Sparse Analysis Model Based Dictionary Learning and Signal Reconstruction

Jing Dong

Submitted for the Degree of Doctor of Philosophy from the University of Surrey

Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

June 2016

© Jing Dong 2016
Abstract

Sparse representation has been studied extensively in the past decade in a variety of applications, such as denoising, source separation and classification. Earlier effort has been focused on the well-known synthesis model, where a signal is decomposed as a linear combination of a few atoms of a dictionary. However, the analysis model, a counterpart of the synthesis model, has not received much attention until recent years. The analysis model takes a different viewpoint to sparse representation, and it assumes that the product of an analysis dictionary and a signal is sparse. Compared with the synthesis model, this model tends to be more expressive to represent signals, as a much richer union of subspaces can be described. This thesis focuses on the analysis model and aims to address the two main challenges: analysis dictionary learning (ADL) and signal reconstruction.

In the ADL problem, the dictionary is learned from a set of training samples so that the signals can be represented sparsely based on the analysis model, thus offering the potential to fit the signals better than pre-defined dictionaries. Among the existing ADL algorithms, such as the well-known Analysis K-SVD, the dictionary atoms are updated sequentially. The first part of this thesis presents two novel analysis dictionary learning algorithms to update the atoms simultaneously. Specifically, the Analysis Simultaneous Codeword Optimization (Analysis SimCO) algorithm is proposed, by adapting the SimCO algorithm which is proposed originally for the synthesis model. In Analysis SimCO, the dictionary is updated using optimization on manifolds, under the $\ell_2$-norm constraints on the dictionary atoms. This framework allows multiple dictionary atoms to be updated simultaneously in each iteration. However, similar to the existing ADL algorithms, the dictionary learned by Analysis SimCO may contain similar atoms. To address this issue, Incoherent Analysis SimCO is proposed by employing a coherence constraint and introducing a decorrelation step to enforce this constraint. The competitive performance of the proposed algorithms is demonstrated in the experiments for recovering synthetic dictionaries and removing additional noise in images, as compared with existing ADL methods.

The second part of this thesis studies how to reconstruct signals with learned dictionaries under the analysis model. This is demonstrated by a challenging application problem: multiplicative noise removal (MNR) of images. In the existing sparsity motivated methods, the MNR problem is addressed using pre-defined dictionaries, or learned dictionaries based on the synthesis model. However, the potential of analysis dictionary learning for the MNR problem has not been investigated. In this thesis, analysis dictionary learning is applied to MNR, leading to two new algorithms. In the first algorithm, a dictionary learned based on the analysis model is employed to form a regularization term, which can preserve image details while removing multiplicative noise. In the second algorithm, in order to further improve the recovery quality of smooth areas in images, a smoothness regularizer is introduced to the reconstruction formulation. This regularizer can be seen as an enhanced Total Variation (TV) term with an additional parameter controlling the level of smoothness. To address the optimization problem of this model, the Alternating Direction Method of Multipliers (ADMM) is adapted and a relaxation technique is developed to allow variables to be updated flexibly. Experimental results show the superior performance of the proposed algorithms as compared with three sparsity or TV based algorithms for a range of noise levels.
**Key words:** Sparse representation, sparse analysis model, sparse synthesis model, sparse coding, analysis pursuit, synthesis dictionary learning, analysis dictionary learning, image denoising, multiplicative noise removal, SAR image, optimization methods, convex optimization.

Email: j.dong@surrey.ac.uk
WWW: http://www.eps.surrey.ac.uk/
Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Wenwu Wang, for his insightful guidance, continuing support, and many invaluable discussions, comments, and suggestions at every stage of my PhD study. He has been always extremely generous with his time and support. I feel grateful to work with him.

I would also like to thank my co-supervisor, Dr. Philip Jackson, for his advice and encouragement. I want to express my sincere thanks to Prof. Jonathon Chambers from Newcastle University, Dr. Wei Dai from Imperial College, Prof. Mark Plumbley, Mr. Zi-Fa Han from City University of Hong Kong, and Prof. Yuxin Zhao from Harbin Engineering University, for their constructive comments and kind help.

This work is part of the University Defence Research Collaboration (UDRC) funded by the Ministry of Defence (MOD) and Engineering and Physical Sciences Research Council (EPSRC), U.K. I acknowledge them for their financial support which allowed me to present the papers at several conferences. I am also grateful to my university, University of Surrey, for offering me the PhD scholarship.

Special thanks go to my friends at Surrey, Xiaoyi Chen, Qingju Liu, Volkan Kılıç, Guosheng Hu, Zhenhua Feng, Hazim F. El-Sharif, Jin Wang, Anju Goel, Yang Yu, Jian Guan, Xiaoqing Luo, Qiaoxi Zhu, for their companionship, help and encouragement. They have made this journey more enjoyable and I also have learned so many things from them. I would also like to thank the CVSSP staff, Elizabeth James, Anna Korzeniowska, and Nan Bennett, for helping me solve my administrative problems.

I would like to express my deepest appreciation to my family. I wish to thank my parents and my sister for their unconditional support, persistent encouragement and endless love. I would also like to thank my husband for his great help and guidance.
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACoSaSP</td>
<td>Analysis Compressive Sampling Matching Pursuit</td>
</tr>
<tr>
<td>ADL</td>
<td>Analysis Dictionary Learning</td>
</tr>
<tr>
<td>ADMM</td>
<td>Alternating Direction Method of Multipliers</td>
</tr>
<tr>
<td>AHTP</td>
<td>Analysis Hard Thresholding Pursuit</td>
</tr>
<tr>
<td>AIHT</td>
<td>Analysis Iterative Hard Thresholding</td>
</tr>
<tr>
<td>AKSVD</td>
<td>Analysis K-SVD</td>
</tr>
<tr>
<td>AOL</td>
<td>Analysis Operator Learning</td>
</tr>
<tr>
<td>ASimCO</td>
<td>Analysis SimCO</td>
</tr>
<tr>
<td>ASP</td>
<td>Analysis Subspace Pursuit</td>
</tr>
<tr>
<td>BG</td>
<td>Backward-Greedy</td>
</tr>
<tr>
<td>BM3D</td>
<td>Block Matching 3D</td>
</tr>
<tr>
<td>BP</td>
<td>Basis Pursuit</td>
</tr>
<tr>
<td>BPDN</td>
<td>Basis Pursuit DeNoising</td>
</tr>
<tr>
<td>BSS</td>
<td>Blind Source Separation</td>
</tr>
<tr>
<td>CoSaMP</td>
<td>Compressive Sampling Matching Pursuit</td>
</tr>
<tr>
<td>CS</td>
<td>Compressed Sensing</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Dictionary</td>
</tr>
<tr>
<td>DFN</td>
<td>Duran, Fadili and Nikolova</td>
</tr>
<tr>
<td>ENL</td>
<td>Equivalent Number of Look</td>
</tr>
<tr>
<td>FOCUSS</td>
<td>FOcal Under-determined System Solver</td>
</tr>
<tr>
<td>GAP</td>
<td>Greedy Analysis Pursuit</td>
</tr>
<tr>
<td>GOAL</td>
<td>GeOmetric Analysis operator Learning</td>
</tr>
<tr>
<td>GP</td>
<td>Gradient Pursuit</td>
</tr>
<tr>
<td>HTP</td>
<td>Hard Thresholding Pursuit</td>
</tr>
<tr>
<td>ICA</td>
<td>Independent Component Analysis</td>
</tr>
<tr>
<td>IHT</td>
<td>Iterative Hard Thresholding</td>
</tr>
<tr>
<td>IK-SVD</td>
<td>Incoherent K-SVD</td>
</tr>
</tbody>
</table>

vii
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN-ASimCO</td>
<td>Incoherent Analysis SimCO</td>
</tr>
<tr>
<td>Iter-NSM</td>
<td>Iterative method using Nonlocal Sparse Model</td>
</tr>
<tr>
<td>LASSO</td>
<td>Least-Absolute-Shrinkage-and-Selection-Operator</td>
</tr>
<tr>
<td>LOST</td>
<td>Learning Overcomplete Sparsifying Transforms</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Programming</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute-deviation Error</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum A Posteriori</td>
</tr>
<tr>
<td>MIDAL</td>
<td>Multiplicative Image Denoising with the Augmented Lagrangian</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>MNR-ADL-SR</td>
<td>Multiplicative Noise Removal using Analysis Dictionary Learning and a Smoothness Regularizer</td>
</tr>
<tr>
<td>MNR-DL-TV</td>
<td>Multiplicative Noise Removal via Dictionary Learning and Total Variation</td>
</tr>
<tr>
<td>MOD</td>
<td>Method of Optimal Directions</td>
</tr>
<tr>
<td>MP</td>
<td>Matching Pursuit</td>
</tr>
<tr>
<td>MSSIM</td>
<td>Mean Structural SIMilarity Index</td>
</tr>
<tr>
<td>(NA)AOL</td>
<td>Noise Aware Analysis Operator Learning</td>
</tr>
<tr>
<td>(NL)AOL</td>
<td>NoiseLess Analysis Operator Learning</td>
</tr>
<tr>
<td>OBG</td>
<td>Optimized-Backward-Greedy</td>
</tr>
<tr>
<td>OMP</td>
<td>Orthogonal Matching Pursuit</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>ROMP</td>
<td>Regularized Orthogonal Matching Pursuit</td>
</tr>
<tr>
<td>RSN-ADL</td>
<td>Removing Speckle Noise by Analysis Dictionary Learning</td>
</tr>
<tr>
<td>SAR</td>
<td>Synthetic Aperture Radar</td>
</tr>
<tr>
<td>SAS</td>
<td>Synthetic Aperture Sonar</td>
</tr>
<tr>
<td>SDL</td>
<td>Synthesis Dictionary Learning</td>
</tr>
<tr>
<td>SimCO</td>
<td>Simultaneous Codeword Optimization</td>
</tr>
<tr>
<td>SP</td>
<td>Subspace Pursuit</td>
</tr>
<tr>
<td>StOMP</td>
<td>Stagewise Orthogonal Matching Pursuit</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>TKSVD</td>
<td>Transform K-SVD</td>
</tr>
<tr>
<td>TV</td>
<td>Total Variation</td>
</tr>
<tr>
<td>UNTF</td>
<td>Uniform Normalized Tight Frame</td>
</tr>
</tbody>
</table>
Symbols

