch with a weighting coefficient equal to 1, the weighting coefficients of the sparse prediction (SP) method are calculated iteratively by the sparse decomposition algorithm OMP.

Below, we first start with formulating the template matching algorithm in terms of our notation in this manuscript and then adapt our proposed sparse prediction method in order to have a fair comparison with simple template matching.

Image prediction based on template matching

Let S denote a region in the image containing the unknown block of size $n \times n$ and its template as shown in Fig. 3.7. The region S contains 4 blocks hence of size $\mathrm{N} = 2n \times 2n$ pixels for running the TM algorithm. Suppose that the columns $\mathbf{a}_m$ of a matrix $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ are constructed by stacking the luminance values of all possible patches, of size $2n \times 2n$ pixels, in a given causal search window W in the reconstructed image region. Here the matrix $\mathbf{A}$ will also be referred as the *dictionary*, however, the columns (atoms) will not be normalized in contrary to the SP method. Assume that the N sample values of the region S are stacked in a column vector $\mathbf{b}$.

Static template prediction: A *static template* is considered as the commonly used conventional template as shown in Fig. 3.7 (also in Fig. 3.5 as Mode 1). Let us suppose that the static template is used for prediction. For the first step, that is the search for a best match of the known pixel values in the template, the matrix $\mathbf{A}$ is modified by masking its rows corresponding to the spatial location of the pixels of the unknown block. A compacted dictionary $\mathbf{A}_c$ of size $3n^2 \times \mathrm{M}$ is obtained in which actually the columns $\mathbf{a}_{c_m}$ correspond to the possible *candidate neighborhoods* in the search window. The vector $\mathbf{b}$ is also compacted in $\mathbf{b}_c$ with the known pixels in the template of size $3n^2$ values. The template matching algorithm then proceeds by calculating

$$m_{opt} = \underset{m \in [1 \ldots \mathrm{M}]}{\arg\min} \left\{ d_m : d_m = \mathrm{DIST}\left(\mathbf{b}_c, \mathbf{a}_{c_m}\right) \right\} \tag{3.23}$$

where the operator DIST denotes a distance metric such as SSE.

The prediction signal $\hat{\mathbf{b}}_t$ is simply assigned by the sample values of the candidate block $\mathbf{a}_{t_{m_{opt}}}$ as $\hat{\mathbf{b}}_t = \mathbf{a}_{t_{m_{opt}}}$. The columns $\mathbf{a}_{t_m}$ of $\mathbf{A}_t$ correspond to the possible *candidate blocks* in the search window, where the matrix $\mathbf{A}_t$ is obtained by masking the rows of the dictionary $\mathbf{A}$ corresponding to the spatial location of the pixels of the template.

Optimized dynamic templates: The optimum dynamic template is selected among seven pre-defined approximation supports as shown in Fig. 3.5. Here also the optimization is conducted according to two criteria, i.e., 1. minimization of the prediction signal SSE (or MSE); 2. minimization of the RD cost function $J(\mathbf{D}, \mathbf{R})$ when the TM algorithm used in a coding scheme.

Image prediction based on sparse representations

The basic principle of the sparse prediction approach (with locally adaptive dictionaries) is to first search for a linear combination of image patches which approximates well the approximation support (template), and then keep the same linear combination of co-located pixels to predict the pixel values in the unknown block.

For the sake of a fair comparison with template matching, the sparse prediction method here is iterated only once. Let the quantity $\mathbf{x}$ denote the sparse vector which will contain the result of the sparse approximation of the current approximation support, i.e., the coefficient of the expansion of the vector $\mathbf{b}_c$ on only one atom of the dictionary matrix $\mathbf{A}_c$. The constrained approximate minimization in (3.19) then becomes

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} \|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 = 1. \tag{3.24}$$

The prediction signal $\hat{\mathbf{b}}_t$ is similarly obtained as $\hat{\mathbf{b}}_t = \mathbf{A}_t\mathbf{x}_{opt}$. Note that the columns of the matrices $\mathbf{A}_c$ and $\mathbf{A}_t$ need to be normalized accordingly as defined earlier.

3.3.2 Weighted linear combination of template matching predictors

Starting with the above definition of simple TM, a weighted linear combination of multiple TM predictors can be seen as a possible extension since there might be more than one candidate blocks which are equally important, or with a varying degree of importance but useful to be used in the prediction process.

3.3.2.1 Average template matching (ATM)

A simple consideration is of the weighting coefficients which are all equal to each other, i.e., all the weights are equal to $1/K$ if there are K number of patches to be used. We refer to this special case with uniform weights as average template matching (ATM).

Let the vector $\hat{\mathbf{b}}_t$ be predicted as a uniform average of the candidate blocks $\mathbf{a}_{t_{m_k}}, k = 1...K$, where the corresponding candidate neighborhoods $\mathbf{a}_{c_{m_k}}, k = 1...K$, which are taken from the search window, are the K nearest neighbors (e.g., in terms of a distance metric such as SSE) of the template $\mathbf{b}_c$. Assuming the columns $\mathbf{a}_{t_m}$ of the dictionary $\mathbf{A}_t$ are independent and identically distributed (i.i.d.) samples, and they have the same mean and variance, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ respectively, the *central limit theorem* states that the average of K candidate blocks tends towards to a Gaussian distribution with a mean $\boldsymbol{\mu}_{avg} = \boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2_{avg} = \frac{1}{K}\boldsymbol{\sigma}^2$. As a result, the average of K candidate blocks has a smaller variance, thus statistically a smaller prediction error can be expected when compared to the prediction with an individual candidate block, i.e., simple TM.

3.3.2.2 Non-local means (NLM)

Although ATM has an easy and straightforward way of obtaining the weighting coefficients, the weights can be calculated by other means than simply using the average. Non-local means (NLM) based approaches have already been applied to different image processing problems such as inpainting [46], restoration [47], and denoising [48, 49, 50]. Here we apply this method to our image prediction problem. Similar to ATM, the NLM based method tries to aggregate multiple image patches as a weighted linear combination, but the weighting coefficients are calculated in a different way. The idea here is to express the weights in terms of the amount of similarity between the candidate neighbor patches and the template, i.e., the contribution weights are calculated with a *patch similarity based kernel function* (e.g., exponential) in order to give more attention to the neighboring patches which are more similar to the template than the others.

Assume here also the vector $\hat{\mathbf{b}}_t$ be predicted as a weighted linear combination of the candidate blocks $\mathbf{a}_{t_{m_k}}, k = 1...K$, where the corresponding candidate neighborhoods $\mathbf{a}_{c_{m_k}}, k = 1...K$, taken from the search window are the K nearest neighbors of the template $\mathbf{b}_c$. By using an exponential kernel function, the weigthing coefficients vector $\boldsymbol{\alpha} = [\alpha_1 \; ... \; \alpha_K]^{\mathrm{T}}$ can then be calculated as follows

$$\alpha_k = \exp\left(-\frac{\mathrm{DIST}\left(\mathbf{b}_c, \mathbf{a}_{c_{m_k}}\right)}{h}\right), \quad k = 1...K, \tag{3.25}$$

where h is a decay coefficient. The calculated weights are finally normalized to sum-to-one, i.e., $\boldsymbol{\alpha} = \boldsymbol{\alpha}/sum(\boldsymbol{\alpha})$, in order to avoid a possible overflow in the predicted values of the pixels.

For the sake of a fair comparison with sparse prediction, the ATM and NLM based methods here are iterated in a loop of increasing $k, k = 1...K$, by keeping track of the SSE or the RD cost function values for the block to be predicted. The optimum number k_{opt} and dynamic template mode type, which minimize the considered criterion, are then transmitted to the decoder similar to SP. The ATM and NLM based image prediction algorithms are summarized in Table 3.2. Please note here that these methods differ from SP in sense that they do not have an optimization on the approximation of the template, whereas SP runs an optimization on the approximation support region (template) for calculating the weighting coefficients.

3.4 Experimental Results

3.4.1 Prediction quality evaluation of adaptive dictionaries

The proposed adaptive dictionary construction method in this chapter has first been assessed against the overcomplete DCT dictionary in terms of visual prediction quality

```
Input:   $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, $K$
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
Initialization: $k = 0$, $\boldsymbol{\alpha} = [\,]$, $\mathbf{A}_c^0 = [\,]$, $\mathbf{A}_t^0 = [\,]$
  repeat until $k = K$
    $k = k + 1$
    $m_k = \arg\min_m \|\mathbf{b}_c - \mathbf{a}_{c_m}\|_2^2$ where $m \notin \{m_1, ..., m_{k-1}\}$
    $\mathbf{A}_c^k = \mathbf{A}_c^{k-1} \cup \{\mathbf{a}_{c_{m_k}}\}$ and $\mathbf{A}_t^k = \mathbf{A}_t^{k-1} \cup \{\mathbf{a}_{t_{m_k}}\}$
      if ATM: $\forall k' \in [1, k]\ \alpha_{k'} = 1/k \to \boldsymbol{\alpha}$
      if NLM: $\forall k' \in [1, k]\ \alpha_{k'} = \exp\left(-h^{-1}\mathrm{DIST}\left(\mathbf{b}_c, \mathbf{a}_{c_{k'}}^k\right)\right) \to \boldsymbol{\alpha}$
    $\boldsymbol{\alpha} = \boldsymbol{\alpha}/sum(\boldsymbol{\alpha})$
    $\mathbf{p}_k = \mathbf{A}_t^k \boldsymbol{\alpha}$
  end repeat
Select the optimum $k$ minimizing the selected criterion
Set $\hat{\mathbf{b}}_t = \mathbf{p}_{k_{opt}}$
```

Table 3.2: Image prediction algorithm based on ATM and NLM - The proposed prediction method has been drawn here for one template, however, its extension to optimized dynamic templates is straightforward.

and also of peak signal-to-noise ratio (PSNR)[1] using both one fixed approximation support as proposed in [92] and dynamic supports selection (see Fig. 3.5). In order to observe the impact on the prediction quality, the optimum iteration number k and also the support mode type (if dynamic selection is enabled) are selected by minimizing the SSE on the block to be predicted $\mathbf{b}_t$ in a lossless encoder/decoder structure.

In the experiments reported here, the OMP algorithm has been used with a maximum allowed iteration number $K = 8$. Block size is fixed to 8×8 pixels, and both DCT and adaptive dictionaries are constructed with a redundancy factor $\eta = 4$, e.g., $\mathbf{A}$ is of size 576×2304 for a region S of size 24×24 pixels. In the case when dynamic supports selection is enabled, it is assumed that the region S contains 16×16 pixels, hence $\mathbf{A}$ is of size 256×1024 for both DCT and adaptive dictionaries.

[1]PSNR (in decibels) is defined as the ratio between the maximum possible power of a signal I and the power of corrupting noise affecting its representation $\tilde{I}$. Formally,

$$\mathrm{PSNR} = 10\log_{10}\left(I_{\max}^2/\mathrm{MSE}\right) \ \ [\mathrm{dB}]$$

where $I_{\max}$ is the maximum possible value of I, i.e., $I_{\max} = 255$ if the signal is represented with 8 bits/sample, and MSE is the mean square error between the signal I and its noisy representation $\tilde{I}$.

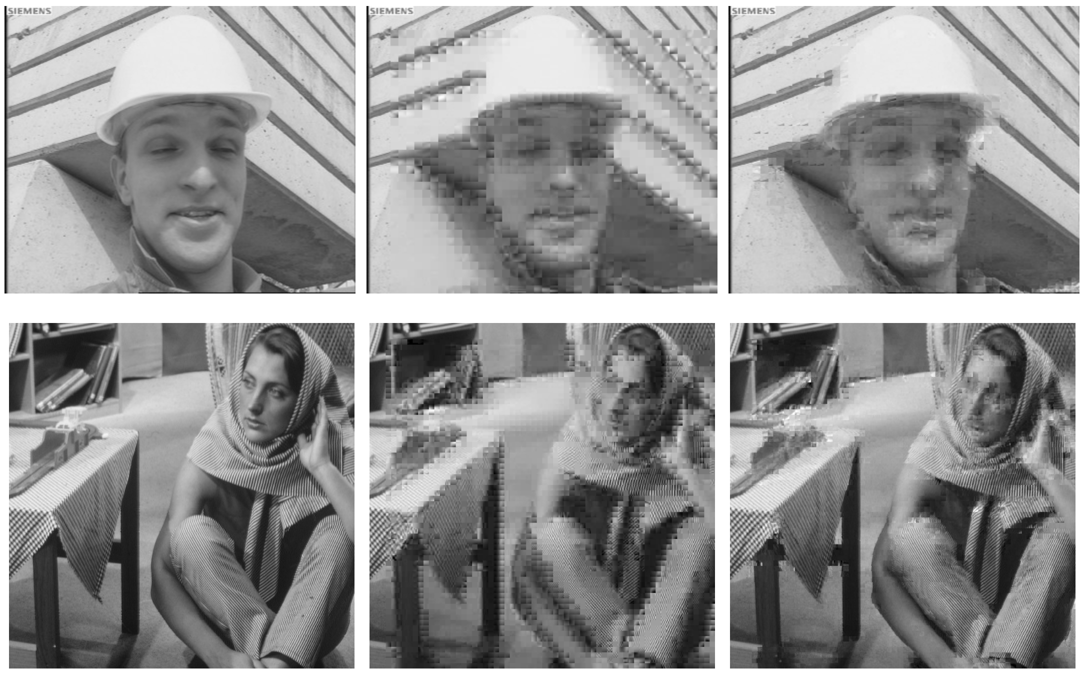


Figure 3.8: Predicted images of (top) Foreman and (bottom) Barbara using overcomplete DCT and adaptive dictionaries with one approximation support as in Martin *et al.* - (left) The original image, (middle) the predicted image with DCT dictionary, and (right) the predicted image with adaptive dictionaries.

	Fixed support [92]		**Dynamic supports**	
Image name	**DCT**	**Adaptive**	**DCT**	**Adaptive**
Barbara (512 × 512)	21.04 dB	23.18 dB	22.84 dB	25.44 dB
Foreman (CIF)	23.01 dB	27.01 dB	25.31 dB	29.43 dB
Roof (512 × 512)	19.80 dB	21.09 dB	20.90 dB	23.31 dB
Cameraman (256 × 256)	18.02 dB	18.73 dB	19.88 dB	20.77 dB

Table 3.3: Prediction PSNR results obtained for various images - Prediction results for Barbara, Foreman, Roof, and Cameraman images. Optimization criterion is the minimization of the prediction signal SSE.

Fig. 3.8 shows the predicted images of Foreman (CIF) and Barbara (512 × 512) using only one fixed approximation support with overcomplete DCT and adaptive dictionaries. Fig. 3.9 demonstrates the prediction results of the same test images using dynamic approximation supports selection.

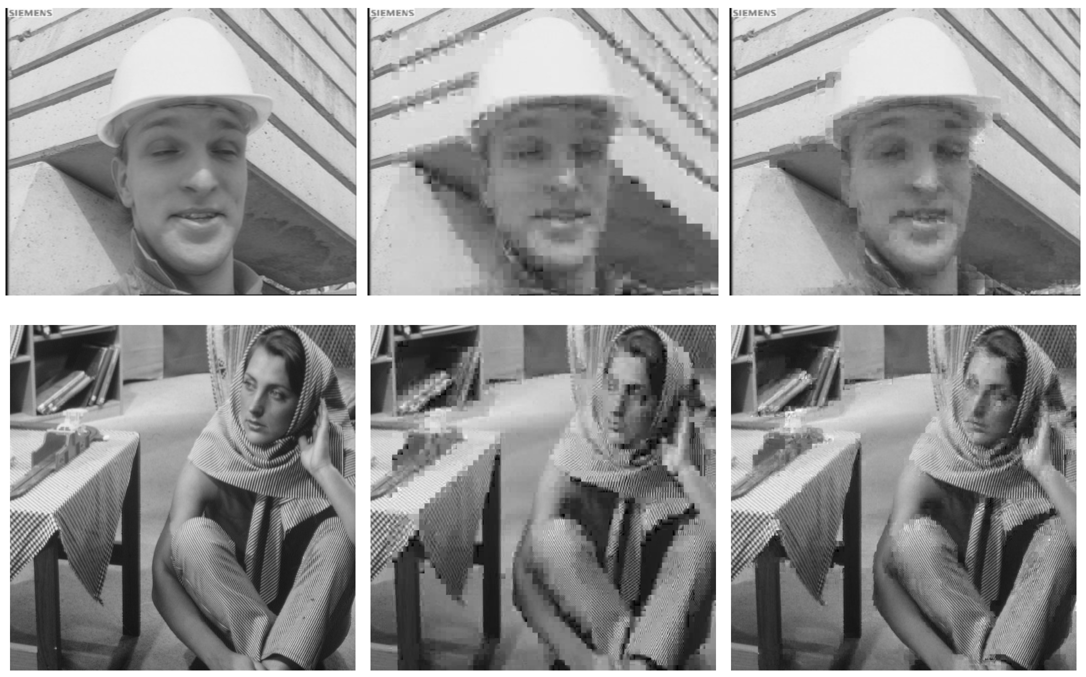


Figure 3.9: Predicted images of (top) Foreman and (bottom) Barbara using overcomplete DCT and adaptive dictionaries with dynamic approximation supports - (left) The original image, (middle) the predicted image with DCT dictionary, and (right) the predicted image with adaptive dictionaries.

The adaptive dictionaries allow to better account for more complex structures, in particular for the areas with contours and discontinuities. The quality of the predicted signal is significantly improved when compared to the DCT dictionary. This fact can be observed clearly from the predicted images. Moreover, the usage of dynamic supports further improves the prediction quality visually and also in terms of PSNR. All the experimental results obtained for various images, in terms of prediction PSNR in decibels (dB), are given in Table 3.3.

3.4.2 Rate distortion evaluation of adaptive dictionaries

The proposed adaptive dictionary construction method has also been assessed in a complete image coding scheme by comparing it to the overcomplete DCT dictionary. The performance assessment is done both in terms of prediction quality and encoding PSNR/bit-rate efficiency.

In order to initialize the prediction/compression process, the top 3 rows and left 3 columns of blocks of size 8×8 pixels are intra encoded with JPEG algorithm. Image blocks are processed in a raster scan order. Once a block has been predicted with the respective prediction method (i.e., based on overcomplete DCT or adaptive dictionaries), the DCT transformed 8×8 residual block is quantized, zig-zag scanned, and encoded with an algorithm similar to JPEG. In this coding structure, a uniform quantization matrix with step size $\Delta = 16$ is weighted by a quality factor. The quality factor (q_f) is increased from 10 to 90 with a step size of 10, and the corresponding weight $\mathbf{w}_{q_f}$ is calculated by means of the following equation

$$\mathbf{w}_{q_f} = \begin{cases} 50/q_f & \text{if } q_f \leq 50 \\ 2 - 0.02q_f & \text{if } q_f > 50 \end{cases} . \tag{3.26}$$

The reconstructed image is obtained by adding the quantized residue to the prediction. The adaptive dictionaries are constructed using the texture patches in the reconstructed image. The redundancy factor η has been kept as 4 and the atoms of the dictionary $\mathbf{A}$ are smoothed with a low-pass filter in the case when adaptive dictionaries are used.

The optimum iteration number and also the optimum approximation support mode is selected by minimizing either the SSE or the RD cost function on the block to be predicted. The RD cost function is defined as of the form $J(\mathbf{D}, \mathbf{R}) = \mathbf{D} + \lambda\mathbf{R}$. Here $\mathbf{D}$ is the SSE distortion of the reconstructed block (after adding the quantized residue to the prediction), and $\mathbf{R}$ is the residual signal encoding cost which can be estimated as $\mathbf{R} = \gamma_0 M'$ for low bit-rate compression [94] with M' being defined as the number of non-zero quantized DCT coefficients, and for a DCT basis $\gamma_0 = 6.5$. By considering the uniform scalar quantizer with a quantization step size Δ (where the deadzone is equal to 2Δ), the relation between the optimum Lagrange multiplier λ_{opt} and the quantization step Δ is given by [77]

$$\lambda_{opt} = \frac{3\Delta^2}{4\gamma_0}. \tag{3.27}$$

The OMP algorithm is limited to iterate at most $K = 8$ iterations, i.e., allowing a maximum 3 bits per block bit-rate cost for signaling the value of k_{opt}. The optimum iteration number and the approximation support type, when transmitted, are Huffman encoded. A skip mode (the corresponding flag is arithmetically encoded) has also been

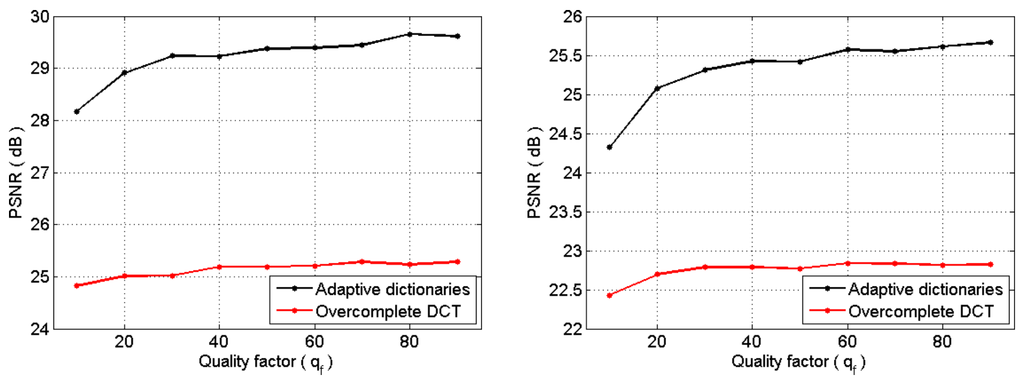


Figure 3.10: Prediction performance curves of (left) Foreman and (right) Barbara using overcomplete DCT and adaptive dictionaries - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE.

included to the encoder to avoid coding the blocks of prediction residue in which all the transformed and quantized coefficients are zero.

Fig. 3.10 shows the prediction performance curves of the proposed image prediction algorithm with adaptive dictionaries in comparison to overcomplete DCT dictionary, where k_{opt} and approximation support selection criterion is the minimization of the prediction signal SSE. The quality of the predicted signal, in terms of PSNR, is significantly improved even in the presence of different levels of quantization noise. A prediction gain up to 2.5 dB and 4 dB has been achieved for Barbara and Foreman images respectively.

Fig. 3.11 demonstrates the encoding PSNR/bit-rate performance curves of the proposed dictionary construction algorithm compared with the overcomplete DCT method in reference to JPEG standard. Here the optimization criterion is the minimization of the RD cost function on the prediction signal. The coding cost of the optimum iteration number, the approximation support type, and the skip mode have been included into total bit-rate. The encoding RD performance curves also show significant improvement when compared with the overcomplete DCT dictionary and the JPEG coding standard.

3.4.3 Sparse prediction vs. template matching based prediction

In this section, we assess comparatively the proposed sparse prediction method to the template matching based prediction techniques (i.e., TM, ATM, and NLM) with

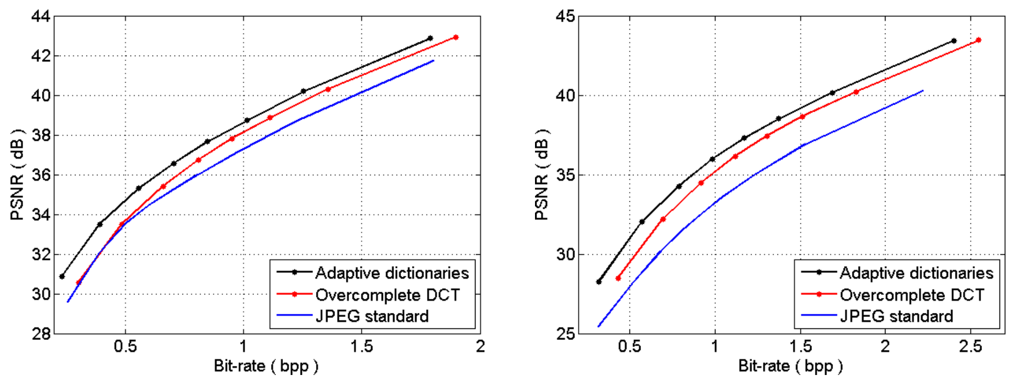


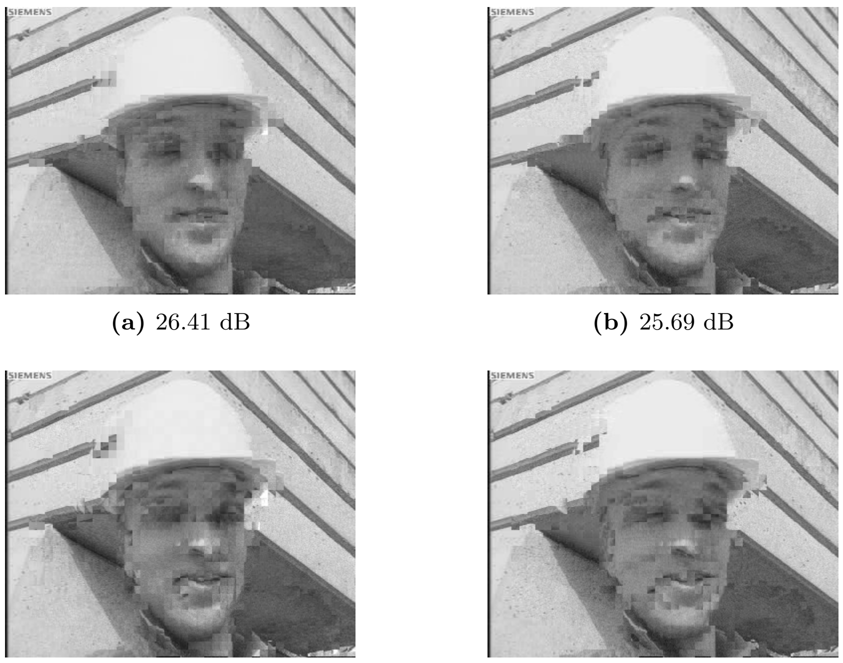
Figure 3.11: Encoding performance curves of (left) Foreman and (right) Barbara using overcomplete DCT and adaptive dictionaries in reference to JPEG standard - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the prediction signal.

a static and optimized dynamic templates (in other words, approximation supports). We plot the results of the experimental observations with a detailed comparison in terms of prediction quality (both visually and in terms of PSNR) and also encoding PSNR/bit-rate efficiency.

3.4.3.1 Sparse prediction vs. simple template matching

In the experiments reported here, block size is fixed to 8×8 pixels and the dictionary redundancy factor $\eta = 4$. In order to have a fair comparison with simple TM, the OMP algorithm in SP is iterated only once (see (3.24)), and the same dictionary $\mathbf{A}$ has been used for both of the prediction methods (except the normalization of the atoms in SP). The distance metric DIST in (3.23) is set to SSE when TM is used.

Fig. 3.12 shows the predicted images of Foreman at $q_f = 30$ where dynamic templates (supports) selection criterion (if it is enabled) is the minimization of the prediction residue SSE. Fig. 3.13 demonstrates the corresponding prediction PSNR performance curves for Foreman and Barbara images. The quality of the predicted signal, both visually and in terms of PSNR, is significantly improved when compared with the conventional template matching based prediction. Fig. 3.14 illustrates the encoding PSNR/bit-rate performance curves of the same test images using both a static and RD optimized dynamic templates. One can observe here also that the proposed method with dynamic templates selection significantly improves the encoding performance with



(a) 26.41 dB **(b)** 25.69 dB **(c)** 24.23 dB **(d)** 23.97 dB

Figure 3.12: Predicted images of Foreman using sparse prediction (one iteration) in comparison to template matching - (a) Dynamic template OMP, (b) dynamic template TM, (c) static template OMP, and (d) static template TM.

respect to the conventional static template matching prediction. A gain up to 3 dB has been achieved in both prediction quality and encoding efficiency.

3.4.3.2 Sparse prediction vs. weighted template matching

Now we turn our attention to a comparison between sparse prediction and the weighted template matching predictors (i.e., ATM and NLM). The same experimental setup (for the block size and dictionary construction) as described above has been kept for the simulations. The algorithms drawn in Table 3.1 and Table 3.2 are limited to iterate at most $K = 8$ iterations. The optimum number $k, k = 1...K$, and the dynamic template mode type, when transmitted, are Huffman encoded. The nearest neighboring patches (for ATM and NLM) are selected using the SSE distance measure. When NLM based method is used, the distance metric DIST in (3.25) has been set to MSE, and the decay coefficient $h = 25$.

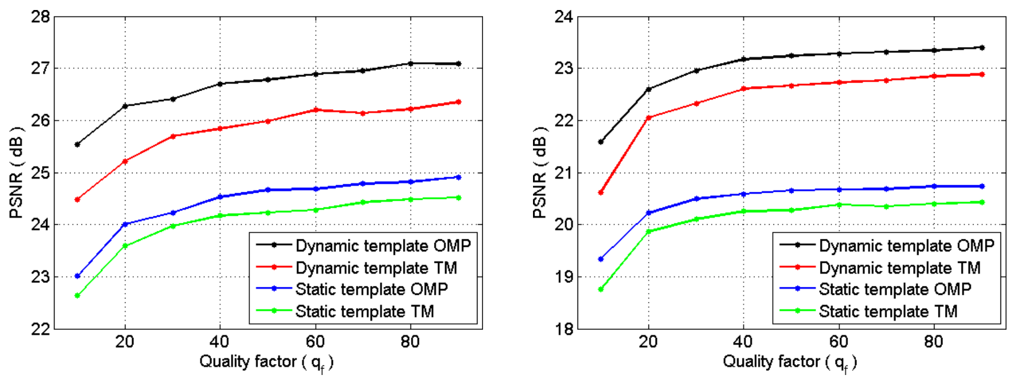


Figure 3.13: Prediction performance curves of (left) Foreman and (right) Barbara using sparse prediction (one iteration) in comparison to template matching - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE.

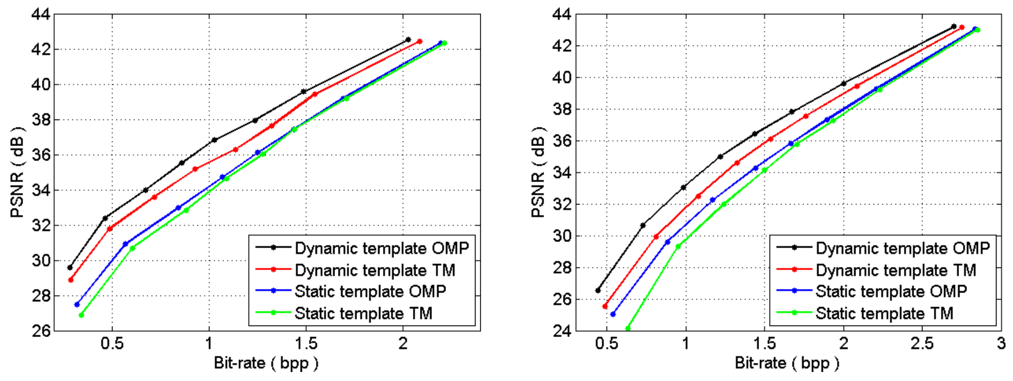


Figure 3.14: Encoding performance curves of (left) Foreman and (right) Barbara using using sparse prediction (one iteration) in comparison to template matching - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the prediction signal.

Fig. 3.15 demonstrates the prediction PSNR performance curves for Foreman and Barbara images, where the optimization criterion is the minimization of the prediction residue SSE. The quality of the predicted signal, in terms of PSNR, is greatly improved when compared with the ATM and NLM based prediction methods. It might be worthy to note here that, in terms of prediction quality, (weighted) average of predictors (i.e., SP, ATM, and NLM) definitely work better than using only one simple TM predictor.

Fig. 3.16 illustrates the encoding PSNR/bit-rate performance curves of the same

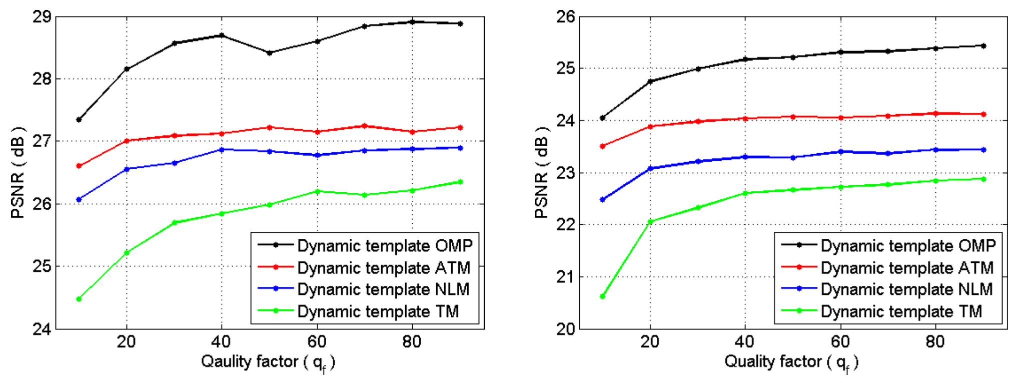


Figure 3.15: Prediction performance curves of (left) Foreman and (right) Barbara using sparse prediction in comparison to ATM and NLM - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE.

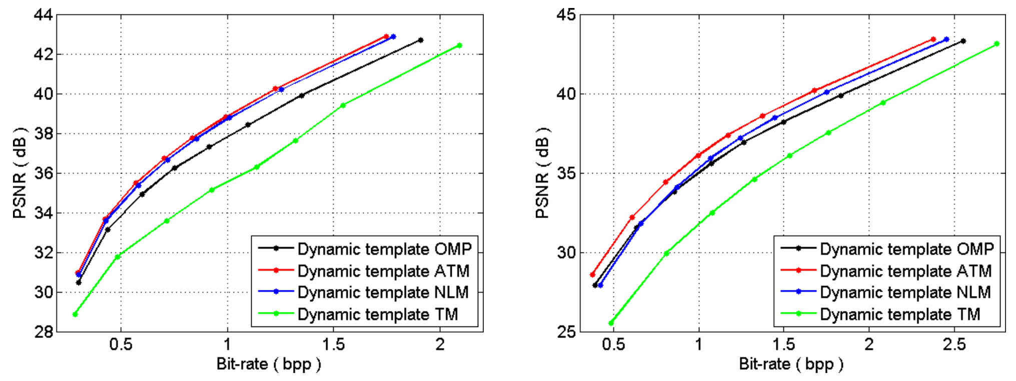


Figure 3.16: Encoding performance curves of (left) Foreman and (right) Barbara using using sparse prediction in comparison to ATM and NLM - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the prediction signal.

test images with RD optimization. In terms of encoding performance, our proposed sparse prediction method is outperformed by the ATM and NLM based methods. The reason is that the iterative sparse decomposition algorithms (e.g., MP and OMP) try to approximate residue signals at each iteration (except for the first iteration) which is not actually well suited to the image prediction problem since the residue on the approximation support (template) is not well correlated to the residue on the block to be predicted as the algorithm iterates, i.e., the spatial correlation between the approx-

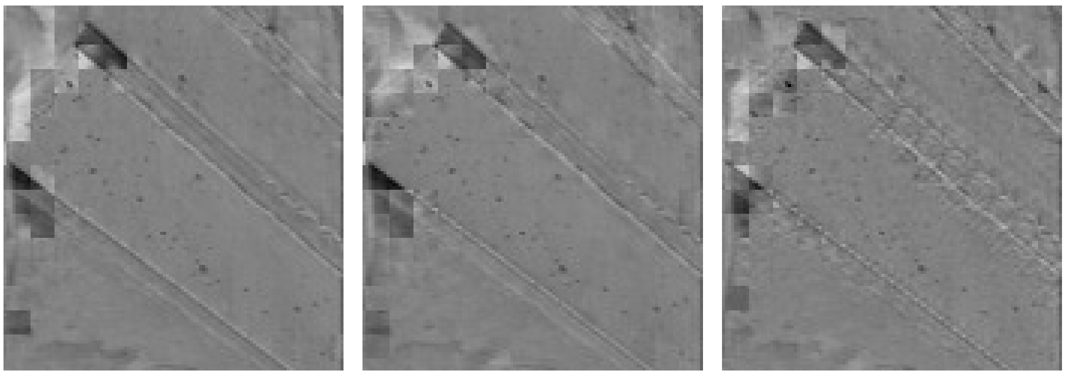


Figure 3.17: Prediction residue images of a region in Foreman using sparse prediction in comparison to ATM and NLM - (left) ATM, (middle) NLM, and (right) SP.

imation support and the block to be predicted gets weaker and weaker along with the iterations of OMP. Therefore, the obtained sparse approximation vector, which is very well suited to the template signal residue, is not likely to be well suited to approximate the residue of the block to be predicted. This fact is not so observable when one uses the SSE criterion for optimization, however, RD criterion results show that the calculated weighting coefficients (except for the first one) on the template introduce high frequency components to the prediction residue (of the block to be predicted) which leads to high bit-rate requirements for encoding.

Fig. 3.17 compares the obtained prediction residue images of a region in Foreman using SP, ATM, and NLM based prediction methods at low bit-rates ($q_f = 10$), where the RD optimization is enabled. In terms of RD criterion, the predicted image residue (which needs to be encoded and then sent to the decoder) contains a lot of high frequency components in the case when the SP method is used. As it can also be observed from Fig. 3.17, the prediction residue images of ATM and NLM are relatively smoother than that of the SP method.

3.5 Computational complexity analysis

Suppose that for a given dictionary matrix $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$, $\mathrm{N} \leqslant \mathrm{M}$, $\mathbf{A}_c$ is of size $\mathrm{N}_1 \times \mathrm{M}$ and $\mathbf{A}_t$ is of size $\mathrm{N}_2 \times \mathrm{M}$ where $\mathrm{N}_1 + \mathrm{N}_2 = \mathrm{N}$. Similarly, the known pixel values in the

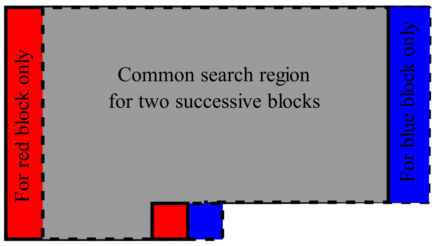


Figure 3.18: Online construction of the locally adaptive dictionaries - The gray shaded area is common for the two successive blocks shown as red and blue. The red shaded area will be combined with the gray shaded area for the prediction of the red block. The dictionary for the blue block can then be adapted from the dictionary constructed for the red block.

approximation support (template) are stored in the column vector $\mathbf{b}_c$ of size N_1. (For 8×8 block size $\mathrm{N}_2 = 64$.)

In the proposed image prediction method, there are K iterations for each approximation support mode. At any given iteration k, $k = 1...K$, of the OMP algorithm, one needs to compute MN_1 multiply-add (correlation) operations, M comparisons to select the maximum correlation, and approximately an order of $k\mathrm{N}_1$, k^2, and $k\mathrm{N}_1$ update operations to update the Gram matrix (i.e., ${\mathbf{A}^k}^{\mathrm{T}}\mathbf{A}^k$ which can be calculated by updating the one obtained at the $(k-1)^{th}$ iteration), the weighting coefficients, and the residual signal, respectively [95]. Finally, the prediction of unknown pixel values needs $k\mathrm{N}_2$ multiply-add operations. This complexity can be reduced by placing a conjugate gradient descent algorithm [96], or simply using MP (instead of OMP) which involves only N_1 update operations. Note here that the value of the parameter N_1 varies with the support type, e.g., for 8×8 block size $\mathrm{N}_1 = 192$ in Mode 1 and $\mathrm{N}_1 = 64$ in Mode 7.

In contrary to a static dictionary, the usage of locally adaptive dictionaries requires an online update step of the dictionary matrix $\mathbf{A}$ for each block to be predicted B. In practice, a local adaptation of the dictionary can be done successively by removing and introducing some atoms (texture patches) which are not in the overlap area between the two successive blocks' causal search windows. Fig. 3.18 illustrates the simple idea of updating the dictionary for two successive blocks, where the gray shaded area is in common and the atoms extracted from this area will be used in both dictionaries. The prediction of the red block will be carried out with the corresponding texture patches in the red shaded area pixels in addition to the gray shaded area. Once a dictionary has been constructed for the red block, the dictionary for the blue block can easily be obtained by removing the red shaded area texture patches from and then introducing the blue shaded area texture patches into the dictionary.

When the proposed image prediction method (SP) is compared with simple TM, the introduced computational load can be neglected as OMP has been iterated only once (please see (3.24)). On the other side, when compared with the ATM and NLM based methods, SP requires more computations since it runs an optimization algorithm for approximating the template (approximation support) in order to calculate the weighting coefficients. In the ATM and NLM based methods however, the weighting coefficients are calculated in a simple way, i.e., they are either uniform, or calculated by means of a similarity based kernel function (as we used an exponential function in the simulations).

3.6 Conclusion

In this chapter, we proposed an adaptive dictionary construction method for sparse signal representations which has been placed into a block-based intra image prediction framework. The proposed method for spatial texture prediction with optimized *dynamic approximation supports* offers new directions for coding of still images (or intra frames). We have shown that it turns out to be an effective alternative for more complex and non-periodic textural areas in terms of prediction quality and also encoding efficiency, especially when it is compared to pre-defined *static* waveform dictionaries such as DCT.

The proposed method can also be regarded as an extension of the well-known template matching based prediction algorithm. Instead of using only one image patch with a weighting coefficient equal to 1, here more than one patch has been combined linearly with the weighting coefficients calculated by a sparse decomposition algorithm such as OMP. An adapted simple version of the proposed prediction method has also been assessed comparatively to the conventional template matching technique with a static and SSE/RD optimized dynamic approximation supports (templates) resulting in a better prediction and encoding performance. However, extensive simulation results (in comparison to ATM and NLM) indicate that the prediction method proposed here suffers from the iterative sparse methods, like MP and OMP. As the OMP algorithm iterates, the residue on the template is not well correlated to the residue of the block to be predicted. Therefore, the calculated sparse representation vector (on the template) is in general not well suited to approximate the block to be predicted especially in the RD sense. This limitation of sparse prediction motivated us to solve the constrained least squares problem formulated in (3.19) with some other methods than greedy pursuit algorithms which would in particular avoid the iterative approximation of residues, but which would "directly" find the best linear combination of the texture patches to approximate the input signal. The following chapters have been devoted for this purpose.

Perspectives: In the prediction method described here, the dictionary $\mathbf{A}$ is constructed by stacking the luminance values of all patches (having the same geometric shape as the region S) in a given causal search window W in the reconstructed image region. When using those kinds of methods which are mainly based on the approximation of a support region (or template), a good representation of the support region does not necessarily lead to a good approximation of the unknown pixel values as the support region and the unknown region pixels may have different characteristics. Furthermore, in an image coding framework, at the decoder side the template information C, as well as the patches stored in the dictionary $\mathbf{A}$, may not be noise-free depending on the prediction quality and the residue signal quantization. Therefore, it is very crucial to analyze and to optimize the prediction quality of the unknown pixel values as a function of the sparsity constraint and the quantization noise. Up to this point in this chapter, we briefly introduced the basics of a heuristic method for sparse intra image prediction with dynamic and locally adaptive dictionaries. In the upcoming chapters, we will continue analyzing this method in order to observe the effect of the sparsity constraint in the case when the signals are corrupted by various levels of quantization noise, and also comparing it with the other image prediction algorithms, e.g., H.264/AVC intra prediction, or the other image prediction methods proposed by us in these chapters.