\( X_{i,:} \) \hspace{1em} The \( i \)-th row of matrix \( X \)
\( X_{:,i} \) \hspace{1em} The \( i \)-th column of matrix \( X \)
\( D_{:,\Gamma} \) \hspace{1em} The sub-matrix of \( D \) with columns indexed by set \( \Gamma \)
\( \Omega_{\Lambda,:} \) \hspace{1em} The sub-matrix of \( \Omega \) with rows indexed by set \( \Lambda \)
\( x_i \) \hspace{1em} Denotes the \( i \)-th element of vector \( x \)
\( x_{i,j} \) \hspace{1em} The element in the \( i \)-th row and the \( j \)-th column of matrix \( X \)
\( y \in \mathbb{R}^m \) \hspace{1em} Signal
\( a \in \mathbb{R}^d \) \hspace{1em} Representation coefficient vector
\( x \in \mathbb{R}^p \) \hspace{1em} Analysis representation vector
\( Y \in \mathbb{R}^{m \times n} \) \hspace{1em} Training signals
\( A \in \mathbb{R}^{d \times n} \) \hspace{1em} Representation coefficient matrix
\( X \in \mathbb{R}^{p \times n} \) \hspace{1em} Analysis representation matrix
\( D \in \mathbb{R}^{m \times d} \) \hspace{1em} Synthesis dictionary
\( \Omega \in \mathbb{R}^{p \times m} \) \hspace{1em} Analysis dictionary
\( s \) \hspace{1em} Sparsity
\( l \) \hspace{1em} Co-sparsity
\( \| \cdot \|_0 \) \hspace{1em} \( \ell_0 \)-norm, number of non-zero entries in \( x \)
\( \| \cdot \|_1 \) \hspace{1em} \( \ell_1 \)-norm, \( \| x \|_1 = \sum_i |x_i| \)
\( \| \cdot \|_2 \) \hspace{1em} \( \ell_2 \)-norm, \( \| x \|_2 = \sqrt{\sum_i x_i^2} \)
\( \| \cdot \|_p \) \hspace{1em} \( \ell_p \)-norm, \( \| x \|_p = \left( \sum_i |x_i|^p \right)^{1/p} \), \( 0 < p < 1 \)
\( \| \cdot \|_F \) \hspace{1em} Frobenius norm, \( \| X \|_F = \sqrt{\sum_{i,j} x_{i,j}} \)
\( (\cdot)^\dagger \) \hspace{1em} Moore-Penrose pseudo-inverse
\( (\cdot)^T \) \hspace{1em} Transpose
\( (\cdot)^{(i)} \) \hspace{1em} Variable in the \( i \)-th iteration
\( (\cdot)^* \) \hspace{1em} Solution to an optimization problem
\( \ln(\cdot), \log(\cdot) \) \hspace{1em} Natural logarithm
\( \text{rank}(\cdot) \) \hspace{1em} Matrix rank
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{det}(\cdot)$</td>
<td>Matrix determinant</td>
</tr>
<tr>
<td>$\text{diag}(X)$</td>
<td>Diagonal matrix with the same diagonal entries as $X$</td>
</tr>
<tr>
<td>$\text{card}(\cdot)$</td>
<td>Set cardinality</td>
</tr>
<tr>
<td>$\Lambda / \Gamma$</td>
<td>The difference between set $\Lambda$ and $\Gamma$</td>
</tr>
<tr>
<td>$\Lambda \cup \Gamma$</td>
<td>The union of set $\Lambda$ and $\Gamma$</td>
</tr>
<tr>
<td>$\text{sng}(\cdot)$</td>
<td>Sign function</td>
</tr>
<tr>
<td>$\langle \cdot, \cdot \rangle$</td>
<td>Inner-product</td>
</tr>
<tr>
<td>$\circ$</td>
<td>Entry-wise product</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>Empty set</td>
</tr>
</tbody>
</table>
List of Figures

2.1 The plot of $|a|^p$ for different values of $p$. ........................................ 12
2.2 The diagram of synthesis dictionary learning algorithms. .................. 15
2.3 The original image, noisy image and denoised images using overcomplete DCT and K-SVD. ......................................................... 37
2.4 The comparison between additive Gaussian noise and multiplicative Gamma noise. ................................................................. 39

3.1 The flow diagram of Analysis SimCO. .............................................. 46
3.2 Objective function value with different initializations. ...................... 58
3.3 The histograms of the elements in the mutual correlation matrices of the dictionaries learned. ......................................................... 60
3.4 Recovery Rate and Average Co-sparsity over iterations. ................... 60
3.5 Recovery Rate and Average Co-sparsity with different co-sparsities. .... 62
3.6 Recovery Rate and Average Co-sparsity with different numbers of training samples. ................................................................. 63
3.7 Recovery Rate and Average Co-sparsity with different numbers of atoms. 64
3.8 Face images. (a) Original face. (b) Noisy face with $\sigma = 12.8$. (c) Noisy face with $\sigma = 45$. ............................................................. 68
3.9 Face image denoising ($\sigma = 12.8$). .................................................. 69
3.10 Face image denoising ($\sigma = 45$). ..................................................... 70
3.11 Test images for natural image denoising. ........................................ 71
3.12 Training images used for learning analysis dictionaries. ................... 71
3.13 Natural image denoising ($\sigma = 12.8$). ........................................... 72
3.14 Natural image denoising ($\sigma = 45$). ............................................. 73

4.1 Training images used to learn analysis dictionaries. ......................... 91
4.2 Test images: Cameraman, Nimes, Fields and Peppers. ...................... 91
List of Figures

4.3 Synthetic noisy images .................................................. 92
4.4 PSNR results for the Cameraman image with different regularization parameters .......... 94
4.5 Denoising results in PSNR, MAE and MSSIM based on 30 noisy realizations for each case ........................................... 96
4.6 Results for Cameraman \((L = 10)\) ........................................ 97
4.7 Results for Cameraman \((L = 4)\) ........................................ 97
4.8 Results for Cameraman \((L = 1)\) ........................................ 98
4.9 Results for Nîmes \((L = 10)\) .............................................. 98
4.10 Results for Nîmes \((L = 4)\) .............................................. 99
4.11 Results for Nîmes \((L = 1)\) .............................................. 99
4.12 Results for Fields \((L = 10)\) ............................................ 100
4.13 Results for Fields \((L = 4)\) ............................................ 100
4.14 Results for Fields \((L = 1)\) ............................................ 101
4.15 Results for Peppers \((L = 10)\) ......................................... 101
4.16 Results for Peppers \((L = 4)\) ......................................... 102
4.17 Results for Peppers \((L = 1)\) ......................................... 102
4.18 Original SAR images ..................................................... 103
4.19 Results for the first SAR image ........................................ 104
4.20 Results for the second SAR image ..................................... 104
4.21 Results for Cameraman \((L = 10)\) .................................... 106
4.22 Results for Cameraman \((L = 4)\) .................................... 106
4.23 Results for Cameraman \((L = 1)\) .................................... 106
List of Tables

2.1 Comparison of different SDL algorithms ................................................. 16
2.2 Comparison of different ADL algorithms ................................................. 35

3.1 Parameters used in the comparison of different ADL algorithms with synthetic data ................................................. 61
3.2 Time of one test with different parameters (in seconds) ................................................. 66
3.3 The best face image denoising results (PSNR in decibels) ................................................. 70
3.4 The best natural image denoising results (PSNR in decibels) ................................................. 72

4.1 Parameters used in the algorithms for multiplicative noise removal ................................................. 95
4.2 ENL for the homogeneous regions in the denoised SAR images ................................................. 105
# Contents

Abstract i

Acronyms vii

Symbols ix

List of Figures x

List of Tables xii

1 Introduction 1
   1.1 Synthesis Model and Analysis Model 2
   1.2 Problem Description 3
   1.3 Contributions 4
   1.4 Thesis Structure 6

2 Literature Review on Sparse Representation and Dictionary Learning 7
   2.1 Sparse Synthesis Model 8
      2.1.1 Sparse Coding 8
      2.1.2 Synthesis Dictionary Learning 14
   2.2 Sparse Analysis Model 21
      2.2.1 Analysis Pursuit 22
      2.2.2 Analysis Dictionary Learning 27
   2.3 Applications of Sparse Models 36
   2.4 Summary 42
# Analysis SimCO Algorithms for Analysis Dictionary Learning

## 3.1 Introduction

## 3.2 Problem Formulation and Optimization Framework

## 3.3 The Analysis SimCO Algorithm

### 3.3.1 Optimization on Matrix Manifolds

### 3.3.2 Analysis Sparse Coding Stage

### 3.3.3 Dictionary Update Stage

### 3.3.4 Convergence

### 3.3.5 Computational Complexity

## 3.4 Incoherent Analysis SimCO

## 3.5 Simulation Results

### 3.5.1 Experiments with Synthetic Data

### 3.5.2 Experiments for Image Denoising

## 3.6 Summary

# Multiplicative Noise Removal Based on Sparse Analysis Model

## 4.1 Introduction

## 4.2 Removing Speckle Noise by Analysis Dictionary Learning

### 4.2.1 Problem Formulation

### 4.2.2 Optimization Method

## 4.3 Sparse Analysis Model Based Multiplicative Noise Removal with Enhanced Regularization

### 4.3.1 Proposed Image Restoration Formulation

### 4.3.2 Optimization Method

## 4.4 Experimental Results

### 4.4.1 Experiments with Synthetic Images

### 4.4.2 Experiments with Real SAR Images

### 4.4.3 Comparison with State-of-the-art Methods

## 4.5 Summary

# Conclusions and Future Research

## 5.1 Conclusions

### 5.1.1 Analysis Dictionary Learning

### 5.1.2 Sparse Analysis Model Based Multiplicative Noise Removal

## 5.2 Future Research
Contents

A  Derivation of the Gradient Descent Step in MNR-ADL-SR 113

B  List of Publications 115

Bibliography 117
Chapter 1

Introduction

Signal models can be regarded as mathematical descriptions of signals to capture their intrinsic characteristics of the signals. They are fundamental tools for practical tasks in signal processing. For instance, in inverse problems where additional prior information is required to regularize the problems, the assumption that the signal of interest satisfies a signal model, e.g. a prior distribution, can serve as the prior information. The quality of the solutions to the inverse problems heavily depends on the selection of the signal model. Actually, the progress made in signal processing during the past several decades is stimulated by an evolution of signal models and their employment in practice [41]. Examples of important models include Principal Component Analysis, $\ell_2$ based regularization and Mixture of Gaussians.