Possible future directions: A future work, which might extend this study, would be dedicated first to adapt an improved contextual dictionary construction method by classifying the image patches according to the local texture characteristics, i.e., whether the image patch is smooth, is highly textured, or contains edge information, and then to optimize the residue block quantization. Besides, directional transform based coding can be used for the residue blocks which have directional edges for more efficient encoding performance. Another interesting application would be the usage of a tree-structured pursuit algorithm [97, 98, 99, 100], instead of MP or OMP, which could adaptively exploit the atom dependencies of two different dictionaries, i.e., in our case these dictionaries are $\mathbf{A}_c$ and $\mathbf{A}_t$, not only in the spatial domain but also in the residue domain at each iteration of the algorithm.

Chapter 4

Dictionary Learning for Image Prediction

A very crucial question in sparse signal approximation algorithms is the choice of the representation dictionary. Although one can use a variety of pre-defined sets of basis functions (*static* waveforms), both the sparsity and the quality of the representation depend on how well the used dictionary is adapted to the data at hand. The problem of dictionary learning for sparse representations –that also goes far beyond the concatenation of a few off-the-shelf bases– has therefore become a key issue for further progress in this area.

In this chapter[1], we first introduce the basic principles of dictionary learning techniques by formulating the underlying optimization problem and then explain various algorithms used for solving this problem. We then describe how the dictionary learning problem can be adapted and applied to image prediction leading to a method for "online learning of prediction dictionaries". Finally we conclude this chapter by presenting the experimental results obtained for the proposed framework for image prediction, which we refer to as **on-the-fly dictionaries (OFD)**.

4.1 Dictionary Learning

In the last decades, the problem of learning redundant and overcomplete dictionaries has gained great importance in sparse signal representations for many basic image

[1]Some part of this chapter is related to our publication in [101] which is nominated for the Best Student Paper Award in 2011 IEEE Int. Conf. on Image Process. (ICIP'11), Sept. 2011.

processing tasks such as image denoising [54, 55], texture modeling [56], image restoration [57, 58], image compression [59, 60], inpainting and zooming [61], and more [62, 63]. The underlying main idea suggests that natural (image) signals can be better approximated sparsely, and therefore compacted more efficiently, as a weighted linear combination of a set of pre-learned dictionary basis functions, so-called the *atoms*, when compared to off-the-self overcomplete (e.g., DCT, DFT, wavelets) bases or dictionaries. The sparsity constraint, that is associated with the learning problem, generally leads to a solution which can fit any practical application (using sparse decomposition algorithms) in terms of approximation accuracy, compaction rate, and computational complexity.

4.1.1 Problem formulation

The dictionary learning problem aims at obtaining an explicit matrix $\mathbf{A} = [\mathbf{a}_1 \,|\, ... \,|\mathbf{a}_\mathrm{M}] \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ (with $\mathrm{N} \leqslant \mathrm{M}$) which is optimally representative of a given set of training samples, under strict sparsity constraints. The matrix $\mathbf{A}$ is known as the *dictionary* and its columns $\mathbf{a}_m, m = 1...\mathrm{M}$, are the *atoms*. Formally, given a set of J training samples stored in a matrix $\mathbf{T} = [\mathbf{t}_1 \,|\, ... \,|\mathbf{t}_\mathrm{J}] \in \mathbb{R}^{\mathrm{N}\times\mathrm{J}}$, in which the columns $\mathbf{t}_j$, $j = 1...\mathrm{J}$, represent the N-dimensional individual training samples, a dictionary learning algorithm searches for an optimum dictionary $\mathbf{A}$ by solving the following constrained minimization

$$\underset{\mathbf{A},\mathbf{Y}}{\arg\min} \|\mathbf{T} - \mathbf{A}\mathbf{Y}\|_F^2 \quad \text{subject to} \quad \|\mathbf{y}_j\|_0 \leqslant K \;\forall j \quad \text{and} \quad \|\mathbf{a}_m\|_2 = 1 \;\forall m \tag{4.1}$$

where $\mathbf{Y} = [\mathbf{y}_1 \,|\, ... \,|\mathbf{y}_\mathrm{J}] \in \mathbb{R}^{\mathrm{M}\times\mathrm{J}}$ is the sparse representation matrix and its columns $\mathbf{y}_j$ denote the sparse representation vectors of the corresponding training samples $\mathbf{t}_j \;\forall j$. Here K represents the targeted sparsity, i.e., the ℓ_0–norm of each sparse vector $\mathbf{y}_j$, and $\|\,.\,\|_F$ denotes the Frobenius norm.[1]

The resulting optimization problem in (4.1) is combinatorial and higly non-convex, and thus one can expect a local minimum at best [79]. The formulation in (4.1) can be rewritten as a joint optimization with respect to the dictionary $\mathbf{A}$ and the sparse

[1]The Frobenius norm (or the Hilbert–Schmidt norm) $\|\mathbf{A}\|_F$ of a matrix $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ is defined as the square root of the sum of the squares of all elements in the matrix: $\|\mathbf{A}\|_F = \sqrt{\sum\limits_{n=1}^{\mathrm{N}} \sum\limits_{m=1}^{\mathrm{M}} |A_{nm}|^2}$.

representation vectors $\mathbf{y}_j \ \forall j$, which is not jointly convex but convex with respect to one of its variables when the other one is fixed, as follows

$$\min_{\mathbf{A}} \sum_{j \in \mathrm{J}} \left\{ \min_{\mathbf{y}_j} \left[\|\mathbf{t}_j - \mathbf{A}\mathbf{y}_j\|_2^2 + \chi_j \|\mathbf{y}_j\|_0 \right] \right\} \tag{4.2}$$

where the sparsity constraint is included in the formulation as a penalty term. Note here that the constraint on the atoms $\mathbf{a}_m$ of the dictionary $\mathbf{A}$ is implicitly assumed to be valid to obtain unit norm atoms, i.e., $\|\mathbf{a}_m\|_2 = 1 \ \forall m$.

The problems formulated in (4.1) and (4.2) are equivalent if the values of the constants $\chi_j \ \forall j$ are selected accordingly [54]. Actually, transforming (4.1) into (4.2) allows us to factorize the whole optimization into two *approximate* convex optimization steps as: a) *sparse coding*: optimizing $\mathbf{y}_j \ \forall j$ by fixing $\mathbf{A}$; b) *dictionary update*: optimizing $\mathbf{A}$ by fixing $\mathbf{y}_j \ \forall j$. A solution can be reached by iteratively solving these two optimization steps.

4.1.1.1 Sparse coding step

Given a fixed dictionary matrix $\mathbf{A}$, the first problem is to obtain the sparse representation matrix $\mathbf{Y}$ of the training samples in $\mathbf{T}$. This optimization is known as *the sparse coding problem*, and it corresponds to the inner optimization in (4.2) which can be reformulated as

$$\mathbf{y}_j^{opt} = \arg\min_{\mathbf{y}_j} \left[\|\mathbf{t}_j - \mathbf{A}\mathbf{y}_j\|_2^2 + \chi \|\mathbf{y}_j\|_0 \right] \tag{4.3}$$

where $\mathbf{y}_j^{opt}$ represents the optimum sparse representation vector $\mathbf{y}_j$ of the training sample $\mathbf{t}_j$ for the given dictionary $\mathbf{A}$, and χ is the sparsity regularization parameter.

The sparse coding problem is usually solved by greedy pursuit algorithms such as MP [66], OMP [67], or BP [65] (with ℓ_1–norm regularization), as presented in Chapter 3.

4.1.1.2 Dictionary update step

Given the sparse representation vectors $\mathbf{y}_j$ calculated in the sparse coding step, the problem is then to optimize/update the dictionary matrix $\mathbf{A}$ by minimizing the representation error of the training samples. This optimization is known as *the dictionary update problem*, and it can be formulated as

$$\mathbf{A}^{opt} = \arg\min_{\mathbf{A}} \sum_{j \in \mathrm{J}} \|\mathbf{t}_j - \mathbf{A}\mathbf{y}_j\|_2^2 \tag{4.4}$$

where $\mathbf{A}^{opt}$ represents the optimum dictionary $\mathbf{A}$ minimizing the representation error over all training samples with respect to the given sparse vectors stored in $\mathbf{Y}$.

The dictionary update problem can be solved by various update techniques which have been developed by various researchers, e.g., [86, 87, 88, 89, 102, 103]. One way of obtaining a solution to (4.4) is to update the whole dictionary matrix at once, by minimizing the representation error over all training samples, using methods such as [86, 88]. On the other hand, another way is to update one dictionary atom at a time by minimizing the representation error for each atom individually while keeping all the other atoms fixed, e.g., as proposed in [89, 102].

4.1.2 Dictionary learning algorithms

One of the earliest dictionary learning schemes is proposed by Olshausen and Field [104]. In their algorithm, a log-likelihood estimate of the optimal dictionary has been carried out by performing a gradient descent method for both sparse coding and dictionary update steps, while assuming a log-prior on the sparse representation coefficients such as a Cauchy distribution or an exponential distribution.

Various dictionary learning schemes have then been proposed in the literature for the sparse signal representations problem. The most recent dictionary learning methods focus on ℓ_0 and ℓ_1 sparsity measures, which potentially leads to simple formulations, hence to efficient techniques.

Non-parametric dictionary learning methods, such as Method of Optimal Directions (MOD) [86] and K–SVD [89], have been developed resulting in non-structural learned dictionaries. These methods are indeed very effective in practice, however, the computational complexity required for learning a non-structured dictionary makes their usage restricted only to low-dimensional problems.

There are also parametric learning structures for such as *translation invariant dictionaries* [105, 106, 107, 108], *multiscale dictionaries* [58, 109], and *sparse dictionaries* [110]. These dictionaries are usually learned by imposing various desired properties on the dictionary leading typically to a more compact representation, hence to a more efficient implementation, when compared to regular (non-parametric) dictionaries [79].

For example, an interesting "shift and scale invariant" approach, which is called image-signature-dictionary (ISD), has been proposed in [107]. The idea of ISD is that a large image can be constructed by linearly combining several carefully chosen patches from a small representative image, i.e., *epitome* [111, 112]. Here the ISD image has been assumed to be a periodic extension of conventional epitomes, and has been shown that it is a promising alternative to the classical dictionary structure.

Unions of orthonormal bases [87, 88] can also be seen as parametric learning methods resulting in structured dictionaries in tight frames. Although their efficiency in learning dictionaries with a reduced complexity, these methods suffer from not being flexible to more complex structures in natural images.

Moreover, online learning algorithms [102, 103], task-driven learning approaches [113], tree-structured hierarchical [114, 115, 116] methods, and iteration-tuned schemes [117, 118] have been introduced into the literature aiming at improving dictionary learning methods and their applications to various image processing tasks.

4.1.2.1 Method of optimal directions (MOD)

The MOD method is based on the observation that the sparse coding problem could be regarded as a generalization of vector quantization [13]. It actually builds upon the K-means process with a sparse coding step using a pursuit algorithm (i.e., OMP) followed by an update step of the dictionary. For the dictionary update step, the quadratic optimization in (4.4) can be rewritten in a compact form as

$$\arg\min_{\mathbf{A}} \|\mathbf{T} - \mathbf{A}\mathbf{Y}\|_F^2 \tag{4.5}$$

which has an analytic solution obtained by forcing its derivative to zero

$$\mathbf{A} = \mathbf{T}\mathbf{Y}^{\mathrm{T}}\left(\mathbf{Y}\mathbf{Y}^{\mathrm{T}}\right)^{-1}, \tag{4.6}$$

thus, solving the dictionary update problem in one step.

MOD generally converges with a few iterations alternating between sparse coding and dictionary update, and indeed is a very effective solution. However, it suffers from the high complexity of the matrix inversion, in particular for large dictionaries.

4.1.2.2 Sparse orthonormal transforms (SOT)

Learning a union of orthobases dictionary with ℓ_1–norm constraints has been initially proposed by Lesage *et al.* [87]. SOT [88] can be regarded as an extension of this method, incorporating ℓ_0–norm constraints, as an application for image compression.

The SOT method considers a dictionary $\mathbf{A}$ of the form $\mathbf{A} = [\mathbf{A}_1 \,|\, ... \,\, |\mathbf{A}_{\mathrm{I}}]$ as a union of learned orthonormal bases $\mathbf{A}_i \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ such that $\mathbf{A}_i^{\mathrm{T}}\mathbf{A}_i = \mathbf{I}_{\mathrm{N}}$, $i = 1...\mathrm{I}$. The method initially classifies the training samples using image gradients and obtains the initial orthonormal transforms via Karhunen–Loeve Transform (KLT) which are optimal for each class. It then alternates between sparse coding and transform optimization (dictionary update) followed by a reclassification step of the training samples, until a joint convergence is reached. The sparse coding process has a constraint for each training sample $\mathbf{t}_j$ on using only a single transform $\mathbf{A}_i$ in $\mathbf{A}$. Let the training sample $\mathbf{t}_j$ belong to class–i, the optimization problem in (4.3) can be constrained and rewritten as

$$\underset{\mathbf{y}_j}{\arg\min} \left[\|\mathbf{t}_j - \mathbf{A}_i\mathbf{y}_j\|_2^2 + \chi\|\mathbf{y}_j\|_0 \right] \text{ subject to } \mathbf{A}_i^{\mathrm{T}}\mathbf{A}_i = \mathbf{I}_{\mathrm{N}}, \tag{4.7}$$

and the solution of (4.7) can be computed directly by hard-thresholding the components of the orthonormal projection of $\mathbf{t}_j$ onto the column space spanned by $\mathbf{A}_i$ such that

$$\mathbf{y}_j = \Upsilon_{\sqrt{\chi}} \left[\mathbf{A}_i^{\mathrm{T}}\mathbf{t}_j\right] \tag{4.8}$$

where $\Upsilon_\sigma\,[\,.\,]$ represents the hard-thresholding operator[1] with a threshold equal to σ.

For the dictionary update step, the minimization in (4.4) can be constrained to the training samples belonging to the class–i (indexed in the set J_i) as

$$\underset{\mathbf{A}_i}{\arg\min} \sum_{j\in\mathrm{J}_i} \|\mathbf{t}_j - \mathbf{A}_i\mathbf{y}_j\|_2^2 \text{ subject to } \mathbf{A}_i^{\mathrm{T}}\mathbf{A}_i = \mathbf{I}_{\mathrm{N}} \tag{4.9}$$

which has an equivalent maximization problem such that

$$\underset{\mathbf{A}_i}{\arg\max}\, Tr\,[\mathbf{H}_i\mathbf{A}_i] \text{ subject to } \mathbf{A}_i^{\mathrm{T}}\mathbf{A}_i = \mathbf{I}_{\mathrm{N}} \tag{4.10}$$

[1]Given a vector $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$ and a threshold σ, the elements b_n of the vector $\mathbf{b}$ can be hard-thresholded as $\Upsilon_\sigma[\mathbf{b}] = \begin{cases} b_n = 0 & \text{if } |b_n| < \sigma \\ b_n = b_n & \text{if } |b_n| \geqslant \sigma \end{cases}$ for $n = 1...\mathrm{N}$.

where $\mathbf{H}_i = \sum_{j \in \mathrm{J}_i} \mathbf{y}_j \mathbf{t}_j^{\mathrm{T}}$ and $Tr\,[\,.\,]$ denotes the trace operator, i.e., the sum of the elements on the main diagonal of a matrix. By letting the singular value decomposition (SVD) of $\mathbf{H}_i$ be denoted as $\mathbf{U}_i \Lambda_i \mathbf{V}_i^{\mathrm{T}}$, the optimum solution to (4.10) is given in [88] as

$$\mathbf{A}_i = \mathbf{V}_i \mathbf{U}_i^{\mathrm{T}}. \tag{4.11}$$

The training samples $\mathbf{t}_j \ \forall j$ are reclassified using the cost function as follows

$$\mathit{class}\,(\mathbf{t}_j) = \arg\min_i \left[\min_{\mathbf{y}_j^i} \left[\left\| \mathbf{t}_j - \mathbf{A}_i \mathbf{y}_j^i \right\|_2^2 + \chi \left\| \mathbf{y}_j^i \right\|_0 \right] \right]. \tag{4.12}$$

4.1.2.3 K–SVD dictionary

The K–SVD algorithm also builds upon a generalization of the K-means clustering process (similar to MOD). It is an iterative method which alternates between sparse coding of training samples and dictionary update for an efficient representation of the data. Formally, after a sparse coding stage which can be solved using any pursuit algorithm, the dictionary update step proceeds by modifying (updating) one atom at a time while keeping all the other atoms fixed. The solution leads to a so-called *rank–1 approximation* (per atom) which can be solved directly with SVD (i.e., updating the corresponding atom by the left singular vector associated to the highest singular value of the corresponding representation error matrix), or some other numerical methods (cf. [110]).

In theory, the overall representation error matrix $\mathbf{E}_m$, $m = 1...\mathrm{M}$, is computed by masking one atom $\mathbf{a}_m$ at a time in the dictionary $\mathbf{A}$ and the corresponding weights stored in the rows $\mathbf{y}_m^r$ of the representation matrix $\mathbf{Y}$, i.e.,

$$\mathbf{E}_m = \mathbf{T} - \sum_{\substack{m'=1 \\ m' \neq m}}^{\mathrm{M}} \mathbf{a}_{m'} \mathbf{y}_{m'}^r. \tag{4.13}$$

The matrix $\mathbf{E}_m$ represents the error for the training set when the m^{th} atom is removed from the dictionary. In order to keep the same (or even smaller) support for all representations, $\mathbf{E}_m$ is restricted to obtain a sub-matrix $\mathbf{E}_m^R$ by choosing the columns which correspond to the group of training samples those use the atom $\mathbf{a}_m$. SVD decomposes $\mathbf{E}_m^R$ into $\mathbf{U}\Lambda\mathbf{V}^{\mathrm{T}}$. The rank–1 solution is then defined for the updated atom $\mathbf{a}_m$ as the first column of $\mathbf{U}$, i.e., $\mathbf{a}_m = \mathbf{u}_1$, and the non-zero coefficients in $\mathbf{y}_m^r$ as the first column of $\mathbf{V}$ multiplied by $\Lambda_{1,1}$.

K–SVD is an effective method, and it is less computational demanding than MOD. However, it also suffers from the high non-convexity of the optimization problem which might lead to a local minima solution or even saddle points [79]. Moreover, learning an example-based non-structured dictionary is very costly, and thus these methods, e.g., MOD and K–SVD, are suitable for signals of relatively small in size.

4.1.2.4 Sparse dictionaries

A novel, efficient, and flexible dictionary structure has been proposed in [110] for sparse and redundant signal representations. This method assumes that each atom $\mathbf{a}_m$ of the dictionary $\mathbf{A}$ to be learned can be characterized with a sparsity model over a fixed base (typically an analytic) dictionary $\mathbf{\Theta}$. Hence the dictionary can be expressed of the form as $\mathbf{A} = \mathbf{\Theta D}$ where $\mathbf{D}$ is the atom representation matrix, assumed to be a sparse matrix having a fixed number of non-zero coefficients per column.

While the choice of the base dictionary $\mathbf{\Theta}$ has an important effect on the success of the method, the above sparse dictionary model can be more effectively learned from the training set of examples in comparison to implicit (analytic) dictionaries, hence it introduces adaptability by modification of the matrix $\mathbf{D}$. Moreover, when compared to explicit dictionaries which are adaptable but not efficient and costly to apply, this method provides compactness and efficiency.

4.1.2.5 Online dictionaries

An appealing approach for online dictionary learning based on stochastic approximations has been recently proposed in Mairal *et al.* [102]. This learning algorithm alternates between a sparse coding step with ℓ_1–norm regularization and a progressive dictionary update step based on block-coordinate descent (BCD) with warm restarts [119]. The sparse coding problem as formulated in (4.14) is solved by a Cholesky-based implementation of the LARS-Lasso algorithm, *a homotopy method* [120, 121], by considering i.i.d. samples of a random variable $\mathbf{t} \sim p(\mathbf{t})$, i.e., at the j^{th} iteration, an observed sample $\mathbf{t}_j$ is coded by solving the following equation

$$\underset{\mathbf{y}_j}{\arg\min}\left[\frac{1}{2}\|\mathbf{t}_j-\mathbf{A}\mathbf{y}_j\|_2^2+\chi\|\mathbf{y}_j\|_1\right]. \tag{4.14}$$

Given the sparse representation vector $\mathbf{y}_j$, each atom $\mathbf{a}_m$ of the dictionary $\mathbf{A}$ is then updated while keeping all the other ones fixed under the constraint $\mathbf{a}_m^{\mathrm{T}}\mathbf{a}_m \leqslant 1$. This constraint prevents $\mathbf{A}$ having arbitrarily large values, and here it amounts to an orthogonal projection onto the ℓ_2–ball. The "past information" (i.e., the information obtained from the observed set of samples up to $\mathbf{t}_{j-1}$) has also been taken into account for the currently observed sample $\mathbf{t}_j$ in two matrices as

$$\begin{aligned}\mathbf{\Gamma}^j &= \sum_j \mathbf{y}_j\mathbf{y}_j^{\mathrm{T}} = \mathbf{\Gamma}^{j-1}+\mathbf{y}_j\mathbf{y}_j^{\mathrm{T}} \quad \text{where} \quad \mathbf{\Gamma}^0=\mathbf{0}\\ \mathbf{\Xi}^j &= \sum_j \mathbf{t}_j\mathbf{y}_j^{\mathrm{T}} = \mathbf{\Xi}^{j-1}+\mathbf{t}_j\mathbf{y}_j^{\mathrm{T}} \quad \text{where} \quad \mathbf{\Xi}^0=\mathbf{0}\end{aligned} \tag{4.15}$$

in order to solve the following dictionary update problem per sample $\mathbf{t}_j$,

$$\underset{\mathbf{A}}{\arg\min}\,\frac{1}{j}\sum_j\frac{1}{2}\|\mathbf{t}_j-\mathbf{A}\mathbf{y}_j\|_2^2, \tag{4.16}$$

which can be rewritten in terms of $\mathbf{\Gamma}^j$ and $\mathbf{\Xi}^j$ as

$$\underset{\mathbf{A}}{\arg\min}\,\frac{1}{j}\left[\frac{1}{2}Tr\left[\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{\Gamma}^j\right]-Tr\left[\mathbf{A}^{\mathrm{T}}\mathbf{\Xi}^j\right]\right] \tag{4.17}$$

where $\mathbf{\Gamma}^j=[\boldsymbol{\gamma}_1\,|\ldots\,|\boldsymbol{\gamma}_{\mathrm{M}}]\in\mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ and $\mathbf{\Xi}^j=[\boldsymbol{\xi}_1\,|\ldots\,|\boldsymbol{\xi}_{\mathrm{M}}]\in\mathbb{R}^{\mathrm{N}\times\mathrm{M}}$.

It is easy to show that (4.17) can be solved with respect to the m^{th} column $\mathbf{a}_m$ $\forall m$ under the constraint $\mathbf{a}_m^{\mathrm{T}}\mathbf{a}_m \leqslant 1$ such that

$$\mathbf{a}_m=\frac{1}{\max\left(\|\boldsymbol{\vartheta}_m\|_2,1\right)}\boldsymbol{\vartheta}_m \quad \text{where} \quad \boldsymbol{\vartheta}_m=\frac{1}{\Gamma^j_{mm}}\left(\boldsymbol{\xi}_m-\mathbf{A}\boldsymbol{\gamma}_m\right)+\mathbf{a}_m. \tag{4.18}$$

All the atoms are iteratively updated for each observed sample with the update rule given in (4.18) until it converges, where convergence to a global optimum is guaranteed [119]. After a few iterations of the algorithm, the dictionary obtained in the $(j-1)^{th}$ iteration can effectively be utilized as a *warm restart* for the current dictionary at the j^{th} iteration, and in this case a single iteration of (4.18) has been found to be sufficient to achieve convergence of the dictionary update step. The advantages of this online scheme include the capability to handle millions of training samples, the adaptivity to newly observed signals, and the flexibility to rescale the past information so that newly processed samples have more weight on the updated dictionary.

Another recent approach for online "recursive least squares" dictionary learning has been proposed in [103]. With a forgetting factor, it is similar to the recursive least squares algorithm for adaptive filtering. Although there is still room for research to be done on this method, the performance analysis shows good representation and convergence properties with easy implementations in comparison to K–SVD and iterative least squares dictionary learning [108].

4.2 Dictionary Learning for Image Prediction

4.2.1 Motivation

The above described dictionary learning algorithms are not very well suited to the image prediction problem. These methods are mainly adapted to the learning of basis functions (atoms) to be used for approximating the input data vectors, but not to the problem of predicting the unknown pixels from noisy observed samples in a causal neighborhood (approximation support, or template). Moreover, the complexity, which results from the number and the dimension of training samples, of these methods often limits their applicability to low-dimensional data analysis problems, and makes them fragile to outliers, i.e., to training samples which do not have a sparse enough representation, like in the case where the training samples are perturbed by noise as quantization noise in compression applications, as the one considered here.

In order to be usable online while performing the encoding of the prediction residue, the learning process must be relatively fast, hence done with a limited (small) number of training samples. In addition, the learned dictionary must be efficient leading to a good approximation of not only the known samples in the approximation support but also of the unknown samples of the block to be predicted. With the observed limitations of SP in the previous chapter, here we present a novel method which, because of its simplicity and the limited number of training samples it requires, can be used for *locally adaptive online learning* of (**on-the-fly**) dictionaries for spatial texture prediction.

Let $\mathbf{A}$ denote the dictionary matrix which is assumed to be composed of two submatrices (subdictionaries) as $\mathbf{A}_c$ and $\mathbf{A}_t$. The rationale behind is to develop a simple but efficient online dictionary learning method which is adapted to the image prediction

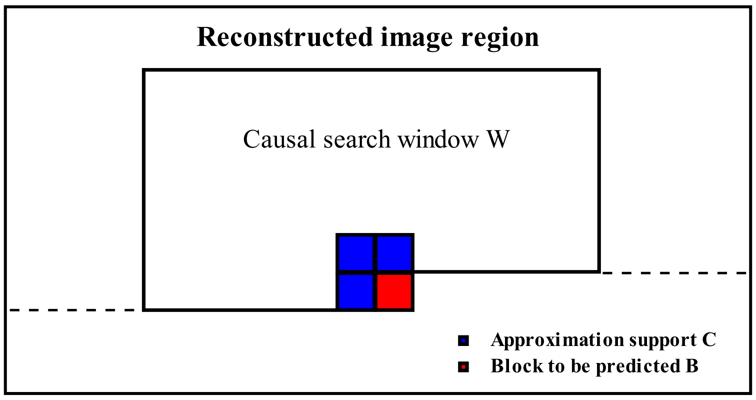


Figure 4.1: Online prediction dictionary learning - C is the approximation support of the current $n \times n$ block to be predicted B. W is the search window from which texture patches are extracted to collect the training samples for learning the dictionary $\mathbf{A}$ to be used for the prediction of B.

problem, that is which will learn both subdictionaries so that sparse representation vector obtained by approximating the known samples in the approximation support using the first subdictionary $\mathbf{A}_c$ will lead to a good approximation of the block to be predicted when used together with the second subdictionary $\mathbf{A}_t$. In order to reduce the complexity of the proposed approach, an orthonormality constraint is imposed on the dictionary $\mathbf{A}_c$ which is used for approximating the known samples in the approximation support. This constraint allows us to use simple projections followed by a standard hard-thresholding to obtain the sparse representation vectors, instead of using costly iterative pursuit algorithms such as MP and OMP.

The training samples are collected by extracting all possible previously endoded and decoded texture patches (blocks of pixels) within a search window W located in a causal neighborhood of the block to be predicted (see Fig. 4.1). That is done by taking in the columns of $\mathbf{T}$ all possible texture patches (in a vectorized form) with a mask shifted by one pixel horizontally and vertically in the search window W. Notice the difference that SP uses these texture patches directly to construct the dictionary $\mathbf{A}$. The size of the search window controls the number of training samples, hence the complexity of the approach. The use of a causal search window guarantees that the decoder can collect exactly the same training samples, thus it can run the same algorithm for learning the dictionary $\mathbf{A}$.

4.2.2 Learning prediction dictionaries: Problem statement

Let S denote a region in the image containing both known pixel values (in the approximation support C) and unknown block to be predicted B (which is initially assumed to be zero of size $n \times n$ pixels) as shown in Fig. 4.1. All the values of the region S are stacked in a column vector $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$. Then, the rows of $\mathbf{b}$ are masked corresponding to the spatial locations of the unknown pixel values in order to obtain $\mathbf{b}_c \in \mathbb{R}^{\mathrm{N}_1}$ representing the subvector for the approximation support C, and let $\mathbf{b}_t \in \mathbb{R}^{\mathrm{N}_2}$ denote the original pixel values of the block to be predicted B. $\mathrm{N} = \mathrm{N}_1 + \mathrm{N}_2$ and $\mathrm{N}_2 = n^2$.

Let $\mathbf{T} = [\mathbf{t}_1 \,|\, ... \,|\mathbf{t}_{\mathrm{J}}] \in \mathbb{R}^{\mathrm{N} \times \mathrm{J}}$ denote a matrix in which J training image patches (having the same geometric shape as S) extracted from the causal search window W are stacked as columns, and suppose that $\mathbf{T}$ is modified by masking its rows corresponding to the spatial locations of the (un)known pixel values in order to obtain the submatrices $\mathbf{T}_c = [\mathbf{t}_{c_1} \,|\, ... \,|\mathbf{t}_{c_{\mathrm{J}}}] \in \mathbb{R}^{\mathrm{N}_1 \times \mathrm{J}}$ and $\mathbf{T}_t = [\mathbf{t}_{t_1} \,|\, ... \,|\mathbf{t}_{t_{\mathrm{J}}}] \in \mathbb{R}^{\mathrm{N}_2 \times \mathrm{J}}$ spatially corresponding to $\mathbf{b}_c$ and $\mathbf{b}_t$ respectively. Note here that the structure of the elements in the matrix $\mathbf{T}$ depends on the spatial location of the area S and the size of the search window W. This feature introduces a *local adaptability* by capturing the important spatial contextual information in the image as training samples.

Let $\mathbf{A} \in \mathbb{R}^{\mathrm{N} \times \mathrm{M}}$ be the *dictionary* matrix to be learned of the form $\mathbf{A} = [\mathbf{a}_1 \,|\, ... \,|\mathbf{a}_{\mathrm{M}}]$, and its columns $\mathbf{a}_m \; \forall m$ are the *atoms*. This matrix is masked in the same manner as $\mathbf{T}$ to obtain $\mathbf{A}_c \in \mathbb{R}^{\mathrm{N}_1 \times \mathrm{M}}$ and $\mathbf{A}_t \in \mathbb{R}^{\mathrm{N}_2 \times \mathrm{M}}$ subdictionaries spatially corresponding to $\mathbf{b}_c$ and $\mathbf{b}_t$ respectively. The columns (atoms) of the submatrix $\mathbf{A}_c$ are used for representing the approximation support $\mathbf{b}_c$ and the atoms of $\mathbf{A}_t$ are used for approximating the block to be predicted $\mathbf{b}_t$. For simplicity, let us consider an orthonormality constraint on the dictionary $\mathbf{A}_c$ as $\mathbf{A}_c^{\mathrm{T}} \mathbf{A}_c = \mathbf{I}_{\mathrm{N}_1}$ which will allow us to use simple orthonormal projections followed by a standard hard-thresholding to calculate the representation vectors of the training samples. This orthonormality constraint implies that $\mathrm{N}_1 = \mathrm{M}$, i.e., $\mathbf{A}_c \in \mathbb{R}^{\mathrm{M} \times \mathrm{M}}$ and $\mathbf{A}_t \in \mathbb{R}^{\mathrm{N}_2 \times \mathrm{M}}$. Starting with this observation, the number of unknown pixels in S is required to be equal to or less than the number of known samples, i.e., $\mathrm{N}_2 \leqslant \mathrm{M} = \mathrm{N}_1$.

Let $\mathbf{Y} = [\mathbf{y}_1 \,|\, ... \,|\mathbf{y}_{\mathrm{J}}] \in \mathbb{R}^{\mathrm{M} \times \mathrm{J}}$ be the sparse representation matrix of the training image patches, and $\mathbf{x} \in \mathbb{R}^{\mathrm{M}}$ denote the sparse representation vector of the approximation support $\mathbf{b}_c$. The sparsity constraint here is enforced by the $\mathrm{Thr}_K\,[\,.\,]$ operator which keeps the K largest representation coefficients (in magnitude) and forces all the others to be zero, i.e., $\|\mathbf{x}\|_0 = \|\mathbf{y}_j\|_0 = K \;\; \forall j$.

As an important remark, the dictionary $\mathbf{A}$ here is considered as a combination of two subdictionaries, $\mathbf{A}_c$ and $\mathbf{A}_t$. The columns (atoms) of $\mathbf{A}_c$ are used for approximating the support region $\mathbf{b}_c$ which is supposed to be known to both encoder and decoder. In the image prediction context, this process refers to **the sparse coding problem**. On the other hand, the subdictionary $\mathbf{A}_t$ is the most crucial one since it is used for predicting the unknown pixel values. The process of calculating the optimum dictionary $\mathbf{A}_t$ refers to **the dictionary update problem**. Therefore, in this work for image prediction, the conventional dictionary learning scheme is divided into its two steps which are operating on two different datasets.

4.2.2.1 Sparse coding step for prediction

After extracting a number of training image patches from the search window W where all the pixel values are available, i.e., in the reconstructed image region (see Fig. 4.1), the subdictionary $\mathbf{A}_c$ is initialized with the KLT of the training set $\mathbf{T}_c$ which contains the pixel values located at the same spatial position as the approximation support $\mathbf{b}_c$. KLT has interesting properties: a) It is orthonormal, and b) it is complete. So, the set of sparse representation vectors $\mathbf{y}_j \; \forall j$ in $\mathbf{Y}$ of the training set $\mathbf{T}_c$ can be calculated directly with the orthonormal projection of $\mathbf{t}_{c_j}$ onto the column space of $\mathbf{A}_c$ followed by standard hard-thresholding according to a given sparsity constraint. The sparse coding problem in (4.3) can be reformulated as

$$\underset{\mathbf{y}_j}{\arg\min} \left[\left\| \mathbf{t}_{c_j} - \mathbf{A}_c \mathbf{y}_j \right\|_2^2 + \chi \|\mathbf{y}_j\|_0 \right] \quad \text{subject to} \quad \mathbf{A}_c^{\mathrm{T}} \mathbf{A}_c = \mathbf{I}_{\mathrm{M}}, \tag{4.19}$$

and an "*approximate*" K–sparse solution can be computed by

$$\mathbf{y}_j = \mathrm{Thr}_K \left[\mathbf{A}_c^{\mathrm{T}} \mathbf{t}_{c_j} \right] \quad \text{where} \quad \mathbf{A}_c = \mathrm{KLT}(\mathbf{T}_c). \tag{4.20}$$

4.2.2.2 Dictionary update step for prediction

Among the two subdictionaries, the most crucial one is $\mathbf{A}_t$ since it is the one used for predicting the unknown pixels in B. $\mathbf{A}_t$ needs to be derived from the set of sparse vectors stored in $\mathbf{Y}$ so that the representation vector computed by approximating the support region $\mathbf{b}_c$ with the dictionary $\mathbf{A}_c$ will lead to an optimum approximation (i.e., according to the collected training samples) of the block to be predicted $\mathbf{b}_t$, when the same representation vector is used with the dictionary $\mathbf{A}_t$. Note here the dictionary $\mathbf{A}_t$ has no constraint on it; it is not forced to be orthonormal, but it is assumed to be *overcomplete*.

Given the set of representation vectors $\mathbf{y}_j$ in $\mathbf{Y}$ calculated in the sparse coding step, the dictionary update problem here aims at optimizing the dictionary $\mathbf{A}_t$ so that the representation error of the corresponding training samples $\mathbf{t}_{t_j}$ in $\mathbf{T}_t$ is minimized as much as possible, i.e.,

$$\arg\min_{\mathbf{A}_t} \sum_{j\in \mathrm{J}} \left\| \mathbf{t}_{t_j} - \mathbf{A}_t \mathbf{y}_j \right\|_2^2, \tag{4.21}$$

or equivalently in the compact form as

$$\arg\min_{\mathbf{A}_t} \| \mathbf{T}_t - \mathbf{A}_t \mathbf{Y} \|_F^2. \tag{4.22}$$

Below we give two least squares based solutions, which are derived for the optimization expressed in (4.21) and (4.22), for obtaining and/or updating the subdictionary $\mathbf{A}_t$. The first method is mainly based on the MOD algorithm, and the second one relies on the BCD method.

MOD Solution

The optimum subdictionary $\mathbf{A}_t$ can be obtained analytically by forcing the derivative of the quadratic function in (4.22) to zero (i.e., similar to MOD) such that

$$\mathbf{A}_t = \mathbf{T}_t \mathbf{Y}^{\mathrm{T}} \left(\mathbf{Y} \mathbf{Y}^{\mathrm{T}} \right)^{-1}. \tag{4.23}$$

Note here that the solution given above can be iterated only once since the sparse representation matrix $\mathbf{Y}$, which has been obtained using the training set $\mathbf{T}_c$, is fixed. Thus, one can expect a suboptimal solution at best. However, with the limited (small) number of training samples as considered here, a reasonably good least squares solution can be achieved even with one iteration since the MOD algorithm generally converges with a few iterations for large numbers of training samples.

BCD Solution

The optimum subdictionary $\mathbf{A}_t$ can also be obtained by the exact block-coordinate descent, i.e., updating one atom $\mathbf{a}_{t_m}$ at a time while keeping all the other atoms and all the sparse vectors $\mathbf{y}_j$ fixed, since the matrix $\mathbf{Y}$ is fixed. Notice here the difference from K–SVD which updates also the sparse representation coefficients while updating each individual atom. A straightforward solution can directly be adapted from (4.18) under the constraint $\mathbf{a}_{t_m}^{\mathrm{T}}\mathbf{a}_{t_m} \leqslant 1\ \forall m$ such that

$$\mathbf{a}_{t_m} = \frac{1}{\max\left(\|\boldsymbol{\vartheta}_{t_m}\|_2, 1\right)}\boldsymbol{\vartheta}_{t_m} \quad \text{and} \quad \boldsymbol{\vartheta}_{t_m} = \frac{1}{\Gamma_{mm}}\left(\boldsymbol{\xi}_{t_m} - \mathbf{A}_t\boldsymbol{\gamma}_m\right) + \mathbf{a}_{t_m} \tag{4.24}$$

where $\boldsymbol{\Gamma} = \mathbf{Y}\mathbf{Y}^{\mathrm{T}} = [\boldsymbol{\gamma}_1\,|\ldots\,|\boldsymbol{\gamma}_{\mathrm{M}}] \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$ and $\boldsymbol{\Xi}_t = \mathbf{T}_t\mathbf{Y}^{\mathrm{T}} = \left[\boldsymbol{\xi}_{t_1}|\ldots|\boldsymbol{\xi}_{t_{\mathrm{M}}}\right] \in \mathbb{R}^{\mathrm{N}_2\times\mathrm{M}}$. All the atoms are iteratively updated until convergence is reached.

4.2.3 Prediction of unknown pixels

After learning/obtaining the subdictionaries $\mathbf{A}_c$ and $\mathbf{A}_t$ from the training sets of image patches $\mathbf{T}_c$ and $\mathbf{T}_t$ respectively, the prediction $\hat{\mathbf{b}}_t$ of the unknown pixel values simply follows by calculating the linear coefficients vector $\mathbf{x}$ by thresholding the projection coefficients of the support region $\mathbf{b}_c$ onto the column space of $\mathbf{A}_c$, i.e.,

$$\mathbf{x} = \mathrm{Thr}_K\left[\mathbf{A}_c^{\mathrm{T}}\mathbf{b}_c\right] \quad \text{where} \quad \mathbf{A}_c = \mathrm{KLT}(\mathbf{T}_c), \tag{4.25}$$

and then using the same representation coefficients with the optimized subdictionary $\mathbf{A}_t$ such that $\hat{\mathbf{b}}_t = \mathbf{A}_t\mathbf{x}$. Finally, one can write

$$\begin{bmatrix}\mathbf{b}_c\\ \mathbf{b}_t\end{bmatrix} \approx \begin{bmatrix}\mathbf{A}_c\\ \mathbf{A}_t\end{bmatrix}\mathbf{x}. \tag{4.26}$$

4.2.4 Optimum approximation support selection

In addition to seven different forms of approximation supports as defined in the previous chapter, here we introduce two new modes, Mode 8 and Mode 9 as shown in Fig. 4.2, which will be activated depending on the availability of the support region pixels.

The optimum dynamic approximation support is selected according to two criteria: 1. minimization of the prediction signal SSE (i.e., $\min \|\mathbf{b}_t - \mathbf{A}_t\mathbf{x}\|_2^2$) in order to observe the impact on the prediction quality, 2. minimization of the RD cost function $J(\mathbf{D}, \mathbf{R})$,

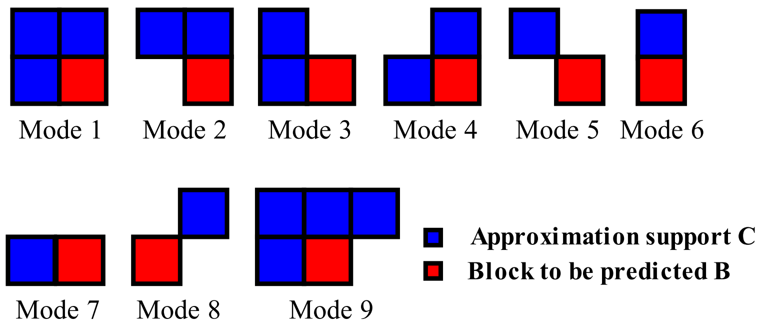


Figure 4.2: Nine possible modes for approximation support selection - The optimum support type is selected according to a given criterion.

```
**Inputs:** $\mathbf{T}_c$, $\mathbf{T}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, $K$
**Output:** Predicted values of unknowns $\hat{\mathbf{b}}_t$
  $\mathbf{A}_c = \mathrm{KLT}(\mathbf{T}_c)$
  $\mathbf{y}_j = \mathrm{Thr}_K \left[\mathbf{A}_c^{\mathrm{T}} \mathbf{t}_{c_j}\right] \; \forall j$ where $\mathbf{Y} = [\mathbf{y}_1 | \ldots |\mathbf{y}_{\mathrm{J}}]$
  $\mathbf{\Gamma} = \mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ and $\mathbf{\Xi}_t = \mathbf{T}_t \mathbf{Y}^{\mathrm{T}}$
  $\mathbf{A}_t = \mathbf{\Xi}_t \mathbf{\Gamma}^{-1}$
  $\mathbf{x} = \mathrm{Thr}_K \left[\mathbf{A}_c^{\mathrm{T}} \mathbf{b}_c\right]$
  $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}$
```

Table 4.1: Image prediction algorithm based on on-the-fly dictionaries using the MOD solution - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

when the proposed prediction scheme is used in a compression algorithm, in order to observe the impact on the encoding efficiency.

The proposed image prediction algorithms are summarized in Table 4.1 (using the MOD solution) and Table 4.2 (using the BCD solution). Note here that the dictionary submatrices $\mathbf{A}_c$ and $\mathbf{A}_t$ are optimized adaptively (per block) according to the spatial location of the block to be predicted B. Since the same learning and prediction procedure can also be done at the decoder, nothing needs to be stored (except warm restarts of dictionaries), hence this method is named as *on-the-fly dictionaries* (OFD).