In the last decade, sparse models have received considerable attention [100], [5], [92], [74], as they are effective for real problems and also appealing in theory. The core assumption of the sparse models is that, with a dictionary, the signal of interest can be represented using a sparse vector which only has a few non-zero elements. This sparse assumption is based on the fact that many natural signals distributed in a high-dimensional space cannot cover the entire space, but actually reside in a union of much lower dimensional subspaces. From this point of view, the sparse models can be regarded as low-dimensional models [107]. The applications of the sparse models for real problems range from denoising [42], [92], inpainting [71], super-resolution [108], compression [19], to separation [110], decomposition [45] and classification [109]. At the same time, the sparse models also lead to a wide set of interesting theoretical questions such as the uniqueness of the sparsest solution of a linear system, the minimal possible error
for representation, and conditions for recovering sparse signals [18].

It should be noted that, as a critical component of the sparse models, the dictionaries play an essential role for the performance of the sparse models. Some analytical dictionaries have been designed depending on the characteristics of a specific family of signals. For example, in image processing, the curvelet transform [21] is developed to approximate curvilinear structures and the ridgelet transform [22] provides a good representation for straight edges [96]. The analytical dictionaries have explicit mathematical descriptions and can be applied efficiently, but they lack adaptivity due to the generic assumptions used in the design process [90]. To better fit a specific class of signals, an alternative route is to learn adaptive dictionaries using signal samples from training data. It has been shown that the learned dictionaries are able to obtain superior results in many applications, compared with the analytical ones citeaharon2006k-svd, [91]. Therefore, dictionary learning has become a popular topic in the last decade.

Another challenging problem related to the sparse models is how to reconstruct a signal given a dictionary. This involves the formulation of an objective function based on sparsity measures and the solution to the arising optimization problem [57]. A typical sparsity measurement is the $\ell_0$-norm which is defined as the number of non-zero elements in a vector. However, this measurement usually results in computationally non-tractable problems because of its combinatorial nature. For this reason, when $\ell_0$-norm is used to model the sparsity, the optimization is usually addressed approximately with greedy algorithms [82], [74]. Another option to model the sparsity is to relax the $\ell_0$-norm as the forms that can be addressed by other optimization methods. For example, if $\ell_1$-norm, a convex surrogate of $\ell_0$-norm, is used, convex optimization can usually be applied [74]. In real applications, how to incorporate the sparse model based prior information and how to address the corresponding optimization problems need to be considered carefully.

### 1.1 Synthesis Model and Analysis Model

There are two types of sparse models available: synthesis model and analysis model [44]. They are classified according to the way used to sparsify the signal. Specifically, the synthesis model decomposes the signal as a linear combination of a few atoms of the dictionary [5] and thus the coefficient vector is sparse. This model can be understood intuitively by seeing the dictionary
atoms as “building blocks” to compose the signal. In contrast, the analysis model assumes the product of a dictionary and a signal to be sparse \[92\]. In other words, the signal satisfying the analysis model lies in the subspace which is orthogonal to the the subspace spanned by some atoms of the dictionary. It has been proven that these two sparse models are equivalent in the complete and undercomplete cases \[44\]. Nevertheless, in the more interesting case where overcomplete dictionaries are utilized, they differ significantly \[44\]. In the subspace point of view, the analysis model is more expressive as it leads to a richer union of subspaces \[41\].

Regarding the dictionary learning and signal reconstruction challenges as mentioned above, the synthesis model has been studied extensively and many algorithms have been proposed \[5\], \[27\]. However, much less attention has been given to the analysis model in the literature. In the past few years, there is a growing interest in the analysis model.

### 1.2 Problem Description

The focus of this thesis is on the analysis model and its two critical challenges: dictionary learning and signal reconstruction. The signal reconstruction problem is demonstrated with a specific task, i.e. multiplicative noise removal. Specifically, the following two problems are considered:

1. **Analysis dictionary learning:**

   Given a set of signal samples, the goal of analysis dictionary learning is to train a dictionary in order to sparsify the samples using the analysis model \[92\], i.e. the product of the dictionary and the signal should be as sparse as possible. This task is usually formulated as an optimization problem with sparsity promoting constraints and the constraints related to the expected dictionary. A good formulation should be able to avoid trivial dictionaries but not to exclude favourable ones. Another issue is to develop a method to address the corresponding optimization problem.

   As mentioned before, the dictionary learning problem is very important to the performance of the analysis model, however, only a few algorithms have been developed, e.g. Analysis K-SVD \[92\], Learning Overcomplete Sparsifying Transform (LOST) \[88\], and
GeOmetric Analysis operator Learning (GOAL) [56]. These algorithms have some limitations. For example, the Analysis K-SVD algorithm updates dictionary atoms sequentially and its computational complexity is also high. Therefore, in the first part of this thesis, the analysis dictionary learning problem is considered and some of the limitations associated with these existing algorithms are addressed.

2. Sparse analysis model based multiplicative noise removal:

Multiplicative noise, also known as speckle noise, arises in real applications, such as Synthetic Aperture Radar (SAR) images [38]. Multiplicative noise removal is more difficult than additive Gaussian noise reduction which is often assumed in conventional image denoising. One reason is that the image is degraded more severely since the noise is multiplied with the original image. In addition, due to the physical process to acquire the images, the statistical property of multiplicative noise is usually modeled with Gamma distribution [38], rather than the Gaussian distribution widely used in image denoising, which raises challenges for optimization.

Classical methods for multiplicative noise removal include spatial filtering [63] and Total Variation (TV) based approaches [11]. Recently, the sparse synthesis model has also been applied to remove multiplicative noise [58]. The method proposed in [58] employs a dictionary learned based on the synthesis model. In contrast, the existing methods based on the analysis model all utilize the pre-defined dictionaries, e.g. wavelet [6] and curvelet [38], rather than a learned dictionary. To this end, the potential of analysis dictionary learning for the multiplicative noise removal task will be studied in the second part of this thesis.

1.3 Contributions

To address the problems raised in Section 1.2, several algorithms are proposed in this thesis. The major contributions are summarised as follows:

1. A new analysis dictionary learning algorithm is proposed, which is referred to as Analysis SimCO. The formulation of Analysis SimCO is to minimize the representation error
1.3. Contributions

with two constraints. The first constraint reflects the sparsity of the signals with respect to the dictionary and the second one guarantees that the atoms of the dictionary have unit $\ell_2$-norm to exclude trivial solutions. The optimization of this formulation involves two kinds of variables, i.e. the representation vectors and the dictionary, and can be addressed by updating them in an alternating manner. For the update of the representation vectors, a hard thresholding operator is applied to enforce the sparsity related constraint. Due to the unit $\ell_2$-norm constraint, the optimization on the dictionary can be regarded as an optimization problem on matrix manifolds, and thus an optimization method on manifolds is proposed.

2. From the experiments using Analysis SimCO, it has been found that the learned dictionaries may have some similar atoms, which may degrade the representation ability of the dictionaries. This issue is also found in some existing algorithms, such as Analysis K-SVD [92] and LOST [88]. To address this issue, a constraint on the coherence of the learned dictionary is introduced to the formulation of Analysis SimCO, which leads to the Incoherent Analysis SimCO algorithm. In this algorithm, the coherence of the dictionary is limited below a given threshold. To enforce this constraint, an extra step is introduced after the dictionary update step of the Analysis SimCO algorithm. In particular, the atom pairs whose correlations are greater than the given threshold are labeled and then decorrelated successively.

3. Based on the analysis model, a novel speckle noise removal algorithm is proposed, which is referred to as Removing Speckle Noise by Analysis Dictionary Learning (RSN-ADL). Specifically, the assumption that the image to be reconstructed satisfies the analysis model is utilized as the prior information for the speckle noise removal problem. In the RSN-ADL algorithm, an analysis dictionary is learned by Analysis SimCO, and then the learned dictionary is employed to form a regularization term for reconstructing the image. The image reconstruction task is formulated as an unconstrained optimization problem which contains a data fidelity term derived from the statistical distribution of the speckle noise and a sparsity promoting regularizer using the learned dictionary. Finally, the Alternating Direction Method of Multipliers (ADMM) method is carefully tailored to optimize the corresponding problem.
4. As can be observed in the experiments of RSN-ADL, the learned dictionaries are adapted to textures but not good at recovering smooth areas. To further improve the recovery quality for the smooth areas, an extended model is proposed by introducing an enhanced regularization to the reconstruction formulation of RSN-ADL. The new algorithm which follows the framework of RSN-ADL but uses the extended reconstruction formulation is referred to as Multiplicative Noise Removal using Analysis Dictionary Learning and a Smoothness Regularizer (MNR-ADL-SR). The smoothness regularizer in the reconstruction formulation of MNR-ADL-SR is defined on the pixel-wise differences with a parameter controlling the smoothness constraint. To address this new formulation, an optimization method is proposed by adapting ADMM and developing a relaxation technique for updating variables flexibly as required.

1.4 Thesis Structure

The remainder of this thesis is organized as follows. In Chapter 2, the background and the related literatures regarding both the sparse synthesis model and the sparse analysis model are reviewed. The proposed analysis dictionary learning algorithms, i.e. Analysis SimCO and Incoherent Analysis SimCO, are presented in Chapter 3. They are tested for recovering synthetic dictionaries and removing additive noise in images. Chapter 4 presents the proposed analysis model based multiplicative noise removal algorithms RSN-ADL and MNR-ADL-SR. Finally, Chapter 5 concludes this thesis and points out the possible future work.
Chapter 2

Literature Review on Sparse Representation and Dictionary Learning

This chapter presents the background related to the sparse synthesis and analysis models. In practical applications of the sparse models, a crucial challenge involved is how to reconstruct a signal using the sparsity prior based on the sparse models. Specifically, the signal reconstruction task based on the synthesis model can be addressed by finding the sparse coefficients of the signal with respect to a given dictionary and recovering the signal as the product of the sparse coefficients and the dictionary. As a result, the key problem for reconstructing a signal based on this model is to represent the signal sparsely using a given dictionary, i.e. the sparse coding problem. In contrast, the analysis model based signal reconstruction leads to the analysis pursuit problem. Another challenge in practice is how to find a dictionary that is adaptive to specific data, which results in the dictionary learning problems for both synthesis and analysis models.

In this chapter, the sparse synthesis and analysis models are introduced in Sections 2.1 and 2.2, respectively, where the existing methods addressing the main challenges of the models as mentioned above are also included. In Section 2.3, the applications of the sparse models are presented.
2.1 Sparse Synthesis Model

The most well-known model in sparse representation is the sparse synthesis model [5], [100], [46], [30]. This model assumes that a signal $y \in \mathbb{R}^m$ can be linearly represented with some atoms (i.e., columns) of a synthesis dictionary $D \in \mathbb{R}^{m \times d}$, where the dictionary is usually overcomplete with $d > m$. The number of atoms used to represent $y$ is much smaller than the total number of atoms in the dictionary, which reflects the sparse property of the signal $y$. Mathematically, this model can be written as

$$y = Da$$

with $\|a\|_0 = s,$ \hspace{1cm} (2.1)

where the $\ell_0$-norm $\| \cdot \|_0$ counts the number of non-zero elements of its argument, and $a$ is the representation coefficient vector with $s$ being its sparsity. The atoms corresponding to the non-zero elements of $a$ are used to synthesise the signal $y$ via their linear combination, which brings about the term “synthesis” in the name of this model. The key feature of this model is that $s$ is expected to be much smaller than $d$. The index set of the non-zero entries of $a$ are referred to as the support of $y$, and the atoms of $D$ indexed by the support are referred to as support atoms.

2.1.1 Sparse Coding

One challenge related to the synthesis model is the sparse coding problem which aims to find the sparsest representation $a$ of a given signal $y$ with respect to a given dictionary $D$. It can be formulated as an $\ell_0$ minimization problem as follows [5]

$$a^* = \arg \min_a \|a\|_0 \quad \text{s.t.} \quad y = Da.$$ \hspace{1cm} (2.2)

Once the sparse vector $a$ is determined, the signal can be reconstructed via $y^* = Da^*$. The linear system of equations $y = Da$ is under-determined as there are more unknowns than equations with $m < d$ and thus it has infinite number of $a$ satisfying the equation. The goal of sparse coding, however, is to find the solution $a$ with the smallest number of non-zeros. Unfortunately, this problem proves to be NP-hard (Non-deterministic Polynomial-time hard).
in general, and thus approximate methods are required in practice. A large number of algorithms have been proposed to tackle the sparse coding problem, and they can be divided generally to two major classes – greedy algorithms \cite{72}, \cite{82} and relaxation algorithms \cite{27}, \cite{53}.

**Greedy Algorithms**

As can be observed in (2.2), the unknown representation coefficients $a$ is composed of two types of elements – the support (non-zero) elements corresponding to the selected atoms to represent the signal and the zero elements. Thus, one way to find $a$ is to focus on the support elements by detecting the support atoms, which leads to the family of greedy algorithms. Generally, the greedy algorithms approximate the signal $y$ iteratively, by selecting one atom or a few atoms in each iteration. In the $i$-th iteration, the approximated signal can be calculated as the product of the dictionary $D$ and current coefficient $a^{(i)}$, and then the representation residual $r^{(i)}$ is evaluated as the difference between $y$ and the current approximation of $y$, that is $r^{(i)} = y - Da^{(i)}$. The atoms selected at each iteration are usually determined using the representation residual in order to find a better approximation.

Matching Pursuit (MP) \cite{72} is one of the simplest greedy algorithms to address the sparse coding problem. The algorithm starts with an all-zero coefficient vector $a$ and updates it iteratively. At each iteration of MP, a dictionary atom is selected to decrease the residual most and then the corresponding element of $a$ is updated. In particular, the inner-products of the dictionary atoms and the current residual is calculated and the atom leading to the maximum inner-product (in magnitude) is selected. After that, the representation vector and the representation residual are updated by projecting the current residual to the selected atom. This algorithm is summarized as Algorithm 1. Note that in MP only the coefficient corresponding to the newly selected atom is updated and other coefficients are not modified (cf. Step 3). This is because the approximation of the residual is considered by projecting to the newly selected atom only and this projection will not affect the coefficients except for the one corresponding to the newly selected atom. However, due to this projection approach, the atoms may be selected repeatedly in different iterations, which leads to a slow convergence speed.

The Orthogonal Matching Pursuit (OMP) \cite{82} algorithm \cite{100} has been proposed as an im-
Chapter 2. Literature Review on Sparse Representation and Dictionary Learning

Algorithm 1 Matching Pursuit (MP) Algorithm

**Input:** $y$, $D$

**Output:** $\hat{a}$

**Initialization:**
- Initialize the iteration counter $i = 1$, the representation residual $r^{(i)} = y$, and the representation coefficient vector $a^{(i)} = 0$. Perform the following steps.

**Main Steps:**

1. Calculate the inner-products between the atoms and the current residual: $b^{(i)} = D^T r^{(i)}$.
2. Determine the index of the atom which leads to the largest inner-product (in magnitude): $\{k^*\} = \arg \max_k |b^{(i)}_k|$.
3. Update the representation coefficients: $a^{(i+1)}_{k^*} = a^{(i)}_{k^*} + b^{(i)}_{k^*}$.
4. Update the representation residual: $r^{(i+1)} = r^{(i)} - b^{(i)}_{k^*} D_{:,k^*}$.
5. If the stopping criterion is satisfied, $\hat{a} = a^{(i+1)}$ and quit the iteration. Otherwise, increase the iteration counter $i = i + 1$ and go back to step 1.

proved version of MP. It uses the same approach as MP to select a new atom in each iteration, but the update of the coefficients and the residual is performed by projecting the residual onto the subspace spanned by the atoms selected in all previous iterations rather than onto the newly selected atom only. Specifically, a set $\Gamma$ saving the indices of the atoms that have already been selected is maintained, and the representation coefficients are updated by projecting the signal onto all the atoms indexed by this set, i.e. $a = D^\dagger :\Gamma y$ with $\dagger$ denoting the Moore-Penrose pseudo-inverse. In this way, the residual is always orthogonal to all previously selected atoms and thus the atoms can only be selected once. This projection approach also leads to the optimum approximation for the signal using the selected atoms, since it actually solves the least squares problem $\{a^*\} = \arg \min_a \|y - D_{:,\Gamma} a\|_2^2$. The OMP algorithm is shown as Algorithm 2. OMP has superior performance compared with MP, but it is more demanding both in terms of computation time and memory requirement [13].

Based on the OMP algorithm, many variants have been proposed, such as Stagewise Orthogonal Matching Pursuit (StOMP) [37], Subspace Pursuit (SP) [29], Compressive Sampling Matching Pursuit (CoSaMP) [75], Regularized Orthogonal Matching Pursuit (ROMP) [76], [77], Gradient Pursuit (GP) [13], Iterative Hard Thresholding (IHT) [14], [15] and Hard Thresholding Pursuit (HTP) [48]. These algorithms employ different ways to select support
2.1. Sparse Synthesis Model

Algorithm 2 Orthogonal Matching Pursuit (OMP) Algorithm

\textbf{Input:} \( y, D \)
\textbf{Output:} \( \hat{a} \)

\textbf{Initialization:}
Initialize the iteration counter \( i = 1 \), the representation residual \( r^{(i)} = y \), the representation vector \( a^{(i)} = 0 \) and the index set \( \Gamma^{(i)} = \emptyset \). Perform the following steps.

\textbf{Main Steps:}

1. Calculate the inner-products between the atoms and the current residual: \( b^{(i)} = D^T r^{(i)} \).
2. Determine the index of the atom which leads to the largest inner-product (in magnitude): \( \{ k^* \} = \text{arg max}_k |b^{(i)}_k| \).
3. Add the index of the selected atom to the index set: \( \Gamma^{(i+1)} = \Gamma^{(i)} \cup \{ k^* \} \).
4. Update the representation coefficients: \( a^{(i+1)} = D_{\cdot, \Gamma^{(i+1)}}^\dagger y \).
5. Update the representation residual: \( r^{(i+1)} = y - D_{\cdot, \Gamma^{(i+1)}} a^{(i+1)} \).
6. If the stopping criterion is satisfied, \( \hat{a} = a^{(i+1)} \) and quit the iteration. Otherwise, increase the iteration counter \( i = i + 1 \) and go back to step 1.

atoms or to update the representation vector. In particular, StOMP \cite{37} selects atoms in each iteration based on a hard thresholding scheme which enables the algorithm to stop after a fixed number of stages. SP \cite{29} and CoSaMP \cite{75} detect a fixed number of atoms in a group to update the list of selected atoms. The number of atoms in the group, which is the main difference between SP and CoSaMP, is determined by the sparsity. ROMP \cite{76}, \cite{77} also detects subsets of atoms as a whole part but it determines which subset to be selected through a regularization step. Its computational complexity is comparable to OMP in theory, but the experiments in \cite{76} demonstrate that the running time of ROMP is often shorter than that of OMP in practice. GP \cite{13} uses a greedy atom selection scheme, similar to MP and OMP, and applies directional optimization to approximate the computationally expensive procedure of orthogonal projection in OMP. This strategy leads to fast approximations to OMP. IHT \cite{14}, \cite{15} uses a simple iterative strategy with hard thresholding to select the support atoms. It can achieve performance that lies between those obtained with MP and OMP. Its computational complexity is however similar to MP. The HTP algorithm \cite{48} is similar to IHT, but it takes a different strategy to estimate the representation vector.
Relaxation Algorithms

Besides detecting the support atoms in a greedy manner, an alternative way to address the sparse coding problem is to approximate the $\ell_0$-norm with other norms or functions that can also promote sparsity but are computationally tractable. The relaxation algorithms adopt this way by using various forms as the replacements of the $\ell_0$-norm.