A similar pixel-wise prediction method which is called "Position Dependent Linear Intra Prediction (PDLIP)" is presented in [122]. In this approach also, the pixels of the block to be predicted are estimated using the least squares technique, hence as a linear combination of spatially neighboring pixels. The weights of the linear combination are

```
**Inputs:** $\mathbf{T}_c$, $\mathbf{T}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, $K$
**Output:** Predicted values of unknowns $\hat{\mathbf{b}}_t$
**Initialization:** $\mathbf{A}_t$ with warm restart
  $\mathbf{A}_c = \mathrm{KLT}(\mathbf{T}_c)$
  $\mathbf{y}_j = \mathrm{Thr}_K\left[\mathbf{A}_c^{\mathrm{T}}\mathbf{t}_{c_j}\right]\ \forall j$ where $\mathbf{Y} = [\mathbf{y}_1 | \dots |\mathbf{y}_{\mathrm{J}}]$
  $\mathbf{\Gamma} = \mathbf{Y}\mathbf{Y}^{\mathrm{T}} = [\gamma_1 | \dots |\gamma_{\mathrm{M}}]$
  $\Xi_t = \mathbf{T}_t\mathbf{Y}^{\mathrm{T}} = \left[\boldsymbol{\xi}_{t_1} | \dots |\boldsymbol{\xi}_{t_{\mathrm{M}}}\right]$
  **repeat** until convergence on $\mathbf{A}_t = [\mathbf{a}_{t_1} | \dots |\mathbf{a}_{t_{\mathrm{M}}}]$
    **for** $m = 1$ to M
      $\boldsymbol{\vartheta}_{t_m} = \frac{1}{\Gamma_{mm}}\left(\boldsymbol{\xi}_{t_m} - \mathbf{A}_t\gamma_m\right) + \mathbf{a}_{t_m}$
      $\mathbf{a}_{t_m} = \frac{1}{\max\left(\|\boldsymbol{\vartheta}_{t_m}\|_2, 1\right)}\boldsymbol{\vartheta}_{t_m}$
    end **for**
  end **repeat**
  $\mathbf{x} = \mathrm{Thr}_K\left[\mathbf{A}_c^{\mathrm{T}}\mathbf{b}_c\right]$
  $\hat{\mathbf{b}}_t = \mathbf{A}_t\mathbf{x}$
```

Table 4.2: Image prediction algorithm based on on-the-fly dictionaries using the BCD solution - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

derived by online learning with the classified data of the previously decoded frame of the video sequence, i.e., using temporal information, for the same prediction direction. The weighting coefficients are calculated for each pixel location (position) and for each prediction direction in accordance with H.264/AVC intra modes, and they are sent as side information to the decoder. The proposed approach here differs first in the way the weighting coefficients are computed block-wise rather than pixel-wise similarly by the decoder and the encoder. In [122], for each prediction mode the weighting coefficients are position dependent, which means that each pixel in a block has its own linear weighting coefficients while pixels located at the same coordinates in all blocks use the same prediction coefficients. In the proposed method, for each prediction mode the weighting coefficients are the same for all pixels in a block while each block has its own prediction coefficients, which can be calculated online (there is no need to send as side information). In addition, the neighborhood (called the approximation support) used for the approximation or the search for the weighting coefficients is here optimized in the SSE or the RD sense.

4.2.5 Impact of sparsity constraint

One of the strengths of the proposed OFD structure is that only a small number of training samples can be sufficient for learning. The counterpart is that forcing a strict sparsity constraint may cause some atoms of $\mathbf{A}_c$ to never get used. This may prevent the corresponding atoms of $\mathbf{A}_t$ from being optimized accordingly. In order to solve this problem, one could purge the dictionary from unused atoms, or replace the unused ones by randomly choosing elements in the training set. However, this would decrease the performance of the proposed approach since the constraints on orthonormality of $\mathbf{A}_c$, overcompleteness of $\mathbf{A}_t$, and the learned relation between the atoms of $\mathbf{A}_c$ and the atoms of $\mathbf{A}_t$ will most probably be destroyed.

One can instead relax the sparsity constraint on the training samples $\mathbf{T}_c$ so that the number of atoms used in $\mathbf{A}_c$ is increased. In many image compression applications, the sparsity is usually forced in order to fulfill the bit-rate requirements of the quantized linear weighting coefficients which are transmitted to the decoder. However, the proposed learning algorithm in this chapter does not require transmitting the coefficients of the sparse representation vector. Hence, one can safely relax the sparsity constraint.

Fig. 4.3 shows the mean prediction PSNR performance curves obtained for Foreman and Barbara with varying sparsity constraints in the case where the approximation support as shown in Fig. 4.1 is not impacted by a quantization noise, i.e., in a lossless decoder/encoder structure. One can observe that the prediction quality is improved by relaxing the sparsity constraint. It improves (in terms of mean PSNR) up to a point where all the atoms in $\mathbf{A}_c$ are being used. After that point, further relaxing the sparsity constraint on the training samples $\mathbf{T}_c$ still improves the prediction accuracy but with a lower slope. In practice, it has been observed that "half sparsity" was leading to the fact that all the atoms in $\mathbf{A}_c$ were used by the training samples, thus $\mathbf{A}_t$ could be safely optimized.

4.2.6 Simplifying the algorithm

Having observed that the sparsity constraint can be relaxed (or even be neglected), the schemes summarized in Table 4.1 and Table 4.2 can further be simplified. Without loss of generality, let us assume that the sparsity constraint hence the thresholding operator $\mathrm{Thr}_K[\,.\,]$ is ignored. Besides, knowing the fact that KLT represents the training data

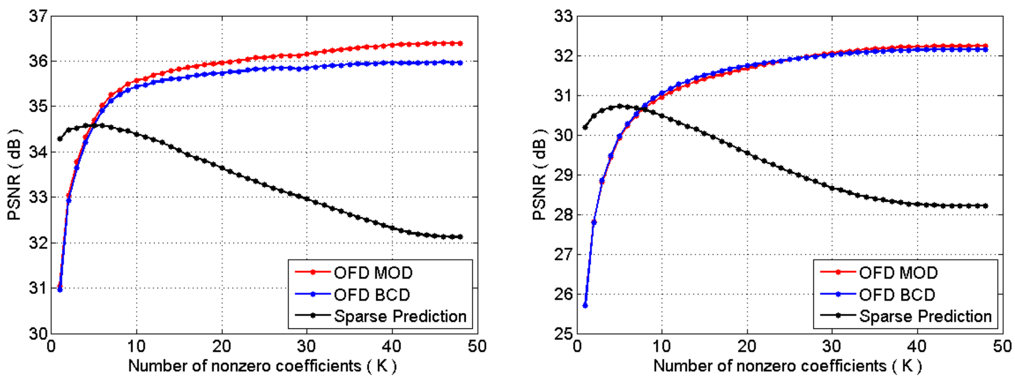


Figure 4.3: Mean approximation PSNR versus sparsity performance curves of (left) Foreman and (right) Barbara for the unknown block using on-the-fly dictionaries - Block size is 4×4 pixels, and the approximation support Mode 1 has been used as illustrated in Fig. 4.2.

```
Inputs: $\mathbf{T}_c$, $\mathbf{T}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
   $\mathbf{Y} = \mathbf{T}_c = [\mathbf{y}_1 \,|\, \ldots \,|\mathbf{y}_J]$
   $\mathbf{\Gamma} = \mathbf{Y}\mathbf{Y}^T$ and $\mathbf{\Xi}_t = \mathbf{T}_t\mathbf{Y}^T$
   $\mathbf{A}_t = \mathbf{\Xi}_t\mathbf{\Gamma}^{-1}$
   $\mathbf{x} = \mathbf{b}_c$
   $\hat{\mathbf{b}}_t = \mathbf{A}_t\mathbf{x}$
```

Table 4.3: Image prediction algorithm based on simplified on-the-fly dictionaries using the MOD solution - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

in a complete manner, it might be replaced with some other orthonormal and complete basis which is simple and fast to implement. Although any frame spanning the signal space (e.g., DCT basis) would work for this purpose, a trivial replacement can be the standard (canonical) basis where the training data can perfectly be represented. In this case, the linear weighting coefficients are simply the pixel values of the training samples. Thus, Eqn. (4.26) can be rewritten as

$$\begin{bmatrix} \mathbf{b}_c \\ \hat{\mathbf{b}}_t \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{N_1} \\ \mathbf{A}_t \end{bmatrix} \mathbf{b}_c, \tag{4.27}$$

and the simplified versions of the OFD learning algorithm are shown in Table 4.3 and

```
Inputs: $\mathbf{T}_c$, $\mathbf{T}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
Initialization: $\mathbf{A}_t$ with warm restart
  $\mathbf{Y} = \mathbf{T}_c = [\mathbf{y}_1 | \ldots | \mathbf{y}_\mathrm{J}]$
  $\mathbf{\Gamma} = \mathbf{Y}\mathbf{Y}^\mathrm{T} = [\boldsymbol{\gamma}_1 | \ldots | \boldsymbol{\gamma}_\mathrm{M}]$
  $\mathbf{\Xi}_t = \mathbf{T}_t\mathbf{Y}^\mathrm{T} = [\boldsymbol{\xi}_{t_1} | \ldots | \boldsymbol{\xi}_{t_\mathrm{M}}]$
  repeat until convergence on $\mathbf{A}_t = [\mathbf{a}_{t_1} | \ldots | \mathbf{a}_{t_\mathrm{M}}]$
    for $m = 1$ to M
      $\boldsymbol{\vartheta}_{t_m} = \frac{1}{\Gamma_{mm}} \left(\boldsymbol{\xi}_{t_m} - \mathbf{A}_t\boldsymbol{\gamma}_m\right) + \mathbf{a}_{t_m}$
      $\mathbf{a}_{t_m} = \frac{1}{\max\left(\|\boldsymbol{\vartheta}_{t_m}\|_2, 1\right)} \boldsymbol{\vartheta}_{t_m}$
    end for
  end repeat
  $\mathbf{x} = \mathbf{b}_c$
  $\hat{\mathbf{b}}_t = \mathbf{A}_t\mathbf{x}$
```

Table 4.4: Image prediction algorithm based on simplified on-the-fly dictionaries using the BCD solution - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

Table 4.4. Note that the subdictionary $\mathbf{A}_t$ still needs to be updated as in the complete method so that, given the representation vector calculated on the approximation support $\mathbf{b}_c$, it will lead to a good approximation of the unknown block to be predicted when used with the same representation vector. Note also that the update equation for the subdictionary $\mathbf{A}_t$ amounts to a block-wise quadratic linear regression for the MOD solution, and to an approximate least squares solution for the BCD solution.

4.2.7 Learning based on patch clustering

The OFD structure described above is indeed a very effective method in approximating the unknown values of the block to be predicted, when compared especially to SP. The initial observations show that the imposed sparsity constraint is actually not required particularly for the proposed prediction framework. However, both complete and simplified (MOD and BCD) solutions suffer from the computational complexity introduced by the size of the search window W. The size of W is a parameter which controls the number of training samples, hence the complexity of the approach. In order to reduce the computational load while keeping the size of W as large as possible, one can use instead a small subset of atoms (image patches) which are carefully chosen from W. For this purpose, we propose below a patch clustering based approach which selects and/or uses a small subset of image patches present in W. A brief performance analysis with advantages and disadvantages will be discussed in the upcoming sections.

Patch clustering and optimum cluster selection

We propose here an initialization phase, that performs a heuristic classification of image blocks in the training set $\mathbf{T}$, which uses image gradients to obtain the class labels of training samples $\mathbf{t}_j\ \forall j$. Assuming that there are I clusters, the training set $\mathbf{T}$ has been divided into I subsets $\mathbf{T}^i,\ i = 1...\mathrm{I}$, consisting of image blocks which have dominant orientations in a given directional range defined as

$$\theta_i = \left[\frac{\pi\,(i-1)}{\mathrm{I}}, \frac{\pi i}{\mathrm{I}}\right]\ \forall i \tag{4.28}$$

where θ_i (in radians) denotes the orientation range of the patches in cluster–i.

After clustering training image patches into I subsets, one can test each individual subset $\mathbf{T}^i$ for a best predicted block $\hat{\mathbf{b}}_t$ by applying the OFD method as described above. In this case, the optimum class number i_{opt} needs to be encoded and then transmitted in order to ensure that the decoder can do the same prediction using the subset $\mathbf{T}^{i_{opt}}$. The optimum class can be selected (as in the approximation support selection scheme) according to two criteria by minimizing either the prediction signal SSE or the RD cost function on the block to be predicted.

On the other hand, an automatic cluster selection scheme could be employed in order to prevent transmitting an extra side information to the decoder. Since the only information available to both encoder and decoder is the approximation support (template) $\mathbf{b}_c$, one can assume that a good representing class–i of the support region $\mathbf{b}_c$ would lead to an "unsupervised" selection of the subset $\mathbf{T}^i$, however at the expense of a possible performance decrease on the prediction quality when compared to the above optimized case. For example, the selection can be done by minimizing the following cost function:

$$\underset{i\in[1,\mathrm{I}]}{\arg\min} \left\| \mathbf{b}_c - \mathbf{T}_c^i {\mathbf{T}_c^i}^{+} \mathbf{b}_c \right\|_2^2 \tag{4.29}$$

```
Input:   $\mathbf{T}_c$, $\mathbf{T}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, I
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
Initialization: Cluster $\mathbf{T}$ into I subsets $\mathbf{T}^i$ using block gradients
    $i_{opt} = \arg\min_{i\in[1,\mathrm{I}]} \left\| \mathbf{b}_c - \mathbf{T}_c^i \mathbf{T}_c^{i+} \mathbf{b}_c \right\|_2^2$
    $\mathbf{Y} = \mathbf{T}_c^{i_{opt}}$
    $\mathbf{\Gamma} = \mathbf{Y}\mathbf{Y}^{\mathrm{T}}$ and $\mathbf{\Xi}_t = \mathbf{T}_t^{i_{opt}}\mathbf{Y}^{\mathrm{T}}$
    $\mathbf{A}_t = \mathbf{\Xi}_t \mathbf{\Gamma}^{-1}$
    $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{b}_c$
```

Table 4.5: Image prediction algorithm based on simplified on-the-fly dictionaries with patch clustering and automatic cluster selection - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

when the simplified OFD method has been taken into consideration whereas its extension to the complete OFD algorithm is straightforward. The clustering based simplified OFD prediction scheme (using the MOD solution) with automatic cluster selection has been summarized in Table 4.5.

4.3 Application to Image Compression

The proposed OFD algorithm has first been assessed comparatively to H.264/AVC intra prediction and also to SP. It has then been assessed in a complete image compression scheme where the prediction residue is transform encoded. In the experiments reported here, both complete and simplified OFD structures have been evaluated together with an extension to training set clustering method. The results obtained show a significant improvement in terms of the quality of the predicted image (up to 3 dB) compared to H.264/AVC intra modes. Moreover, significant RD gains have been achieved on the reconstructed image, after encoding and decoding the prediction residue, up to 2 dB compared with H.264/AVC intra modes and to the SP approach.

4.3.1 Encoder structure

In order to be able to initialize the prediction/compression process, the top 4 rows and left 4 columns of blocks of size 4×4 are intra predicted with H.264/AVC. Once a block has been predicted with the respective prediction method (e.g., SP, OFD, or H.264/AVC

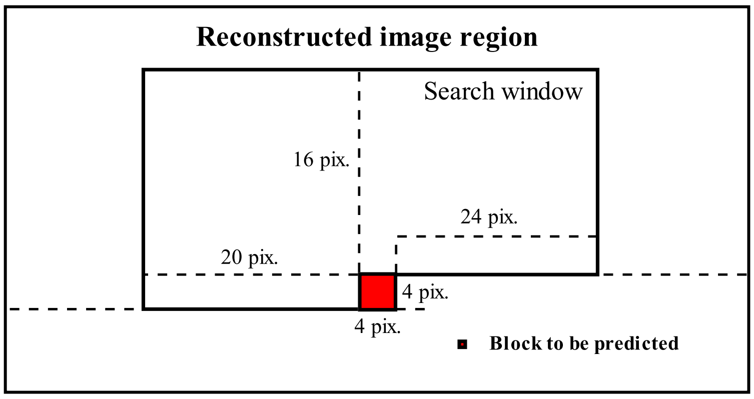


Figure 4.4: The configuration of the search window used in the simulations - The configuration of the search window for 4×4 block size. All possible image patches are extracted from the search window in order to construct the training matrix $\mathbf{T}$.

intra), the DCT transformed 4×4 residual block is encoded with an algorithm similar to JPEG, in which a uniform quantization matrix with $\Delta = 16$ is weighted by $\mathbf{w}_{q_f}$ where $q_f = 10, 20, ..., 90$ (please see (3.26)). Image blocks are processed in a raster scan order, and the reconstructed image is obtained by adding the quantized residue to the prediction. A skip mode has also been included into the encoder to avoid coding the blocks of prediction residue in which all the transformed and quantized coefficients are zero. The skip mode flag, which is binary, is arithmetically encoded.

In conventional methods, usually clean (noise-free) training signals –extracted from some other natural images– are used for dictionary learning. However, this is not suitable for the prediction problem in an image coding context since at the decoder side the approximation support (template) may not be noise-free depending on the prediction quality and the residue signal quantization. In this case, learning a locally adaptive dictionary –as a function of quality factor– with a limited number of contextual training samples extracted from a search window in the reconstructed image region would give more reasonable prediction and also compression results than using a fixed size dictionary learned offline with noise-free samples. With this motivation, the training samples (image patches) in the matrix $\mathbf{T}$ are collected from the previously decoded and encoded region (i.e., search window) which is located in a causal neighborhood of the block to be predicted B. All possible image patches in the search window are extracted to construct the matrix $\mathbf{T}$. Fig. 4.4 shows the configuration of the search window for

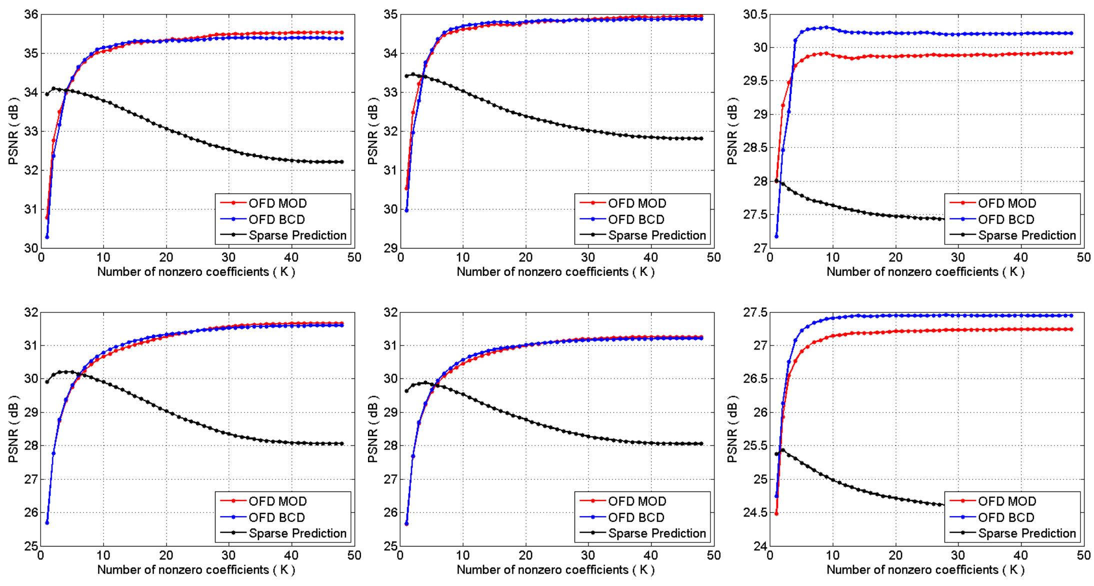


Figure 4.5: Mean prediction PSNR versus sparsity performance curves of noise corrupted (top) Foreman and (bottom) Barbara for the unknown block using on-the-fly dictionaries - With (left) low ($q_f = 70$), (middle) medium ($q_f = 50$), and (right) high ($q_f = 10$) level of quantization noise corruption. Block size is 4×4 pixels, and the approximation support Mode 1 has been used as shown in Fig 4.2.

4×4 block size that is used in the experimentations reported here.

The other advantage of this online structure is that it provides flexibility in adapting the size of dictionaries to varying numbers of pixels, i.e., offering a full compatibility to the possibly varying number of known pixels in the approximation support as well as in the unknown area to be predicted. As here allowing us to optimize the prediction using nine different modes of approximation supports (with different dimensions) as shown in Fig. 4.2. The SSE/RD optimized support mode type is signaled to the decoder using Huffman codes.

In the SP method, OMP is limited to $K = 8$ iterations, the best iteration number k_{opt} is Huffman encoded when transmitted to the decoder, and the same configuration of the search window (see Fig. 4.4) is used for constructing the dictionary $\mathbf{A}$.

4.3.2 Impact of sparsity constraint with quantization noise

Fig. 4.5 illustrates the mean prediction PSNR performance curves obtained for Foreman and Barbara with varying sparsity constraints in the case where image signals (i.e.,

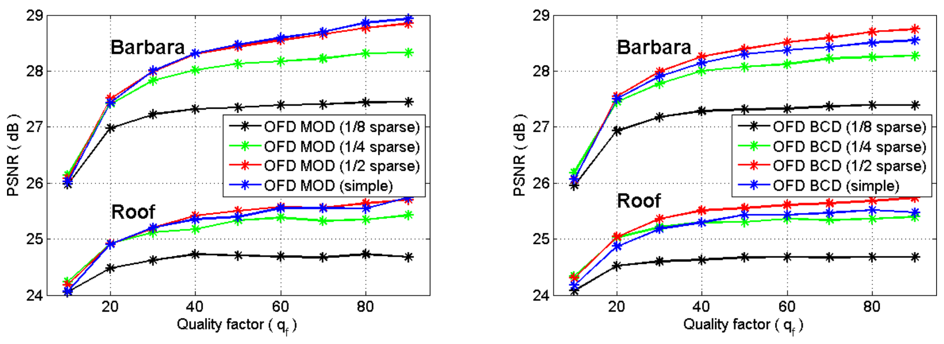


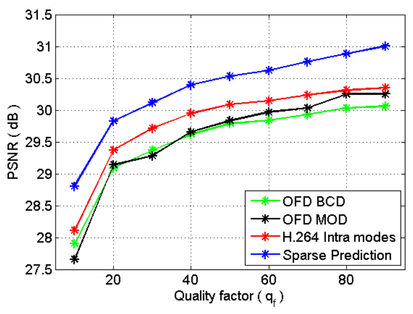
Figure 4.6: Prediction performance curves of Barbara and Roof using on-the-fly dictionaries: (left) MOD and (right) BCD with various sparsity constraints - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE.

image blocks) are corrupted by a low, a medium, and a high level of quantization noise, i.e., image residue blocks are encoded in a lossy manner with the quantization scheme as defined previously. The observed prediction performance behavior is very similar to the lossless encoding case which is shown in Fig. 4.3. The prediction quality improves by relaxing the sparsity constraint, even in the presence of the high level of noise. Note here that the proposed OFD structures with two different solutions, MOD and BCD, have almost the same performance under low and medium noise corruption. However, the BCD based method works relatively better when there exists a high level of noise corruption in the image. So, it can be an effective alternative to the MOD based method for low bit-rate prediction/compression.

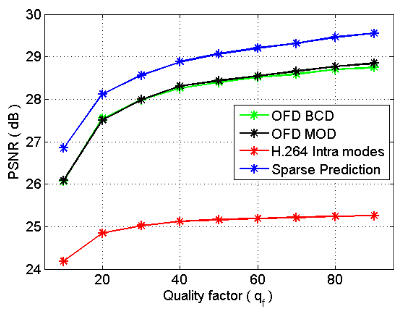
4.3.3 Experimental results

4.3.3.1 Prediction performance with SSE criterion

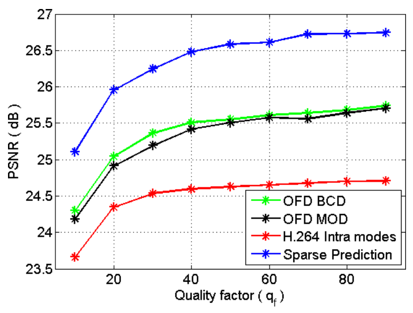
Fig. 4.6 shows the prediction PSNR performance curves obtained for Barbara and Roof using the complete OFD algorithm with various quantization levels (in the complete coding scheme) as well as with the simplified method. The sparsity here is enforced by keeping a number (a ratio to the length of the representation vector) of significant coefficients and forcing the others to be zero. For example, in 1/4–sparse case, one-fourth of the (large valued) coefficients have been kept and the others are made zero. Relaxing the sparsity constraint (up to half sparsity) leads to better prediction in most



(a)



(b)



(c)

Figure 4.7: Prediction performance curves of (a) Foreman, (b) Barbara, and (c) Roof using on-the-fly dictionaries in comparison to H.264/AVC intra and SP - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE.

of the cases. One can observe that the complete OFD method with sparsity constraints leads to better prediction especially at low quality factors (low bit-rates), but that in some cases, the simplified version gives very close performance results.

Fig. 4.7 compares the prediction PSNR performance of (half sparsity constrained) OFD with H.264/AVC intra and SP, where the optimization criterion is the minimization of the prediction signal SSE in the complete coding scheme. The quality of the predicted signal, both visually (see Fig. 4.8) and in terms of PSNR, is significantly improved when compared to H.264/AVC intra prediction especially for the images which contain textural regions. A gain more than 3 dB has been achieved for Barbara. Since H.264/AVC intra modes are highly aligned with the orientation of the contours in the Foreman image, it works slightly better than OFD in terms of prediction quality. Note here that SP outperforms all the other methods since the optimization is done in two

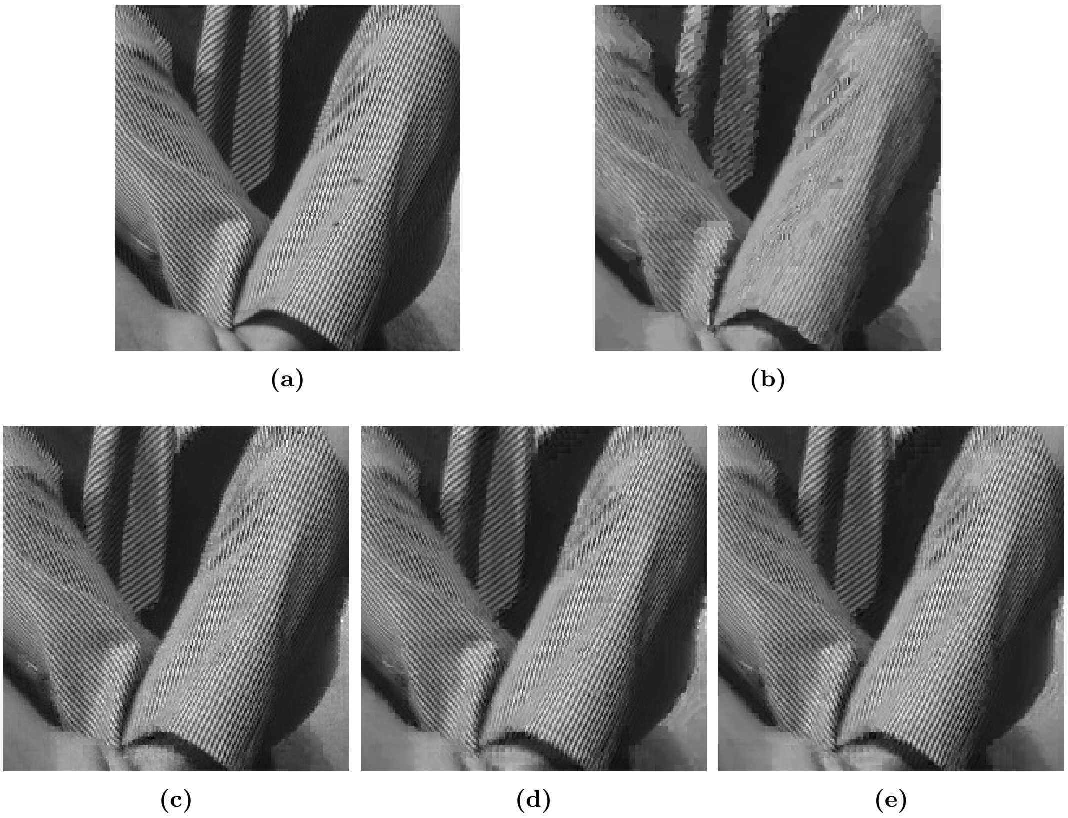
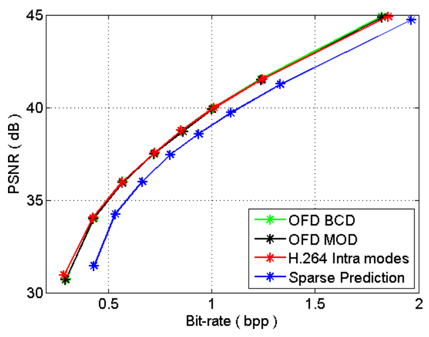


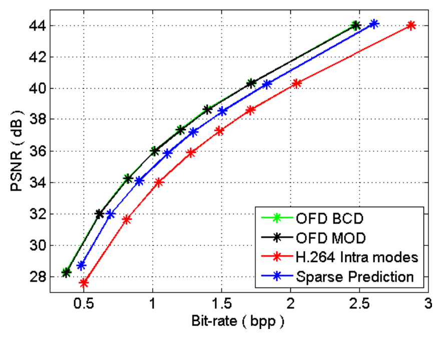
Figure 4.8: Predicted images of a textural region in Barbara at low bit-rates ($q_f = 20$) - (a) Original image, (b) H.264 intra modes, (c) SP, (d) OFD–MOD, and (e) OFD–BCD based prediction methods with the SSE criterion.

steps, i.e., first for the selection of iteration number, and then for the selection of the approximation support type. In terms of encoding, one needs to add the coding cost of the extra side information, i.e., here of the best iteration number k_{opt} of OMP.

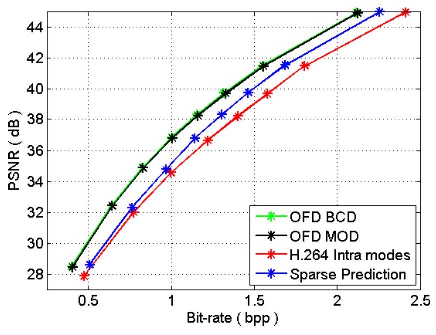
Note that in Fig. 4.8, the prediction residue has been encoded with a quality factor $q_f = 20$. Thus, the approximation support of the block to be predicted as well as the collected training patches (or the elements in the dictionary for SP) contain quantization noise which is similar for all prediction methods tested here. However, these methods do not lead to same overall bit-rate since the residue obtained and the side information to be transmitted are not the same with all methods. Fig. 4.6 and Fig. 4.7 also show prediction performance curves for different quality levels of q_f in the same setup.



(a)



(b)

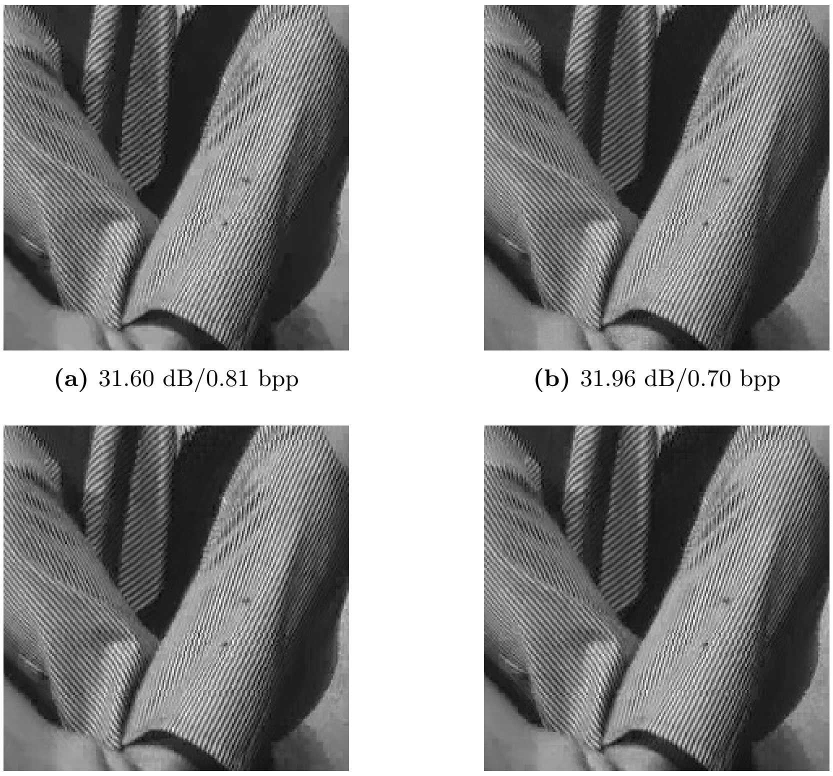


(c)

Figure 4.9: Encoding performance curves of (a) Foreman, (b) Barbara, and (c) Roof using on-the-fly dictionaries in comparison to H.264/AVC intra and SP - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the predicted signal.

4.3.3.2 Encoding performance with RD criterion

Fig. 4.9 demonstrates the total encoding PSNR/bit-rate performance curves for the test images where the optimization criterion is the minimization of the RD cost function. The coding cost of the optimum iteration number (for SP), the selected prediction mode (for H.264/AVC intra), the approximation support type (for SP and OFD), and the skip mode have been included into total bit-rate. One can observe that the proposed learning based prediction methods (using both MOD and BCD solutions) lead to comparable compression results and improve the encoding performance of the images containing more complex and textural regions. A gain up to 2 dB has been achieved when compared to H.264/AVC intra and the sparse prediction method. Although SP outperforms all the other methods in terms of prediction, encoding cost of an ex-



(a) 31.60 dB/0.81 bpp

(b) 31.96 dB/0.70 bpp

(c) 31.97 dB/0.61 bpp

(d) 31.99 dB/0.61 bpp

Figure 4.10: Reconstructed images of a textural region in Barbara - (a) H.264 intra modes, (b) SP, (c) OFD–MOD, and (d) OFD–BCD based prediction methods with the RD criterion.

tra side information, i.e., the best iteration number of OMP, decreases drastically the PSNR/bit-rate performance in some images (e.g., like Foreman) even below H.264/AVC intra prediction. Fig. 4.10 shows the reconstructed textural region of Barbara image (corresponding to Fig. 4.8) in comparison to H.264/AVC intra and SP.

4.3.3.3 Performance analysis of patch clustering

Here we will briefly analyse the effect of image gradients based training set clustering on prediction quality as well as encoding performance. To begin with, we classify the training image patches into I = 6 clusters in which the elements (image blocks) of the subsets $\mathbf{T}^i$, $i = 1...6$, have similar directional orientations in a given range defined as $\theta_i = [30^o\,(i-1)\,, 30^o i]\ \forall i$ (in degrees). Two sets of tests have then been carried out.

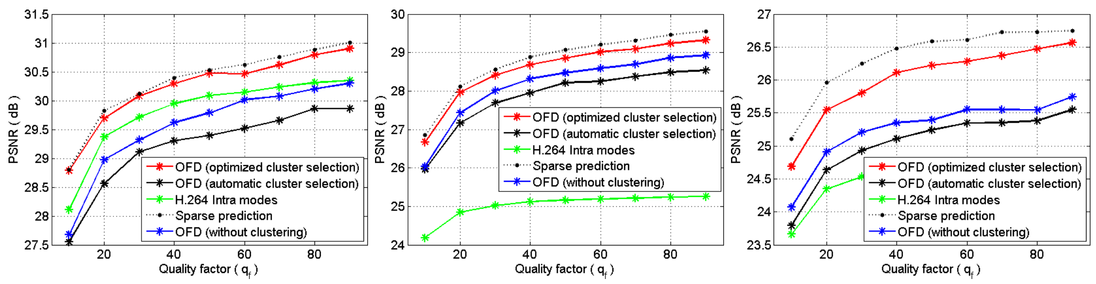
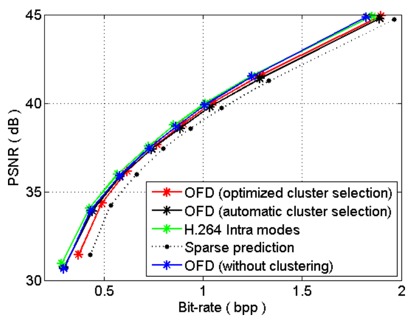


Figure 4.11: Prediction performance curves of (left) Foreman, (middle) Barbara, and (right) Roof using on-the-fly dictionaries with different clustering methods in comparison to H.264/AVC intra and SP - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE.

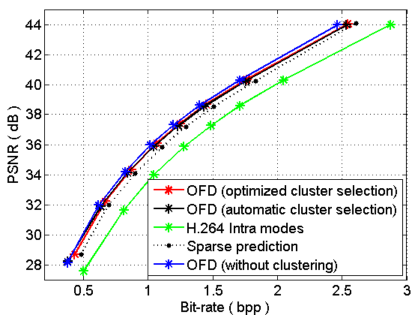
The first one makes use of an optimization scheme for selecting the best subset $\mathbf{T}^i$ in terms of the SSE/RD value of the block to be predicted. In this setup, in addition to the optimum approximation support mode, the selected optimal cluster number i_{opt} needs to be transmitted to the decoder. This information is signaled using Huffman codes. The second set of experiments prevents transmitting this extra information hence a cluster has been automatically selected by means of (4.29) using the approximation support information. All the experimentations given here are based on the simplified OFD algorithm using the MOD solution. Note here that we implicitly assume a union of canonical bases (for the subdictionary $\mathbf{A}_c$) whereas its extension is straightforward for the complete method (i.e., a union of KLT bases which is very similar to [88], or even a union of directional DCT bases can be used).

Fig. 4.11 compares the prediction PSNR performance of the different cluster selection methods with simplified OFD as well as the H.264/AVC intra prediction modes and SP. The optimization criterion here is the minimization of the prediction signal SSE. In terms of prediction quality, as one can expect, the optimized cluster selection improves the performance up to 1 dB (note here that the coding cost of i_{opt} has to be taken into account for encoding). However, the automatic cluster selection process degrades the prediction performance since cluster selection has been done by minimizing the support region (or template) approximation error.

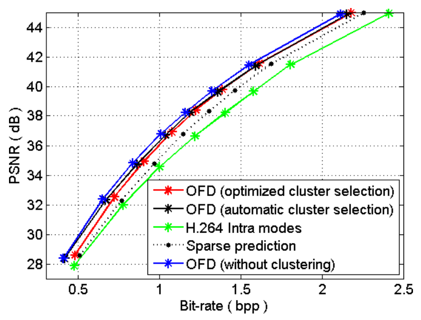
Fig. 4.12 shows the encoding PSNR/bit-rate performance curves for the same test images where the optimization criterion is the minimization of the RD cost function (in



(a)



(b)



(c)

Figure 4.12: Encoding performance curves of (a) Foreman, (b) Barbara, and (c) Roof using on-the-fly dictionaries with different clustering methods in comparison to H.264/AVC intra and SP - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the predicted signal.

the complete encoding scheme). One can observe that the proposed clustering based methods lead to comparable compression results. Although there is a little performance decrease, the overall encoding performance still outperforms H.264/AVC intra and also SP for Barbara and Roof images. Note here that the clustering scheme has been applied on the training samples which are effected by the quantization noise, hence a perfect clustering is almost impossible to obtain.

4.3.3.4 A hybrid encoder/decoder model

Fig. 4.13 demonstrates the selected blocks by the OFD algorithm in a hybrid prediction and encoding model which is a combination of the simplified OFD and H.264/AVC prediction modes. Here the criteria of selecting the prediction method (either OFD

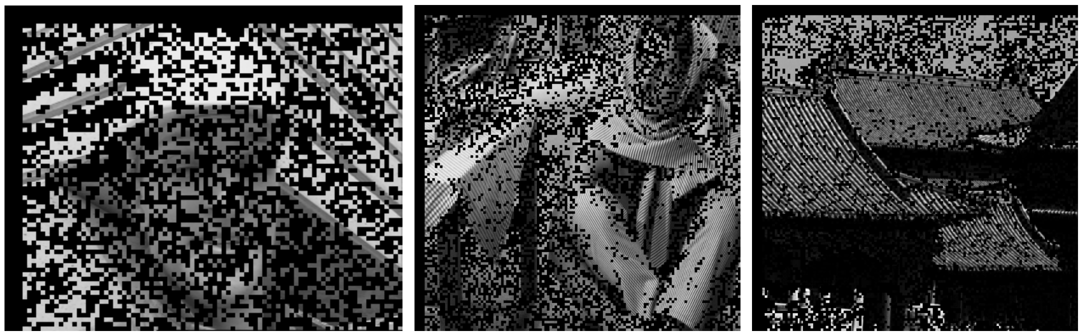


Figure 4.13: Competitive RD selection of the hybrid model at $q_f = 20$ - (Left) Foreman, (middle) Barbara, and (right) Roof. The blocks shown are selected by the (simplified) OFD algorithm: 45.50%, 58.81%, and 55.95% for Foreman, Barbara, and Roof respectively.

or H.264/AVC intra) and the prediction mode (out of nine modes for each) is the minimization of the RD cost function (at $q_f = 20$) as described earlier in this chapter. The blocks which are shown in Fig. 4.13 are selected competitively by the proposed OFD algorithm in the hybrid model. The selection percentages are 45.50%, 58.81%, and 55.95% for Foreman, Barbara, and Roof respectively. One can see that the proposed method works better for more complex, non-periodic and highly textural regions.

4.4 Computational complexity analysis

In H.264/AVC intra prediction, the prediction of each pixel within an unknown block to be predicted is obtained as a linear weighted summation (i.e., interpolation, extrapolation) of the surrounding spatial neighboring pixels. The prediction coefficients of each pixel for each prediction direction is calculated in advance and fixed, which are supposed to be known both at the encoder and decoder. In this case, the computational load required for prediction is very low, however, this kind of fixed extrapolation technique can achieve good results for only simple textures, and it can not adapt to the variational content and the complex structures in the image. Observing this trade-off between computational complexity and prediction/encoding performance, adaptive prediction techniques, as defined in this chapter, inevitably have more complexity and require more computational load for prediction. For all the methods described here, the encoder is more complex than the decoder since some side information is encoded and sent to the decoder, e.g., the best prediction mode, etc.

Suppose that for a given training samples matrix $\mathbf{T} \in \mathbb{R}^{\mathrm{N}\times\mathrm{J}}$, $\mathbf{T}_c$ is of size $\mathrm{N}_1 \times \mathrm{J}$ and $\mathbf{T}_t$ is of size $\mathrm{N}_2 \times \mathrm{J}$ where $\mathrm{N}_1 + \mathrm{N}_2 = \mathrm{N}$. Similarly, the known pixel values in the approximation support are stored in the column vector $\mathbf{b}_c$ of size N_1, and for 4×4 block size $\mathrm{N}_2 = 16$. The complexity of the OFD method is directly related with the configuration of the search window W, hence the number J, and the dimension N of training samples stored in the columns of the matrix $\mathbf{T}$. The first step of the method is the calculation of the KLT basis of the training samples stored in the matrix $\mathbf{T}_c$. KLT here has an approximate complexity of $\mathcal{O}(\mathrm{N}_1^4)$, however, it might be replaced with a trivial canonical basis as proposed in the simplified version, or even with an other static basis such as DCT or DFT. The calculation of the subdictionary $\mathbf{A}_t$ requires $\mathrm{J}\mathrm{N}_1^2 + \mathrm{J}\mathrm{N}_1\mathrm{N}_2 + \mathrm{N}_1^2\mathrm{N}_2$ multiply-add operations with an $\mathrm{N}_1 \times \mathrm{N}_1$ matrix inversion (for the MOD solution).