As $\|a\|_0$ reflects the number of non-zero elements in $a$, the $\ell_0$-norm can also be denoted as

$$\|a\|_0 = \lim_{p \to 0} \sum_{i=1}^{d} |a_i|^p.$$  \hspace{1cm} (2.3)

Fig. 2.1 shows the behavior of $|a|^p$ for a scalar $a$. This illustrates that when $p$ goes to zero, the curve becomes an indicator function, being 0 for $a = 0$ and 1 for every other value. Note that the value $p = 1$ leads to a convex function, while $0 < p < 1$ gives non-convex functions. Thus, the $\ell_1$-norm which is defined as

$$\|a\|_1 = \sum_{i=1}^{d} |a_i|,$$  \hspace{1cm} (2.4)

is the closest convex surrogate of the $\ell_0$-norm [40].

The Basis Pursuit (BP) [27] relaxes the $\ell_0$-norm in the sparse coding problem (2.2) as $\ell_1$-norm.
2.1. Sparse Synthesis Model

and thus leading to the formulation as follows

\[
a^* = \arg\min_a \|a\|_1 \quad \text{s.t.} \quad y = Da.
\]  

(2.5)

The above BP optimization problem (2.5) can be equivalently reformulated as a Linear Programming (LP) problem and any algorithm from the LP literature can be employed for solving this problem, for example, Dantzig’s simplex algorithm and interior point methods [27]. For the case where only noisy data is available, the BP algorithm is adapted as the Basis Pursuit DeNoising (BPDN) algorithm [27] by considering the error between the noisy signal and the reconstructed signal obtained by \( Da \). The BP and BPDN algorithms are much more sophisticated than the greedy algorithms reviewed earlier, as they pursue the global solutions to well-defined optimization problems. However, this also leads to higher computational complexities, as compared to the greedy algorithms.

The Least-Absolute-Shrinkage-and-Selection-Operator (LASSO) [98] also uses the \( \ell_1 \)-norm to promote sparsity, but in a different way from BP. In LASSO, the \( \ell_1 \)-norm is employed as a constraint for the minimization of residual sum of squares, which results in the following model

\[
a^* = \arg\min_a \|y - Da\|_2^2 \quad \text{s.t.} \quad \|a\|_1 \leq s,
\]  

(2.6)

where the parameter \( s \) controls the level of sparsity. LASSO was originally proposed for linear regression in the context of statistics, but it has also been applied to solve the sparse coding problem.

Another option for relaxing \( \ell_0 \)-norm is the \( \ell_p \)-norm defined as

\[
\|a\|_p = \left( \sum_{i=1}^d |a_i|^p \right)^{1/p}
\]  

(2.7)

with \( 0 < p < 1 \). FOcal Under-determined System Solver (FOCUSS) [53], [87] employs the \( \ell_p \)-norm as the measurement of sparsity and formulates the sparse coding task as an \( \ell_p \) minimization problem, that is

\[
a^* = \arg\min_a \|a\|_p \quad \text{s.t.} \quad y = Da.
\]  

(2.8)
Chapter 2. Literature Review on Sparse Representation and Dictionary Learning

The solution to this problem can be approximated iteratively by solving a sequence of weighted $\ell_2$-norm minimization problems with the weights depending on the solution obtained in the last iteration [87]. In particular, suppose the solution in the $i$-th iteration is denoted by $a^{(i)}$, and the weight will be constructed as $W^{(i+1)} = \text{diag}(a^{(i)})$ where $\text{diag}(\cdot)$ returns a diagonal matrix with the elements of its operand on the main diagonal. In the $(i + 1)$-th iteration, the solution will be updated by solving the problem

$$a^{(i+1)} = \arg\min_a \| (W^{(i+1)})^\dagger a \|_2^2 \quad \text{s.t.} \quad y = Da,$$

which has the closed-form solution $a^{(i+1)} = W^{(i+1)} (D W^{(i+1)})^\dagger y$. The FOCUSS algorithm has been proved to converge to a fixed sparse solution absolutely for any starting point [53].

2.1.2 Synthesis Dictionary Learning

The choice of the dictionary in the synthesis model is crucial for the performance of this model, and the second challenge for the synthesis model is to design or learn a dictionary $D$ to represent a set of signals as sparsely as possible. Many analytical dictionaries have been developed [91], but dictionaries learned from a set of training signals have the potential to fit these signals better than the analytical dictionaries [5]. As a result, the synthesis dictionary learning (SDL) problem has become one of the most popular topics in sparse representation.

In particular, the goal of SDL is to seek the dictionary $D$ that leads to the best representations for a given set of training signals. Let $Y \in \mathbb{R}^{m \times n}$ denote the set of training signals, where each column of $Y$ is one training signal. Assuming that every signal in this set can be approximated with the atoms of the dictionary $D$, the SDL problem can be formulated as [5]

$$\arg\min_{\{D, A\}} \sum_i \| A_{:,i} \|_0 \quad \text{s.t.} \quad \| Y - DA \|_F^2 \leq \epsilon, \quad \forall i,$$

where $A_{:,i}$ denotes the coefficient vector corresponding to the signal $Y_{:,i}$, the $i$-th column of $Y$. The notation $\| \cdot \|_F$ stands for the Frobenius norm, defined as $\|X\|_F = \sqrt{\sum_{i,j} x_{i,j}^2}$ where $x_{i,j}$ is the $i, j$-th element of $X$.

Many algorithms have been proposed to address the SDL problem [46], [5], [30]. These algorithms typically formulate the dictionary learning task as an optimization problem which can
be addressed by alternating iteratively between the update of the coefficients $A$ and the update of the dictionary $D$. These two steps are commonly followed by the synthesis dictionary learning algorithms, as shown in Fig. 2.2.

In the following subsection, several popular dictionary learning algorithms are introduced, i.e. Maximum Likelihood (ML) based dictionary learning \[80\], Method of Optimal Directions (MOD) \[46\], K-SVD \[5\] and Simultaneous Codeword Optimization (SimCO) \[30\]. The main differences between these algorithms are their formulations and the corresponding optimization methods, as summarized in Table 2.1. The ML-based method \[80\] formulates the SDL task in a probabilistic framework where the sparsity property of $A$ is embedded in the prior distribution. Other algorithms minimize the representation errors directly and restrict the sparsity of $A$ using an $\ell_0$-norm constraint. The constraint that all columns in $D$ have unit $\ell_2$-norm is considered in all algorithms to avoid trivial solutions. As mentioned below, the optimization methods associated with these algorithms can be divided as the sparse coding stage and dictionary update stage. In the sparse coding stage, the dictionary $D$ is fixed and the coefficient matrix $A$ is updated using gradient descent or sparse coding algorithms. In the dictionary update stage, $D$ is updated, and $A$ is fixed in the ML-based method \[80\] and MOD \[46\], updated in K-SVD \[5\] and SimCO \[30\]. More details of these algorithms will be presented in the following.

**Maximum Likelihood Based Dictionary Learning**

The method reported in \[80\] considers the dictionary learning task with probabilistic reasoning. In particular, the proposed method learns the dictionary by maximizing the likelihood function $P(Y|D)$. Assuming the training samples are independent, $P(Y|D)$ can be written as

$$P(Y|D) = \prod_{i=1}^{n} P(Y_{i}|D).$$  \hspace{1cm} (2.11)
Table 2.1: Comparison of different SDL algorithms

<table>
<thead>
<tr>
<th>SDL Algorithm</th>
<th>Formulation</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sparse Coding</td>
</tr>
<tr>
<td>ML-based</td>
<td>(1) Probabilistic framework; (2) All columns in $D$ have unit $\ell_2$-norm.</td>
<td>$D$: Gradient descent</td>
</tr>
<tr>
<td>MOD [46]</td>
<td>(1) Minimize representation error; (2) All columns in $D$ have unit $\ell_2$-norm; (3) $\ell_0$-norm constraint on $A$.</td>
<td>$D$: Sparse coding algorithms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A$: Fixed</td>
</tr>
</tbody>
</table>

The likelihood function with respect to each training sample $Y_{:,i}$ can be computed via

$$P(Y_{:,i}|D) = \int P(Y_{:,i}|A_{:,i}, D)P(A_{:,i}) dA_{:,i}. \quad (2.12)$$

To calculate the required distributions $P(Y_{:,i}|A_{:,i}, D)$ and $P(A_{:,i})$, two assumptions are made. The first assumption is that the representation error $\|Y_{:,i} - DA_{:,i}\|_2^2$ is Gaussian distributed with zero mean and variance $\sigma^2$, which results in the following density function

$$P(Y_{:,i}|A_{:,i}, D) = c \cdot \exp\left(-\frac{1}{2\sigma^2}\|Y_{:,i} - DA_{:,i}\|_2^2\right) \quad (2.13)$$

with $c$ being a constant scale factor. The second assumption is that the entries of $A_{:,i}$ are distributed independently with Cauchy distribution, i.e.

$$P(A_{:,i}) = c \cdot \exp\left(-\beta \sum_{j=1}^{d} \ln(1 + a_{j,i})\right). \quad (2.14)$$

Substituting the functions (2.13) and (2.14) into the expression of the likelihood function (2.12), the SDL problem can be formulated as

$$\{D^*\} = \arg \max_D \sum_{i=1}^{n} \arg \max_{A_{:,i}} \left\{ \ln \left( P(Y_{:,i}|A_{:,i}, D) P(A_{:,i}) \right) \right\}$$

$$= \arg \min_D \sum_{i=1}^{n} \arg \min_{A_{:,i}} \left\{ \|Y_{:,i} - DA_{:,i}\|_2^2 + \lambda \sum_{j=1}^{d} \ln(1 + a_{j,i}) \right\} \quad (2.15)$$
2.1. Sparse Synthesis Model

with $\lambda = 2\sigma^2\beta$. This problem is solved by an iterative method consisting of two alternating steps. In the first step, the sparse coefficients $A$ are updated using a gradient descent method with the dictionary being fixed. In the second step, $A$ is fixed and the dictionary $D$ is updated via gradient descent.

MOD

MOD presented in [46] minimizes the reconstruction error between the training signals $Y$ and their sparse approximations $DX$ with a sparsity constraint, which can be stated mathematically as

$$\{D^*, A^*\} = \arg \min_{\{D,A\}} \|Y - DA\|_F^2 \quad \text{s.t.} \quad \|A_{:,i}\|_0 \leq s, \ \forall i. \quad (2.16)$$

This problem is addressed by updating $D$ and $A$ alternatively, with the other fixed. For a fixed $D$, as $\|Y - DA\|_F^2 = \sum_{i=1}^n \|Y_{:,i} - DA_{:,i}\|_2^2$, the optimization on $A$ can be decoupled as a set of sub-problems with respect to each column of $A$. The sub-problem for optimizing the $i$-th column $A_{:,i}$ can be written as

$$\{A^*_{:,i}\} = \arg \min_{A_{:,i}} \|Y_{:,i} - DA_{:,i}\|_2^2 \quad \text{s.t.} \quad \|A_{:,i}\|_0 \leq s, \quad (2.17)$$

which can be solved by sparse coding algorithms. For the update of $D$ with a fixed $A$, MOD aims to estimate $D$ by solving the following quadratic problem

$$\{D^*\} = \arg \min_D \|Y - DA\|_F^2. \quad (2.18)$$

The optimal solution to this problem can be obtained by setting the gradient of the objective function with respect to $D$ as zero, i.e.

$$-2(Y - DA)A^T = 0, \quad (2.19)$$

and thus the updated dictionary can be given by the closed-form solution to equation $2.19$, which can be expressed as

$$D = YA^T(AA^T)^{-1} = YA^\dagger \quad (2.20)$$

with $A^\dagger = A^T(AA^T)^{-1}$. 
K-SVD

The K-SVD algorithm \[5\] also addresses problem (2.16) as MOD \[46\]. The coefficient matrix \(A\) is also updated using sparse coding algorithms, however, the method for updating the dictionary is modified in K-SVD. In particular, the dictionary is updated atom-by-atom, rather than as a whole matrix, and the corresponding coefficients are also updated at the same time. For updating the \(j\)-th atom \(D_{\cdot,j}\) together with its corresponding coefficients \(A_{j,:}\), the approximation error can be rewritten as

\[
\|Y - DA\|_F^2 = \|Y - \sum_{i=1}^d D_{\cdot,i}A_{i,:}\|_F^2 \\
= \left\| \left( Y - \sum_{i \neq j} D_{\cdot,i}A_{i,:} \right) - D_{\cdot,j}A_{j,:} \right\|^2_F \\
= \|R_j - D_{\cdot,j}A_{j,:}\|_F^2, \tag{2.21}
\]

where the matrix \(R_j\) represents the representation error when the \(j\)-th atom is removed. In this form, the multiplication \(DA\) is expanded as the sum of the products of two rank-1 matrices, i.e. \(DA = \sum_{i=1}^d D_{\cdot,i}A_{i,:}\).

Assuming \(d - 1\) atoms are fixed, K-SVD only updates the \(j\)-th atom \(D_{\cdot,j}\) and its corresponding coefficients \(A_{j,:}\) at one time to minimize the approximation error (2.21). This can be achieved by applying Singular Value Decomposition (SVD) to the matrix \(R_j\). However, the direct application of SVD may result in non-sparse vector \(A_{j,:}\), as the sparsity constraint is not enforced. To address this problem, the non-zero pattern (support) of \(A_{j,:}\) is fixed by only updating the values of the non-zero entries in \(A_{j,:}\). The reduced vector \(A_{j,:}^E\), which only contains the non-zero elements of the vector \(A_{j,:}\), can be extracted by multiplying a matrix \(E_j\), and the approximation error (2.21) can be rewritten as the following form by applying the matrix \(E_j\)

\[
\|R_j^E - D_{\cdot,j}A_{j,:}^E\|_F^2 = \|R_j^E - D_{\cdot,j}A_{j,:}^E\|_F^2, \tag{2.22}
\]

where \(R_j^E = R_jE_j\) and \(A_{j,:}^E = A_{j,:}E_j\). As a result, the update of \(D_{\cdot,j}\) and \(A_{j,:}\) by minimizing the approximation error (2.21) can be transformed to minimizing (2.22) with the zero entries...
2.1. Sparse Synthesis Model

of $A_{j,:}$ being unchanged, that is

\[
\{D_{:,j}^*, A_{j,:}^{E*}\} = \arg \min_{\{D_{:,j}, A_{j,:}^E\}} \|R_j^E - D_{:,j}A_{j,:}^E\|_F^2. \tag{2.23}
\]

Suppose the SVD of $R_j^E$ is $R_j^E = U \Sigma V^T$, and then the optimal solutions to the above problem can be obtained via $D_{:,j}^* = U_{:,1}$ and $A_{j,:}^{E*} = \Sigma_{1,:} V_{:,1}^T$. The K-SVD algorithm is summarized in Algorithm 3.

**Algorithm 3 K-SVD**

**Input:** training samples $Y$, sparsity $s$

**Output:** $D^*$

**Initialization:**
Initialize the iteration counter $k = 1$ and the dictionary $D^{(k)}$. Perform the following steps.

**Main Iterations:**

1. Sparse coding:
   For each training signal $Y_{:,i}$, use a sparse coding algorithm to update the sparse representation $A_{:,i}^{(k)}$, by approximating the solution of (2.17).

2. Dictionary update:
   For each column $j = 1, 2, \ldots, d$ in $D^{(k)}$, update it as follows
   
   (a) Compute the representation error matrix $R_j$, i.e. $R_j = Y - \sum_{i \neq j} D_{:,i}^{(k)} A_{:,i}^{(k+1)}$.

   (b) Restrict $R_j$ by selecting only the columns corresponding to the non-zero elements in $A_{:,i}^{(k+1)}$, i.e. $R_j^E \leftarrow R_j$.

   (c) Compute the SVD of $R_j^E$, i.e. $R_j^E = U \Sigma V^T$, update the $j$-th column of $D^{(k)}$ via $D_{:,j}^{(k+1)} = U_{:,1}$, and update the non-zero elements in $A_{j,:}^{(k+1)}$ to be $\Sigma_{1,:} V_{:,1}^T$.

3. If the stopping criterion is satisfied, $D^* = D^{(k+1)}$ and quit the iteration. Otherwise, increase the iteration counter $k = k + 1$ and go back to step 1.

To obtain an incoherent dictionary, the standard K-SVD algorithm [5] has been extended to Incoherent K-SVD (IK-SVD) [2]. In particular, an extra step to decrease the correlations between the dictionary atoms is employed after updating all atoms in K-SVD. This step is achieved by solving the minimization problem as follows

\[
\hat{D} = \arg \min_D \|DD^T - I\|_F^2, \tag{2.24}
\]
which can be addressed by the gradient descent method.

Other variants of the K-SVD algorithm have also been developed to address the problems in real applications. For example, the label consistent K-SVD algorithm [61] associates label information with the dictionary atoms during the dictionary learning process so that a discriminative dictionary can be learned for recognition. The algorithm proposed in [85] employs linear extension of Graph Embedding to optimize K-SVD in terms of computational complexity for classification problems. As large variations in the update of the dictionary may overshoot the optimal solution and slow down the convergence process, the dictionary learning algorithm presented in [83] introduces a regularization term controlling the modifications of the dictionaries to improve the convergence speed.

**SimCO**

SimCO proposed in [30] can update multiple atoms and their corresponding coefficients simultaneously with a fixed sparsity pattern, which gives rise to the term Simultaneous Codeword Optimization (SimCO). In particular, the dictionary learning problem is formulated as

$$
\mathbf{D}^* = \arg \min_{\mathbf{D} \in \mathcal{D}} f(\mathbf{D}) = \arg \min_{\mathbf{D} \in \mathcal{D}} \min_{\mathbf{A} \in \mathcal{A}} \left\| \mathbf{Y} - \mathbf{DA} \right\|_F^2.
$$

(2.25)

In this formulation, \( \mathbf{D} \) is assumed to contain unit \( \ell_2 \)-norm columns, which is addressed by the constraint \( \mathbf{D} \in \mathcal{D} \) with \( \mathcal{D} \) representing the set of all matrices that contain unit \( \ell_2 \)-norm columns, i.e. \( \mathcal{D} = \{ \mathbf{D} \in \mathbb{R}^{m \times d} : \|\mathbf{D}_{:,i}\|_2 = 1, \; \forall i = 1, 2, \ldots, d \} \). The positions of the non-zero elements of the coefficient matrix \( \mathbf{A} \) are fixed, achieved with the constraint \( \mathbf{A} \in \mathcal{A} \).

To solve the optimization problem (2.25), SimCO follows the conventional two-stage optimization process – a sparse coding stage and a dictionary update stage. This algorithm is summarized as Algorithm 4. As in MOD and K-SVD, the sparse coding stage determines the sparse representations \( \mathbf{A} \) of the signals in \( \mathbf{Y} \) for a given dictionary \( \mathbf{D} \), using various sparse coding algorithms such as OMP [100]. In the dictionary update stage, multiple atoms and their corresponding coefficients are updated with the positions of the non-zeros in \( \mathbf{A} \) unchanged. Note the Stiefel manifold is defined as \( \mathcal{S}_{m,1} = \{ \mathbf{u} \in \mathbb{R}^m : \mathbf{u}^T \mathbf{u} = 1 \} \) and the columns of \( \mathbf{D} \) are constrained to have unit \( \ell_2 \)-norm, and thus the update of \( \mathbf{D} \) can be addressed by the opti-
2.2. Sparse Analysis Model

Algorithm 4 SimCO

**Input:** training samples $Y$, sparsity $s$

**Output:** $D^*$

**Initialization:**
Initialize the iteration counter $k = 1$ and the dictionary $D^{(k)}$. Perform the following steps.

**Main Iterations:**

1. **Sparse coding:**
   For each training signal $Y_{:,i}$, use a sparse coding algorithm to update the sparse representation $A^{(k)}_{:,i}$, by approximating the solution of (2.17).