An interesting property of the proposed prediction method here is that it can be initialized with *warm restarts* [102] since any two successive blocks have a large common search region in which the training samples are the same (please see Fig. 3.18). E.g., after predicting one block in the algorithm, the obtained subdictionary $\mathbf{A}_t$ for that block can effectively be used as an initial dictionary for the next block, and so on. Here the dictionary update problem reduces to a relatively simple problem of introducing new training samples while forgetting old ones gradually. In this case, a few iterations of (4.24) could be sufficient for convergence in the BCD solution. Moreover, adapting a recursive least squares method as in [103] will prevent explicit inversion and multiplication of large matrices for the MOD solution.

The complexity can further be reduced by using less number of training samples either by restricting the configuration of the search window W, or by selecting a subset of image patches from the search window, or even by selecting a fixed number of patches from the search window (e.g., a limited number of closest patches in Euclidean space). In the experiments reported in this chapter, all image patches (or a selected subset with patch clustering) in the search window are tested for learning.

4.5 Conclusion

In this chapter, a novel spatial texture prediction method based on online learning of dictionaries has been introduced. The image prediction problem has been put into a dictionary learning framework by learning two “related” subdictionaries, $\mathbf{A}_c$ and $\mathbf{A}_t$. In other words, a convential dictionary learning scheme has been divided into its two main steps (sparse coding and dictionary update) which are operating on two “related” training datasets, $\mathbf{T}_c$ and $\mathbf{T}_t$. The learning of $\mathbf{A}_c$ is done to lead to a sparse approximation of the known samples in the approximation support. At the same time, the goal is to make sure that the sparse vectors computed in the approximation with $\mathbf{A}_c$ of the known samples will also lead to a good approximation of the block to be predicted but this time using $\mathbf{A}_t$, hence the need for optimizing $\mathbf{A}_t$ given the corresponding samples of the set of training patches. This is the key point behind the proposed method and this, to our knowledge is a novel contribution with respect to classical dictionary learning methods. It has been shown that the proposed prediction method offers a powerful and efficient tool leading to very good performance both in terms of prediction quality and in terms of RD performance when integrated in a complete image compression algorithm. For more complex and non-periodic texture areas, it turns out to be an effective alternative to H.264/AVC and sparse approximation based prediction methods.

The computational complexity is always an issue for learning methods, hence most of the dictionary learning schemes are generally evaluated offline. In order to reduce the computational load, here $\mathbf{A}_c$ has been fixed and constrained to be orthonormal in order to be able use simple orthonormal projections followed by a thresholding to calculate the sparse representation vectors of the training samples, which is much more simple than using iterative greedy methods like MP or OMP. This is why this constraint has been used even if it is indeed more restrictive. Alternatively, a classical dictionary learning algorithm can be employed for $\mathbf{A}_c$, however at the expense of the required computational load. In the proposed method, $\mathbf{A}_t$ is the crucial subdictionary for the prediction problem since it is one used for predicting the unknown values in the block to be predicted. Hence $\mathbf{A}_t$ is not restricted and it is optimized using different least squares approaches. The nature of the problem allows reducing the dictionary update problem to a relatively simple problem of introducing new training samples while forgetting old ones gradually, which leads to further reduce the computational requirements.

As a final remark, because of its simplicity, the OFD structure can easily be extended to any other texture synthesis problems such as image denoising and image inpainting.

Chapter 5

Image Prediction based on Neighbor Embedding Methods

With the recent developments in computer technology, most of the machine learning and pattern recognition applications need to deal with increasing quantities of data signals which are described by large numbers of features. One of the crucial problems in dealing with high-dimensional data is that, not all the collected measurements are perfect because of the noise corruption and other degradations. Therefore, dimensionality reduction methods are applied to investigate the underlying exact characteristics of the data of interest. These methods are usually applied as a pre-processing step or a part of the data analysis tools in order to simplify the high-dimensional data into a lower dimension. The common aspects of these approaches include computational efficiency, noise removal, data and text mining, and so on. Unfortunately, there is not any record in the literature analyzing the potential power of some of these methods in terms of signal extrapolation, i.e., synthesizing/predicting unknown pixel values from observed samples in a neighborhood (approximation support, or template).

In this chapter[1], we first describe the main principles of certain dimensionality reduction methods by presenting their solutions given in the literature. Next we place the image prediction problem in the context of data dimensionality reduction. More precisely, we develop two new intra image prediction methods based on two data dimensionality reduction techniques: non-negative matrix factorization (NMF) and locally

[1]The content of this chapter is related to our publications in [123, 124]. A related recent submission, which is not included into this chapter, is available in [125].

linear embedding (LLE), which we may refer to as **neighbor embedding methods**. These two methods aim at approximating a block to be predicted in the image as a linear combination of K nearest neighbors determined on the known pixels in the approximation support of the block. The variable K can be seen as a parameter controlling some sort of sparsity constraint of the approximation vector. The impact of this parameter as well as of the non-negativity (because of NMF) and sum-to-one (because of LLE) constraints for the addressed prediction problem has been analyzed. The prediction and rate-distortion performances of these two new image prediction methods have then been evaluated in a complete image coding and decoding algorithm. Simulation results show gains up to 2 dB in terms of PSNR of the reconstructed signal after coding and decoding of the prediction residue, when compared to H.264/AVC intra prediction modes, up to 3 dB when compared with template matching, and up to 1 dB when compared with the sparse prediction method.

5.1 Data Dimensionality Reduction

A dataset in high-dimensional space can often be transformed into a lower dimension with a very little or no loss of important information for a better analysis, regression, presentation, and visualization of the data. The classical and widely utilized method for linear dimensionality reduction is principle component analysis (PCA) [90], which is also known as KLT or the *Hotelling* transform. PCA aims at preserving the covariance structure (up to rotation) [126]. The high-dimensional data is represented with a subspace in a lower dimension that is spanned by the components of largest variance in the dataset. Formally, PCA projects N-dimensional input signals $\mathbf{b}_j,\ j = 1...\mathrm{J}$, onto an M-dimensional subspace which is spanned by the orthonormal basis functions $\mathbf{a}_m,\ m = 1...\mathrm{M}$ and $\mathrm{N} \geqslant \mathrm{M}$, by minimizing the reconstruction error given by

$$\sum_{j\in \mathrm{J}} \left\| \mathbf{b}_j - \sum_{m=1}^{\mathrm{M}} \left(\mathbf{b}_j^{\mathrm{T}} \mathbf{a}_m\right) \mathbf{a}_m \right\|_2^2 . \tag{5.1}$$

PCA can be calculated by the eigenvalue decomposition of the data covariance matrix[1], or SVD of the data matrix, assuming the mean of the dataset for each attribute is cen-

[1]Given a dataset $\mathbf{B} = [\mathbf{b}_1 \,| \ldots \,| \mathbf{b}_\mathrm{J}\,] \in \mathbb{R}^{\mathrm{N}\times \mathrm{J}}$, the N×N data covariance matrix is defined as $Cov\left(\mathbf{B}\right) = \frac{1}{J}\sum_{j=1}^{\mathrm{J}} \mathbf{b}_j \mathbf{b}_j^{\mathrm{T}}$, assuming the mean of the dataset for each attribute is centered on the origin, $\sum_{j=1}^{\mathrm{J}} \mathbf{b}_j = \mathbf{0}$.

tered on the origin. The M-dimensional subspace is defined by the top M-eigenvectors of the data covariance matrix, and the outputs of PCA are simply the coordinates of the input signals in this subspace, i.e., orthonormal projection of input signals onto M-dimensional subspace obtained by PCA. Note in (5.1) that the signal $\mathbf{b}_j$ is approximated by a linear combination of basis functions $\mathbf{a}_m$, where the weighting coefficients correspond to the orthonormal projection of $\mathbf{b}_j$ onto the directions defined by $\mathbf{a}_m \; \forall m$.

There are several linear and non-linear data dimensionality reduction methods operating under different constraints. E.g., metric multidimensional scaling (MDS) [127] is a linear method preserving the inner products between the data points in the dataset. Yet another linear method which is called non-negative matrix factorization (NMF) [128] can be regarded as a generalization of PCA with non-negativity constraints, where only additive linear combinations are allowed in the notion of "combining parts to form a whole" [129]. In detail, NMF is a subspace approximation algorithm which finds a suitable low-rank representation of the high-dimensional data. In many data analysis tasks the data to be analysed is non-negative, and classical tools, e.g., PCA, independent component analysis (ICA) [130], can not guarantee to maintain the non-negativity property of the original data in the low-dimensional space [131]. NMF is a recent method for obtaining such a non-negative representation, which is indeed helpful for physical interpretation of the results in many data analysis tools such as dimensionality reduction, data mining, and noise removal.

When the high-dimensional data is confined to a low-dimensional subspace, linear dimensionality reduction methods give reliable representations of the data in the lower dimension. However, if the high-dimensional data lies on or near to a low-dimensional submanifold, linear methods tend to fail since the data might be highly non-linear. Most of the linear methods can be extended to a non-linear framework in order to handle the non-linear characteristics of the data. The generalization is usually done by applying the so-called *kernel trick*. Kernel PCA (KPCA) [132] can be seen as a non-linear generalization of PCA (also of MDS) which replaces the Euclidean dot product space with a space defined by a *kernel* function. A kernel function can be viewed as a non-linear similarity measure [126]. Typically polynomial or Gaussian kernels are used with KPCA, however, these kernels are not very well adapted for manifold learning.

ISOMAP [133] and locally linear embedding (LLE) [134] are the two recent graph based manifold learning methods relying on the non-linear generalizations of PCA.

These methods can also be seen as an extension of KPCA because their kernels (as graphs) are derived from the data instead of using a pre-defined kernel function. ISOMAP tries to represent high-dimensional data in a lower dimension by preserving the pairwise distances of the data points, whereas the LLE method preserves the local linear structure of the neighboring data points. More precisely, LLE solves a constrained optimization problem by obtaining a global transformation of the high-dimensional coordinates into low-dimensional ones by exploiting the locally linear characteristics of the non-linear data. It aims at preserving the local linear structure of the high-dimensional data in the lower-dimensional space.

In this section, we mainly focus on the basic principles of NMF and LLE algorithms by presenting their problem formulations and solutions given in the literature. Our underlying main motivation of selecting these two methods is that they can be easily adapted, either completely or partially, to the image prediction problem which was defined in the previous chapters. Originating from the PCA formulation in (5.1), these two dimensionality reduction methods operate under different constraints, hence, give different results. Several dimensionality reduction methods, other than NMF and LLE, can definitely be envisaged for the image prediction problem. In this study, we restrict ourselves with these two methods in order to investigate the potential capacity of the dimensionality reduction algorithms in a signal extrapolation framework since, to the best of our knowledge, this chapter contains novel approaches for the image prediction problem. A detailed technical motivation will be given shortly.

5.1.1 Non-negative matrix factorization (NMF)

In theory, given a non-negative matrix $\mathbf{B} \in \mathbb{R}^{\mathrm{N}\times\mathrm{J}}$ and a positive integer $\mathrm{M} < \min\{\mathrm{N},\mathrm{J}\}$, the aim is to find non-negative matrix factors $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ and $\mathbf{X} \in \mathbb{R}^{\mathrm{M}\times\mathrm{J}}$, such that $\mathbf{B} \approx \mathbf{AX}$ where the reconstruction error between $\mathbf{B}$ and $\mathbf{AX}$ is minimized. In order to calculate a good approximation of $\mathbf{B}$, two cost functions are defined and optimized for the quality of the factorization: (i) squared Euclidean distance; (ii) Kullback–Leibler divergence [135]. The most widely used cost function is the squared Euclidean distance, i.e.,

$$\min_{\mathbf{A},\mathbf{X}} \left[\frac{1}{2}\|\mathbf{B}-\mathbf{AX}\|_F^2\right] \quad \text{subject to} \quad \mathbf{A} \geqslant \mathbf{0} \text{ and } \mathbf{X} \geqslant \mathbf{0}, \tag{5.2}$$

and NMF algorithm is optimized with the *multiplicative* update equations as

$$A_{nm} \leftarrow A_{nm} \frac{(\mathbf{B}\mathbf{X}^{\mathrm{T}})_{nm}}{(\mathbf{A}\mathbf{X}\mathbf{X}^{\mathrm{T}})_{nm} + \varepsilon}, \qquad X_{mj} \leftarrow X_{mj} \frac{(\mathbf{A}^{\mathrm{T}}\mathbf{B})_{mj}}{(\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{X})_{mj} + \varepsilon} \tag{5.3}$$

where ε is a small constant equal to 10^{-9} to avoid divide by zero in the update equations. Here A_{nm} (or X_{mj}) represents the n^{th} (or m^{th}) row and m^{th} (or j^{th}) column elements of the corresponding matrices, $\mathbf{A}$ and $\mathbf{X}$, respectively. In the standard algorithm, the matrices $\mathbf{A}$ and $\mathbf{X}$ are initialized with random non-negative values, and the Euclidean distance $||\mathbf{B} - \mathbf{A}\mathbf{X}||_F^2$ is *non-increasing* under the above alternating update rules as proven in [128]. *The Euclidean distance is invariant under these updates if and only if* $\mathbf{A}$ *and* $\mathbf{X}$ *are at a stationary point of the distance* [128].

The product $\mathbf{A}\mathbf{X}$ is called an NMF of $\mathbf{B}$, and the underlying features of $\mathbf{B}$ are extracted as basis vectors in $\mathbf{A}$ which can then be used in data analysis tools, e.g., for classification or identification. The matrix $\mathbf{X}$ contains the coordinates of the high-dimensional data (stored in the columns of $\mathbf{B}$) in a lower dimensional space which is spanned by the columns of $\mathbf{A}$. Note that $\mathbf{B} \approx \mathbf{A}\mathbf{X}$ can be rewritten in the vector form as $\mathbf{b} \approx \mathbf{A}\mathbf{x}$ where $\mathbf{b}$ and $\mathbf{x}$ represent the corresponding columns of $\mathbf{B}$ and $\mathbf{X}$ respectively. One can interpret that a column vector $\mathbf{b}$ of $\mathbf{B}$ is approximated by a linear combination of the columns of the dictionary $\mathbf{A}$, weighted by the column vector $\mathbf{x}$ of $\mathbf{X}$, i.e.,

$$\min_{\mathbf{A},\mathbf{x}} \left[\frac{1}{2} ||\mathbf{b} - \mathbf{A}\mathbf{x}||_2^2 \right] \quad \text{subject to} \quad \mathbf{A} \geqslant \mathbf{0} \text{ and } \mathbf{x} \geqslant \mathbf{0}. \tag{5.4}$$

NMF can also be extended to include some other auxiliary constraints on $\mathbf{A}$ and $\mathbf{X}$, in addition to the non-negativity constraint, depending on the requirements, e.g., sparseness constraint [136], in a specific application. A more detailed analysis of NMF with multiplicative update methods, gradient descent methods, and alternating least squares methods has been given in [131] by explaining sample applications.

5.1.2 Locally linear embedding (LLE)

LLE supposes that each high-dimensional data point and its neighbors lie on or close to a locally linear patch of a smooth non-linear manifold, so that the local geometry of these patches can be characterized linearly with the coefficients that approximate each data point from its neighbors [134]. Formally, given J high-dimensional data points consisting of N-real valued column vectors $\mathbf{b}_j$, the LLE method consists of the following steps:

1. It first identifies K nearest neighbors $\mathbf{b}_{j_k}$ per data point $\mathbf{b}_j$ for all $j, j \neq j_k$. The usual measure is the Euclidean distance.

2. It then searches for the weights W_{jk} so that each data point $\mathbf{b}_j$ is approximated by its neighbors $\mathbf{b}_{j_k}$, $k = 1...K$. The algorithm thus aims at minimizing a cost function $E(\mathbf{W})$ which is given as

$$E(\mathbf{W}) = \sum_{j \in \mathrm{J}} \left\| \mathbf{b}_j - \sum_{k=1}^{K} W_{jk} \mathbf{b}_{j_k} \right\|_2^2 . \tag{5.5}$$

The weights W_{jk} represent the contribution of the k^{th} neighboring point to the reconstruction of the j^{th} data point $\mathbf{b}_j$, and they are constrained to sum-to-one[1], i.e., $\sum_k W_{jk} = 1 \; \forall j$. Each data point $\mathbf{b}_j$ can only be reconstructed (approximated) from its neighbors (i.e., $W_{jk} = 0$ if the data point $\mathbf{b}_{j_k}$ does not belong to the neighbors of $\mathbf{b}_j$). The optimal weights W_{jk} satisfying these constraints are obtained by solving the constrained least squares problem per data point $\mathbf{b}_j$ as

$$E_j(W_{jk}) = \left\| \sum_k W_{jk} \left(\mathbf{b}_j - \mathbf{b}_{j_k} \right) \right\|_2^2 \quad \text{subject to} \quad \sum_k W_{jk} = 1 \; \forall j. \tag{5.6}$$

Let $\mathbf{B}_j = [\mathbf{b}_{j_1} | \ldots \; |\mathbf{b}_{j_K}]$ denote a data matrix in which the columns correspond to the K nearest neighbors obtained for the data point $\mathbf{b}_j$. The optimum weighting coefficients in $\mathbf{w}_j = [W_{j1} | \ldots \; |W_{jK}]$ can be calculated by

$$\mathbf{w}_j^{\mathrm{T}} = \frac{[Cov_{loc}(\mathbf{B}_j)]^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}[Cov_{loc}(\mathbf{B}_j)]^{-1}\mathbf{1}} \tag{5.7}$$

where $Cov_{loc}(\mathbf{B}_j)$ denotes the *local covariance matrix*[2] of $\mathbf{B}_j$ in reference to $\mathbf{b}_j$, and $\mathbf{1}$ is the column vector of ones.

[1] For any data point $\mathbf{b}_j$, the "*sum-to-one*" constraint on weighting coefficients enforces the invariance to translations of that data point and its neighbors. Note also that the invariance to rotations and rescalings follows from (5.5). So, with these symmetry properties, the weights characterize the intrinsic geometric properties of each neighborhood [134].

[2] Given a data point $\mathbf{b}_j$ and its K neigbors stored in the matrix $\mathbf{B}_j = [\mathbf{b}_{j_1} | \ldots \; |\mathbf{b}_{j_K}] \in \mathbb{R}^{\mathrm{N} \times \mathrm{K}}$, the $\mathrm{K} \times \mathrm{K}$ *local covariance matrix* is defined as $Cov_{loc}(\mathbf{B}_j) = \hat{\mathbf{B}}_j^{\mathrm{T}} \hat{\mathbf{B}}_j$ where $\hat{\mathbf{B}}_j = [\mathbf{b}_{j_1} - \mathbf{b}_j | \ldots \; |\mathbf{b}_{j_K} - \mathbf{b}_j]$.

3. Finally, an embedding cost function $E(\mathbf{Y})$ is minimized in order to obtain the low-dimensional global internal coordinates $\mathbf{y}_j \in \mathbb{R}^{\mathrm{M}}$, $\mathrm{N} \geqslant \mathrm{M}$, as

$$E\left(\mathbf{Y}\right) = \sum_{j \in \mathrm{J}} \left\| \mathbf{y}_j - \sum_{k=1}^{K} W_{jk} \mathbf{y}_{j_k} \right\|_2^2 . \tag{5.8}$$

The vectors $\mathbf{y}_j$ $\forall j$ are computed in the sense of a best reconstruction, by fixing the weights W_{jk}. The quadratic function in (5.8) is minimized by its bottom non-zero eigenvectors with respect to two constraints such that

$$\sum_{j=1}^{\mathrm{J}} \mathbf{y}_j = \mathbf{0} \qquad \text{and} \qquad \frac{1}{\mathrm{J}} \sum_{j=1}^{\mathrm{J}} \mathbf{y}_j \mathbf{y}_j^{\mathrm{T}} = \mathbf{I}_{\mathrm{M}} \tag{5.9}$$

in order to remove the freedom on translation and to avoid degenerate solutions respectively. Please see [134] for more information.

There always exists a local distance preserving LLE for a given high-dimensional dataset. However, LLE can not guarantee a one-to-one mapping. For more information on LLE and its variants, please refer to [137, 138, 139].

5.2 Image Prediction based on Neighbor Embedding

5.2.1 Motivation

The use of sparse approximations in various image processing problems is motivated by the assumption that natural images are composed of only a few structural “primitives”. One has thus to first learn these primitives and then decompose the image (or its texture patches) on the set of primitives to extract the representative features of the image. In Chapter 3, we have considered dynamic and locally adaptive dictionaries formed by atoms derived from texture patches present in a large causal neighborhood of the block to be predicted, and this in order to exploit self-similarities within the image. Note that the self-similarity property has been largely exploited and shown to be beneficial in a number of image processing tasks such as in the NLM approach used for image denoising in [140] or for texture synthesis in [11]. Image texture patches are therefore taken as atoms and stacked as columns in a matrix $\mathbf{A}$ called the dictionary.

The proposed sparse prediction method (in Chapter 3) however suffers from the fact that iterative sparse approximation methods, like MP and OMP, trying to approximate residues at each iteration (except for the first one) is not well suited to the chosen dictionary with positive atoms. In addition, they turn out to be not well suited to our problem since the residue on the approximation support is not well correlated to the residue on the block to be predicted, therefore the approximation vectors are likely to be quite different. The idea instead to consider other approximation methods which would "directly" find the best combination of the input non-negative "texture primitives" (the atoms in the dictionary) to approximate the input non-negative patch. This problem falls naturally within both frameworks of NMF and LLE.

In NMF, the weights of the approximation vector are forced to be non-negative, in order to construct parts-based representation of objects [129] or texture patches in an additive manner, and leading to prediction vectors which will also be non-negative to approximate an input non-negative vector. Similarly, for LLE, the goal is to reconstruct each input texture patch from its K nearest neighbors. The constraint that the sum of the weights is equal to "1" forces the reconstruction of each data point (here input patch) to lie in the subspace spanned by its nearest neighbors.

5.2.2 Problem statement

We now turn our attention to the image prediction problem. Let S denote a region in the image containing a block B of $n \times n$ pixels and its causal neighborhood C used as the approximation support (template) as shown in Fig. 5.1. In Fig. 5.1 the region S contains 5 blocks, hence is of size $\mathrm{N} = 5n^2$ pixels for running the prediction algorithm. However, as introduced in the previous chapter(s), different forms of approximation support C can also be considered (please see Fig. 5.2). In any case, in the region S there are known pixel values (the support region C) and unknown samples (the values of the pixels of the block B to be predicted). The principle of the proposed prediction approach here is to first search for a good approximation for the known pixels in C and keep the same parameters (e.g., weighting coefficients) to approximate the unknown pixel values in B using the data dimensionality reduction methods, i.e., NMF and LLE, as described above.

Let the N sample values of the region S be stacked in a column vector $\mathbf{b}$ (assuming the unknown values in B are equal to zero). The vector $\mathbf{b}$ is then compacted in $\mathbf{b}_c$ of size

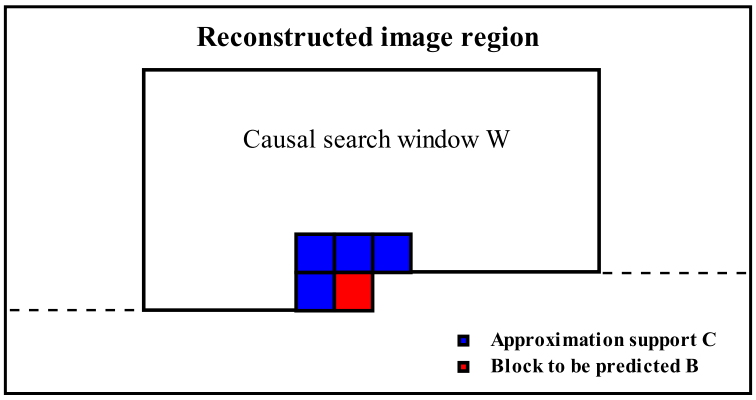

Reconstructed image region
Causal search window W
🟦 Approximation support C
🟥 Block to be predicted B

Figure 5.1: Dictionary construction - C is the approximation support of the current $n \times n$ block to be predicted B. W is the search window from which texture patches are extracted to construct the dictionary $\mathbf{A}$ to be used for the prediction of B.

$4n^2$ known (observed) values, thus $\mathbf{b}_c$ corresponds to the support region (template) C organized in a vector form, and assume that the vectors $\mathbf{b}_t$ and $\hat{\mathbf{b}}_t$ represent the actual (original) and the predicted values of the block B respectively. Note that the original pixel values of the block to be predicted B are known at the encoder, but they are not known at the decoder.

Let $\mathbf{A}$ denote the so-called *dictionary* represented by a matrix of dimension $\mathrm{N} \times \mathrm{M}$, where $\mathrm{N} \leqslant \mathrm{M}$. In all prediction methods presented below (TM, SP, NMF, and LLE), the columns of the “locally adaptive” dictionary $\mathbf{A}$ is constructed by stacking the luminance values of all patches (having the same geometric shape as S) in a given causal search window W in the reconstructed image region as shown in Fig. 5.1. The use of the causal window here guarantees that the decoder can construct exactly the same dictionary. The dictionary matrix $\mathbf{A}$ is then assumed to be formed by two submatrices (subdictionaries) $\mathbf{A}_c$ and $\mathbf{A}_t$ as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_c \\ \mathbf{A}_t \end{bmatrix} \tag{5.10}$$

where the subdictionary $\mathbf{A}_c$ of size $4n^2 \times \mathrm{M}$ corresponds to the spatial location of the pixels of the area C (or equivalently to the vector $\mathbf{b}_c$), and the other submatrix $\mathbf{A}_t$ (of size $n^2 \times \mathrm{M}$) as the spatial subdictionary corresponding to the spatial location of the pixels of the area B to be predicted, or equivalently to the vector $\mathbf{b}_t$.

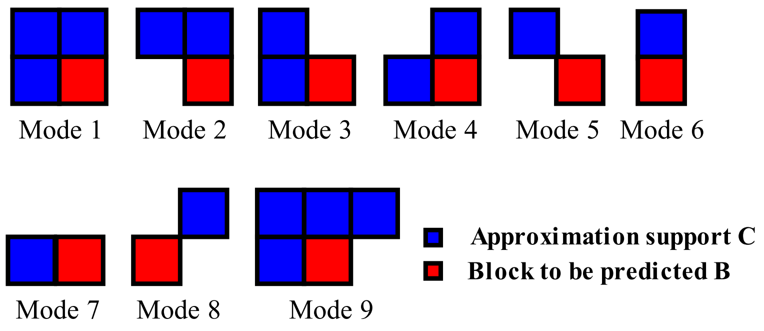


Figure 5.2: Nine possible forms for approximation support - The optimum support type is selected according to a given criterion. This selection of the best approximation support according to either the SSE or an RD criterion allows improving the prediction or compression performance.

The following part of this section describes how NMF and LLE algorithms can be adapted and then applied to the image prediction problem with some sort of “sparsity” constraint as in the sparse prediction case.

5.2.3 Image prediction based on NMF

Given a fixed non-negative dictionary $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ formed by texture patches, as explained above, and the (non-negative) data vector $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$, the underlying basic idea is to first obtain an NMF representation $\mathbf{x}$ of the support region C and keep the same representation parameters (i.e., weighting coefficients in $\mathbf{x}$) to approximate the unknown pixel values in the block to be predicted B. The non-negativity constraints are satisfied for both $\mathbf{A}$ and $\mathbf{b}$, similarly for $\mathbf{A}_c$ and $\mathbf{b}_c$, since the values in the spatial domain range between 0 and 255.

5.2.3.1 The algorithm

Assuming the dictionary matrix $\mathbf{A}_c$ is fixed for the data vector $\mathbf{b}_c$, the NMF formulation for the representation vector $\mathbf{x}$ of $\mathbf{b}_c$ can be written as

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} \left[\frac{1}{2}\left\|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\right\|_2^2\right] \quad \text{subject to} \quad \mathbf{x} \geqslant \mathbf{0}, \tag{5.11}$$

and the *multiplicative* update equation for $\mathbf{x}$ becomes

```
**Inputs:** $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, $T$
**Output:** Predicted values of unknowns $\hat{\mathbf{b}}_t$
**Initialization:** $t = 0$, $\mathbf{x} \geqslant \mathbf{0}$
  **iterate** until $t = T$, or change in $\mathbf{x}$ is small
    $t = t + 1$
    $\mathbf{x} \leftarrow \mathbf{x} \otimes (\mathbf{A}_c^{\mathrm{T}} \mathbf{b}_c) \oslash (\mathbf{A}_c^{\mathrm{T}} \mathbf{A}_c \mathbf{x} + 10^{-9})$
  end **iterate**
  $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}$
```

Table 5.1: Image prediction algorithm based on NMF - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

$$x_m \leftarrow x_m \frac{(\mathbf{A}_c^{\mathrm{T}} \mathbf{b}_c)_m}{(\mathbf{A}_c^{\mathrm{T}} \mathbf{A}_c \mathbf{x})_m + \varepsilon}, \quad m = 1...\mathrm{M}. \tag{5.12}$$

The representation vector $\mathbf{x}$ is initialized with random non-negative values. In order to obtain the optimum weighting vector $\mathbf{x}_{opt}$, the update equation in (5.12) is iterated until a pre-defined iteration number T is reached, or the total change in the elements of the vector $\mathbf{x}$ is very small between two consecutive iterations. The prediction signal $\hat{\mathbf{b}}_t$ is then calculated by multiplying the dictionary matrix $\mathbf{A}_t$ by $\mathbf{x}_{opt}$ as $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}_{opt}$. The NMF based image prediction algorithm is summarized in Table 5.1. Note here that, as in the sparse prediction case, the dictionary matrices $\mathbf{A}_c$ and $\mathbf{A}_t$ are updated adaptively (per block) depending on the spatial location of the block to be predicted B.

The above proposed approach actually leads to a very efficient compression performance (a PSNR gain of up to 3 dB when compared with TM and SP, please see the experimental results section). Furthermore, it does not require sending extra (side) information to the decoder, e.g., as the optimum iteration number k in the SP method. On the other hand, it suffers from high computational complexity, because of the size of the dictionary and the costly update equations since the dictionary is fixed, which makes difficult the use of several forms of approximation supports. Moreover, aggregating more than a certain number of image patches might lead to blurring effects on the prediction.

5.2.3.2 Adapted algorithm with a sparsity constraint

Here we push further the study of NMF based image prediction. To reduce the complexity of this promising approach, in addition to non-negativity, let us now impose some sort of additional sparsity constraint by limiting the number of used atoms (columns) in the dictionary. Note that NMF can be solved with strict ℓ_0–norm (or ℓ_1–norm) sparsity constraints by using methods such as the local NMF (LNMF) algorithm [141], or the method based on the projected gradient descent algorithm, which keeps the ℓ_2–norm unchanged and sets the corresponding ℓ_1–norm to desired sparsity, as described in [136]. However, these sparse NMF algorithms, incorporating additional constraints, are known to require more iterations to converge, compared to the standard NMF algorithm, which is already quite time consuming. Note that here "sparsity" comes instead from the fact that the input patch will be approximated by a linear combination of K nearest neighbors. The dictionary is thus kept fixed and the number of atoms used is limited to K texture patches, hence the parameter K controls the sparsity of the data approximation. Note here that this constraint exists implicitly in LLE.

The main idea explored here is again to search for a linear combination of K nearest neighbor texture patches (taken from a causal window in the image) to approximate the known pixel values in the approximation support C, and then keep the same weighting coefficients to predict the unknown pixels in B as a linear combination of the co-located pixels in these K neighbor patches. The K-NN distance is computed between the approximation support of the block to be predicted and the co-located pixels of the reference patches in the causal window of the image, so that the decoder can similarly find the same K neighbors.

The proposed sparsity constraint can directly be imposed into the optimization (5.11) by limiting the number of non-zero coefficients in $\mathbf{x}$, i.e., that is the same with limiting the number of used atoms in $\mathbf{A}_c$. The atom selection is done iteratively by choosing k, $k = 1...K$, patches which are close to $\mathbf{b}_c$ in the Euclidean distance. At the k^{th} iteration the algorithm identifies k atoms $\left[\mathbf{a}_{c_{m_1}}, ..., \mathbf{a}_{c_{m_k}}\right]$ in $\mathbf{A}_c$, and let $\mathbf{A}_c^k$ denote the compacted matrix containing all the selected atoms. One then solves

$$\min_{\boldsymbol{\alpha}_k} \left[\frac{1}{2}\left\|\mathbf{b}_c - \mathbf{A}_c^k \boldsymbol{\alpha}_k\right\|_2^2\right] \quad \text{subject to} \quad \boldsymbol{\alpha}_k \geqslant \mathbf{0} \tag{5.13}$$

by updating the randomly initialized non-negative elements of $\boldsymbol{\alpha}_k$ as

```
Input: $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, $K$, $T$
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
Initialization: $k = 0$, $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{A}_c^0 = [\ ]$
  repeat until $k = K$
    $k = k + 1$
    $m_k = \arg\min_m \|\mathbf{b}_c - \mathbf{a}_{c_m}\|_2^2$ where $m \notin \{m_1, ..., m_{k-1}\}$
    $\mathbf{A}_c^k = \mathbf{A}_c^{k-1} \cup \{\mathbf{a}_{c_{m_k}}\}$
    initialize $t = 0$, $\boldsymbol{\alpha}_k \geqslant \mathbf{0}$
    iterate until $t = T$, or change in $\boldsymbol{\alpha}_k$ is small
      $t = t + 1$
      $\boldsymbol{\alpha}_k \leftarrow \boldsymbol{\alpha}_k \otimes \left(\mathbf{A}_c^{k^{\mathrm{T}}} \mathbf{b}_c\right) \oslash \left(\mathbf{A}_c^{k^{\mathrm{T}}} \mathbf{A}_c^k \boldsymbol{\alpha}_k + 10^{-9}\right)$
    end iterate
    $\mathbf{x}_k \leftarrow \boldsymbol{\alpha}_k$
    $\mathbf{p}_k = \mathbf{A}_t \mathbf{x}_k$
  end repeat
Select the optimum $k$ minimizing the selected criterion
Set $\hat{\mathbf{b}}_t = \mathbf{p}_{k_{opt}}$
```

Table 5.2: Image prediction algorithm based on NMF with a sparsity constraint - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

$$\alpha_{k_a} \leftarrow \alpha_{k_a} \frac{\left(\mathbf{A}_c^{k^{\mathrm{T}}} \mathbf{b}_c\right)_a}{\left(\mathbf{A}_c^{k^{\mathrm{T}}} \mathbf{A}_c^k \boldsymbol{\alpha}_k\right)_a + \varepsilon}, \quad a = 1...k, \tag{5.14}$$

where $\boldsymbol{\alpha}_k$ contains the weighting coefficients of all the atoms selected in the k^{th} iteration. Initially $\mathbf{x}_0 = \mathbf{0}$, and the optimal weights calculated in $\boldsymbol{\alpha}_k$ are introduced into $\mathbf{x}_{k-1}$ to yield $\mathbf{x}_k$. Notice that here $\mathbf{x}_k$ is the sparse vector of coefficients at the k^{th} iteration and all the coefficients assigned to the selected atoms are recomputed at each iteration, like in the case of SP.

The algorithm at the encoder iterates until the number K is reached with an increasing number of $k, k = 1...K$, by keeping track of the SSE or the RD cost function values obtained for the block to be predicted $\mathbf{b}_t$, and finally selects the optimum number k_{opt} of *atoms* (i.e., image patches used) which minimizes the considered criterion, leading to an "optimum" sparse vector denoted $\mathbf{x}_{k_{opt}}$. The value of the number k_{opt} is then transmitted to the decoder which can similarly search for the same NMF approximation of the approximation support with the signaled number of atoms. In order to obtain the optimum weighting vector $\boldsymbol{\alpha}_k$, the update equation in (5.14) is iterated until a pre-defined iteration number T is reached, or the total change in the elements of the vector $\boldsymbol{\alpha}_k$ is very small between two consecutive iterations $t-1$ and t, $t = 1...T$. The predicted signal $\hat{\mathbf{b}}_t$ is then calculated by multiplying the dictionary $\mathbf{A}_t$ by $\mathbf{x}_{k_{opt}}$ as $\hat{\mathbf{b}}_t = \mathbf{A}_t\mathbf{x}_{k_{opt}}$. The complete NMF based image prediction algorithm with the sparsity constraint is summarized in Table 5.2.

5.2.4 Image prediction based on LLE

The LLE algorithm is here first applied (partially; steps 1 and 2) on the approximation support C. One thus searches for an approximation of the support region by a linear combination of its nearest neighbor image patches within the search window W, and then keeps the same weighting coefficients in the linear combination of the co-located pixels in order to estimate the unknown values of the block to be predicted B. In terms of LLE, our prediction problem can be written as

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} \|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\|_2^2 \quad \text{subject to} \quad \sum_m x_m = 1. \tag{5.15}$$

A sparsity constraint has been imposed (also implicitly by LLE) into the problem by iteratively choosing $k, k = 1...K$, closest patches to $\mathbf{b}_c$ in the Euclidean space, similar to the NMF based approach. Suppose that $\mathbf{A}_c^k$ denotes the submatrix which contains the selected k atoms (texture patches) from $\mathbf{A}_c$ at the k^{th} iteration. The algorithm then tries to solve the constrained minimization as

$$\min_{\boldsymbol{\alpha}_k} \left\|\mathbf{b}_c - \mathbf{A}_c^k\boldsymbol{\alpha}_k\right\|_2^2 \quad \text{subject to} \quad \sum_{a=1}^{k} \alpha_{k_a} = 1, \tag{5.16}$$

and the optimal weighting coefficients in $\boldsymbol{\alpha}_k$ are computed as

$$\boldsymbol{\alpha}_k = \frac{\mathbf{D}_k^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{D}_k^{-1}\mathbf{1}} \tag{5.17}$$

where $\mathbf{D}_k$ denotes the local covariance matrix of the selected patches in $\mathbf{A}_c^k$ in reference to $\mathbf{b}_c$, and $\mathbf{1}$ is the column vector of ones. In practice, instead of an explicit inversion of the matrix $\mathbf{D}_k$, the linear system of equations $\mathbf{D}_k\boldsymbol{\alpha}_k = \mathbf{1}$ is solved, then the weights

```
**Input:** $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, $K$
**Output:** Predicted values of unknowns $\hat{\mathbf{b}}_t$
**Initialization:** $k = 0$, $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{A}_c^0 = [\,]$
  **repeat** until $k = K$
    $k = k + 1$
    $m_k = \arg\min_m \|\mathbf{b}_c - \mathbf{a}_{c_m}\|_2^2$ where $m \notin \{m_1, ..., m_{k-1}\}$
    $\mathbf{A}_c^k = \mathbf{A}_c^{k-1} \cup \{\mathbf{a}_{c_{m_k}}\}$
    $\mathbf{D}_k = Cov_{loc}(\mathbf{A}_c^k)$ in reference to $\mathbf{b}_c$
    Solve $\mathbf{D}_k \boldsymbol{\alpha}_k = \mathbf{1}$ for $\boldsymbol{\alpha}_k$
    $\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_k / sum(\boldsymbol{\alpha}_k)$ and $\mathbf{x}_k \leftarrow \boldsymbol{\alpha}_k$
    $\mathbf{p}_k = \mathbf{A}_t \mathbf{x}_k$
  end **repeat**
Select the optimum $k$ minimizing the selected criterion
Set $\hat{\mathbf{b}}_t = \mathbf{p}_{k_{opt}}$
```

Table 5.3: Image prediction algorithm based on LLE - The proposed prediction method has been drawn here for one approximation support, however, its extension to dynamic approximation supports is straightforward.

are rescaled so that they sum to one. Note that the sum-to-one constraint of weights forces the reconstruction of each data point (here the input patch) to lie in the subspace spanned by its neighbors. Since the input texture patch samples are all non-negative, the predicted block will also be non-negative. However, the calculated weights can be positive or negative.

Here also $\mathbf{x}_0$ is set to $\mathbf{0}$ initially, and the optimal weights calculated in $\boldsymbol{\alpha}_k$ are introduced into $\mathbf{x}_{k-1}$ to yield $\mathbf{x}_k$ by replacing the coefficients with the previous ones. All the coefficients assigned to the selected atoms are recomputed at each iteration.

The algorithm at the encoder keeps track of the SSE or the RD cost function values obtained for the block to be predicted, and finally selects the number k_{opt} of *used patches* which minimizes the considered criterion, leading to an "optimum" sparse vector $\mathbf{x}_{k_{opt}}$. The value of the optimum number k_{opt} is then transmitted to the decoder which can similarly search for the same LLE approximation of the approximation support with the signaled number. The predicted signal $\hat{\mathbf{b}}_t$ is then calculated by multiplying the dictionary $\mathbf{A}_t$ by $\mathbf{x}_{k_{opt}}$ as $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}_{k_{opt}}$. The complete LLE based image prediction algorithm is summarized in Table 5.3.

5.3 Experimental Results

5.3.1 Primary results with NMF

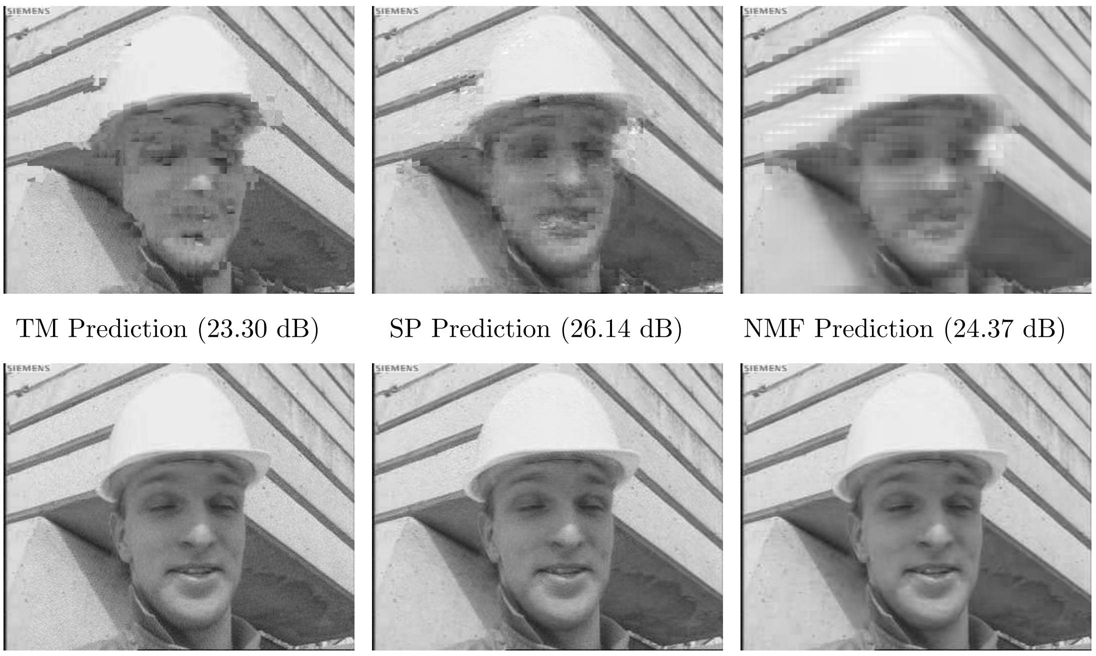
Our primary setup assesses comparatively the proposed NMF based image prediction scheme (Table 5.1) to TM and SP methods in terms of prediction quality and encoding efficiency. In the experiments reported below, only one approximation support as shown in Fig. 5.1 has been used with blocks of size 8×8 pixels.

In order to initialize the prediction/compression process, the top 3 rows and left 3 columns of blocks of size 8×8 are intra encoded with JPEG standard. Once a block has been predicted with the respective prediction method (i.e., TM, SP, or NMF), the DCT transformed residual block is quantized, zig-zag scanned, and encoded with JPEG. The quality factor (q_f) is increased from 10 to 90 with a step size of 10. Image blocks are processed in a raster scan order, and the reconstructed image is obtained by adding the quantized residue to the prediction. The redundancy factor of the dictionary $\mathbf{A}$ is set to 1, i.e., $\eta = 1$.

The NMF based algorithm is iterated until the residual energy on the approximation support is very small, or the maximum allowed iteration number $T = 500$ is reached. The SP algorithm (using OMP) is limited to iterate at most $K = 8$ iterations, and the optimum iteration number k_{opt}, when transmitted to decoder, is Huffman encoded. The optimization criterion is set to the minimum SSE on the block to be predicted.

Fig. 5.3 shows the predicted and reconstructed images of the NMF based prediction algorithm for Foreman in comparison to TM and SP based methods at $q_f = 30$. Fig. 5.4 demonstrates the prediction and encoding performance curves for Foreman, Cameraman, and Barbara images. Both visually and in terms of PSNR/bit-rate, the quality of the reconstructed image is significantly improved when compared with TM and SP based methods. A gain of up to 3 dB has been achieved in terms of encoding efficiency.

It is experimentally observed that although the prediction quality of SP has better results, the irregularities (non-smoothness) of the residual signals lead to higher bit-rates with lower PSNR values in the reconstructed images when compared to NMF prediction. The reason is that, especially in the case of signal prediction with a support region approximation, the atoms may not approximate (span) residue signals –of the block to be predicted– very well even though the dictionary has been well adapted in



TM Prediction (23.30 dB) SP Prediction (26.14 dB) NMF Prediction (24.37 dB)

TM Reconstruction (31.29 dB/0.56 bpp) SP Reconstruction (32.63 dB/0.53 bpp) NMF Reconstruction (33.68 dB/0.46 bpp)

Figure 5.3: (Top) Predicted and (bottom) reconstructed images of Foreman using NMF in comparison to template matching and sparse prediction - (left) Template matching, (middle) sparse prediction, and (right) non-negative matrix factorization methods.

the spatial domain. Furthermore, in terms of encoding, one needs to add the coding cost of the optimum iteration number k_{opt}. Finally, notice also here that the prediction of NMF is smooth (and blur) since there is no sparsity constraint on the number of used atoms (image patches) for the approximation, i.e., the algorithm aggregates all the patches which are taken from the search window W.

5.3.2 Application to image compression

Let us now turn our attention back to sparsity constrained NMF (Table 5.2) and LLE (Table 5.3) based image prediction algorithms. The impact of sparsity, non-negativity, and sum-to-one constraints has first been analyzed on the prediction quality. The proposed methods have then been evaluated in a complete image (or intra frame) coding scheme with both strict and relaxed sparsity constraints controlled by the parameter

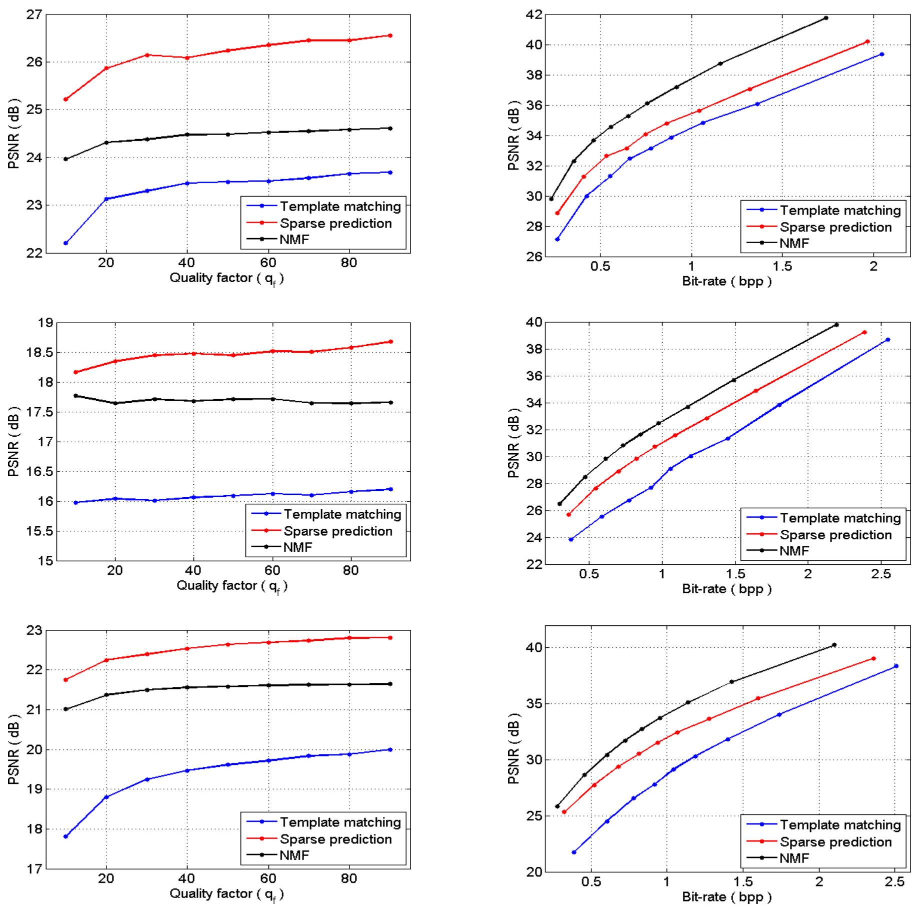


Figure 5.4: (Left) Prediction and (right) encoding performance curves of (top) Foreman, (middle) Cameraman, and (bottom) Barbara using NMF in comparison to template matching and sparse prediction - Performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE.

$k \in [1, K]$. A detailed analysis has been carried out on the prediction quality and the encoding PSNR/bit-rate efficiency in comparison to TM, SP and also H.264/AVC intra image prediction modes. The results obtained show a significant improvement in terms of rate-distortion gain of the reconstructed image, after coding and decoding the prediction residue (up to 2 dB, up to 1 dB, and up to 3 dB) when compared to H.264/AVC intra prediction modes, to SP, and to TM respectively. Further RD gains have been achieved with the LLE based prediction method by relaxing the sparsity constraints and setting the parameter k to 100, at the expense however of extra complexity.

5.3.2.1 Encoder structure

In order to initialize the prediction/compression process, the top 4 rows and left 4 columns of blocks of size 4×4 are predicted with H.264/AVC intra modes. Once a block has been predicted with the respective prediction method, the 4×4 residual block is encoded with an algorithm similar to JPEG. In this coding structure, a uniform quantization matrix with $\Delta = 16$ is weighted by $\mathbf{w}_{q_f}(q_f)$ (see (3.26)). Image blocks are processed in the raster scan order, and the reconstructed image is obtained by adding the quantized residue to the prediction. A skip mode has also been included into the encoder to avoid coding the blocks of prediction residue in which all the transformed and quantized coefficients are zero. The corresponding flag is arithmetically encoded.

Nine possible forms of *approximation supports* are considered as shown in Fig. 5.2. The optimum support mode, as well as the iteration number k for SP, and the number of used patches (also referred to here as k) for NMF and LLE, is selected according to two criteria: 1. minimization of the prediction signal SSE (i.e., $\min \|\mathbf{b}_t - \mathbf{A}_t\mathbf{x}\|_2^2$) in order to observe the impact on the prediction quality; 2. minimization of the RD cost function $J(\mathbf{D}, \mathbf{R})$ in order to observe the impact on the encoding efficiency, when the proposed prediction scheme is used in the compression algorithm.

Note that the optimization is done in two steps, i.e., first for the selection of iteration number k in the case of SP, or the optimal number of used patches k in the case of NMF and LLE based methods, and then for the selection of the optimum support mode type. In terms of encoding, one needs to add the encoding cost of the side information, here it is the k value and the support mode type. This information is signaled to the decoder using Huffman codes.

Fig. 5.5 shows the configuration of the search window for 4×4 block size that is used in the simulations reported below. All possible image patches in the search window are extracted to construct the dictionary matrix $\mathbf{A}$, i.e., that is, by taking in its columns all possible texture patches (in a vectorized form) with a mask shifted by one pixel horizontally and vertically in the search window W.

5.3.2.2 Impact of sparsity constraint and quantization noise

In the prediction methods described above, the dictionary $\mathbf{A}$ is constructed by stacking the luminance values of all patches (having the same geometric shape as S) in a given

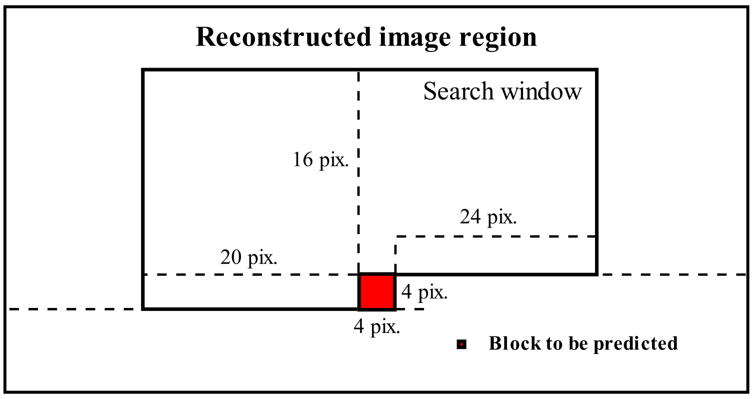


Figure 5.5: The configuration of the search window used in the simulations - The configuration of the search window for 4×4 block size. All possible image patches are extracted from the search window in order to construct the dictionary $\mathbf{A}$.

causal search window W in the reconstructed image region. When using those methods which are mainly based on approximations of a support region, a good approximation of the support region does not necessarily lead to a good approximation of the unknown pixel values as the approximation support and the unknown region pixels may have different characteristics. Furthermore, in an image coding context, at the decoder side the approximation support C, as well as the image patches stored in the dictionary $\mathbf{A}$, may not be noise-free depending on the prediction quality and the residue signal quantization. Therefore, it is crucial to analyse and to optimize the prediction quality of the unknown pixel values as a function of sparsity and the quantization noise. Here we briefly analyse the effect of the sparsity constraint on TM, SP, NMF, and LLE based prediction methods in the case where the image signals (i.e., blocks) are corrupted by various levels of quantization noise, i.e., both the region C and the columns of $\mathbf{A}$ contain quantization noise.

Fig. 5.6 demonstrates the mean prediction PSNR performance curves obtained for Barbara and Foreman images with a varying sparsity constraint $k \in [1, 50]$ in the case where image signals are clean (i.e., image residue blocks are encoded with a lossless encoder/decoder structure), or corrupted by a low, a medium, and a high level of quantization noise (i.e., image residue blocks are encoded in a lossy manner with the quantization scheme described in the previous subsection). The quality of the predicted signal, in terms of average PSNR, is significantly improved (up to 1.3 dB) with the

NMF based prediction method when compared to TM and SP, even in the presence of high quantization noise. In the LLE and SP based prediction methods, the mean prediction PSNR has its maximum when $k \in [1, 10]$ whereas NMF based prediction has its maximum PSNR when $k \in [1, 20]$. Notice here that the mean performance curves of the NMF based method look noisy because of the randomization of the weighting coefficients. For each simulation point k, the vector $\boldsymbol{\alpha}_k$ is initialized with the same seed of Mersenne twister random number generator.

Another interesting point might be further relaxing the sparsity constraint k, which will increase the computational complexity however, for the sake of a complete analysis of the proposed methods' global characteristics as a function of sparsity and quantization noise. Fig. 5.7 shows the mean prediction PSNR performance curves obtained for Barbara and Foreman images for $k \in [51, 100]$. The mean prediction performance for the LLE based method increases with the number k of neighbors used in the algorithm. In this method, the prediction performance first goes up and then drops (this behavior is similar to the one of NMF and SP for lower values of k). But it finally goes up again with increasing values of k. The number k of neighbors is indeed a parameter which is known to impact significantly the accuracy of the linear data approximation in LLE. Several methods [142, 143] have been proposed in the literature to find the optimum value of k for LLE. The authors in [142] show that the reconstruction error (computed on face images) first decreases with k and then, as k grows, the reconstruction error increases and then starts to alternate (going up and down). One explanation of these authors is that the Euclidean distance is not a good measure of proximity of data points on the underlying manifold. This is even more the case for the prediction problem, since this measure is used to determine the neighbors of the approximation support data point, and these neighbors may not be the neighbors of the block to be predicted.

On the other side, adding more neighbors has negative effect on the performance of the NMF method. One reason here is that increasing the number of neighbors considered in the approximation leads to taking neighbors which are not similar to the block to be predicted and may then degrade the performance. Similar to the SP method, the best approximation of the template does not necessarily lead to a good approximation of the block to be predicted. Hence the idea of running the approximation for several values of k and then keeping the one which leads to the lowest approximation error, or

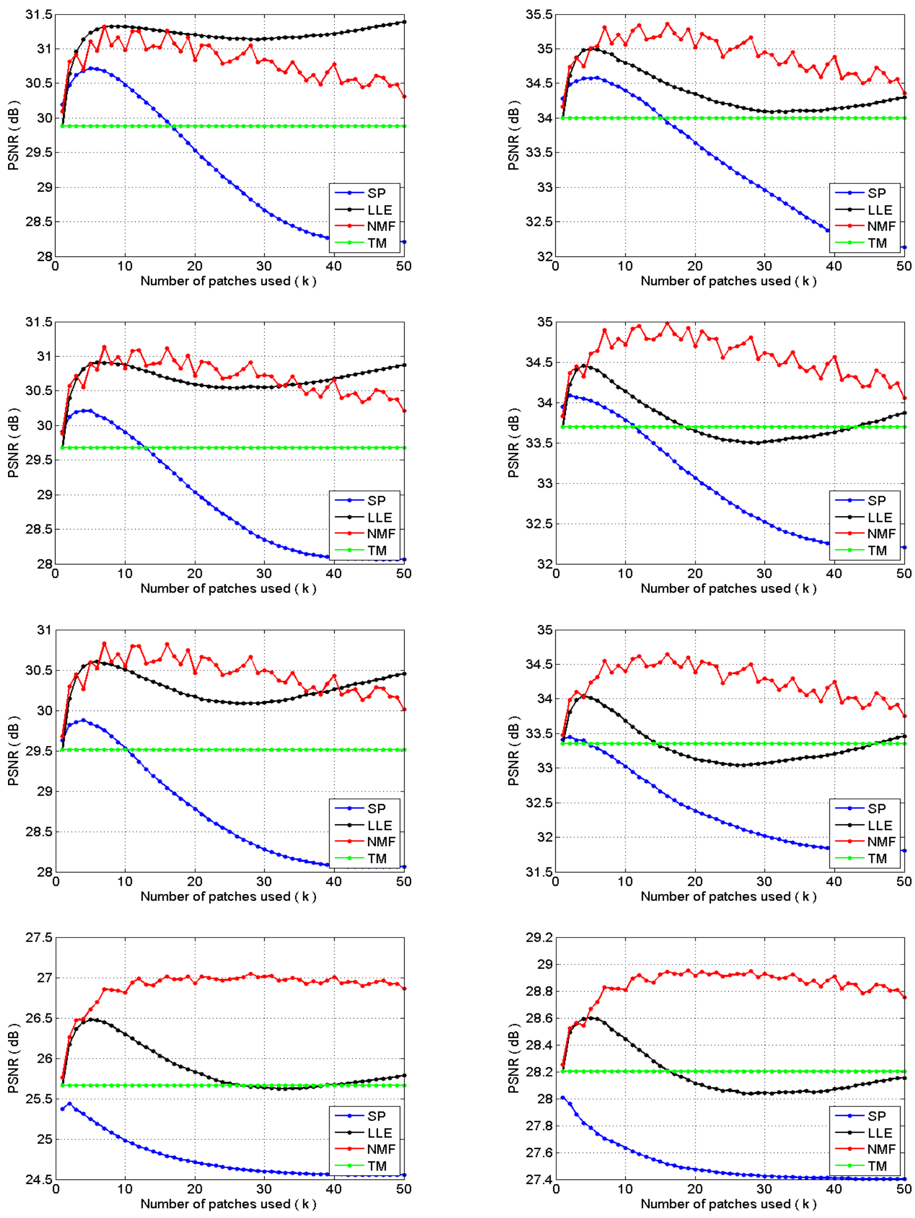


Figure 5.6: Mean prediction PSNR versus sparsity performance curves of (left) Barbara and (right) Foreman for the unknown block using TM, SP, NMF, and LLE based image prediction algorithms - With (top–row) none, (second–row) low ($q_f = 70$), (third–row) medium ($q_f = 50$), and (bottom–row) high ($q_f = 10$) level of quantization noise corruption. Block size is 4×4 pixels, the approximation support Mode 1 has been used as shown in Fig 5.2, and $k \in [1, 50]$.

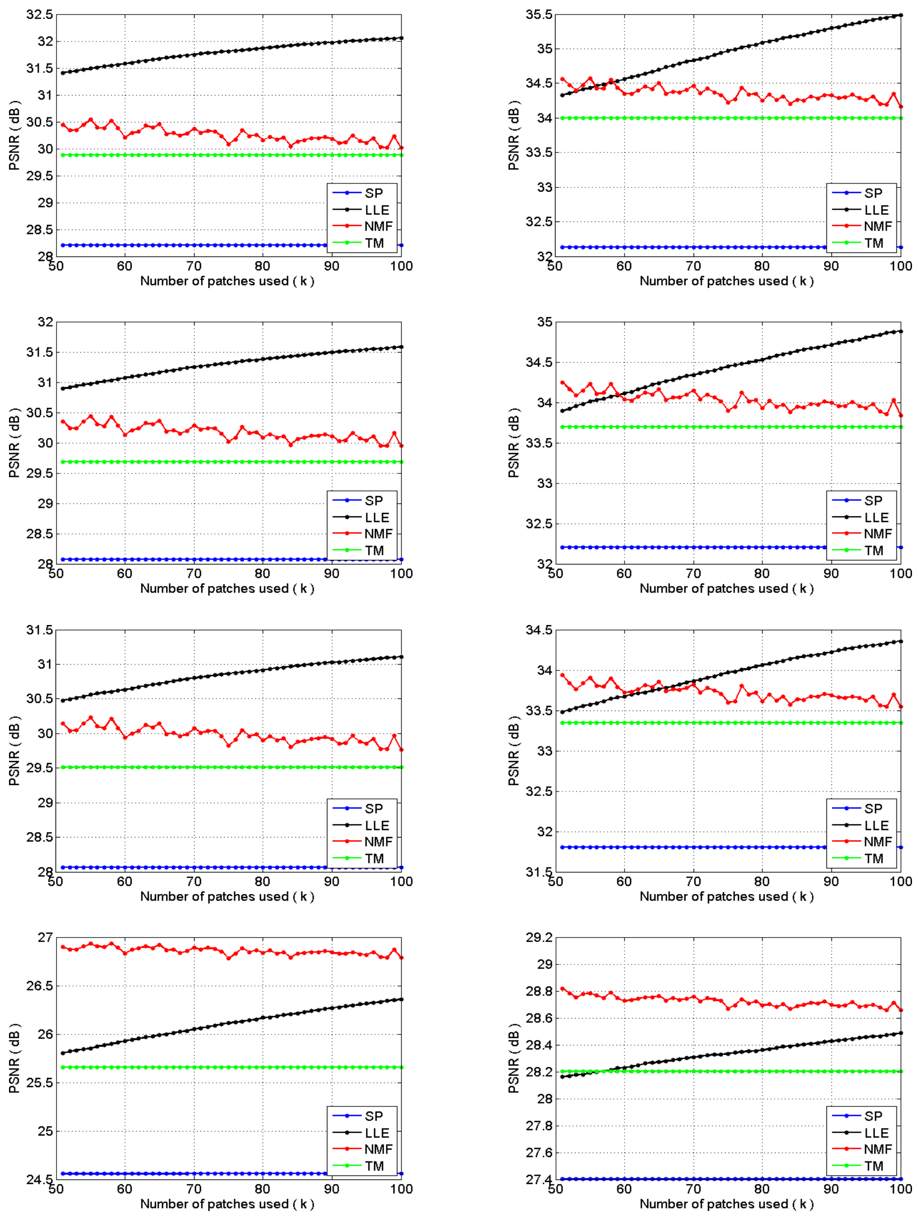


Figure 5.7: Mean prediction PSNR versus sparsity performance curves of (left) Barbara and (right) Foreman for the unknown block using TM, SP, NMF, and LLE based image prediction algorithms - With (top–row) none, (second–row) low ($q_f = 70$), (third–row) medium ($q_f = 50$), and (bottom–row) high ($q_f = 10$) level of quantization noise corruption. Block size is 4×4 pixels, the approximation support Mode 1 has been used as shown in Fig 5.2, and $k \in [51, 100]$.

to the best RD value, on the block to be predicted. Note here SP stops iterating for $k > 48$ since the template Mode 1 has 48 known pixels in C for 4×4 block size. At iteration $k = 48$, the OMP algorithm constructs a complete orthogonal basis for the known values in C and the algorithm stops.

5.3.2.3 Impact of non-negativity and sum-to-one constraints

This subsection aims at analyzing the effect of the non-negativity and sum-to-one constraints imposed in the NMF and LLE frameworks respectively. The problems to be solved as formulated in (5.13) and (5.16) are indeed least squares problems with different constraints and which are solved with different optimization methods.

We have first considered comparing the prediction and encoding (RD) performance of the methods with sum-to-one constraint (with sparsity) when solving this constrained least squares problem with two approaches: the Lagrangian resolution of the problem whose solution is given by (5.17) in the LLE based approach, and the resolution of the problem using quadratic programming [144, 145]. The two corresponding curves given in Fig. 5.8 and Fig. 5.9, for prediction and RD performance results respectively, show that the two optimization methods lead to comparable results.

We have then considered the least squares problem with the non-negativity constraint (as imposed in the NMF framework) plus the sum-to-one constraint (as imposed in the LLE approach), with a sparsity constraint controlled via the number k, $k = 1...K$, of neighbors considered for the reconstruction of each patch. This problem has then been solved using quadratic programming. One can see from the plots (Fig. 5.8 and Fig. 5.9) that adding the non-negativity constraint to the sum-to-one constraint with sparsity tends to degrade the prediction performance. Actually, as explained in [146], adding a non-negativity constraint to (5.16), that is having the two constraints (non-negativity and sum-to-one) forces the reconstruction of each data point (input patch) within the convex hull of its neighbors. This additional constraint is known to degrade the reconstruction of data points not belonging to the convex hull of their neighbors. Since, here the nearest neighbors are determined by computing distances between templates (approximation supports) but not directly on the block to be approximated, one can assume that many input patches that we aim at approximating (the blocks to be predicted) are not belonging to the convex hull defined by the found k nearest neighbors.

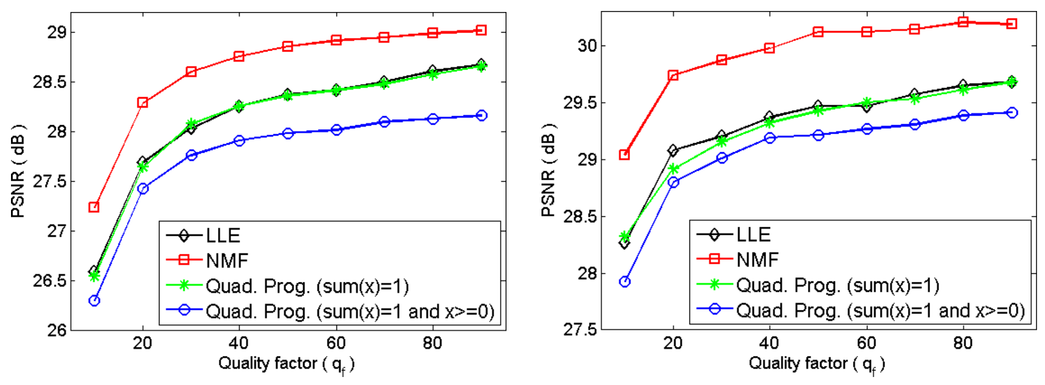


Figure 5.8: Prediction PSNR evaluation of (left) Barbara and (right) Foreman images with different optimization constraints - The sum-to-one constraint (with sparsity) with two optimization methods (Lagrangian resolution as in LLE and quadratic programming), and the three constraints (sparsity, sum-to-one, non-negativity) with quadratic programming. The quality factor for encoding the prediction residue has been varied form 10 to 90, and the block size is 4×4.

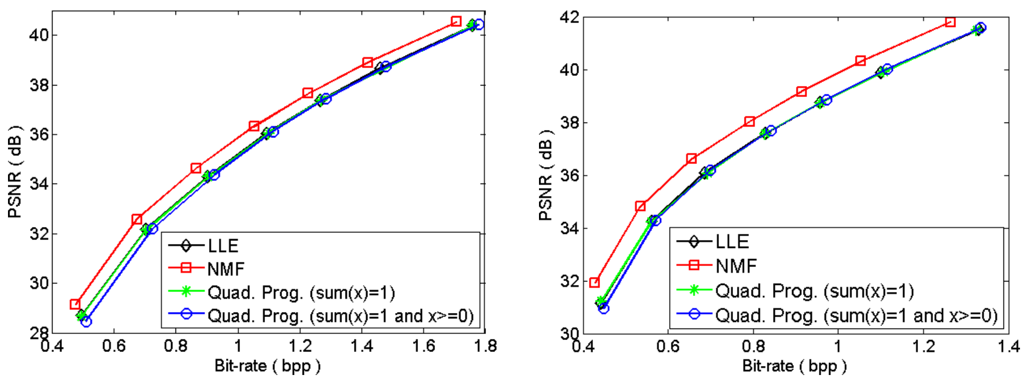


Figure 5.9: Encoding performance evaluation of (left) Barbara and (right) Foreman images with different optimization constraints - The sum-to-one constraint (with sparsity) with two optimization methods (Lagrangian resolution as in LLE and quadratic programming), and the three constraints (sparsity, sum-to-one, non-negativity) with quadratic programming. The quality factor for encoding the prediction residue has been varied form 10 to 90, and the block size is 4×4.

5.3.3 Experimental setup

Five test images have been chosen for the simulations as shown in Fig. 5.10. Foreman can be regarded as the image which has mainly diagonal edges and smooth regions. Barbara contains a combination of smooth and textural regions as well as edges. Roof

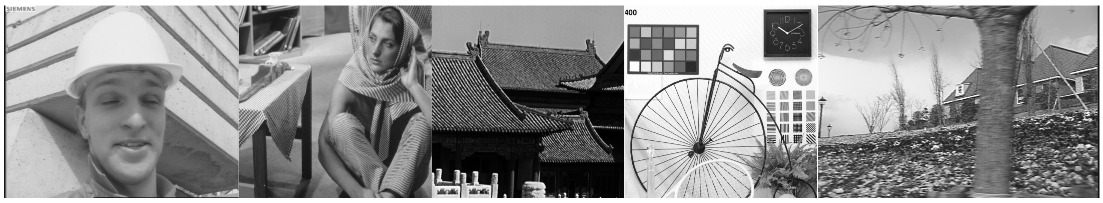


Figure 5.10: Test images used in the simulations - (a) Foreman (CIF), (b) Barbara (512×512), (c) Roof (512×512), (d) Bike (512×512), and (e) Flowergarden (704×480).

contains highly textural regions. Bike and Flowergarden can be regarded as the images composed of different structures and high frequencies. In all tested prediction methods (TM, SP, NMF, LLE), the dictionary $\mathbf{A}$ is constructed by taking in its columns all possible texture patches (shifted by "1" pixel horizontally and vertically) in the search window (see Fig. 5.5 for the search window configuration).

Two sets of tests have been carried out. The first one makes use of approximations with strict sparsity constraints, in the sense that the number of used patches considered in the k-NN search, or equally the number of iterations k, is varied from 1 to 8, i.e., $K = 8$. The optimal k value k_{opt} in terms of selected criterion (i.e., either the SSE or the RD) is then signaled to the decoder. The second set of experiments relaxes the sparsity constraint, and takes $k = K = 100$. In this case, there is no need to signal the value of k since it is fixed and assumed to be known also at the decoder.

In the NMF based prediction approach, the maximum allowed iteration number T is set to 100 (see Table 5.2). However, we have experimentally observed that the total change in the coefficients of $\boldsymbol{\alpha}_k, k = 1...K$, gets very small for $t > 5$ in the smooth areas, and for $t > 50$ in the edgel areas. In the highly textural areas the algorithm iterates up to $t = T = 100$ iterations.

5.3.4 Prediction performance with SSE criterion

Fig. 5.11 and Fig. 5.12 illustrate visually the prediction performance for the test images, an edgel region in Foreman and a textural region in Barbara images respectively. The optimization criterion is the minimization of the predicted block SSE. The prediction residue has been encoded with a quality factor $q_f = 10$ (i.e., at low bit-rates). Therefore, the template (approximation support) of the block to be predicted as well as the patches

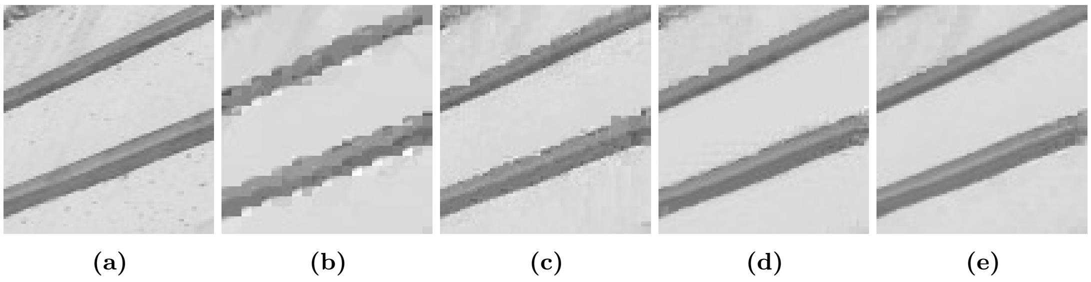


Figure 5.11: Predicted images of an edgel region in Foreman at low bit-rates (q_f = 10) - (a) Original image, (b) H.264 intra modes, (c) SP, (d) LLE, and (e) NMF based prediction methods with the SSE criterion and a strict sparsity constraint $k \in [1, 8]$.

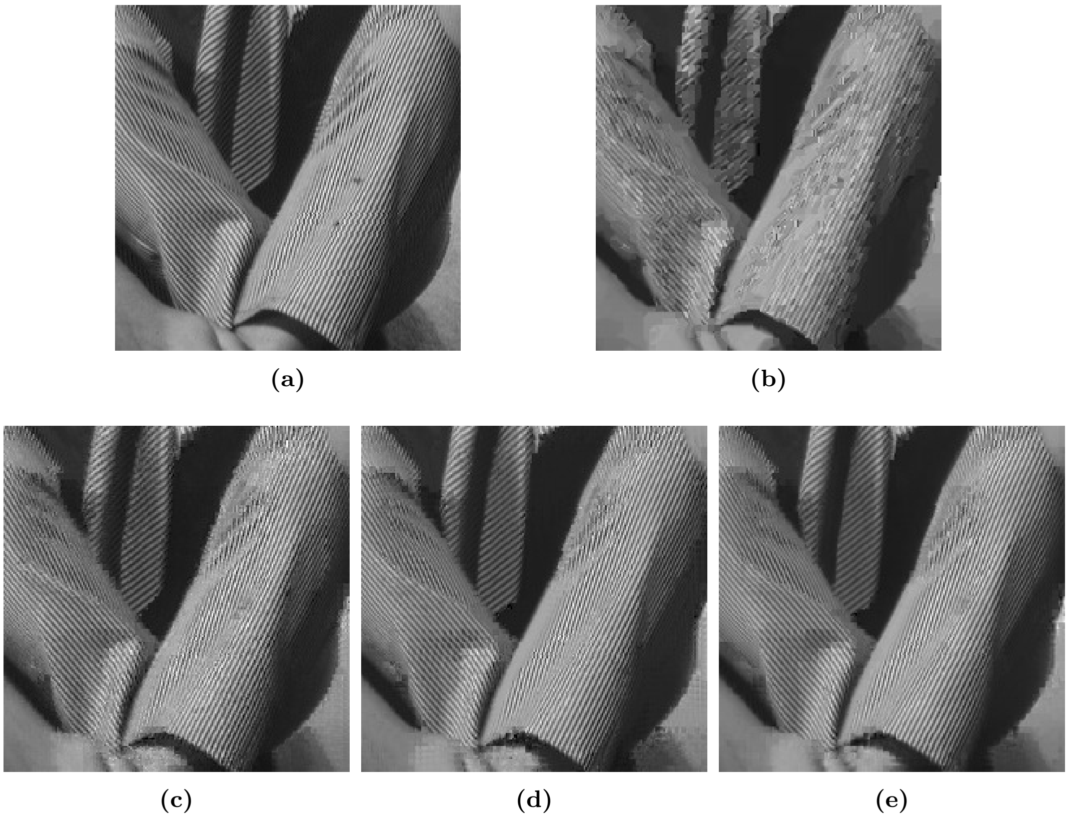


Figure 5.12: Predicted images of a textural region in Barbara at low bit-rates (q_f = 10) - (a) Original image, (b) H.264 intra modes, (c) SP, (d) LLE, and (e) NMF based prediction methods with the SSE criterion and a strict sparsity constraint $k \in [1, 8]$.

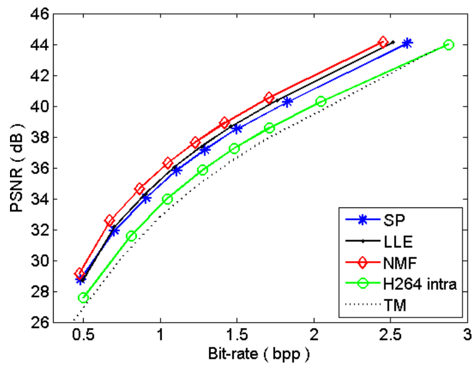
forming the dictionary contain quantization noise. A sparsity constraint has been used for these experiments, k is varied from 1 to 8 for the SP, NMF, and LLE based methods.

The predicted signal, in terms of visual quality, is significantly improved by LLE, and even further by the NMF based method when compared to SP and H.264/AVC based intra prediction especially for the image regions which contain complex textural structures and edges. Note here that the quantization noise present in both approximation support (template) and dictionary elements (patches), which is similar for all solutions tested since the quality factor, which is fixed for all methods to $q_f = 10$, does not significantly impact the performance of the prediction algorithms. The prediction images shown here, as said above, are obtained by encoding the residue with the same quality factor, however they do not lead to the same overall bit-rate since the residue obtained is not the same with all methods.

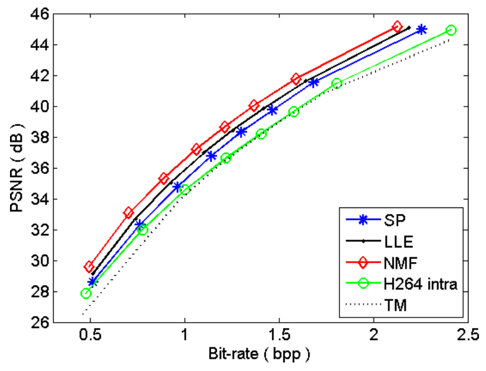
5.3.5 Encoding performance with RD criterion

Fig. 5.13 demonstrates the total encoding PSNR/bit-rate performance curves for the test images with sparsity constraints where the optimization criterion is the minimization of the RD cost function. One can observe that the proposed prediction methods with NMF and LLE improve the encoding performance of the images especially containing textural regions when compared to TM, SP, and H.264/AVC intra prediction. A gain up to 2 dB has been achieved by the NMF based method when compared to H.264/AVC, up to 1 dB in comparison with the SP method, and up to 3 dB in comparison to TM. Furthermore, the LLE based method outperforms the prediction methods including TM, SP, and H.264/AVC.

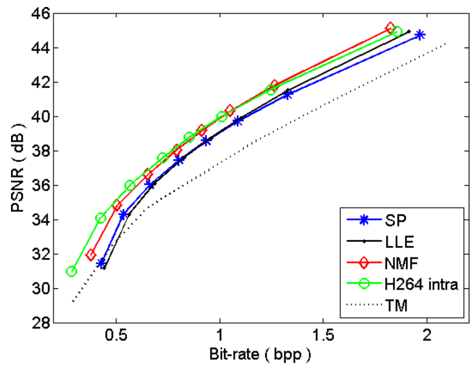
Notice that for H.264/AVC intra prediction, only the selected prediction mode (out of nine intra modes) is signaled to the decoder. For TM, the best template (among those shown in Fig. 5.2) is signaled to the decoder. For the other methods (i.e., SP, LLE, and NMF) the optimum number of used patches k (equally the iteration number in SP) is signaled in addition to the optimal approximation support type. Thus, the gain in prediction might not compensate the coding cost of an extra side information for the images (e.g., like Foreman) which contain mostly smooth regions, and especially directional contours which are highly aligned with the H.264/AVC intra prediction modes. H.264/AVC intra prediction works relatively well for this particular image but is outperformed by the other methods for the other test images.



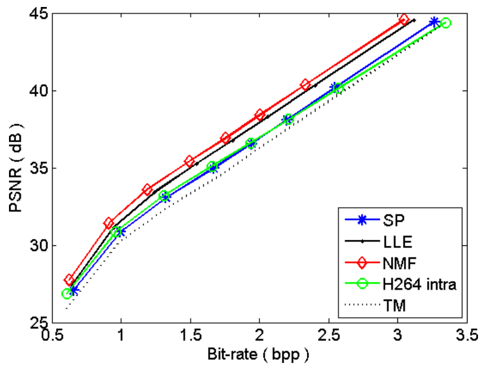
(a)



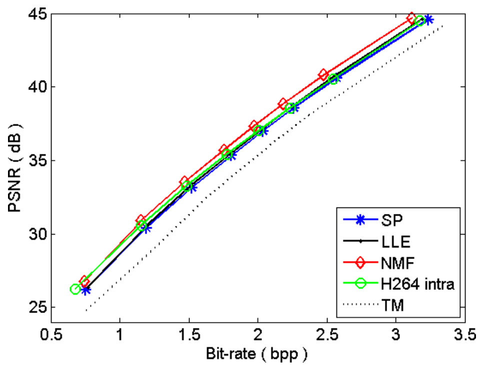
(b)



(c)



(d)

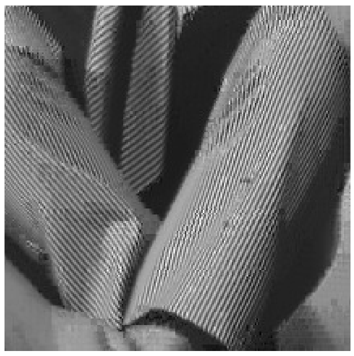


(e)

Figure 5.13: Encoding performance curves of (a) Barbara, (b) Roof, (c) Foreman, (d) Bike, and (e) Flowergarden using NMF and LLE with strict sparsity constraints (searching for the best k) - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the predicted signal and a sparsity constraint $k \in [1, 8]$

(a) 27.56 dB/0.50 bpp

(b) 28.69 dB/0.48 bpp


(c) 28.68 dB/0.49 bpp

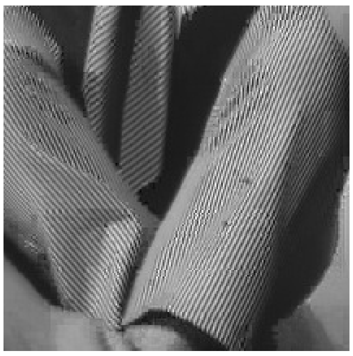
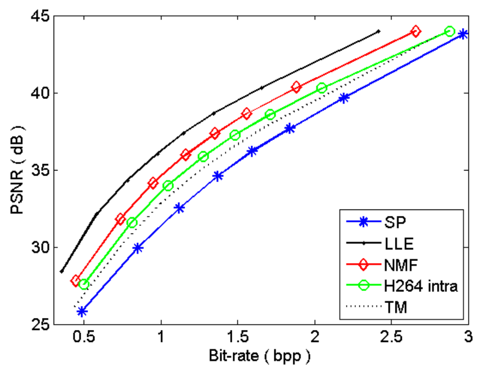

(d) 29.14 dB/0.48 bpp

Figure 5.14: Reconstructed images of a textural region in Barbara with strict sparsity constraints (searching for the best k) - (a) H.264 intra modes, (b) SP, (c) LLE, and (d) NMF based image prediction methods with the RD criterion and a sparsity constraint $k \in [1, 8]$.

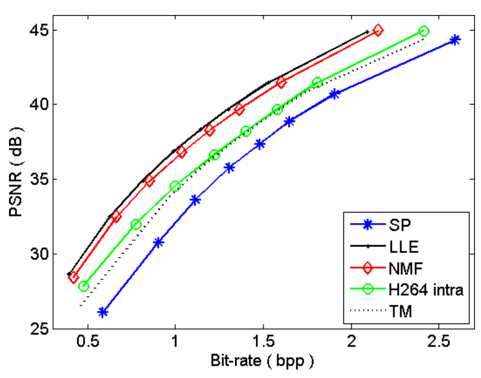
Fig. 5.14 shows a reconstructed textural region of Barbara image with the H.264/AVC intra, SP, LLE, and NMF based image prediction methods.