2. **Dictionary update:**
   (a) Use optimization on manifolds to update the dictionary $D^{(k+1)} \leftarrow D^{(k)}$.

   (b) For each column of $A_{:,i}$, determine the index set of non-zero elements $\Psi$, and update the non-zero elements using equation (2.27).

3. If the stopping criterion is satisfied, $D^* = D^{(k+1)}$ and quit the iteration. Otherwise, increase the iteration counter $k = k + 1$ and go back to step 1.

Optimization methods on manifolds, for example, the gradient descent algorithm and line search Newton-Conjugate Gradient (LSNCG) algorithm [78]. Once the dictionary $D$ is updated, the coefficient matrix $A$ is also updated by approximating the solution to the following problem

$$A^* = \arg \min_{A \in \mathcal{A}} \| Y - DA \|_F^2.$$  \hspace{1cm} (2.26)

Note that the positions of the non-zero elements in $A$ are kept unchanged, using the constraint $A \in \mathcal{A}$. Similar to K-SVD, this can also be achieved by only updating the values of the non-zero entries in $A$. Let $\Psi$ denote the index set associated with the non-zero elements in $A_{:,i}$, and then the non-zero elements in $A$ can be updated by

$$A_{\Psi, i} = \left( D_{:, \Psi}^T D_{:, \Psi} \right)^{-1} D_{:, \Psi} Y_{:, i}. \hspace{1cm} (2.27)$$

2.2 Sparse Analysis Model

Based on the sparsity assumption, there exists an alternative sparse model which is known as the sparse analysis model or co-sparse analysis model [44], [74]. In contrast to the synthesis
model, the sparse analysis model uses an *analysis dictionary* \( \Omega \in \mathbb{R}^{p \times m} \) with \( p > m \) to “analyze” the signal \( y \in \mathbb{R}^m \). Specifically, it assumes that the product of \( \Omega \) and \( y \) is sparse, i.e.

\[
\mathbf{x} = \Omega \mathbf{y} \quad \text{with} \quad \|\mathbf{x}\|_0 = p - l,
\]

(2.28)

where \( 0 \leq l \leq p \) is the number of zeros in \( \mathbf{x} \in \mathbb{R}^p \). The matrix \( \Omega \) is usually referred to as an *analysis dictionary* [92] or *analysis operator* [107], [56], with each row of \( \Omega \) being an *atom*. The vector \( \mathbf{x} \) is the *analysis representation* of the signal \( y \) with respect to \( \Omega \), and the number of zeros \( l \) is called the *co-sparsity* of the signal \( y \) with respect to \( \Omega \) [74].

Let \( \Lambda = \{i : x_i = 0\} \) denote the index set of the rows in \( \Omega \) corresponding to the zero elements in \( \mathbf{x} \) (thus, \( \text{card}(\Lambda) = l \)) and let \( \Omega_{\Lambda,:} \) denote the sub-matrix of \( \Omega \) containing only the rows indexed by \( \Lambda \). The set \( \Lambda \) is called the *co-support* of \( y \). For the analysis model, we have \( \Omega_{\Lambda,:} \mathbf{y} = \mathbf{0} \), meaning that the \( l \) atoms indexed by \( \Lambda \) are orthogonal to the signal \( y \). From the subspace point of view, \( y \) lies in the subspace which is orthogonal to the subspace spanned by the rows of \( \Omega_{\Lambda,:} \). Even though the description of the sparse analysis model may seem similar to its synthesis counterpart, these two models differ significantly if the dictionaries are overcomplete [44].

### 2.2.1 Analysis Pursuit

If the signal \( y \) is known, its analysis representation with respect to a given \( \Omega \) can be obtained via multiplying \( y \) by \( \Omega \). However, when the measurements of \( y \) are noisy or incomplete, the original signal \( y \) has to be estimated first in order to get its analysis representation, which leads to the *analysis pursuit problem* [92]. Suppose the measurements of the signal \( y \) are given by \( \mathbf{z} = \mathbf{M} \mathbf{y} + \mathbf{v} \), where the measurement matrix \( \mathbf{M} \in \mathbb{R}^{q \times m} \) with \( q < m \) is incomplete and \( \mathbf{v} \) is an additive noise that satisfies \( \|\mathbf{v}\|_2^2 \leq \epsilon \). The analysis pursuit problem aims to reconstruct the signal \( y \) from its measurements \( \mathbf{z} \), which can be formulated as

\[
\mathbf{y}^* = \arg \min_{\mathbf{y}} \|\Omega \mathbf{y}\|_0 \quad \text{s.t.} \quad \|\mathbf{z} - \mathbf{M} \mathbf{y}\|_2^2 \leq \epsilon.
\]

(2.29)

Solving this problem has been shown to be NP-complete [50], [74], just as the sparse coding problem for the synthesis model, and thus approaches to addressing it approximately are re-
2.2. Sparse Analysis Model

quired. Similar to the case of sparse coding algorithms, the analysis pursuit algorithms can also be classified as greedy methods [92], [74] and relaxation methods [44], [74], according to the way to approximate the problem.

**Greedy Methods**

The greedy algorithms attempt to solve the original analysis pursuit problem (2.29) directly. The critical point is to determine the co-support of the signal in a greedy fashion and update the estimated signal using the detected co-support. Typical greedy methods are Backward-Greedy (BG) [92], Optimized-Backward-Greedy (OOG) [92] and Greedy Analysis Pursuit (GAP) [74].

The BG and OOG algorithms aim to address the analysis pursuit problem (2.29) in the case that \( M = I \), i.e.

\[
\begin{align*}
\mathbf{y}^* = \arg \min_{\mathbf{y}} \| \mathbf{\Omega y} \|_0 \quad \text{s.t.} \quad \| \mathbf{z} - \mathbf{y} \|_2^2 \leq \epsilon. 
\end{align*}
\]  

(2.30)

In BG, the above problem is considered from the subspace point of view. Suppose that the dimension of the signal subspace is \( r \), with \( r \ll m \), and the dimension of the subspace spanned by the rows of \( \mathbf{\Omega}_\Lambda \) can be denoted by \( m - r \), because of the orthogonality between the signal and the co-support rows \( \mathbf{\Omega}_\Lambda \). As a result, the problem (2.30) can be reformulated as

\[
\begin{align*}
\{ \mathbf{y}^*, \Lambda^* \} = \arg \min_{\{ \mathbf{y}, \Lambda \}} \| \mathbf{z} - \mathbf{y} \|_2^2 \\
\text{s.t.} \quad \mathbf{\Omega}_{\Lambda^*} \mathbf{y} = \mathbf{0}, \\
\text{rank}(\mathbf{\Omega}_{\Lambda^*}) = m - r.
\end{align*}
\]  

(2.31)

To approximate the optimal solution, BG selects co-support rows and estimates the signal iteratively. It starts with an empty co-support set and the estimated signal is initialized as the given signal \( \mathbf{z} \). In each iteration, the atom which has the minimal inner-product with the estimated signal is selected as a co-support row with its index being added to the co-support set, and then the estimated signal is updated by projecting the noisy measurement onto the null space of all detected co-support rows so far. After that, the inner-products of the atoms in \( \mathbf{\Omega} \) and the new estimated signal are calculated, based on which the co-support set is refined as the indices of the rows that lead to the inner-products below some given threshold \( \epsilon_0 \). The iteration process described above is repeated until the target subspace dimension is achieved. The BG algorithm
Chapter 2. Literature Review on Sparse Representation and Dictionary Learning

is summarized as Algorithm 5.

**Algorithm 5** Backward-Greedy (BG) Algorithm

**Input:** $z \in \mathbb{R}^m$, $\Omega \in \mathbb{R}^{p \times m}$, $r$, $\epsilon_0$

**Output:** $\hat{y}$

**Initialization:**
Initialize the iteration counter $i = 1$, the estimated signal $y^{(i)} = z$, and the co-support set $\Lambda^{(i)} = \emptyset$. Perform the following steps.

**Main Steps:**

1. Calculate the inner-products: $b^{(i)} = \Omega y^{(i)}$.

2. Determine the index of the atom which leads to the smallest inner-product (in magnitude): $\{k^*\} = \text{arg min}_{k \notin \Lambda^{(i)}} |b_k^{(i)}|$.

3. Add the index of the selected atom to the index set: $\Lambda^{(i+1)} = \Lambda^{(i)} \cup \{k^*\}$.

4. Update the estimated signal by projection: $y^{(i+1)} = (I - \Omega_{\Lambda^{(i)}},: \Omega_{\Lambda^{(i)}}:)z$.

5. Refine the co-support set: $\Lambda^{(i+1)} = \{k : 1 \leq k \leq p, |\langle \Omega_{k,:}, y^{(i+1)} \rangle| < \epsilon_0\}$.

6. If $i = m - r$, the estimated signal is $\hat{y} = y^{(i+1)}$ and quit the iteration. Otherwise, increase the iteration counter $i = i + 1$ and go back to step 1.

The OBG algorithm [92] also addresses the problem (2.31). It is an improved version of BG by considering a more sophisticated approach to selecting the co-support rows in its iteration. In particular, rather than simply selecting the atom with minimal inner-product as the co-support row as in BG (cf. Step 2 of Algorithm 5), OBG performs the projection step (cf. Step 4 of Algorithm 5) using the new co-supports that are created by adding each row respectively, and chooses the one which leads to the smallest decrease in the signal energy after the projection. The OBG algorithm is presented in Algorithm 6.