5.3.6 Relaxing the sparsity constraint with RD criterion

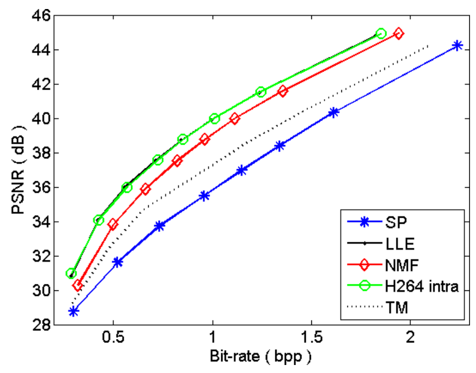
In all SP, NMF, and LLE based prediction methods, the sparsity constraint can be relaxed (or even fixed at the expense of a possible decrease in the prediction quality, please see Fig. 5.6 and Fig. 5.7) in order to save signalling cost of the number k of used atoms, i.e., texture patches. Fig. 5.15 shows the total encoding PSNR/bit-rate performance curves for the test images where the optimization criterion is the minimization of the RD cost function with $k = K = 100$ (which is fixed) for the NMF and LLE



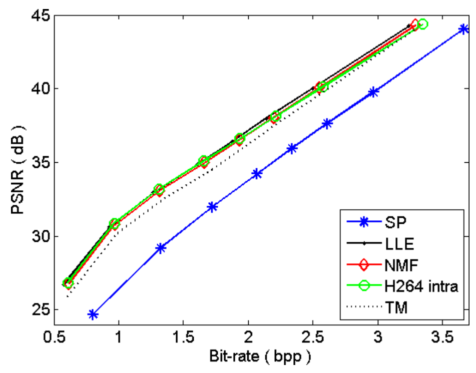
(a)



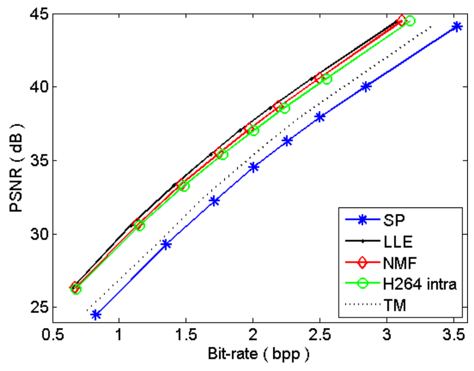
(b)



(c)

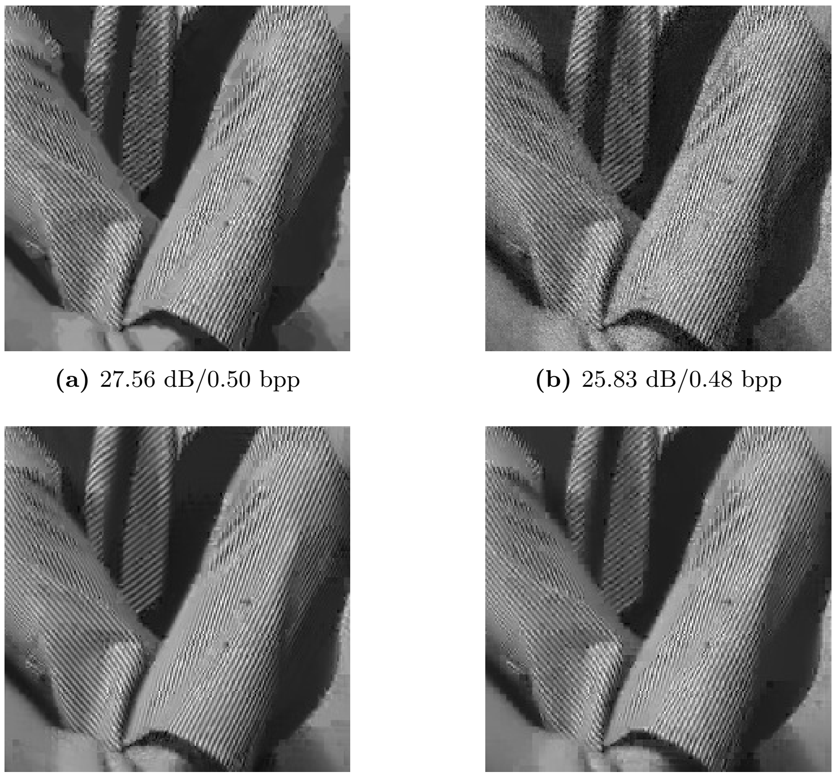


(d)



(e)

Figure 5.15: Encoding performance curves of (a) Barbara, (b) Roof, (c) Foreman, (d) Bike, and (e) Flowergarden using NMF and LLE with relaxed sparsity constraints - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the predicted signal and a relaxed sparsity constraint $k = K = 100$.



(a) 27.56 dB/0.50 bpp **(b)** 25.83 dB/0.48 bpp

(c) 28.44 dB/0.35 bpp **(d)** 27.84 dB/0.44 bpp

Figure 5.16: Reconstructed images of a textural region in Barbara with relaxed sparsity constraints - (a) H.264 intra modes, (b) SP, (c) LLE, and (d) NMF based image prediction methods with the RD criterion and a relaxed sparsity constraint $k = K = 100$.

based methods. In the SP algorithm, the last iteration of OMP is used for prediction where the stopping criterion is a very low residual energy of the approximation support signal.

One can observe that the proposed LLE based prediction method has an increasing RD performance with the number of used image patches (atoms) whereas NMF and SP performance degrades. In this setup, the method based on LLE outperforms the other algorithms described in this chapter. Fig. 5.16 demonstrates the reconstructed textural region of Barbara image with the H.264/AVC intra, SP, LLE, and NMF based image prediction methods for $k = K = 100$, and Fig. 5.17 compares the effect of strict and relaxed sparsity constraints on the proposed NMF and LLE based prediction methods in reference to H.264/AVC intra prediction performance.

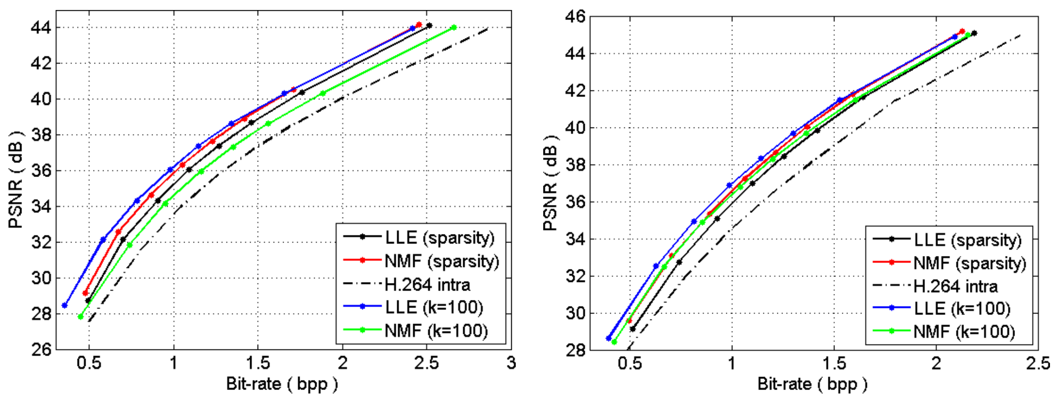


Figure 5.17: Encoding performance analysis of strict and relaxed sparsity constraints - (Left) Barbara and (right) Roof images in reference to the H.264/AVC intra prediction method.

5.3.7 A comparison with weighted template matching

Here we assess comparatively the proposed NMF and LLE based prediction methods to the weighted template matching based algorithms, i.e., ATM and NLM (please refer to Table 3.2), with SSE/RD optimized dynamic approximation supports. Please note here that the NMF and LLE methods differ from ATM and NLM in the sense that they run a constrained optimization on the approximation support (template) signal in order to calculate the weighting coefficients, whereas in the ATM and NLM based methods the weighting coefficients are calculated in a simple way, for example the coefficients are uniform in ATM.

We have evaluated the performance both in terms of prediction quality and encoding PSNR/bit-rate efficiency. The same encoder structure as described above has been kept with the search window configuration in Fig 5.5 for 4×4 block size. The nearest neighboring patches (for all methods) are selected with the SSE distance measure. For the NLM based method, the distance metric DIST in (3.25) has been set to MSE, and the decay coefficient $h = 25$. The parameters for the NMF and LLE methods have been kept the same with the previous experimentations.

Fig. 5.18 demonstrates the PSNR performance curves for Foreman, Barbara, and Roof images where the optimization criterion is the minimization of the prediction signal SSE. Fig. 5.19 shows the corresponding encoding PSNR/bit-rate performance curves of the same test images using the RD optimization criterion. These simulations

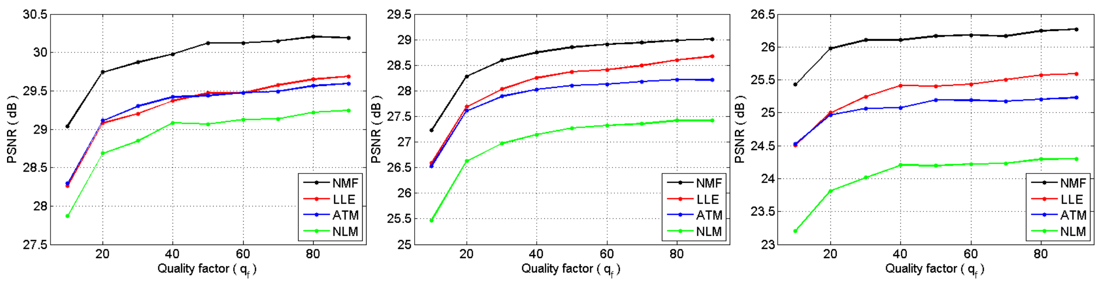


Figure 5.18: Prediction performance curves of (left) Foreman, (middle) Barbara, and (right) Roof using NMF and LLE in comparison to ATM and NLM - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE and a sparsity constraint $k \in [1, 8]$.

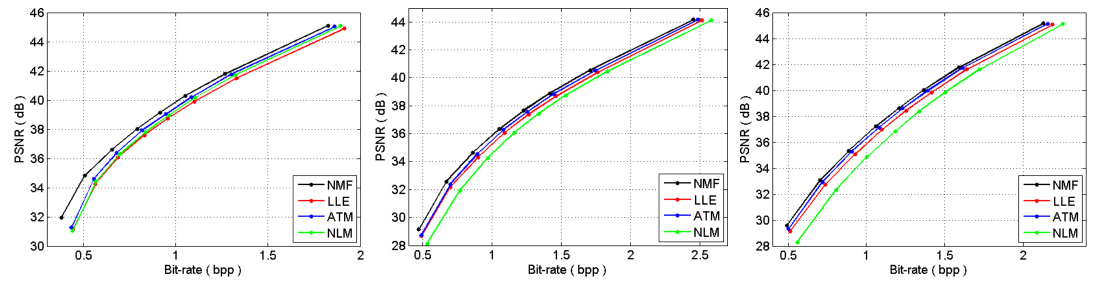


Figure 5.19: Encoding performance curves of (left) Foreman, (middle) Barbara, and (right) Roof using NMF and LLE in comparison to ATM and NLM - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the prediction signal and a sparsity constraint $k \in [1, 8]$.

have been carried out with a sparsity constraint $k \in [1, 8]$. The coding cost of the optimum value of k, as well as the optimal approximation support type, has been included into total bit-rate. One can clearly observe that the NMF based method outperforms the other methods both in terms of prediction quality and encoding efficiency.

The second experimental setup consists of relaxing the value of k and fixing it to $K = 100$ (it is assumed to be known also at the decoder so that no signaling required). Fig. 5.20 and Fig. 5.21 show the performance results obtained for the same test images as above. The first observation one can make is that, as expected, the method based on LLE outperforms the other methods both in terms of prediction quality and encoding efficiency. The second observation is the increase in the performance of the NLM based

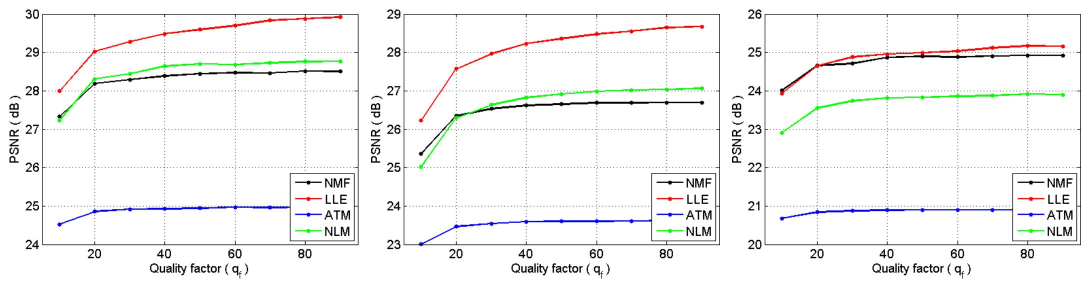


Figure 5.20: Prediction performance curves of (left) Foreman, (middle) Barbara, and (right) Roof using NMF and LLE in comparison to ATM and NLM - PSNR performance curves with different quality levels in the image coding scheme. Optimization criterion is the minimization of the prediction signal SSE and a relaxed sparsity constraint $k = K = 100$.

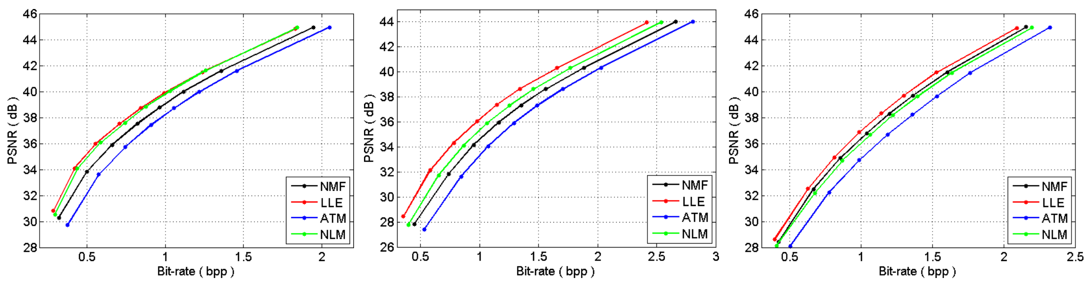


Figure 5.21: Encoding performance curves of (left) Foreman, (middle) Barbara, and (right) Roof using NMF and LLE in comparison to ATM and NLM - PSNR/bit-rate performance curves in the image coding scheme. Optimization criterion is the minimization of the RD cost function on the prediction signal and a relaxed sparsity constraint $k = K = 100$.

method in reference to ATM. Apparently, with the increasing number of used image patches, NLM becomes an effective alternative to ATM since the weighting coefficients have been calculated in a wise manner which prevents over-smoothing effects of simply averaging $k = 100$ patches.

5.4 Computational complexity analysis

Suppose that for a given dictionary matrix $\mathbf{A} \in \mathbb{R}^{\mathrm{N} \times \mathrm{M}}$, $\mathrm{N} \leqslant \mathrm{M}$, $\mathbf{A}_c$ is of size $\mathrm{N}_1 \times \mathrm{M}$ and $\mathbf{A}_t$ is of size $\mathrm{N}_2 \times \mathrm{M}$ where $\mathrm{N}_1 + \mathrm{N}_2 = \mathrm{N}$. The known pixel values in the approximation support are stored in the column vector $\mathbf{b}_c$ of size N_1. (For 4×4 block size $\mathrm{N}_2 = 16$.)

Unlike SP[1], in the NMF based prediction method $K = 8$ patches can be selected in the beginning, and then be used in a loop of increasing $k, k = 1...K$. In order to select K image patches out of M, one needs roughly $2\mathrm{MN}_1$ additions and MN_1 multiplications, followed by M comparisons. At any given iteration k, the update of the weighting vector $\boldsymbol{\alpha}_k$ needs $t(k^2(\mathrm{N}_1 + 1) + k\mathrm{N}_1)$ correlations and $t(2k)$ elementwise multiplications. Finally, the prediction of unknown pixel values needs $k\mathrm{N}_2$ multiply-add (correlation) operations.

Similarly for the LLE based method, $K = 8$ patches can be selected in the beginning. Then, at any given iteration k, the calculation of the local covariance matrix $\mathbf{D}_k$ requires $k\mathrm{N}_1$ additions and $k^2\mathrm{N}_1$ multiply-add operations. To solve the linear system of equations problem $\mathbf{D}_k\boldsymbol{\alpha}_k = \mathbf{1}$, one needs $\mathcal{O}(k^3)$ arithmetic operations (with Gaussian elimination). Here also the prediction of unknown pixel values needs $k\mathrm{N}_2$ multiply-add operations.

In the proposed NMF and LLE based methods, the prediction algorithm can be run only for a fixed k value at once, whereas in the sparse prediction method one needs to iterate until the $(k-1)^{th}$ iteration in order to obtain the results for the k^{th} iteration which is dependent on the residual signal $\mathbf{r}_{k-1}$ obtained at the iteration $k-1$. Therefore, the proposed methods here offer independency between two consecutive iterations $k-1$ and k which may allow to set and then fix k to high values (e.g., we have observed that the LLE based method works better with high k values) without need of $k-1$ processes. Moreover, selection of k patches can be done by using sum of absolute distance measure instead of sum of squared distance in order to further reduce the complexity.

To complete the analysis, Table 5.4 gives the execution times of the different prediction algorithms both on the encoder and decoder sides, measured with our matlab code on a 3GHz CPU. These values have been averaged per block and then given, in average, for an image of size 256×256 pixels.

Note here that the complexity of all above methods is directly related also to the size of the search window W, hence the number M. So, the search window configuration can be readjusted to satisfy the computational load requirements, however at the expense of a possible performance decrease.

[1]Please refer to previous chapter(s) for the computational complexity analysis of the sparse prediction (SP) method.

	with sparsity		relaxed sparsity	
Prediction method	**Encoder**	**Decoder**	**Encoder**	**Decoder**
Template matching	4.70 sec	0.62 sec	4.70 sec	0.62 sec
Sparse prediction	44.43 sec	3.26 sec	392.42 sec	34.14 sec
LLE	32.87 sec	1.19 sec	22.22 sec	2.75 sec
NMF	27.40 sec	2.29 sec	24.23 sec	4.29 sec
H.264 Intra	0.94 sec	0.20 sec	0.94 sec	0.20 sec

Table 5.4: A run-time comparison between different prediction methods both at decoder and encoder - The execution times are measured with a matlab code and could thus be further optimized in C/C++ implementations. The values have been averaged per 4×4 block and than given, in average, for an image of size 256×256 pixels.

5.5 Conclusion

In this chapter, we have introduced two new block-based spatial image prediction methods based on neighbor embedding algorithms. Experimental results obtained show that the proposed methods offer better prediction quality and also compression efficiency when compared to the (weighted) template matching approach(es), the sparse approximations based method, and H.264/AVC intra prediction modes. For more complex and non-periodic textural images, they turn out to be an effective alternative to H.264/AVC prediction modes, however at the expense of complexity increase as shown in this chapter. Note that their complexity largely depends on the size of the search window for constructing the dictionary and of the number of neighbors k considered for the approximation. The proposed methods can be seen as a generalization of template matching (TM), in the sense that they search to approximate the template with a linear combination of co-located pixels in similar patches whereas the template matching approximates the template with one similar patch and with a weighting coefficient equal to one. However, the simulation results show significant performance gain of the proposed methods against template matching (and its simple extensions with ATM and NLM) for intra prediction.

The proposed methods can be applied in a similar manner as TM has been applied to inter prediction in [45]. The search window will thus not be limited to a causal intra window but will consider patches in reference frames for inter prediction. The authors in [45] have shown that the use of TM in this context can lead to a bit-rate gain of around

11% when compared with conventional motion compensation based codecs. Given the superior performances of the proposed methods against TM for intra prediction, it is reasonable to also expect further gains for inter prediction with the proposed NMF and LLE based methods. As a last remark, note that these techniques can also be applied to any other texture synthesis problems such as image inpainting and loss concealment.

Chapter 6

Image Inpainting via Neighbor Embedding

This chapter[1] describes exemplar-based image inpainting algorithms relying on *neighbor embedding methods* using locally linear embedding (LLE) and non-negative matrix factorization (NMF). It first introduces an augmented patch priority which defines the patch fill-in order. In addition to so-called *confidence* and *data* terms, this priority measure takes into account also an edge term, aiming at giving more priority to patches containing strong edges in order to prevent possible artifacts in the continuation of structures. The patch is then filled-in by a weighted linear combination of K closest patches, instead of using a single "best" patch. The nearest neighbors are taken from the source image and determined on the known pixel values of the input patch. An adaptive way of choosing the number K is also described. Finally an extension to the LLE based method has been proposed by exploiting the neighborhood and local geometry preserving properties of LLE. The proposed methods have been assessed for two different applications: object removal in a context of image editing and completion of missing regions in a context of loss concealment. Experiments show the effectiveness of the proposed approaches in providing natural looking images with less visually annoying artifacts in comparison to exemplar-based inpainting methods using template matching (TM), average template matching (ATM), and non-local means (NLM) approach. The inpainting results on several natural images are also compared to other inpainting methods using exemplar- and diffusion-based approaches in the literature.

[1]The content of this chapter is related to our work in [147].

6.1 Introduction

Image inpainting refers to methods which consist of filling-in missing regions (holes) in an image [148]. Inpainting techniques find applications in a number of image processing problems: image editing (e.g., object removal), image restoration, object disocclusion in image based rendering, image coding, loss concealment after impaired transmission. Existing methods can be classified into two main categories. The first category concerns diffusion-based approaches which propagate level lines or linear structures (so-called *isophotes*) via diffusion based on partial differential equations (PDE) [148, 149, 150, 151] and variational methods [152]. In other words, they tend to prolong isophotes arriving at the border of the region to be filled. The diffusion-based methods tend to introduce some blur when the hole to be filled-in is large.

The second type of approach concerns exemplar-based methods which sample and copy best match texture patches from the known image neighborhood [43, 153, 154, 155, 156, 157, 158, 159]. These methods have been inspired from texture synthesis techniques [11, 12, 15] and are known to work well in cases of regular textures. The first attempt to use exemplar-based techniques for object removal has been reported in [154]. The authors in [155] improve the search for similar patches by introducing an a priori rough estimate of the inpainted values using a multi-scale approach which then results in an iterative approximation of the missing regions from coarse to fine levels. In addition, the candidate patches for the match also include rotated, scaled, and mirrored version of texture patches taken from the image.

The two types of methods (diffusion- and exemplar-based) can efficiently be combined so that a PDE-based diffusion technique is applied on structural patches and exemplar-based methods are used for synthesizing texture with finer details. The authors in Criminisi *et al.* [43] introduce an exemplar-based method in which the filling order is defined by a priority function which depends on the angle between the isophote direction and the normal direction of the local filling front. The goal of this priority function is to ensure that the linear structures are propagated before texture filling.

A sparse approximation method is considered in [160] together with a sparsity-based priority function. The results obtained with this method show sharp inpainting results with efficient and consistent inference of structures and textures. As stated in Chapter 5, the use of sparsity concept in various image processing tasks is motivated by the assumption that natural images are composed of only a few structural "primitives". After obtaining (or learning) these primitives, one can decompose the image (or its texture patches) on the set of primitives to extract the representative features. One can even think of texture patches (present in a large neighborhood of an input patch) as primitives, and this in order to exploit self-similarities within the image. The self-similarity property has been largely exploited and shown to be beneficial in a number of image processing tasks, e.g., for image denoising in [140].

In traditional exemplar-based inpainting techniques, the filled-in values of the input patch are obtained by sampling the "best" match patch from the source image as Markov Random Field based texture synthesis. This is the same process with *template matching* (TM) which is defined for image prediction. One can also linearly combine several texture patches instead of using a single best patch. The ATM method is hence applicable to image inpainting. Recently, an NLM based approach [161] has also been proposed for image inpainting. Notice here that TM and its extensions (ATM and NLM) have already been used for the image prediction problem in the previous chapters. There is indeed some strong similarity between the inpainting problem and the prediction problem. Prediction methods also use TM and even more recently methods based on sparse approximations of the known samples (as described in Chapter 3). However, there are at the same time important differences, in particular related to the facts that in inpainting, the known samples are not restricted to be in a causal neighborhood and the texture patches can be treated in a different order than the raster scan. Exemplar-based inpainting methods actually compute a priority for the different patches to be filled so that the algorithm can start by first propagating important structures in the image. This was the key contribution of the approach in [43]. However, in prediction, the processing order is the raster scan, the known samples are located only in a causal neighborhood of the block to be predicted. The priority order as used in inpainting can not be used as such (or not with as much freedom) for the prediction problem. On the other hand, in inpainting, there is no block-wise computation and coding of the prediction residue, hence errors in texture filling are very critical and can propagate easily.

We consider here two methods of data dimensionality reduction for inpainting: locally linear embedding (LLE) [134] and non-negative matrix factorization (NMF) [128]. The underlying main motivation is to find a best linear combination of a small number of non-negative "texture primitives" (texture patches) to approximate the input non-negative patch using a constrained optimization rather than a heuristic calculation of weighting coefficients as in ATM and NLM based methods. This problem falls naturally within both frameworks of LLE and NMF. In NMF, the weighting coefficients are forced to be non-negative in order to construct representations of non-negative texture patches in an additive manner. Similarly, for LLE, the goal is to reconstruct each input texture patch from its K nearest neighbors (K-NN). The sum-to-one constraint on the weighting coefficients forces the reconstruction of each data point (here input patch) to lie in the subspace spanned by its nearest neighbors.

Each patch in the image can be approximated by a linear combination of its K-NN. For both LLE and NMF based inpainting methods, the first idea explored here is to search for this linear combination of K texture patches taken from the source image to approximate the known part (so-called the template) of the patch to be filled, and keep the same weights to estimate the unknown pixels as a linear combination of the colocated pixels in the K-NN patches. The patch selection is carried out by choosing K number of most similar patches to the template of the input patch (by comparing the colocated pixels of the reference patches in the source image). Note here that the main procedure of estimating the unknown pixel values in a patch remains the same as proposed for image prediction in Chapter 5. The methods proposed here can also be seen as a generalization of TM. TM is a special case when only a single image patch is used with a weighting coefficient equal to 1.

The techniques described in this chapter improve upon the exemplar-based inpainting method described in [43], by introducing the following key contributions:

- A new patch priority term which takes into account the edgeness (the amount of pixels belonging to an edge) of the patch to be filled-in;
- The use of a linear combination of K candidate patches to fill-in the missing region, using LLE and NMF based optimization methods instead of using TM, ATM, or NLM in existing exemplar-based inpainting solutions.

The rest of the chapter is organized as follows. We first introduce the notations and summarize the main steps of the proposed inpainting method. Then we describe a new patch priority function which accounts for the edgeness of the patch to be filled-in.

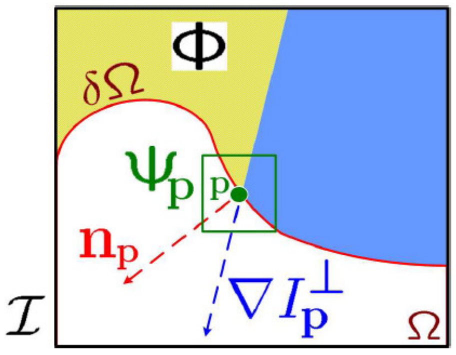


Figure 6.1: Image inpainting algorithm overview - Given the patch Ψ_p, $\mathbf{n}_p$ is the normal to the contour $\delta\Omega$ of the target region Ω and $\nabla I_p^{\perp}$ is the isophote (direction and intensity) at point p. This illustration is taken from [43].

Next we recall the main principles of the background sampling approaches (including TM, ATM, and NLM for inpainting) followed by the proposed NMF and LLE based neighbor embedding algorithms. We then present experimental results obtained with the proposed methods in comparison to TM, ATM, and NLM based methods for two different applications as object removal in a context of image editing and missing region completion in a context of loss concealment. The methods are also compared to other state-of-the-art exemplar- and diffusion-based inpainting algorithms in the literature for object removal. Finally we conclude this work with a brief conclusion.

6.2 Algorithm Overview

Let I be the image and Ω be the region to be filled-in. Let $\phi = I - \Omega$ be the known (or source) region in the image and $\delta\Omega$ be the border (or fill front) of the region to be filled in (see Fig. 6.1). The algorithms based on neighbor embedding methods iterate the following steps until all missing pixels have been filled:

1. Identification of the fill front $\delta\Omega$.

2. Computation of the priority

$$P(p) = C(p)D(p)E(p), \quad \forall p \in \delta\Omega,$$

by taking into account also the edgeness $E(p)$ of the patch (for each patch to be filled centered on a pixel p located on the fill front $\delta\Omega$).

3. Selection of the patch $\Psi_{\hat{p}}$ (centered on a pixel $\hat{p}$ located on the fill front) with the highest priority, i.e.,

$$\hat{p} = \arg\max_{p \in \delta\Omega} P(p).$$

4. Calculation of the number K for the patch $\Psi_{\hat{p}}$ and selection of the K-NN patches $\Psi_{q_k}, k = 1...\text{K}$, from the source image $\phi = I - \Omega$. These patches are determined with the known pixel values of $\Psi_{\hat{p}}$.

5. Computation of the weighting coefficients with neighbor embedding methods by optimizing the approximation of the known pixel values of $\Psi_{\hat{p}}$ using its K-NN.

6. Estimation of the unknown pixel values to be filled-in by a linear combination of the colocated pixels in the K-NN patches.

7. Update of the confidence term $C(p)$ (and also the edge map for $E(p)$) used in the patch priority computation.

Note that the steps above are very similar to the method described in [43] but the algorithm differs first in computation of the patch priority, and then in estimating the unknown (fill-in) values of the patch $\Psi_{\hat{p}}$.

6.3 Edge-based Patch Priority

Given a patch Ψ_p centered at the point p (known pixel) located on the front line (fill front), the filling order (also called patch priority) is defined as a product of three terms: $P(p) = C(p)D(p)E(p)$. The first term, so-called *confidence term* is given by [43],

$$C(p) = \frac{\sum_{q \in \Psi_p \cap (I-\Omega)} C(q)}{|\Psi_p|}, \tag{6.1}$$

measures the ratio between the number of known pixels with respect to the total number of pixels in the patch to be filled-in. And the second term $D(p)$, so-called *data term*, is given as

$$D(p) = \frac{|\nabla I_p^{\perp} \mathbf{n}_p|}{\alpha} \tag{6.2}$$

where α is a normalization factor and $\mathbf{n}_p$ is a unit vector orthogonal ($\perp$) to the filling front $\delta\Omega$ at the point p as shown in Fig. 6.1. This second term increases the priority of the patch having isophotes perpendicular to the filling front. However, it does not really reflect the predominance of an edge within the patch. Therefore, we introduce here a third term $E(p)$ which is the ratio of the amount of known pixels of the patch which belong to an edge with respect to the total number of known pixels in the patch. Thus $E(p)$ can be defined as

$$E(p) = \frac{\sum_{q \in \Psi_p \cap (I - \Omega)} \delta(q \in \mathbf{E})}{|\Psi_p \cap (I - \Omega)|} \tag{6.3}$$

where $\delta()$ is a binary function which returns 1 when its argument is true and 0 otherwise. $\mathbf{E}$ is the set of binary edge pixels which is determined by using a Canny edge detector.

6.4 Inpainting with Neighbor Embedding

6.4.1 Background approaches

This subsection briefly revises the conventional texture synthesis (sampling) methods used for the inpainting problem including TM, ATM as well as the NLM based method. Note that we have already studied these methods for the image prediction problem in the previous chapters. For the sake of a complete analysis, we reformulate these methods for image inpainting.

6.4.1.1 Template matching for inpainting

The TM method has basically been influenced by MRF which models the texture patterns as a realization of a local and stationary random process. A so-called *template* is assumed to be formed with the known pixel values in the input patch $\Psi_{\hat{p}}$ to be filled-in. The best match between the template and the texture patches present in the source image allows sampling of the unknown fill-in pixel values. Assuming that the current patch $\Psi_{\hat{p}}$ centered around the pixel $\hat{p}$ is composed of two regions denoted as $\Psi_{\hat{p}}^o$ and $\Psi_{\hat{p}}^u$ representing the observed (known) pixel values in the template and the unknown (to be filled-in) values respectively. One searches for a patch in the source image ϕ centered at a point $\hat{q}$ such that

$$\hat{q} = \arg\min_{q \in \phi} \left\| \Psi_{\hat{p}}^o - \Psi_q^o \right\|_2^2, \tag{6.4}$$

and the fill-in region has been estimated with the colocated values of $\Psi_{\hat{q}}$ as

$$\Psi_{\hat{p}}^{u} = \Psi_{\hat{q}}^{u}. \tag{6.5}$$

6.4.1.2 Average template matching for inpainting

ATM is a simple extension of TM where several patches are combined with uniform weights. The patch selection process generally proceeds by choosing K number of most similar patches to the template in the source image. After obtaining K nearest neighboring patches $\Psi_{\hat{q}_k}, k = 1...K$, one uniformly combines colocated pixels of these patches in order to estimate the fill-in region values as follows

$$\Psi_{\hat{p}}^{u} = \frac{1}{K}\sum_{k=1}^{K} \Psi_{\hat{q}_k}^{u}. \tag{6.6}$$

6.4.1.3 Non-local means for inpainting

The weights can be calculated by other means than simply the average. NLM tries to aggregate multiple image patches as a weighted linear combination, but the weighting coefficients are calculated in a different, yet another heuristic, manner. The contribution weights are calculated with a patch similarity based kernel function (e.g., an exponential function) in order to give more weights to the patches which are more similar to the template than the others. After obtaining K nearest neighboring patches $\Psi_{\hat{q}_k}, k = 1...K$, determined in the same manner as ATM, the synthesis process follows by

$$\Psi_{\hat{p}}^{u} = \sum_{k=1}^{K} \alpha_k \Psi_{\hat{q}_k}^{u} \tag{6.7}$$

where the weighting coefficients α_k are calculated by means of

$$\alpha_k = \exp\left(- \frac{\left\| \Psi_{\hat{p}}^{o} - \Psi_{\hat{q}_k}^{o} \right\|_2^2}{h} \right) \tag{6.8}$$

and h is the decay coefficient. The calculated weights are finally normalized to sum-to-one.

6.4.2 Neighbor embedding methods for inpainting

The above described methods either copy and paste a single best patch, or calculate weighting coefficients in a heuristic manner which depends on a distance measure with the template and the colocated pixel values of the reference patches in the source image. However, these approaches do not search to minimize an approximation error on the template signal. Neighbor embedding techniques such as LLE or NMF allow us to formulate the inpainting problem as a least-squares problem (as in the prediction problem in Chapter 5) with different types of constraints. For the main principles of NMF and LLE methods please refer to Chapter 5. Below we give their applications to the image inpainting problem.

6.4.2.1 Inpainting based on NMF

Here the main idea is to run NMF optimization algorithm for approximating the known (observed) pixel values of the patch $\Psi_{\hat{p}}$ using K-NN patches $\Psi_{\hat{q}_k}$, selected from the source image. However, unlike the direct application of NMF to these samples, the matrix $\mathbf{A}$ in (5.4) is assumed to be fixed and it is constructed by stacking the colocated values of these K patches in the columns of the matrix $\mathbf{A}$, such that $\mathbf{A} = [\mathbf{a}_1|...|\mathbf{a}_K]$ where $\mathbf{a}_k$ represents the vectorized forms of $\Psi^o_{\hat{q}_k}$, $\forall k$. The NMF method then solves the following optimization as

$$\min_{\boldsymbol{\alpha}} \left[\frac{1}{2} \|\mathbf{b} - \mathbf{A}\boldsymbol{\alpha}\|_2^2 \right] \quad \text{subject to} \quad \boldsymbol{\alpha} \geq 0 \tag{6.9}$$

by updating the randomly initialized non-negative elements of $\boldsymbol{\alpha}^{\mathrm{T}} = [\alpha_1|...|\alpha_K]$ as

$$\alpha_k \leftarrow \alpha_k \frac{(\mathbf{A}^{\mathrm{T}}\mathbf{b})_k}{(\mathbf{A}^{\mathrm{T}}\mathbf{A}\boldsymbol{\alpha})_k + \varepsilon}, \quad k = 1...K, \tag{6.10}$$

where $\mathbf{b}$ represents the vectorized form of the template information $\Psi^o_{\hat{p}}$, and ε is a small constant equal to 10^{-9} to avoid divide-by-zero in the update equation.

In order to obtain the optimum weighting vector $\boldsymbol{\alpha}$, the update equation in (6.10) is iterated until a pre-defined iteration number T is reached, or the total change in the elements of the vector $\boldsymbol{\alpha}$ is very small between two consecutive iterations $t-1$ and t, $t = 1...T$. Finally the synthesis process follows by

$$\Psi^u_{\hat{p}} = \sum_{k=1}^{K} \alpha_k \Psi^u_{\hat{q}_k} \tag{6.11}$$

where the weights α_k have been obtained from the last iteration of the NMF optimization. Note here the similarity between ATM and NLM based methods in the sense of using K-NN patches. However, the calculation of coefficients completely differs since an optimization has been run on the template signal rather than using a heuristic method.

6.4.2.2 Inpainting based on LLE

The LLE method is first applied (partially) on the template $\Psi^o_{\hat{p}}$. One thus searches for an approximation of the template by a linear combination of its K-NN patches in the source image and then keeps the same weighting coefficients in the linear combination of the colocated pixels in order to estimate the unknown values of $\Psi^u_{\hat{p}}$. In terms of LLE, the inpainting problem can be written as

$$\min_{\boldsymbol{\alpha}} \|\mathbf{b} - \mathbf{A}\boldsymbol{\alpha}\|_2^2 \quad \text{subject to} \quad \sum_k \alpha_k = 1, \tag{6.12}$$

where $\mathbf{b}$ represents the vectorized form of the template signal $\Psi^o_{\hat{p}}$, and the matrix $\mathbf{A}$ contains the colocated pixel values of K-NN patches in its columns similar to the NMF based inpainting method. The optimal weighting coefficients α_k in $\boldsymbol{\alpha}$ are computed by

$$\boldsymbol{\alpha} = \frac{\mathbf{D}^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{1}} \tag{6.13}$$

where $\mathbf{D}$ denotes the local covariance (Gram) matrix (i.e., in reference to $\mathbf{b}$) of the selected K-NN patches in $\mathbf{A}$, and $\mathbf{1}$ is the column vector of ones. Finally the synthesis process follows by

$$\Psi^u_{\hat{p}} = \sum_{k=1}^{K} \alpha_k \Psi^u_{\hat{q}_k}. \tag{6.14}$$

6.4.2.3 Inpainting based on LLE with subspace mappings

LLE computes low-dimensional neighborhood preserving embeddings of high-dimensional data points. Each high-dimensional data point and its neighbors are assumed to lie on or close to locally linear patch of a low-dimensional manifold. Each data point can then be approximated by a linear combination of its K-NN. We thus further exploit the neighborhood and local geometry preserving properties of LLE when embedding high-dimensional data points into a lower-dimensional manifold. Let N be the total number of pixels in a patch to be filled-in. A set of image patches taken from the source image in a given window are used to train parametric functions which map data points corresponding to vectors of pixels colocated to observed (known) and missing (unknown) parts of the patch to be filled, into data points in $\mathbb{R}^{\mathrm{N}}$. The parametric mapping functions are computed using a multivariate linear regression. Note that such training has already been considered in [162] to perform 3D object recognition in a lower-dimensional space. These parametric mapping functions are then used here to select the K-NN patches to be considered in the LLE approximation. The goal is, with the help of the linear mapping from the high-dimensional space of input data to a low-dimensional space, and vice versa, to find the best K-NN of the known pixels, for which the corresponding complete patches will also be included in K-NN of the patch to be filled-in.

Given the highest priority patch $\Psi_{\hat{p}}$, we constitute a training set of M image patches taken from the source image ϕ. A training matrix has been constructed by stacking these patches into the columns of $\mathbf{X} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$. Each patch $\mathbf{X}_m \in \mathbb{R}^{\mathrm{N}}, m = 1...\mathrm{M}$, of the training set is assumed to be formed by an observed part $\mathbf{X}_m^o$ (set of N_1 pixels colocated with the known pixels $\Psi_{\hat{p}}^o$ of $\Psi_{\hat{p}}$) and an unknown part $\mathbf{X}_m^u$ (set of N_2 pixels colocated with the unknown pixels $\Psi_{\hat{p}}^u$ of $\Psi_{\hat{p}}$). $\mathrm{N} = \mathrm{N}_1 + \mathrm{N}_2$.

Since there will be only the template information (in $\mathbb{R}^{\mathrm{N}_1}$) available during the inpainting process, the goal is to learn a transformation which relates the set of data patches $\mathbf{X}_m^o$ in $\mathbb{R}^{\mathrm{N}_1}$ to the set of data patches $\mathbf{X}_m^u$ in $\mathbb{R}^{\mathrm{N}_2}$, given that these two signals $\mathbf{X}_m^o$ and $\mathbf{X}_m^u$ correspond to the same patch $\mathbf{X}_m$ in $\mathbb{R}^{\mathrm{N}}$. We thus define the following "relationships" (mappings) between training patches $\mathbf{X}_m^o$, $\mathbf{X}_m^u$, and $\mathbf{X}_m$ for all m such that

$$\mathbf{X}_m^o \xrightarrow{A_1} \mathbf{X}_m^u \xrightarrow{A_2} \mathbf{X}_m \qquad \text{and} \qquad \mathbf{X}_m^o \xrightarrow{A_3} \mathbf{X}_m. \tag{6.15}$$

The ultimate goal is to guarantee that the selected K-NN of $\mathbf{X}_m^o$ will also be K-NN of $\mathbf{X}_m$ and $\mathbf{X}_m^u$.

The transformation (mapping) matrices A_1 between $\mathbf{X}_m^o \in \mathbb{R}^{\mathrm{N}_1}$ and $\mathbf{X}_m^u \in \mathbb{R}^{\mathrm{N}_2}$, A_2 between $\mathbf{X}_m^u \in \mathbb{R}^{\mathrm{N}_2}$ and $\mathbf{X}_m \in \mathbb{R}^{\mathrm{N}}$, and A_3 between $\mathbf{X}_m^o \in \mathbb{R}^{\mathrm{N}_1}$ and $\mathbf{X}_m \in \mathbb{R}^{\mathrm{N}}$ are computed using a multivariate linear regression as follows

$$\begin{aligned} A_1 &= \bar{\mathbf{X}}^u \bar{\mathbf{X}}^{o\mathrm{T}} \left(\bar{\mathbf{X}}^o \bar{\mathbf{X}}^{o\mathrm{T}}\right)^{-1} \\ A_2 &= \bar{\mathbf{X}} \bar{\mathbf{X}}^{u\mathrm{T}} \left(\bar{\mathbf{X}}^u \bar{\mathbf{X}}^{u\mathrm{T}}\right)^{-1} \\ A_3 &= \bar{\mathbf{X}} \bar{\mathbf{X}}^{o\mathrm{T}} \left(\bar{\mathbf{X}}^o \bar{\mathbf{X}}^{o\mathrm{T}}\right)^{-1} \end{aligned} \tag{6.16}$$

where the training matrix $\mathbf{X}$ has been assumed to be composed of two submatrices as

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}^o \\ \mathbf{X}^u \end{bmatrix} \tag{6.17}$$

and $\bar{\mathbf{X}}$, $\bar{\mathbf{X}}^o$, and $\bar{\mathbf{X}}^u$ denote the centered data matrices of $\mathbf{X}$, $\mathbf{X}^o$, and $\mathbf{X}^u$ respectively.

Let us now consider the patch $\Psi_{\hat{p}}$ to be filled-in. Suppose that the observed values $\Psi_{\hat{p}}^o$ have been vectorized in $\mathbf{b} \in \mathbb{R}^{\mathrm{N}_1}$. The K-nearest neighbors $\mathbf{X}_k^o, k = 1...K$, of the signal $\mathbf{b}$ are first determined. The obtained K neighbors $\mathbf{X}_k^o$ and $\mathbf{b}$ are then projected to the space of dimension N via A_3. The projected version of the template vector $\mathbf{b}$, which will be denoted as $\hat{\mathbf{b}}$, is used as a "reference" point in $\mathbb{R}^{\mathrm{N}}$.

$\mathbf{b}$ is further projected to another point $\hat{\mathbf{b}}_u$ via A_1 into a subspace of dimension N_2, i.e., $\hat{\mathbf{b}}_u = A_1\mathbf{b}$, and K-nearest neighbors $\mathbf{X}_k^u, k = 1...K$, of $\hat{\mathbf{b}}_u$ are searched in $\mathbb{R}^{\mathrm{N}_2}$. The obtained K patches $\mathbf{X}_k^u$ are also projected to the space of dimension N via A_2.

We finally search for the K-NN of the "reference" vector $\hat{\mathbf{b}}$ in $\mathbb{R}^{\mathrm{N}}$ among the projected $2K$ points corresponding to $\mathbf{X}_k^o$ and $\mathbf{X}_k^u, k = 1...K$. After determining these K patches to be used in the LLE method, the weighting coefficients are obtained by (6.13) to approximate the known pixels of the patch $\Psi_{\hat{p}}$ and then the same weights are used for inpainting (estimating) the unknown pixels $\Psi_{\hat{p}}^u$ using (6.14). Fig. 6.2 summarizes the underlying idea behind the overall technique based on LLE with parametric mapping.

6.5 Adaptive selection of the number K

The number K of neighboring patches used is indeed a parameter which impacts significantly the accuracy of inpainting. Several methods could be envisaged for determining the value of K. One can use iterative greedy methods based on sparse approximations (for an approximation to template) such as matching pursuit [66], orthogonal matching

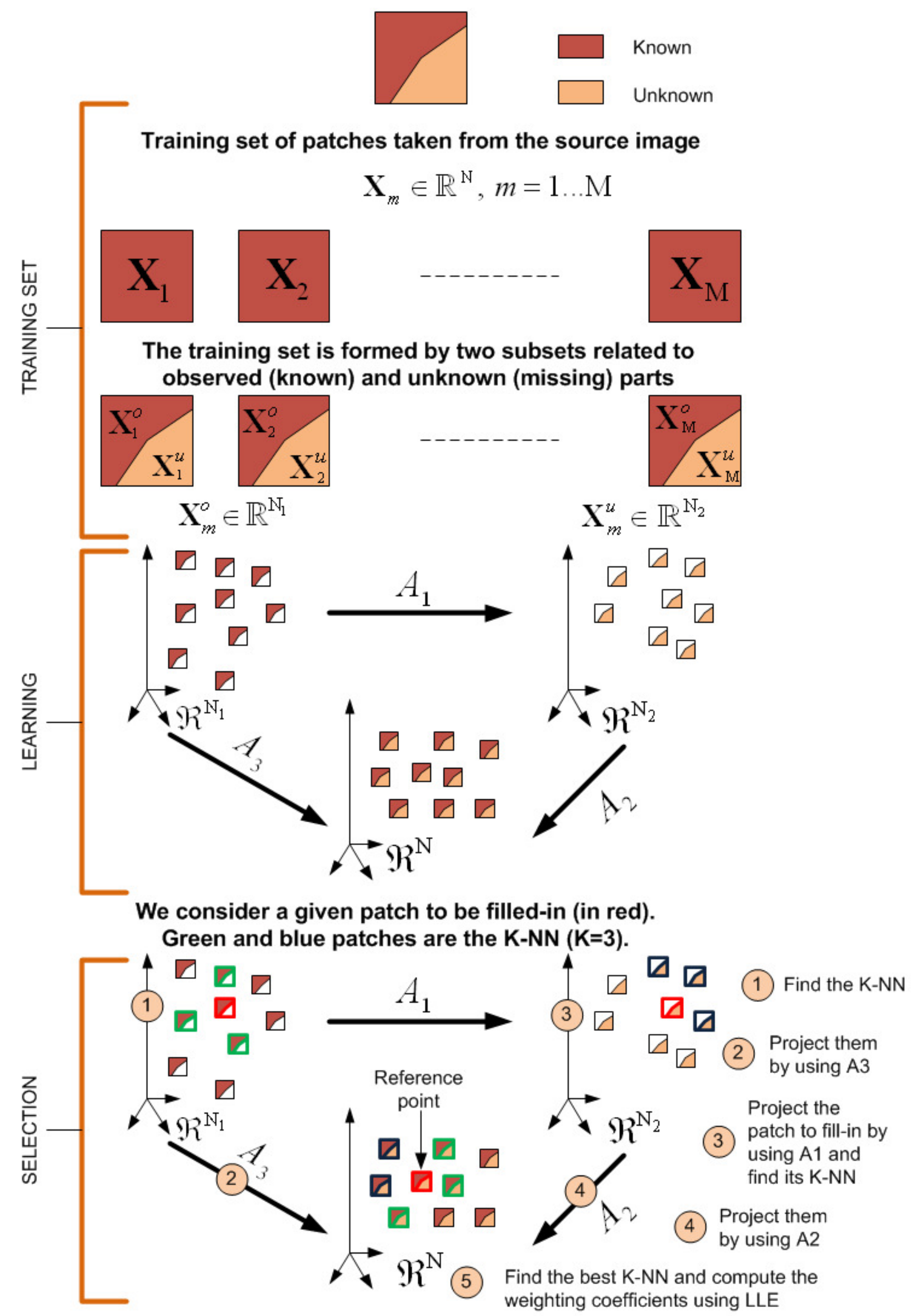

Known
Unknown
Training set of patches taken from the source image
$\mathbf{X}_m \in \mathbb{R}^N, m = 1 \ldots M$
$\mathbf{X}_1$
$\mathbf{X}_2$
$\mathbf{X}_M$
The training set is formed by two subsets related to observed (known) and unknown (missing) parts
$\mathbf{X}_1^o$ $\mathbf{X}_1^u$
$\mathbf{X}_2^o$ $\mathbf{X}_2^u$
$\mathbf{X}_M^o$ $\mathbf{X}_M^u$
$\mathbf{X}_m^o \in \mathbb{R}^{N_1}$
$\mathbf{X}_m^u \in \mathbb{R}^{N_2}$
TRAINING SET
LEARNING
SELECTION
A_1
A_2
A_3
$\Re^{N_1}$
$\Re^{N_2}$
$\Re^N$
We consider a given patch to be filled-in (in red). Green and blue patches are the K-NN (K=3).
Reference point
1 Find the K-NN
2 Project them by using A3
3 Project the patch to fill-in by using A1 and find its K-NN
4 Project them by using A2
5 Find the best K-NN and compute the weighting coefficients using LLE

Figure 6.2: Inpainting with LLE via parametric mapping - Algorithm overview (courtesy of Dr. Olivier Le Meur).

pursuit [67], or basis pursuit [65], however at the expense of a computational complexity. On the other side, fixing K to a value might not be the optimum, such that a low value may lead to less accurate results, hence to the propagation of errors; a high K value may lead to smoothing effects on inpainting which will also propagate. Moreover, a fixed K value will not be optimum for all blocks in an image because of the local characteristics of the image. Here we propose a relatively simple method which keeps K most similar patches (to the template) according to the minimum distance patch. Let us assume that d_{min} represents the mean squared distance of the most similar patch to the template such that

$$d_{min} = \min_{q \in \phi} \frac{\left\| \Psi^{o}_{\hat{p}} - \Psi^{o}_{q} \right\|_2^2}{|\Psi^{o}_{\hat{p}}|}, \tag{6.18}$$

all the patches which are in a ball centered around $\Psi^{o}_{\hat{p}}$ having a radius of length γd_{min} ($\gamma \geq 1$) are selected to be used in the inpainting process. Hence the number K can vary from one patch to another depending on the distance of the other patches in the source image. Note here that as a special case, when $\gamma = 1$, all the methods described above reduce to a simple TM.

6.6 Experimental Results

6.6.1 Effect of edge term on priority function

Fig. 6.3 compares the inpainted (Bungee) images with simple template matching when using the priority function augmented with the “edge term” with respect to the original priority function (as in [43]). The proposed priority function with edge term significantly changes the filling order by giving more priority to structural blocks containing edge information. As it can also be seen from Fig. 6.3, after 100 iteration steps, the shape of the mask is different from the original priority function. Thus, the structures with strong edges are propagated first to prevent any annoying visible artifacts in the continuation of the edgel areas in the image. Here the patch size is 9×9 pixels and γ is fixed to 1.2.

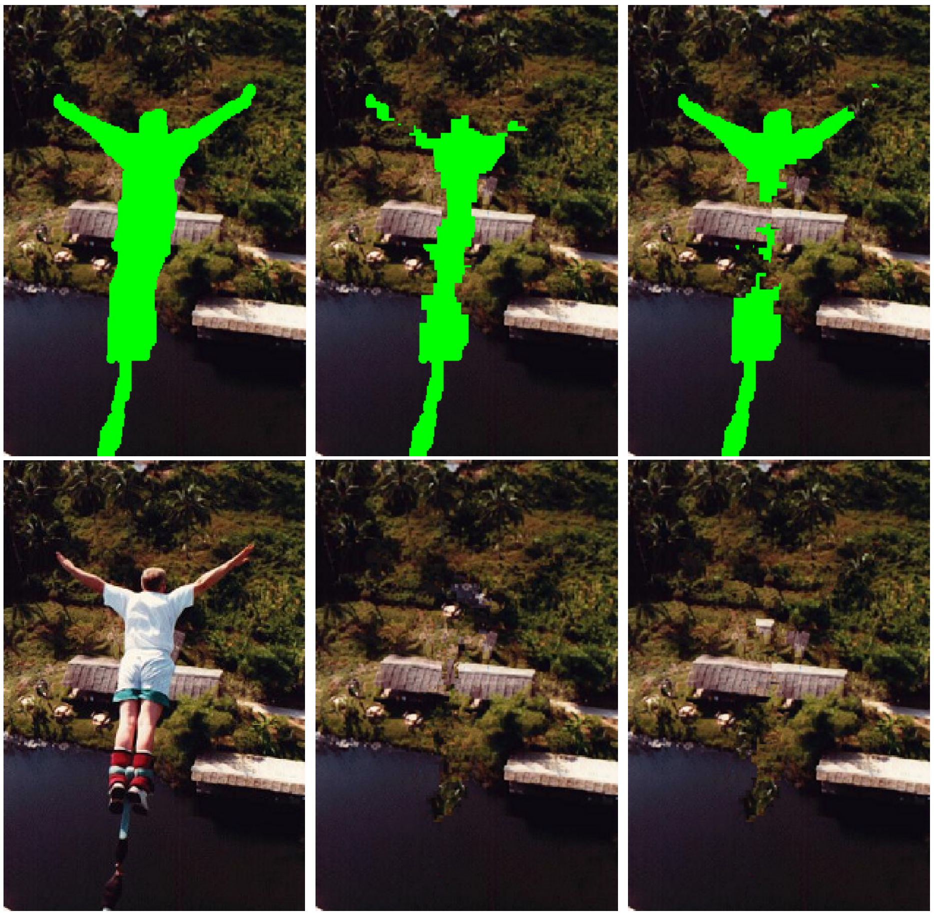


Figure 6.3: Effect of the edge term in the priority function - From top-to-bottom left-to-right: The mask for the inpainting algorithm and the original Bungee image; the inpainting process after 100 patches with priority function of [43] and the final inpainted image using TM; the inpainting process after 100 patches with the augmented priority function and the final inpainted image using TM.

6.6.2 Comparison between neighbor embedding and (weighted) template matching

The first set of experiments aims at assessing the performance of the proposed inpainting algorithms to template matching based methods for an object removal application. In this case, the missing region to be filled-in can be large (e.g., see Fig. 6.3 (left)) and the ground truth pixel values of the missing region are unknown. The only possible way for assessing the results is to check out whether the inpainted image looks natural or not.

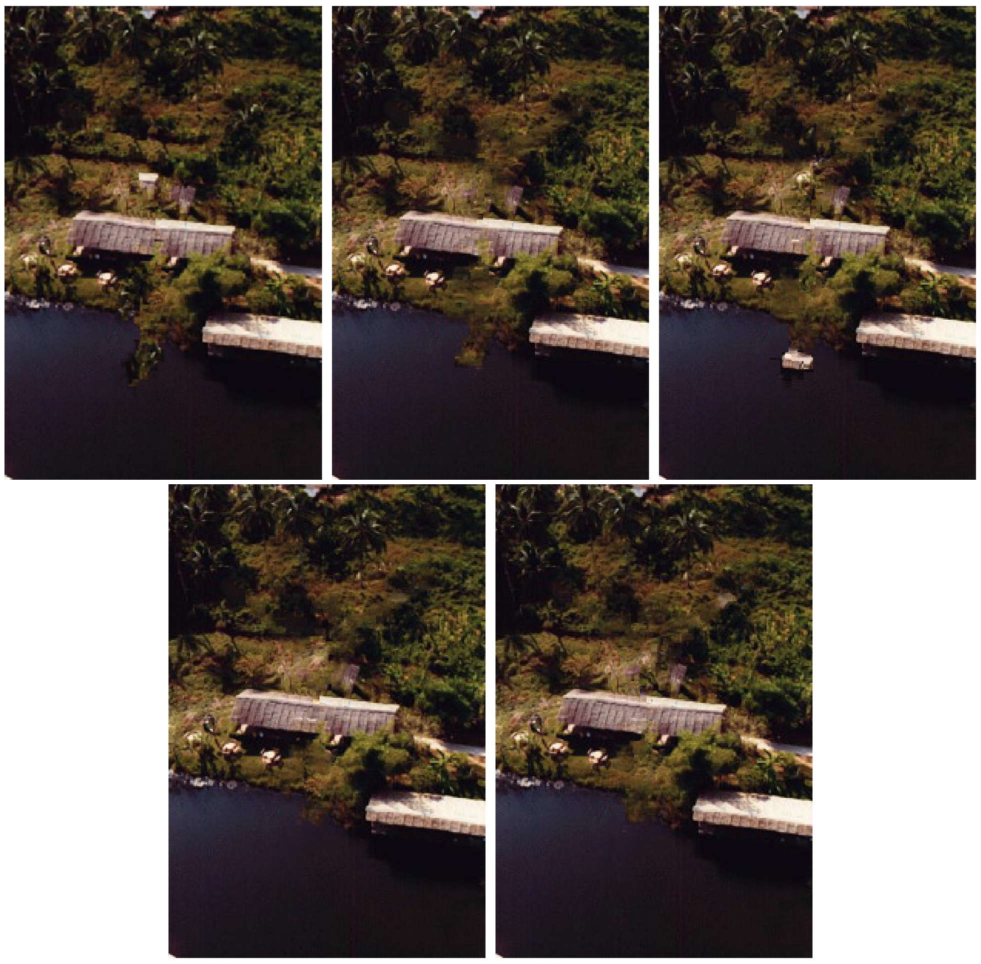


Figure 6.4: Inpainting results with different synthesis methods - From left-to-right top-to-bottom: template matching (TM), average template matching (ATM), non-local means (NLM), NMF, and LLE based methods with the augmented priority function.

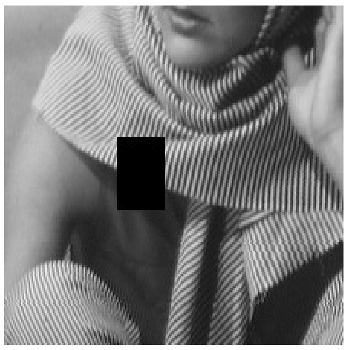
In Fig. 6.4 we compare the inpainting results of the proposed NMF and LLE based neighbor embedding methods with (weighted) template matching (i.e., TM, ATM, and NLM) using the augmented priority function with $E(p)$. The patch size is 9×9 pixels and γ is fixed to 1.2. For NLM, the decay coefficient h is set to $25|\Psi^{o}_{\hat{p}}|$ in (6.8). One can observe that the LLE and NMF based neighbor embedding approaches lead to more realistic inpainted images. In Fig. 6.4, one can see that the bushes are propagated into the sea with TM, ATM, and NLM whereas the proposed embedding methods prevent this propagation. Furthermore, the visual quality of the fill-in region is significantly improved. The results also show, at this point, the importance of the used texture synthesis method for filling-in the unknown samples in addition to the priority function. Since there is no any means of error correction, even small inpainting errors can propagate easily leading to serious problems for the calculation of subsequent priorities and texture estimation. Hence, the used texture synthesis algorithm plays a key role on the inpainting quality as well as on the priority function. For example, in the NLM method, an irrelevant structure has been produced in the sea (as an extension of the bushes) which does not look very natural.

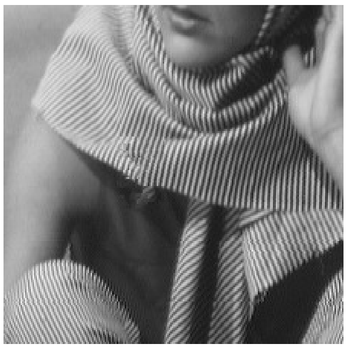
6.6.3 Application to missing region completion

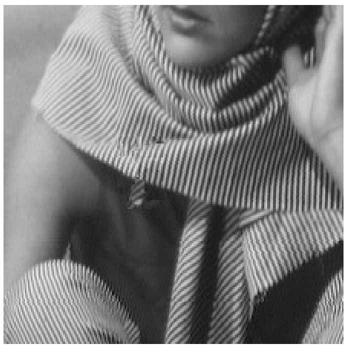
This second set of experiments aims at assessing the performance of the proposed methods in an application where a region (or a block) is missing in the image, e.g., in a loss concealment application. In this case, the missing region ground truth pixel values are known, hence the results can be analysed quantitatively.

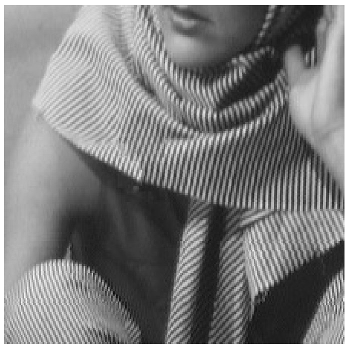
Fig. 6.5, Fig. 6.6, Fig. 6.7 and Fig. 6.8 show a comparison between proposed NMF and LLE based neighbor embedding methods with TM, ATM, and NLM for missing region completion. The effectiveness of neighbor embedding methods can clearly be observed, both visually and in terms of PSNR, from the inpainted images especially for transition areas from textures to edges and edges to textural regions. On the other hand, one can observe that the proposed "edge term" in the priority function fails when there are inpainting errors in filling-in regions which produce irrelevant structures (leading to unwanted edges detected with the Canny edge detector) resulting in lower PSNR results. For example, for the House image in Fig. 6.8, the TM method with augmented priority leads to lower PSNR results because of the propagation of a catastrophic inpainting error. This can be prevented by using a more robust texture synthesis method. Here again one can observe the importance of the fill-in method in addition to the priority function. The main idea proposed in [43] that is the importance of fill-in order hence can significantly be improved by using some optimization techniques for missing pixels estimation when compared to TM, ATM, or NLM based methods. Please note that here the tested missing regions are larger than a block size conventionally used for image or video coding, hence one can expect also good results for missing block completion such as in an error concealment application.

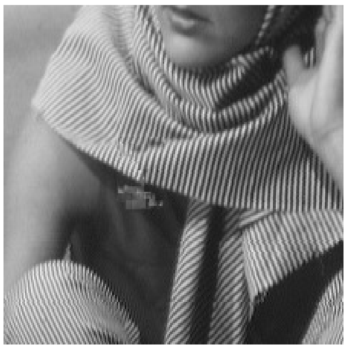
(a) Ground truth

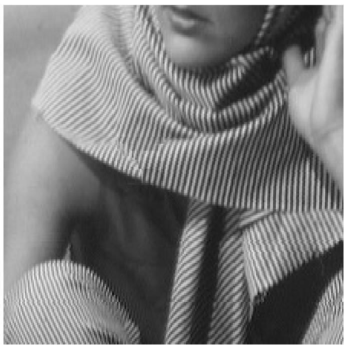

(b) 21.25 dB


(c) 32.89 dB


(d) 33.74 dB


(e) 33.90 dB


(f) 31.75 dB


(g) 34.71 dB

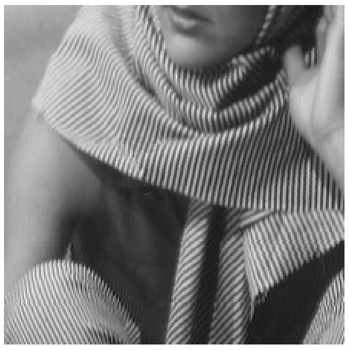
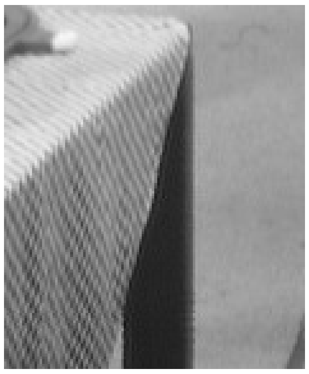
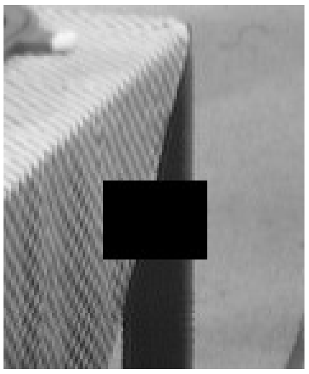
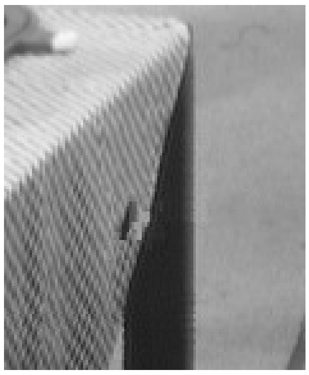
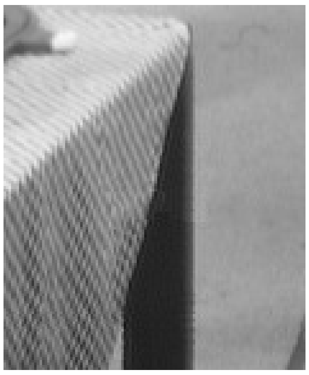

(h) 35.06 dB

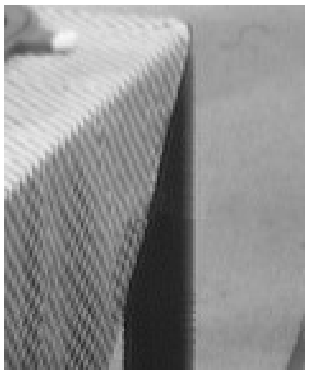
Figure 6.5: Missing region completion PSNR results for Barbara image (I) - (a) Ground truth image, (b) inpainting mask, (c) TM with priority of [43], (d) TM with augmented priority, (e) ATM with augmented priority, (f) NLM with augmented priority, (g) NMF with augmented priority, and (h) LLE with augmented priority.

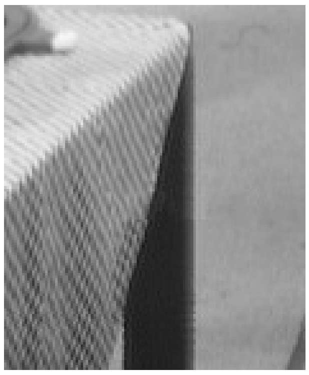

(a) Ground truth

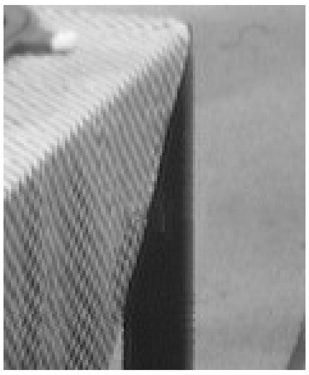

(b) 18.62 dB


(c) 33.70 dB


(d) 35.76 dB


(e) 35.95 dB


(f) 35.87 dB


(g) 36.21 dB

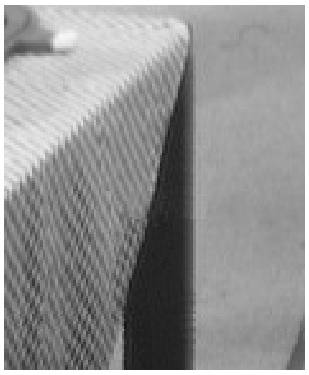

(h) 36.46 dB

Figure 6.6: Missing region completion PSNR results for Barbara image (II) - (a) Ground truth image, (b) inpainting mask, (c) TM with priority of [43], (d) TM with augmented priority, (e) ATM with augmented priority, (f) NLM with augmented priority, (g) NMF with augmented priority, and (h) LLE with augmented priority.

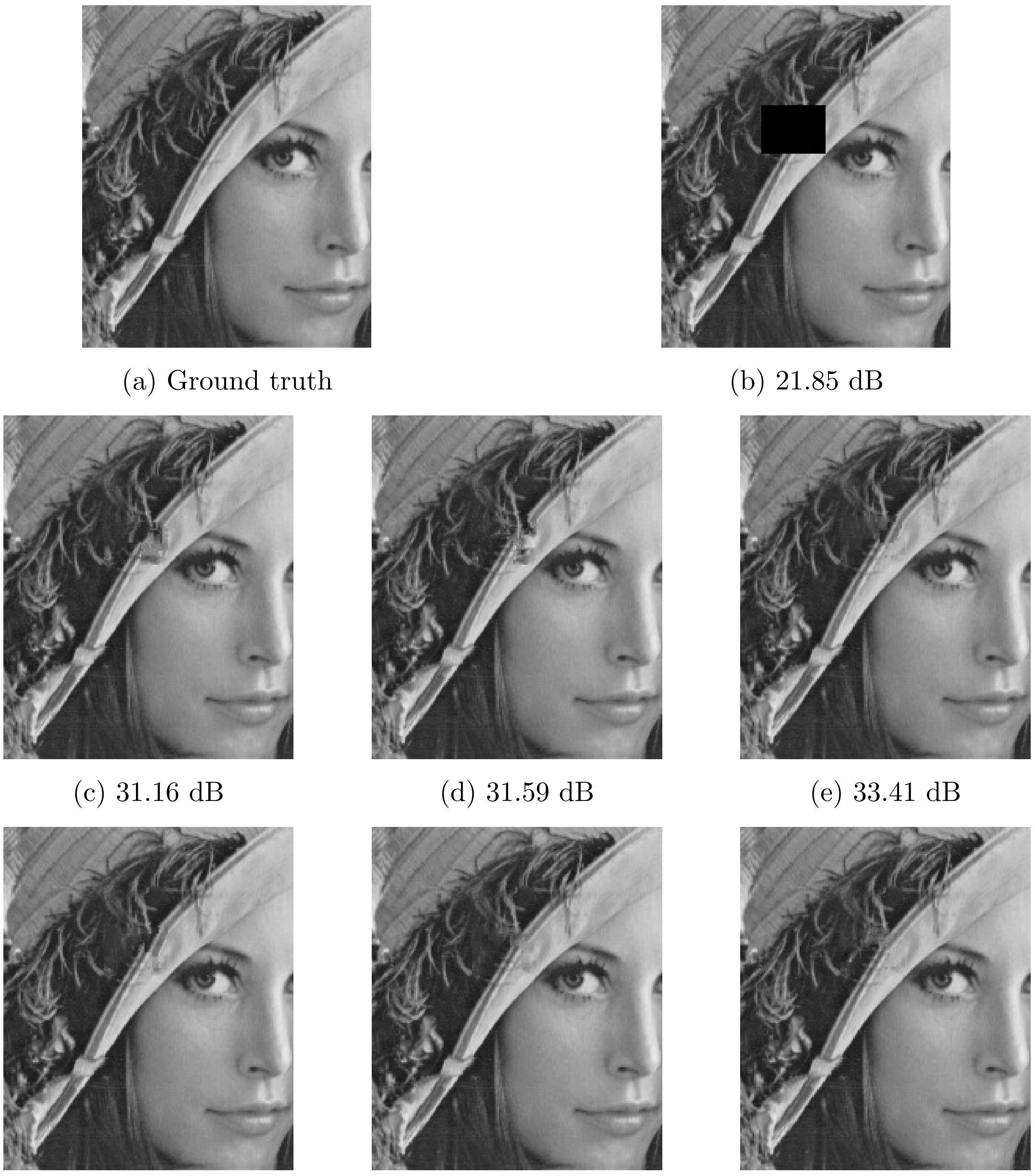
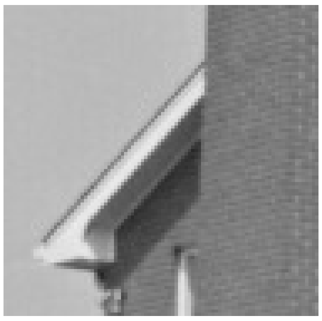
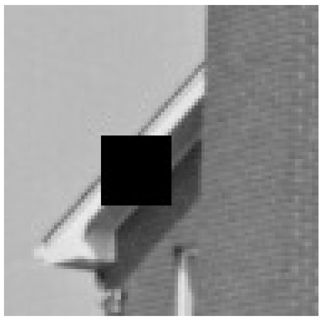
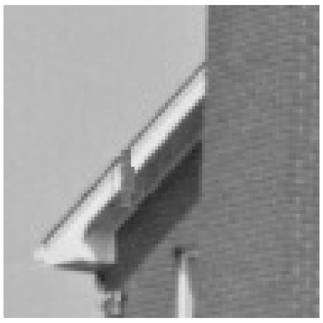
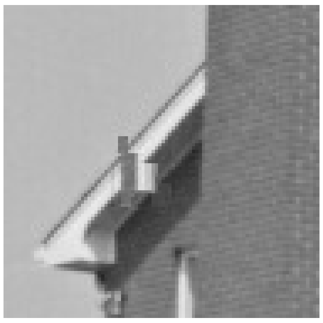


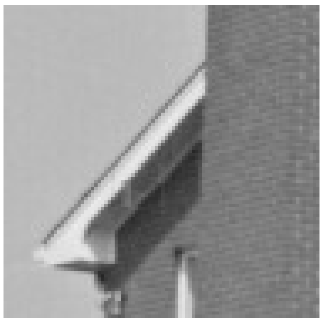
Figure 6.7: Missing region completion PSNR results for Lena image - (a) Ground truth image, (b) inpainting mask, (c) TM with priority of [43], (d) TM with augmented priority, (e) ATM with augmented priority, (f) NLM with augmented priority, (g) NMF with augmented priority, and (h) LLE with augmented priority.

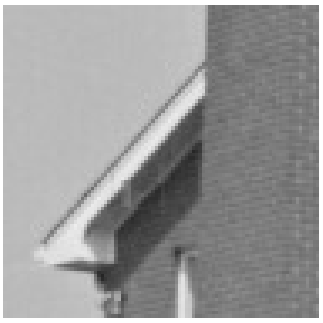

(a) Ground truth

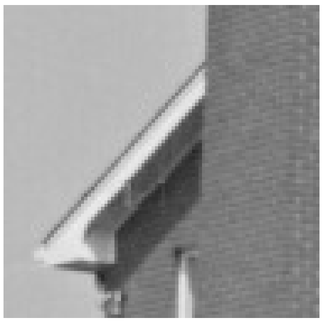

(b) 17.43 dB


(c) 32.70 dB


(d) 27.77 dB


(e) 39.02 dB


(f) 39.00 dB


(g) 39.04 dB

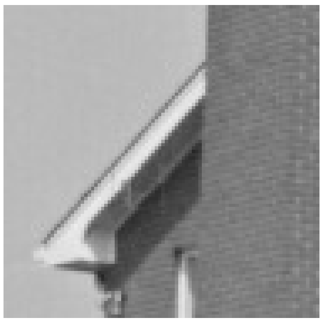

(h) 38.97 dB

Figure 6.8: Missing region completion PSNR results for House image - (a) Ground truth image, (b) inpainting mask, (c) TM with priority of [43], (d) TM with augmented priority, (e) ATM with augmented priority, (f) NLM with augmented priority, (g) NMF with augmented priority, and (h) LLE with augmented priority.

6.6.4 Comparison with other inpainting methods

In this section, the performance of the proposed neighbor embedding (with LLE and NMF) based inpainting algorithms using the augmented patch priority computation have been assessed on several test images, and compared with the state-of-the-art diffusion-based approaches in [150, 151] and exemplar-based methods in [43, 159].[1] Fig. 6.9, Fig. 6.10, and Fig. 6.11 illustrate some of the results obtained for the test images: "Sydney", "Terrasse", and "Bike" respectively. We selected natural images with large holes to be filled-in in order to test our proposed inpainting algorithms for object removal. Since the holes to be filled-in are quite large, the inpainting task is not so easy and diffusion-based approaches introduce blur into the image. When we compare our method with the other exemplar-based methods including [43] and [159], we see comparable results with the natural looking and structure preserving capacity of the proposed method in this chapter.

6.6.5 Performance analysis of subspace mappings on LLE inpainting

Fig. 6.12 compares the inpainting results of Bungee obtained by using a simple LLE and LLE with parametric mapping method using the edge term augmented priority function. In this simple experimentation, K is fixed to 10 and patch size is 9×9 pixels. One can observe that the LLE algorithm with learned mappings gives comparable (and even more natural) inpainting result when compared with a simple LLE. The inpainted patches are relatively better approximated (in the sense of natural looking) by the learned mappings which helps selecting more appropriate K-NN patches. As a final illustration, Fig. 6.13 shows the inpainting results obtained for the test images (Sydney, Terrasse, and Bike) in the same setup.

6.7 Conclusion

In this chapter, we have extended our work in Chapter 5 by describing two new exemplar-based image inpainting methods based on neighbor embedding algorithms with an augmented patch priority function. Experimental results show that the proposed methods offer better inpainting quality when compared to other exemplar-based techniques using template matching, average template matching, and the non-local means approach. The proposed methods can be seen as a generalization of template matching, in the sense that they search to approximate the template signal (via a constrained optimization in contrary to heuristic based methods such as ATM and

[1]The author would like to thank Dr. Olivier Le Meur for providing the inpainting results of the methods in [43, 150, 151, 159].

NLM) with a linear combination of colocated pixels in similar patches, whereas the template matching approximates the template with one similar patch with a weighting coefficient equal to one. The simulation results show significant performance gain (in different applications such as missing region completion in a context of loss concealment or large object removal in a context of image editing) of the proposed methods against (weighted) template matching and other exemplar- and diffusion-based inpainting approaches in the literature. A heuristic extension of the LLE based inpainting method, which exploits the neighborhood and local geometry preserving properties of LLE, has finally been proposed through parametric mapping functions. These mapping functions are learned from a set of training image patches taken from the source image, and they allow another way of selecting the K patches to be used in the inpainting process, i.e., leading relatively better approximated and natural looking inpainted images.

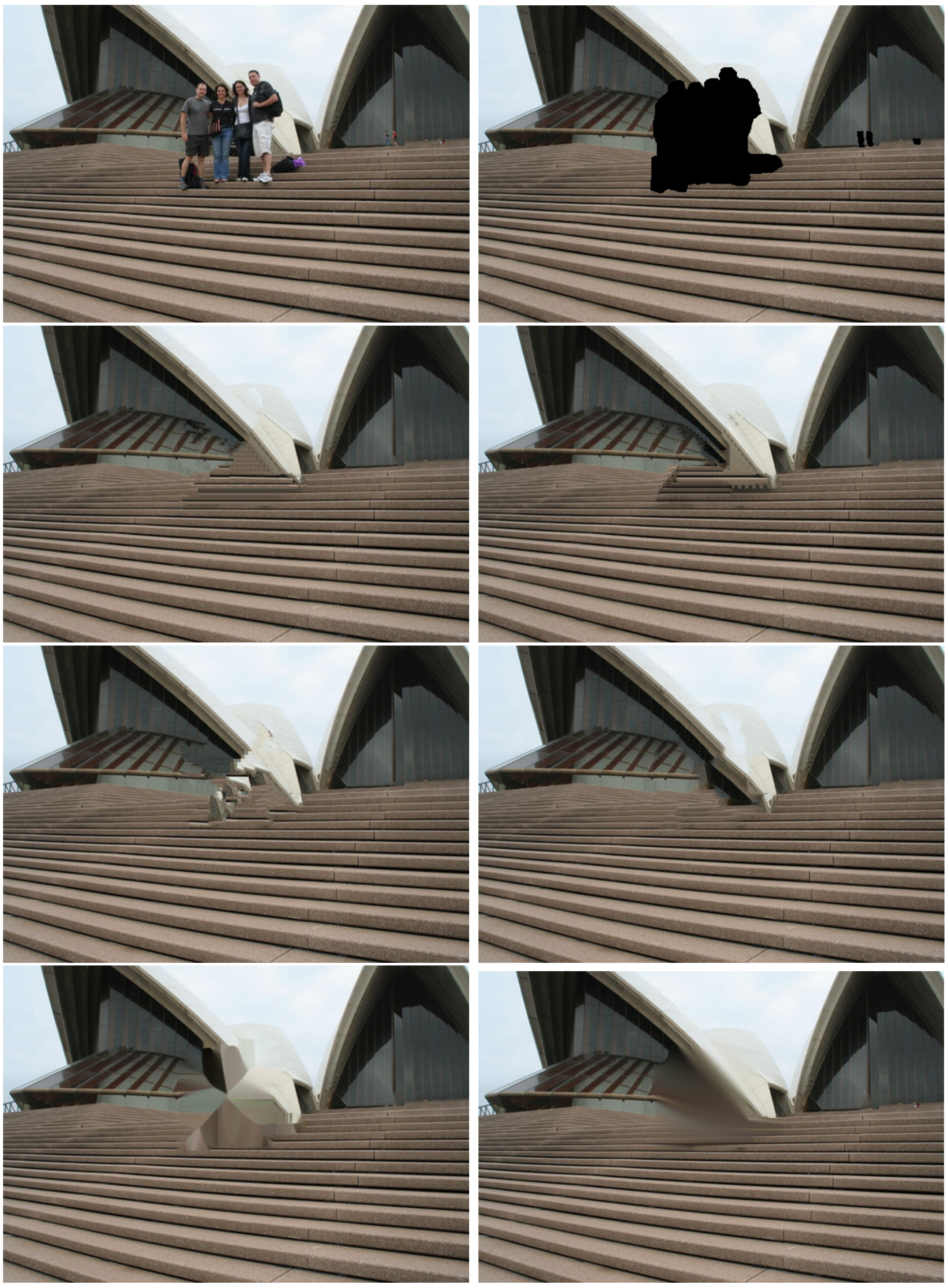


Figure 6.9: Inpainting results for Sydney image - From left-to-right top-to-bottom: Original image, inpainting mask; our method with LLE, our method with NMF; method in [43], method in [159]; method in [150], and method in [151].

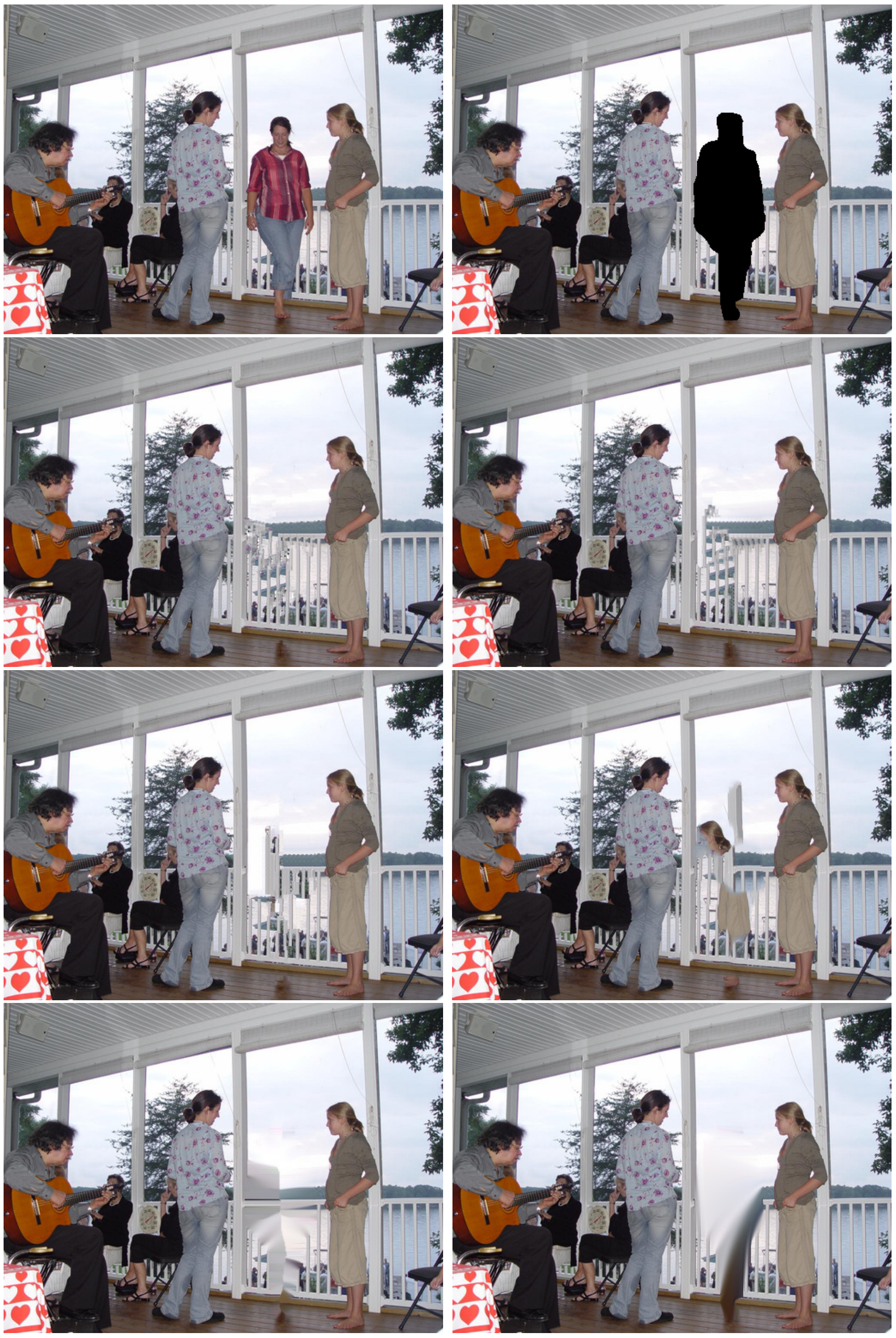


Figure 6.10: Inpainting results for Terrasse image - From left-to-right top-to-bottom: Original image, inpainting mask; our method with LLE, our method with NMF; method in [43], method in [159]; method in [150], and method in [151].

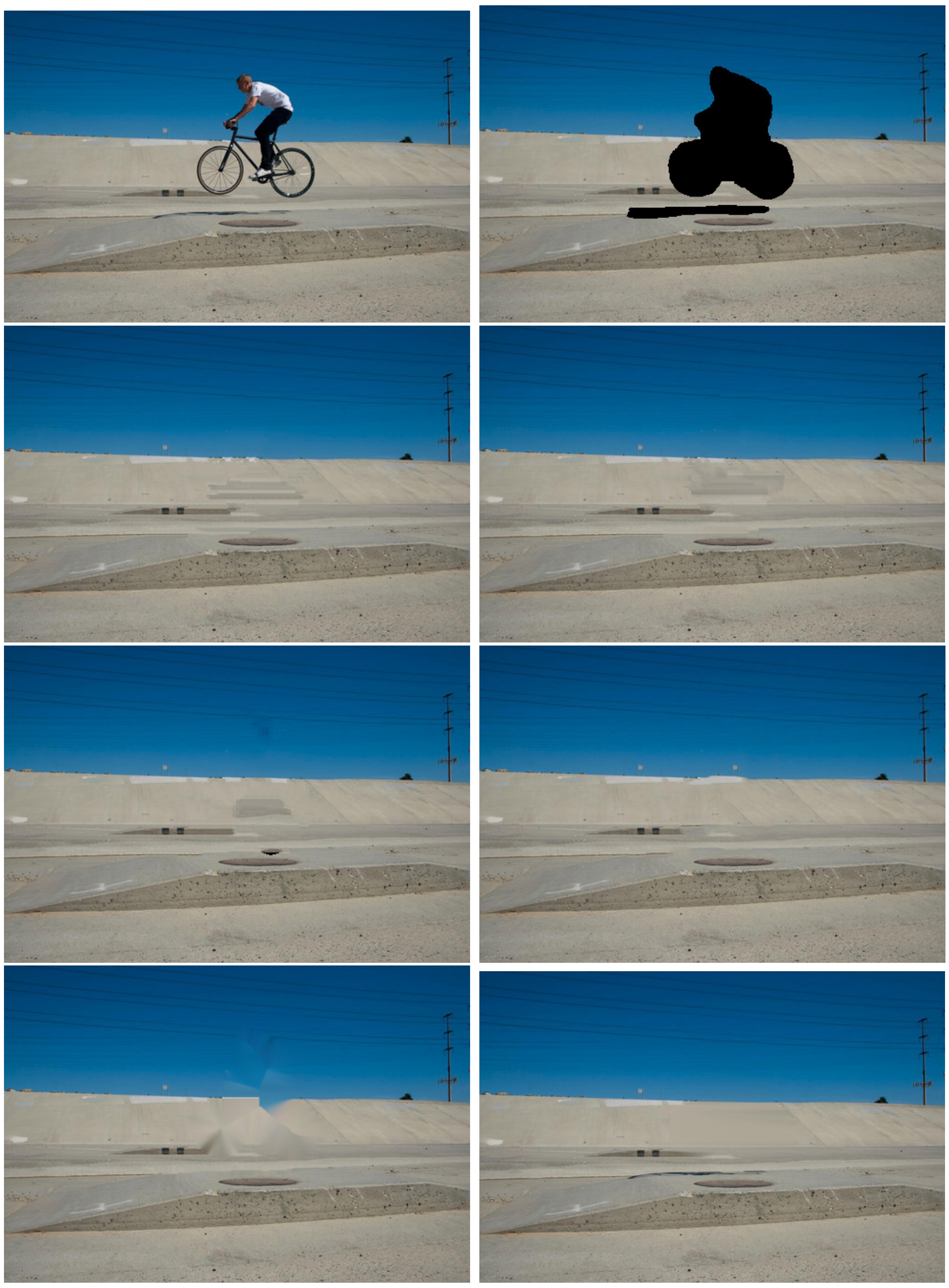


Figure 6.11: Inpainting results for Bike image - From left-to-right top-to-bottom: Original image, inpainting mask; our method with LLE, our method with NMF; method in [43], method in [159]; method in [150], and method in [151].

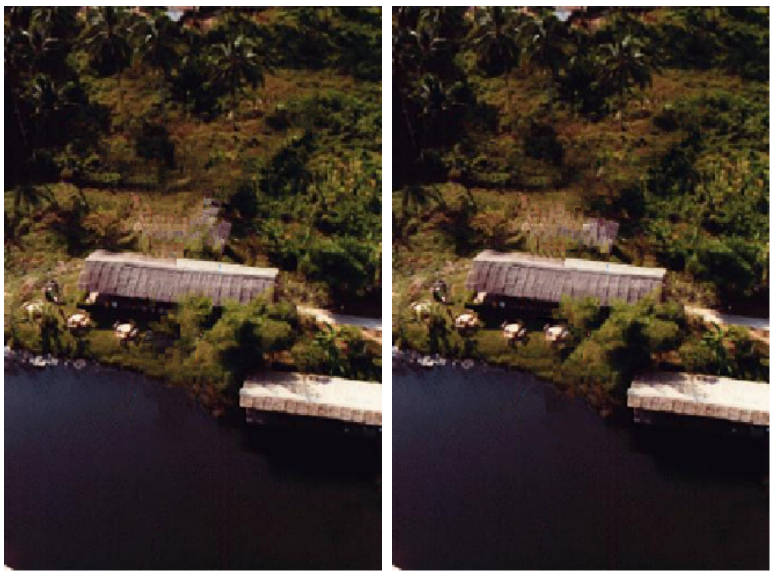


Figure 6.12: Effect of subspace mappings on LLE inpainting - (left) Inpainting with LLE with augmented priority, and (right) inpainting with LLE using learned mappings with augmented priority. (K is fixed to 10.)

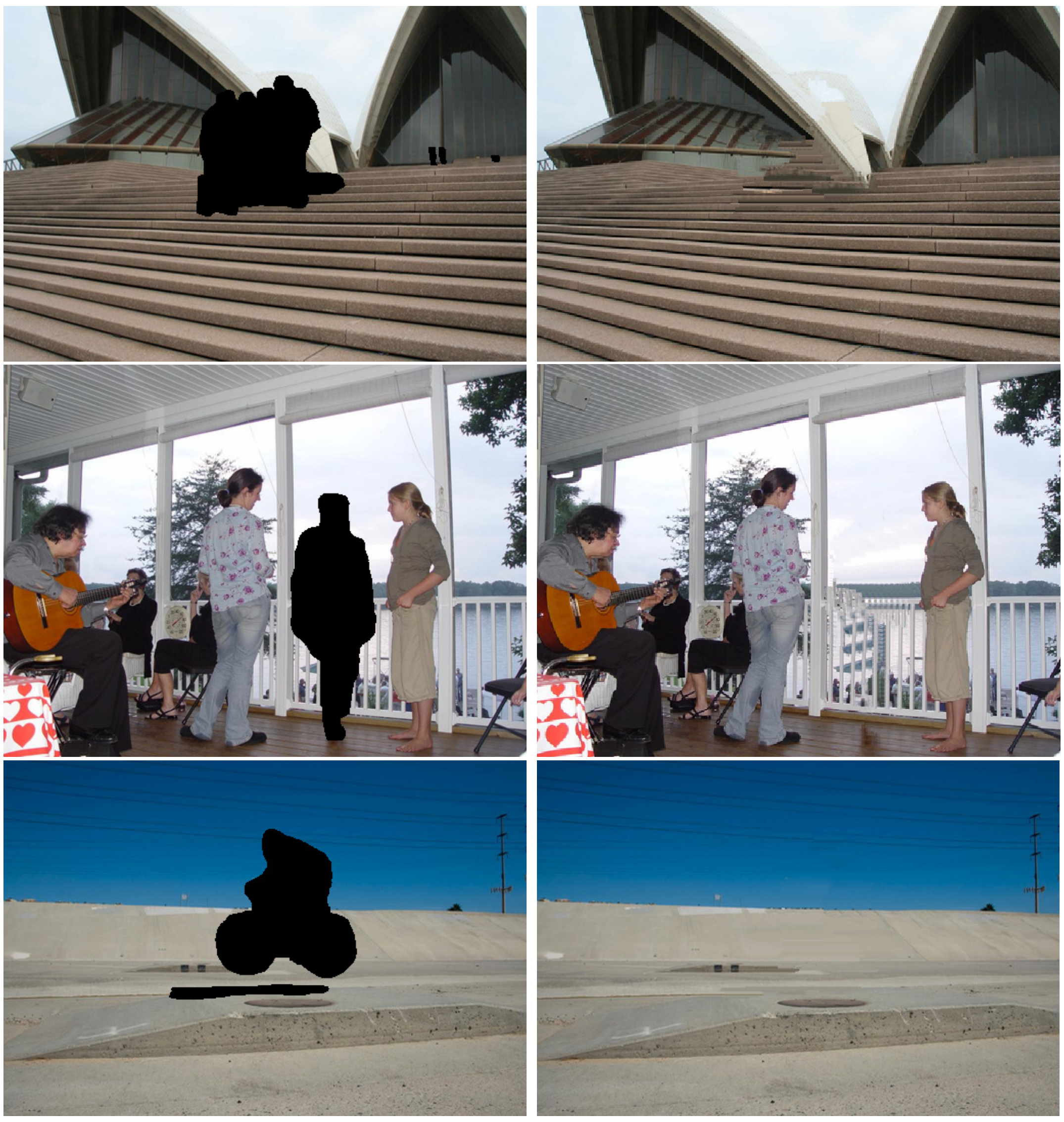


Figure 6.13: Inpainting results for the test images using subspace mappings with LLE - From left-to-right per image: Inpainting mask and the result of inpainting.

Chapter 7

Conclusion and Perspectives

In this manuscript we have placed the image prediction (i.e., predictive coding) problem into different frameworks by formulating an optimization on the approximation of the template signal. The proposed methods in this work can be seen as extensions of the well-known template matching method. We start with placing the image prediction problem into sparse representations framework by approximating the template with a sparsity constraint. The proposed sparse prediction method with locally and adaptive dictionaries show great performance improvement when compared to a model which uses static dictionaries (such as DCT or DFT), and also to template matching. On the other hand, the limitations and drawbacks of the sparse prediction method have later led us to envisage some other means to solve the same optimization problem with different techniques and constraints. We have later put the image prediction problem into a dictionary learning framework by adapting conventional dictionary learning approaches for image prediction. To the best of our knowledge, the proposed online prediction dictionary learning method is quite novel with respect to traditional dictionary learning algorithms, and it offers better performance when compared to sparse prediction and H.264/AVC intra. A simple version of this method has also been proposed which can be regarded as a simple least-squares optimization for prediction through learning. Finally, we have integrated the image prediction problem into a neighbor embedding framework by adapting two different dimensionality reduction methods. The experimental results demonstrate the effectiveness of the underlying idea in both image prediction and inpainting applications.

Regarding to the sparse prediction method in Chapter 3, the proposed method here is indeed original but it suffers from the iterative sparse approximation algorithms such as OMP. As OMP iterates for approximating the template, the residue on the template gets less correlated with the residue of the block to be predicted, hence the final calculated weighting coefficients are in general not well optimized for prediction. Moreover, OMP recomputes all the coefficients assigned to the selected atoms at each iteration. At any iteration, a simple error on selecting an atom would lead to a complete change in the previous and also upcoming calculation of the coefficients which will definetely not be optimum for prediction. One sub-optimal solution could be the use of the MP method instead of OMP. Inspite the fact that MP also approximates residue signals at each iteration (the same drawback still holds as in OMP), it does only update one coefficient at a time. So one might expect relatively better performance. Another interesting solution would be the use of a tree-structured pursuit algorithm, instead of MP or OMP, which can handle adaptively the atom dependencies of two different parts of the dictionary, i.e., corresponding to the template ($\mathbf{A}_c$) and the unknown block ($\mathbf{A}_t$), not only in the spatial domain but also in the residual domain at each iteration of the algorithm.

The sparse prediction algorithm (with OMP) finally turned out to be not really optimal for the prediction problem. However it has some important aspects and perspectives for future study. The locally adaptive dictionary concept indeed is new and useful, and can be replaced with traditional dictionaries in different applications such as for image denoising. In a denoising application all the pixel values of a noisy block needs to be approximated and usually represented in a sparse domain. Thus proposed dictionary concept suits very well for this purpose. A locally adaptive dictionary can be constructed for a noisy block to be denoised by extracting either already denoised texture patches or even the noisy ones from a large neighborhood of that block. The rest is just a question of adapting the constrained optimization and its parameters for this special application with locally adaptive dictionaries.

Regarding to the online prediction dictionary learning method in Chapter 4, the proposed method is novel but it has some underlying issues which should be further investigated. First of all, the imposed sparsity constraint in sparse coding step has not really been reflected to the prediction performance, as the simplified method shows comparable results. This might be caused by fixing the dictionary $\mathbf{A}_c$ and constraining it to be an orthonormal basis, which is indeed too restrictive. It is very crucial to increase the effect of sparsity constraint on this method. One solution would be employing one of the classical dictionary learning algorithms with strict sparsity constraints to learn an optimum $\mathbf{A}_c$ using the training set $\mathbf{T}_c$, however in exchange for the computational complexity. In this way, this method could be made more dependent on the sparsity constraint together with a block-coordinate descent update (which will preserve the sparseness of the learned model for $\mathbf{A}_c$) of the subdictionary $\mathbf{A}_t$. In this kind of specific application for prediction, the computational load can be reduced greatly using recursive learning methods by initializing the dictionaries $\mathbf{A}_c$ and $\mathbf{A}_t$ with warm restarts, i.e., the dictionaries calculated for the previously predicted block.

When we talk about dictionary learning for prediction, the drawbacks of sparse approximation methods (as in Chapter 3) still hold a problematic place. To overcome these limitations, a possible future study which might extend this framework would be the use of an iteration-tuned and tree-structured approach. In this sense, tuning (or learning) level based dictionaries characterized by each iteration of the sparse approximation method (such as OMP) would help improving the performance. This process can be seen as a decorrelation process of recursive iterations (and the coefficients) of the sparse coding step for obtaining optimized prediction dictionaries for residual signals (such that the k^{th} branch of the tree-structured dictionaries is constructed with $\mathbf{A}_c^k$ and $\mathbf{A}_t^k$ for the coefficients of iteration k of the sparse coding step).

Online learning is crucial in many applications. Once the problem is posed and its formulation is defined, the rest is just a question of collecting relevant training samples. Extracting local texture patches as training samples is very efficient for capturing the local content contained in an image. In this way a limited number of samples would be enough for an efficient dictionary model. The proposed learning technique here can also be applied for image inpainting. Although we have proposed some sort of simple learning method for inpainting in Chapter 6, there is still room for research on this topic for an efficient learning mechanism for improved inpainting results.

Regarding to the neighbor embedding methods in Chapters 5–6, we think that the ideas as well as the contibutions given in these chapters should open new doors and show new directions for many image processing applications. Much work has been carried out especially for image prediction and image inpainting, however there are still a few remarks which should be analysed carefully. For example, the selection of K nearest neighbors is carried out using the template signal, and these neighbors may not be the neighbors of the block to be predicted, or the pixels to be inpainted. This is a very important point and one can partially solve this problem by employing some learning methods using the local context (the texture pathces in a close neighborhood). Another important point is related with NMF and LLE based methods and their variants in the literature. For example, the NMF framework has several variants of NMF algorithms based on multiplicative updates (as utilized in this study), projected gradients, graph regularization, and so on. Although these variants intend to solve almost the same optimization problem, either the ways which lead to solutions are quite different or they have different constraints between a point and its neighbors. Hence one should further take consideration of these solutions to analyse which solution is better for the applied texture synthesis problem. As a final remark, the presented neighbor embedding ideas can further be extended to other texture synthesis based applications such as image denoising and super-resolution.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical J.*, vol. 27, pp. 379–423 and 623–656, July and October 1948. 2

[2] E. E. Catmull, "A subdivision algorithm for computer display of curved surfaces," Ph.D. dissertation, Univ. of Utah, 1974. 16

[3] J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images," *Commun. ACM*, vol. 19, no. 10, pp. 542–547, Oct. 1976. 16

[4] L. Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics 2009, State of the Art Report, EG-STAR*, 2009. 16, 17, 18, 21

[5] Y. Liu, W.-C. Lin, and J. H. Hays, "Near-regular texture analysis and manipulation," *ACM Trans. Graphics (SIGGRAPH 2004)*, vol. 23, no. 3, pp. 368–376, Aug. 2004. 17

[6] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *J. Royal Statistical Soc. B*, vol. 36, no. 2, pp. 192–236, 1974. 18

[7] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 5, no. 1, pp. 25–39, Jan. 1983. 18

[8] A. C. Popat, "Conjoint probabilistic subband modeling," Ph.D. dissertation, Massachusetts Institute of Technology, 1997. 18

[9] R. Paget, "Strong markov random field model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 3, pp. 408–413, Mar. 2004. 18

[10] S. C. Zhu, Y. Wu, and D. Mumford, "Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling," *Int. J. Comput. Vision*, vol. 27, no. 2, pp. 107–126, Apr. 1998. 18

[11] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. IEEE Int. Conf. Computer Vis.*, vol. 2, 1999, pp. 1033–1038. 18, 19, 109, 142

[12] L. Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. ACM Comp. Graphics Interactive Tech. (SIGGRAPH 2000)*, 2000, pp. 479–488. 19, 20, 142

[13] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Kluwer Academic, 1992. 19, 71

[14] L. Y. Wei, "Deterministic texture analysis and synthesis using tree structure vector quantization," in *Proc. Brazilian Symp. Compt. Graphics Image Process.*, 1999, pp. 207–213. 19

[15] M. Ashikhmin, "Synthesizing natural textures," in *ACM Symp. Interactive 3D Graph.*, 2001, pp. 217–226. 19, 142

[16] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped textures," in *Proc. ACM Comp. Graphics Interactive Tech. (SIGGRAPH 2000)*, 2000, pp. 465–470. 21, 22

[17] L. Liang, C. Liu, Y. Q. Xu, B. Guo, and H. Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Trans. Graphics (SIGGRAPH 2001)*, vol. 20, no. 3, pp. 127–150, Jul. 2001. 21, 22

[18] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. ACM Comp. Graphics Interactive Tech. (SIGGRAPH 2001)*, 2001, pp. 341–346. 21, 22

[19] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graphics (SIGGRAPH 2003)*, vol. 22, no. 3, pp. 277–286, Jul. 2003. 21, 22

[20] Y. Q. Xu, B. Guo, and H. Shum, "Chaos mosaic: Fast and memory efficient texture synthesis," Microsoft, USA, Tech. Rep. MSR-TR-2000-32, Apr. 2000. 21

[21] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 795–802, Jul. 2005. 21

[22] R. Paget and I. D. Longstaff, "Texture synthesis via a noncausal nonparametric multiscale markov random field," *IEEE Trans. Image Process.*, vol. 7, no. 6, pp. 925–931, Jun. 1998. 21

[23] J. S. D. Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images," in *Proc. ACM Comp. Graphics Interactive Tech. (SIGGRAPH 1997)*, 1997, pp. 361–368. 21

[24] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. ACM Comp. Graphics Interactive Tech. (SIGGRAPH 1995)*, 1995, pp. 229–238. 21

[25] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Computer Vis.*, vol. 40, no. 1, pp. 49–71, 2000. 21

[26] L. Y. Wei, J. Han, K. Zhou, H. Bao, B. Guo, and H. Y. Shum, "Inverse texture synthesis," *ACM Trans. Graphics (SIGGRAPH 2008)*, vol. 27, no. 3, pp. 1–9, Aug. 2008. 21, 23

[27] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003. 24, 26

[28] G. J. Sullivan, P. N. Topiwala, and A. Luthra, "The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions," in *Proc. SPIE Conf. App. Digital Image Process. XXVII*, 2004. 24

[29] "Draft ITU-T Rec. and Final Draft Int. Standard of Joint Video Spec." ITU-T Rec. H.264. ISO/IEC 14496-10 AVC, 2003. 24

[30] M. Wien, "Variable block-size transforms for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 604 – 613, Jul. 2003. 26

[31] C. Dai, O. D. Escoda, P. Yin, X. Li, and C. Gomila, "Geometry-adaptive block partitioning for intra prediction in image video coding," in *Proc. IEEE Int. Conf. Image Process.*, vol. 6, 2007, pp. 85–88. 26

[32] A. Robert, I. Amonou, and B. P. Popescu, "Improving intra mode coding in H.264/AVC through block oriented transforms," in *Proc. IEEE Workshop Mult. Signal Process.*, 2006, pp. 382–386. 26

[33] S. Matsuo and S. Takamura, "Extension of intra prediction using multiple reference lines," in *ITU-T SG16/Q.6 VCEG-AF05*, 2007. 27

[34] T. Tsukuba, T. Yamamoto, Y. Tokumo, and T. Aono, "Adaptive multidirectional intra prediction," in *ITU-T Q.6/SG16 VCEG-AG05*, 2007. 27

[35] J. Yang, B. Yin, Y. Sun, and N. Zhang, "A block-matching based intra frame prediction for H.264/AVC," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2006, pp. 705–708. 27

[36] L. Song, Y. Xu, C. Xiong, and L. Traversoni, "Improved intra-coding methods for H.264/AVC," *EURASIP J. Advances in Signal Process.*, 2009. 27

[37] Y. Piao and H. W. Park, "An adaptive divide-and-predict coding for intra-frame of H.264/AVC," in *Proc. IEEE Int. Conf. Image Process.*, 2009, pp. 3421–3424. 27

[38] Y. Ye and M. Karczewicz, "Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning," in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 2116–2119. 27

[39] T. K. Tan, C. S. Boon, and Y. Suzuki, “Intra prediction by template matching,” in *Proc. IEEE Int. Conf. Image Process.*, 2006, pp. 1693–1696. 27, 47

[40] ——, “Intra prediction by averaged template matching predictors,” in *Proc. IEEE Consumer Comm. Network. Conf.*, 2007, pp. 405–409. 27, 28, 47

[41] Y. Zheng, P. Yin, O. D. Escoda, X. Li, and C. Gomila, “Intra prediction using template matching with adaptive illumination compensation,” in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 125–128. 27, 28, 47

[42] Y. Guo, Y. K. Wang, and H. Li, “Priority-based template matching intra prediction,” in *Proc. IEEE Int. Conf. Multimedia Expo.*, 2008, pp. 1117–1120. 27, 28, 47

[43] A. Criminisi, P. Perez, and K. Toyama, “Region filling and object removal by examplar-based image inpainting,” *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004. 28, 47, 142, 143, 144, 145, 146, 154, 155, 157, 158, 159, 160, 161, 162, 164, 165, 166

[44] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, “Simultaneous structure and texture image inpainting,” *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 882–889, Aug. 2003. 28, 47

[45] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and C. S. Boon, “Inter frame coding with template matching spatio-temporal prediction,” in *Proc. IEEE Int. Cong. Image Process.*, 2004, pp. 465–468. 28, 139

[46] A. Wong and J. Orchard, “A nonlocal-means approach to exemplar-based inpainting,” in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 2600–2603. 28, 51

[47] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, “Non-local sparse models for image restoration,” in *Proc. IEEE Int. Conf. Computer Vis.*, 2009, pp. 2272–2279. 28, 51

[48] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” in *Proc. IEEE Comp. Soc. Conf. Comp. Vis. Pattern Recogn.*, vol. 2, 2005, pp. 60–65. 28, 51

[49] J. Wang, Y. Guo, Y. Ying, Y. Liu, and Q. Peng, “Fast non-local algorithm for image denoising,” in *Proc. IEEE Int. Conf. Image Process.*, 2006, pp. 1429–1432. 28, 51

[50] M. Mahmoudi and G. Sapiro, “Fast image and video denoising via nonlocal means of similar neighborhoods,” *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 839–842, Dec. 2005. 28, 51

[51] M. Turkan and C. Guillemot, “Sparse approximation with adaptive dictionary for image prediction,” in *Proc. IEEE Int. Conf. Image Process.*, 2009, pp. 25–28. 31

[52] ——, "Image prediction: Template matching vs. sparse approximation," in *Proc. IEEE Int. Conf. Image Process.*, 2010, pp. 789–792. 31

[53] A. Dremeau, M. Turkan, C. Herzet, C. Guillemot, and J. J. Fuchs, "Spatial intra-prediction based on mixtures of sparse representations," in *Proc. IEEE Workshop Mult. Signal Process.*, 2010, pp. 345–349. 31

[54] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006. 31, 68, 69

[55] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 27–35, Jan. 2009. 31, 68

[56] G. Peyre, "Sparse modeling of textures," *J. Math. Imag. Vis.*, vol. 34, no. 1, pp. 17–31, May 2009. 31, 68

[57] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, Jan. 2008. 32, 68

[58] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *SIAM Multiscale Model. Simul.*, vol. 7, no. 1, pp. 214–241, Apr. 2008. 32, 68, 70

[59] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *J. Visual Commun. Image Represent.*, vol. 19, no. 4, pp. 270–283, May 2008. 32, 68

[60] L. Peotta, L. Granai, and P. Vandergheynst, "Image compression using an edge adapted redundant dictionary and wavelets," *Signal Process.*, vol. 86, no. 3, pp. 444–456, Mar. 2006. 32, 68

[61] M. J. Fadili, J. L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *Computer J.*, vol. 52, no. 1, pp. 64–79, Jul. 2007. 32, 68

[62] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *Proc. IEEE Comp. Soc. Conf. Comp. Vis. Pattern Recog.*, 2008, pp. 1–8. 32, 68

[63] H. Y. Liao and G. Sapiro, "Sparse representations for limited data tomography," in *Proc. IEEE Int. Symp. Biomed. Imag.: Nano Macro (ISBI)*, 2008, pp. 1375–1378. 32, 68

[64] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive Approx.*, vol. 13, no. 1, pp. 57–98, 1997. 33

[65] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Comput.*, vol. 20, no. 1, pp. 33–61, 1998. 33, 69, 154

[66] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993. 33, 40, 69, 152

[67] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Asimolar Conf. Signals Systems Compt.*, 1993, pp. 40–44. 33, 69, 154

[68] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE Signal Process. Lett.*, vol. 9, no. 4, pp. 137–140, Apr. 2002. 33, 36

[69] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," 2007. 33, 35

[70] T. Blumensath and M. E. Davies, "Gradient pursuits," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2370–2382, Jun. 2008. 33

[71] G. Rath and C. Guillemot, "Sparse approximation with an orthogonal complementary matching pursuit algorithm," in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, 2009, pp. 3325–3328. 33, 34

[72] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *J. Royal Statistical Soc. B*, vol. 58, no. 1, pp. 267–288, 1996. 36

[73] S. Mallat, *A Wavelet Tour of Signal Processing*, 3rd ed. Academic Press, 2008. 37, 40

[74] E. J. Candes and D. L. Donoho, *Curvelets - A Surprisingly Effective Nonadaptive Representation for Objects with Edges.* Vanderbilt University Press, (Curve and Surface Fitting: St-Malo, pp. 105-120), 1999. 37

[75] M. N. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, Dec. 2005. 37

[76] D. Labate, W. Lim, G. Kutyniok, and G. Weiss, "Sparse multidimensional representation using shearlets," in *Proc. SPIE: Wavelets XI*, vol. 5914, 2005, pp. 254–262. 37

[77] E. L. Pennec and S. Mallat, "Sparse geometric image representations with bandelets," *IEEE Trans. Image Process.*, vol. 14, no. 4, pp. 423–438, Apr. 2005. 37, 56

[78] D. L. Donoho, "Wedgelets: Nearly minimax estimation of edges," *Annals of Statistics*, vol. 27, no. 3, pp. 859–897, 1999. 37

[79] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proc. IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010. 40, 68, 70, 74

[80] J. B. Allen and L. R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, pp. 1558–1564, Nov. 1977. 40

[81] D. Gabor, "Theory of communication," *J. Inst. Electr. Eng.*, vol. 93, no. 26, pp. 429–457, Nov. 1946. 40

[82] M. J. Bastiaans, "Gabor's expansion of a signal into Gaussian elementary signals," *Proc. IEEE*, vol. 68, no. 4, pp. 538–539, Apr. 1980. 40

[83] A. Janssen, "Gabor representation of generalized functions," *J. Math. Anal. Applic.*, vol. 83, no. 2, pp. 377–394, 1981. 40

[84] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, Jul. 1989. 40

[85] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 7, pp. 710–732, Jul. 1992. 40

[86] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, vol. 5, 1999, pp. 2443–2446. 41, 70

[87] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, "Learning unions of orthonormal bases with thresholded singular value decomposition," in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, vol. 5, 2005, pp. 293–296. 41, 70, 71, 72

[88] O. G. Sezer, O. Harmanci, and O. G. Guleryuz, "Sparse orthonormal transforms for image compression," in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 149–152. 41, 70, 71, 72, 73, 96

[89] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006. 41, 70

[90] I. T. Jolliffe, *Principle Component Analysis*, 2nd ed. Springer, 2002. 41, 104

[91] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1945–1959, Dec. 2005. 41

[92] A. Martin, J. J. Fuchs, C. Guillemot, and D. Thoreau, "Sparse representation for image prediction," in *European Signal Process. Conf.*, 2007. 41, 44, 53, 54

[93] J. J. Fuchs, "On the application of the global matched filter to DOA estimation with uniform circular arrays," *IEEE Trans. Signal Process.*, vol. 49, no. 4, pp. 702–709, Apr. 2001. 43

[94] S. Mallat and F. Falzon, "Analysis of low bit rate image transform coding," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 1027–1042, Apr. 1998. 56

[95] B. Mailhe, R. Gribonval, F. Bimbot, and P. Vandergheynst, “A low complexity orthogonal matching pursuit for sparse signal approximation with shift-invariant dictionaries,” in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, 2009, pp. 3445–3448. 63

[96] T. Blumensath and M. E. Davies, “In greedy pursuit of new directions: (Nearly) orthogonal matching pursuit by directional optimisation,” in *European Signal Process. Conf.*, 2008. 63

[97] C. La and M. N. Do, “Tree-based orthogonal matching pursuit algorithm for signal reconstruction,” in *Proc. IEEE Int. Conf. Image Process.*, 2006, pp. 1277–1280. 66

[98] J. Wang, Q. Wan, A. Huang, and T. Gan, “Tree-based multiscale pursuit,” in *Proc. IEEE Int. Conf. Commun. Circuits Syst.*, 2009, pp. 521–524. 66

[99] A. J. Bernal and S. E. Ferrando, “Tree structured pursuit for simultaneous image approximation,” in *Proc. Canadian Conf. Elect. Computer Eng.*, 2008, pp. 1069–1072. 66

[100] P. Jost, P. Vandergheynst, and P. Frossard, “Tree-based pursuit: Algorithm and properties,” *IEEE Trans. Signal Process.*, vol. 54, no. 12, pp. 4685–4697, 2006. 66

[101] M. Turkan and C. Guillemot, “Online dictionaries for image prediction,” in *Proc. IEEE Int. Conf. Image Process.*, 2011, pp. –. 67

[102] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *J. Machine Learning Research*, vol. 11, no. 1, pp. 19–60, Jan. 2010. 70, 71, 74, 99

[103] K. Skretting and K. Engan, “Recursive least squares dictionary learning algorithm,” *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2121–2130, Apr. 2010. 70, 71, 76, 99

[104] B. A. Olshausen and D. J. Field, “Natural image statistics and efficient coding,” *Network: Compt. Neural Syst.*, vol. 7, no. 2, pp. 333–339, Feb. 1996. 70

[105] T. Blumensath and M. Davies, “Sparse and shift-invariant representations of music,” *IEEE Trans. Speech Audio Process.*, vol. 14, no. 1, pp. 50–57, Jan. 2006. 70

[106] P. Jost, P. Vandergheynst, S. Lesage, and R. Gribonval, “MoTIF: An efficient algorithm for learning translation invariant dictionaries,” in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, vol. 5, 2006, pp. 857–860. 70

[107] M. Aharon and M. Elad, “Sparse and redundant modeling of image content using an image-signature-dictionary,” *SIAM J. Imaging Sciences*, vol. 1, no. 3, pp. 228–247, Jul. 2008. 70, 71

[108] K. Engan, K. Skretting, and J. H. Husoy, “Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation,” *Digit. Signal Process.*, vol. 17, no. 1, pp. 32–49, Jan. 2007. 70, 76

[109] P. Sallee and B. A. Olshausen, “Learning sparse multiscale image representations,” in *Adv. Neural Information Process. Syst.*, vol. 15, 2003, pp. 1327–1334. 70

[110] R. Rubinstein, M. Zibulevsky, and M. Elad, “Double sparsity: Learning sparse dictionaries for sparse signal approximation,” *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, Mar. 2010. 70, 73, 74

[111] N. Jojic, B. J. Frey, and A. Kannan, “Epitomic analysis of appearance and shape,” in *Proc. IEEE Int. Conf. Computer Vis.*, 2003, pp. 34–41. 71

[112] V. Cheung, B. J. Frey, and N. Jojic, “Video epitomes,” in *Proc. IEEE Comp. Soc. Conf. Comp. Vis. Pattern Recogn.*, 2005, pp. 42–49. 71

[113] J. Mairal, F. Bach, and J. Ponce, “Task-driven dictionary learning,” INRIA, France, Tech. Rep. RR-7400, Sep. 2010. 71

[114] G. Monaci, P. Jost, and P. Vandergheynst, “Image compression with learnt tree-structured dictionaries,” in *Proc. IEEE Workshop Mult. Signal Process.*, 2004, pp. 35–38. 71

[115] M. Nakashizuka, H. Nishiura, and Y. Iiguni, “Sparse image representations with shift-invariant tree-structured dictionaries,” in *Proc. IEEE Int. Conf. Image Process.*, 2009, pp. 2145–2148. 71

[116] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach, “Proximal methods for hierarchical sparse coding,” *J. Machine Learning Res.*, vol. 12, pp. 2297–2334, 2011. 71

[117] J. Zepeda, “Novel sparse representation methods; application to compression and indexation of images,” Ph.D. dissertation, INRIA, France, 2010. 71

[118] J. Zepeda, C. Guillemot, and E. Kijak, “Image compression using sparse representations and the iteration-tuned and aligned dictionary,” *IEEE J. Selected Topics Signal Process.*, vol. 5, no. 5, pp. 1061–1073, Sep. 2011. 71

[119] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999. 74, 75

[120] M. R. Osborne, B. Presnell, and B. A. Turlach, “A new approach to variable selection in least squares problems,” *IMA J. Numerical Analysis*, vol. 20, no. 3, pp. 389–403, 2000. 74

[121] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least angle regression,” *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004. 74

[122] L. Zhang, S. Ma, and W. Gao, “Position dependent linear intra prediction for image coding,” in *Proc. IEEE Int. Conf. Image Process.*, 2010, pp. 2877–2880. 82, 83

[123] M. Turkan and C. Guillemot, “Image prediction based on non-negative matrix factorization,” in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, 2011, pp. –. 103

[124] ——, “Image prediction based on neighbor embedding methods,” *IEEE Trans. Image Process.*, 2011, accepted for publication. 103

[125] C. Guillemot and M. Turkan, “Neighbor embedding with non-negative matrix factorization for image prediction,” in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, 2012, accepted for publication. 103

[126] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee, *Semi-Supervised Learning: Spectral Methods for Dimensionality Reduction*, O. Chapelle, B. Schölkopf, and A. Zien, Eds. MIT Press, 2006. 104, 105

[127] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*, 2nd ed. Chapman and Hall, 2001. 105

[128] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” *Advances in Neural Information Process. Syst. (NIPS)*, 2000. 105, 107, 143

[129] ——, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, pp. 788–791, Oct. 1999. 105, 110

[130] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. Wiley, 2001. 105

[131] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, “Algorithms and applications for approximate nonnegative matrix factorization,” *Computational Statistics Data Analysis*, vol. 52, no. 1, pp. 155–173, 2007. 105, 107

[132] B. Schölkopf, A. Smola, and K. R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998. 105

[133] J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, pp. 2319–2323, 2000. 105

[134] S. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, pp. 2323–2326, 2000. 105, 107, 108, 109, 143

[135] S. Kullback and R. A. Leibler, “On information and sufficiency,” *Annals Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, Mar. 1951. 106

[136] P. O. Hoyer, “Non-negative matrix factorization with sparseness constraints,” *J. Machine Learning Research*, vol. 5, pp. 1457–1469, 2004. 107, 114

[137] F. C. Wu and Z. Y. Hu, “The LLE and a linear mapping,” *Pattern Recognition*, vol. 39, no. 9, pp. 1799–1804, 2006. 109

[138] W. Chojnacki and M. J. Brooks, "A note on the locally linear embedding algorithm," *Int. J. Pattern Recogn. Artif. Intell.*, vol. 23, no. 8, pp. 1739–1752, 2009. 109

[139] Z. Zhang and J. Wang, "MLLE: Modified locally linear embedding using multiple weights," in *NIPS*, 2006, pp. 1593–1600. 109

[140] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *SIAM J. Mult. Scale Modeling Simul.*, vol. 4, no. 2, pp. 490–530, 2005. 109, 143

[141] S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng, "Learning spatially localized, parts-based representation," in *Proc. IEEE Comp. Soc. Conf. Comp. Vis. Pattern Recog.*, vol. 1, 2001, pp. 207–212. 114

[142] O. Kouropteva, O. Okun, and M. Pietikainen, "Selection of the optimal parameter value for the locally linear embedding algorithm," in *Proc. Int. Conf. Fuzzy Syst. Knowledge Discovery*, 2002, pp. 359–363. 123

[143] A. A. Meza, J. V. Aguirre, G. D. Santacoloma, and G. C. Dominguez, "Global and local choice of the number of nearest neighbors in locally linear embedding," *Pattern Recog. Lett.*, vol. In Press, Corrected Proof, May 2011. 123

[144] T. F. Coleman and Y. Li, "A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables," *SIAM J. Optimization*, vol. 6, no. 4, pp. 1040–1058, Apr. 1996. 126

[145] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization.* Academic Press, 1981. 126

[146] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," *J. Mach. Learn. Res.*, vol. 4, pp. 119–155, Dec. 2003. 126

[147] C. Guillemot, M. Turkan, and O. L. Meur, "Constrained least squares neighbor embeddings for image inpainting," *IEEE Trans. Image Process.*, 2012, submitted. 141

[148] M. Bertalmio, G. Sapiro, C. Ballester, and V. Caselles, "Image inpainting," in *Proc. ACM Comp. Graphics Interactive Tech. (SIGGRAPH 2000)*, 2000, pp. 417–424. 142

[149] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proc. IEEE Comp. Soc. Conf. Comp. Vis. Pattern Recog.*, 2001, pp. 355–362. 142

[150] A. Telea, "An image inpainting technique based on the fast marching method," *J. Graphics Tools*, vol. 9, no. 1, pp. 23–34, 2004. 142, 162, 164, 165, 166

[151] D. Tschumperle, “Fast anisotrophic smoothing of multi-valued images using curvature-preserving PDE’s,” *Int. J. Computer Vis.*, vol. 68, no. 1, pp. 65–82, Jun. 2006. 142, 162, 164, 165, 166

[152] T. Chan and J. Shen, “Local inpainting models and TV inpainting,” *SIAM J. Appl. Math.*, vol. 62, no. 3, pp. 1019–1043, 2001. 142

[153] R. Bornard, E. Lecan, L. Laborelli, and J. Chenot, “Missing data correction in still images and image sequences,” in *ACM Int. Conf. Multimedia*, 2002, pp. 355–361. 142

[154] P. Harrison, “A non-hierarchical procedure for re-synthesis of complex textures,” in *Proc. Int. Conf. Central Europe Computer Graph., Visua. Computer Vis.*, 2001, pp. 190–197. 142

[155] I. Drori, D. Cohen-Or, and H. Yeshurun, “Fragment-based image completion,” *ACM Trans. Graphics (SIGGRAPH 2003)*, vol. 22, no. 3, pp. 303–312, Jul. 2003. 142

[156] J. Jia and C. K. Tang, “Image repairing: Robust image synthesis by adaptive ND tensor voting,” in *Proc. IEEE Comp. Soc. Conf. Comp. Vis. Pattern Recog.*, 2003, pp. 643–650. 142

[157] Y. J. Zhang, J. J. Xiao, and M. Shah, “Region completion in a single image,” in *EUROGRAPHICS*, 2004. 142

[158] J. Sun, L. Yuan, J. Jia, and H. Y. Shum, “Image completion with structure propagation,” *ACM Trans. Graphics (SIGGRAPH 2005)*, vol. 24, no. 3, pp. 861–868, Jul. 2005. 142

[159] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “Patchmatch: A randomized correspondence algorithm for structural image editing,” *ACM Trans. Graphics (SIGGRAPH 2009)*, vol. 28, no. 3, Jul. 2009. 142, 162, 164, 165, 166

[160] Z. Xu and J. Sun, “Image inpainting by patch propagation using patch sparsity,” *IEEE Trans. Image Process.*, vol. 19, no. 5, pp. 1153–1165, May 2010. 142

[161] A. Wong and J. Orchard, “A nonlocal-means approach to exemplar-based inpainting,” in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 2600–2603. 143

[162] X. Zhang, Y. Liu, C. Gao, and J. Liu, “An efficient algorithm of learning the parametric map of locally linear embedding,” in *Int. Symp. Intelligent Information Tech. App.*, 2008, pp. 52–56. 151

Publications

[1] M. Turkan and C. Guillemot, "Image prediction based on neighbor embedding methods," *to appear in IEEE Trans. Image Process.*, 2011.

[2] M. Turkan and C. Guillemot, "Online dictionaries for image prediction," *in Proc. IEEE Int. Conf. on Image Process. (ICIP)*, 2011, pp.–.

[3] M. Turkan and C. Guillemot, "Image prediction based on non-negative matrix factorization," *in Proc. IEEE Int. Conf. Acous. Speech Signal Process. (ICASSP)*, 2011, pp.–.

[4] M. Turkan and C. Guillemot, "Image prediction: Template matching vs. Sparse approximation," *in Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2010, pp. 789-792.

[5] A. Drémeau, M. Turkan, C. Herzet, C. Guillemot and J.-J. Fuchs, "Spatial intra-prediction based on mixtures of sparse representations," *in Proc. IEEE Int. Workshop Multimedia Signal Process. (MMSP)*, 2010, pp. 345-349.

[6] M. Turkan and C. Guillemot, "Sparse approximation with adaptive dictionary for image prediction," *in Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2009, pp. 25-28.

[7] C. Guillemot and M. Turkan, "Neighbor embedding with non-negative matrix factorization for image prediction," *accepted to IEEE Int. Conf. Acous. Speech Signal Process. (ICASSP)*, 2012.

Under review

[8] C. Guillemot, M. Turkan and O. Le Meur, "Constrained least squares neighbor embeddings for image inpainting," *submitted to IEEE Trans. Image Process.*, 2012.

[9] M. Turkan and C. Guillemot, "Locally linear embedding based texture synthesis for image prediction and error concealment," *submitted to IEEE Int. Conf. Image Process. (ICIP)*, 2012.

[10] M. Turkan *et al.*, "Anonymous CVPR submission," *submitted to Int. Conf. Comptuter Vis. Pattern Recog. (CVPR)*, 2012.

Résumé Cette thèse présente de nouvelles méthodes de synthèse de texture basées sur l'*exemple* pour les problèmes de prédiction d'images (c'est à dire, codage prédictif) et d'inpainting d'images. Les principales contributions de cette étude peuvent aussi être vues comme des extensions du *template matching*. Cependant, le problème de synthèse de texture tel que nous le définissons ici se situe plutôt dans un contexte d'optimisation formalisée sous différentes contraintes. Le problème de prédiction d'images est d'abord situé dans un contexte de représentations parcimonieuses par l'approximation du *template* sous contraintes de parcimonie. La méthode de prédiction proposée avec les dictionnaires adaptés localement montrent de meilleures performances par rapport aux dictionnaires classiques (tels que la transformée en cosinus discrète (TCD)), et à la méthode du *template matching*. Le problème de prédiction d'images est ensuite placé dans un cadre d'apprentissage de dictionnaires en adaptant les méthodes traditionnelles d'apprentissage pour la prédiction de l'image. Les observations expérimentales montrent une meilleure performance comparativement à des méthodes de prédiction parcimonieuse et des prédictions intra de type H.264/AVC. Enfin un cadre *neighbor embedding* est proposé pour la prédiction de l'image en utilisant deux méthodes de réduction de dimensionnalité: la factorisation de matrice non négative (FMN) et le *locally linear embedding* (LLE). Ce cadre est ensuite étendu au problème d'inpainting d'images. Les évaluations expérimentales démontrent l'efficacité des idées sous-jacentes pour la compression via la prédiction d'images et l'inpainting d'images.

Abstract This thesis presents novel exemplar-based texture synthesis methods for image prediction (i.e., predictive coding) and image inpainting problems. The main contributions of this study can also be seen as extensions to simple template matching, however the texture synthesis problem here is well-formulated in an optimization framework with different constraints. The image prediction problem has first been put into sparse representations framework by approximating the template with a sparsity constraint. The proposed sparse prediction method with locally and adaptive dictionaries has been shown to give better performance when compared to static waveform (such as DCT) dictionaries, and also to the template matching method. The image prediction problem has later been placed into an online dictionary learning framework by adapting conventional dictionary learning approaches for image prediction. The experimental observations show a better performance when compared to H.264/AVC intra and sparse prediction. Finally a neighbor embedding framework has been proposed for image prediction using two data dimensionality reductions methods: non-negative matrix factorization (NMF) and locally linear embedding (LLE). This framework has then been extended to the image inpainting problem. The experimental evaluations demonstrate the effectiveness of the underlying ideas in both image prediction and inpainting applications.