Unlike BG/OBG [92], the GAP algorithm [73], [74] considers the more general analysis pursuit problem (2.29) without the precondition $M = I$. Another main difference between GAP and BG/OBG is that GAP detects the rows that are corresponding to the non-zero entries in $\Omega y$ rather than the zeros. Specifically, the co-support set is initialized with all row indices of $\Omega$ and then the rows having large inner-products with the estimated signal are removed iteratively.
Algorithm 6 Optimized-Backward-Greedy (OBG) Algorithm

**Input:** \(z \in \mathbb{R}^m, \Omega \in \mathbb{R}^{p \times m}, r, \epsilon_0\)

**Output:** \(\hat{y}\)

**Initialization:**

Initialize the iteration counter \(i = 1\), the estimated signal \(y^{(i)} = z\), and the co-support set \(\Lambda^{(i)} = \emptyset\). Perform the following steps.

**Main Steps:**

1. For each \(k \notin \Lambda^{(i)}\), perform the steps (a)-(c) successively:
   
   (a) Develop a temporary co-support set: \(\tilde{\Lambda} = \Lambda^{(i)} \cup \{k\}\).
   
   (b) Perform projection: \(\tilde{y} = (I - \Omega_{\Lambda^{(i)}}^\dagger \Omega_{\Lambda^{(i)}})z\).
   
   (c) Calculate the approximation error: \(e_k = \|\tilde{y} - y^{(i)}\|_2\).

2. Determine the index of the atom which leads to the smallest approximation error:
   
   \(\{k^*\} = \arg \min_{k \notin \Lambda^{(i)}} e_k\).

3. Add the index of the selected atom to the index set: \(\Lambda^{(i+1)} = \Lambda^{(i)} \cup \{k^*\}\).

4. Update the estimated signal by projection: \(y^{(i+1)} = (I - \Omega_{\Lambda^{(i)}}^\dagger \Omega_{\Lambda^{(i)}})z\).

5. Refine the co-support set: \(\Lambda^{(i+1)} = \{k : 1 \leq k \leq p, |\langle \Omega_{k,:}, y^{(i+1)} \rangle| < \epsilon_0\}\).

6. If \(i = m - r\), the estimated signal is \(\hat{y} = y^{(i+1)}\) and quit the iteration. Otherwise, increase the iteration counter \(i = i + 1\) and go back to step 1.

and the signal will be updated as the solution of the problem as follows

\[
y^{(i)} = \arg \min_y \lambda \|\Omega_{\Lambda^{(i)}} y\|_2^2 + \|z - My\|_2^2, \tag{2.32}\]

where \(\lambda\) is the Lagrangian parameter depending on the energy of the noise. The optimal solution to the above problem (2.32) can be computed by setting the gradient as zero, which is given by

\[
y^{(i)} = (M^T M + \lambda \Omega_{\Lambda^{(i)}}^T \Omega_{\Lambda^{(i)}})^{-1} M^T z. \tag{2.33}\]

We refer to the above equation to calculate \(y^{(i)}\) using \(\Omega_{\Lambda^{(i)}}\) as the following function with respect to \(\Omega_{\Lambda^{(i)}}\), i.e.

\[
g(\Omega_{\Lambda^{(i)}}) = (M^T M + \lambda \Omega_{\Lambda^{(i)}}^T \Omega_{\Lambda^{(i)}})^{-1} M^T z. \tag{2.34}\]

The GAP algorithm is summarized as Algorithm 7. Note that the indices to be removed from
the co-support set are determined using a selection factor $0 < t \leq 1$, which allows the selection of multiple rows at once.

**Algorithm 7 Greedy Analysis Pursuit (GAP) Algorithm**

**Input:** $\mathbf{z} \in \mathbb{R}^m$, $\mathbf{\Omega} \in \mathbb{R}^{p \times m}$, $\mathbf{M} \in \mathbb{R}^{q \times m}$, $t \in (0, 1]$

**Output:** $\hat{\mathbf{y}}$

**Initialization:**

Initialize the iteration counter $i = 1$, the co-support set $\Lambda(i) = \{1, 2, 3, ..., p\}$, and the estimated signal $\mathbf{y}(i) = g(\mathbf{\Omega}_{\Lambda(i)}, \cdot)$. Perform the following steps.

**Main Steps:**

1. Calculate the inner-products: $\mathbf{b}(i) = \mathbf{\Omega}\mathbf{y}(i)$.
2. Find the largest entries: $\Gamma(i) = \{k : |b_k(i)| \geq t \max_j |b_j(i)|\}$.
3. Update the co-support: $\Lambda(i+1) = \Lambda(i) \setminus \Gamma(i)$.
4. Update the estimated signal by projection: $\mathbf{y}(i+1) = g(\mathbf{\Omega}_{\Lambda(i+1)}, \cdot)$.
5. If the stopping criterion is satisfied, $\hat{\mathbf{y}} = \mathbf{y}(i+1)$ and quit the iteration. Otherwise, increase the iteration counter $i = i + 1$ and go back to step 1.

Other existing greedy methods include Analysis Iterative Hard Thresholding (AIHT), Analysis Hard Thresholding Pursuit (AHTP), Analysis Compressive Sampling Matching Pursuit (ACoSaMP) and Analysis Subspace Pursuit (ASP) [51], which are the analysis counterparts of the synthesis model based sparse coding algorithms IHT [15], HTP [48], CoSaMP [75] and SP [29]. These analysis versions approximate a variant of (2.29) with a given co-sparsity $l$, that is

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \|\mathbf{z} - \mathbf{M}\mathbf{y}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{\Omega}\mathbf{y}\|_0 \leq p - l. \quad (2.35)$$

Based on this formulation, these algorithms follow the same procedures as in their corresponding synthesis counterparts in general. Nevertheless, the operations in some steps are adapted to the analysis model accordingly. For example, the detection of the new support elements in the synthesis case is replaced with a co-support selection operator which chooses the indices of the $l$ smallest (in magnitude) elements after applying $\mathbf{\Omega}$ on $\mathbf{z}$, the signal estimate step is changed to approximate the projection to the orthogonal subspace of the subspace spanned by the co-support rows, and the set operation to combine the co-supports detected in the current and previous iterations is changed from “union” to “intersection”. Theoretical results related
to the stability of these greedy algorithms have been given in [51]. Comparing performances of the algorithms for reconstructing signals, ACoSaMP and ASP achieve better results than AHTP and AIHT, and AHTP outperforms AIHT.

### Relaxation Methods

A natural relaxation of the analysis pursuit problem is to utilize the convex surrogate of the \(\ell_0\)-norm, i.e. \(\ell_1\)-norm, to promote sparsity [44], [74], which can be formulated as

\[
y^* = \arg\min_y \|\Omega y\|_1 \quad \text{subject to} \quad \|z - My\|_2^2 \leq \epsilon.
\]

(2.36)

This is a convex optimization problem, which admits computationally tractable algorithms to solve it. Typical algorithms to address this problem can be found in [20], [86]. The Matlab cvx package [54] is also available to address it.

#### 2.2.2 Analysis Dictionary Learning

In a similar way to the dictionary in the synthesis model, the analysis dictionary \(\Omega\) also plays an important role in the analysis representation of the signal \(y\), and the dictionaries learned from a set of training signals show some advantages compared with pre-defined dictionaries [92]. In the past few years, the analysis dictionary learning (ADL) problem has begun to attract much attention [92], [107], [56]. Given a set of training signals \(Y \in \mathbb{R}^{m \times n}\) with each column being one training sample, the goal of ADL is to design an analysis dictionary \(\Omega \in \mathbb{R}^{p \times m}\) which can provide the sparsest representation for the training samples in \(Y\). This can be modeled as

\[
\Omega^* = F(\Omega Y),
\]

(2.37)

where \(F(\cdot)\) is the function to promote the co-sparsity. Typical choices for \(F(\cdot)\) include

\[
F(\Omega Y) = \|\Omega Y\|_0
\]

(2.38)

and its \(\ell_1\) relaxation version

\[
F(\Omega Y) = \|\Omega Y\|_1.
\]

(2.39)
Compared to the dictionary learning for the synthesis model, less work has been done for the ADL problem, and the first part of our work focus on this problem as will be presented in Chapter 3. For now, the existing ADL algorithms will be reviewed as follows.

Analysis K-SVD

The Analysis K-SVD algorithm [92] assumes that the training samples are noisy signals $Z \in \mathbb{R}^{m \times n}$ with $Z = Y + V$ and aims to learn an analysis dictionary which can represent the original signals $Y$ sparsely. It minimizes the error between the training samples and the signals estimated using the learned dictionary, i.e.

$$\left\{ \Omega^*, Y^*, \{\Lambda_i^*\}_{i=1}^n \right\} = \arg \min_{\{\Omega, Y, \{\Lambda_i\}_{i=1}^n\}} \|Z - Y\|_F^2$$

s.t. $\Omega_{\Lambda_i}: Y_{:,i} = 0$, $\forall 1 \leq i \leq n$

$$\text{rank} (\Omega_{\Lambda_i,:}) = m - r$, $\forall 1 \leq i \leq n$$

$$\|\Omega_{j,:}\|_2 = 1$, $\forall 1 \leq j \leq p,$

where $\Lambda_i$ denotes the co-support of the $i$-th signal in $Y$, as demonstrated with the constraints $\Omega_{\Lambda_i}: Y_{:,i} = 0$, $\forall 1 \leq i \leq n$, and $r$ is the dimension of the subspace that the noiseless signals lie in. Note that this formulation also takes the subspace point of view by assuming that the signals reside in a subspace with dimension $r$, as in the analysis pursuit algorithms BG and OBG [92], and this leads to the rank constraints, i.e. $\text{rank} (\Omega_{\Lambda_i,:}) = m - r$, $\forall 1 \leq i \leq n$. The atoms of $\Omega$ are assumed to have unit $\ell_2$-norm to avoid trivial solutions. The optimization of (2.40) is addressed by an iterative two-phase approach. In the first phase, the signals $Y$ and the co-supports $\{\Lambda_i\}_{i=1}^n$ are optimized with a fixed $\Omega$ and in the second phase $\Omega$ is updated. This process is repeated until some stopping criterion is met, as summarized in Algorithm 8.

Starting with an initial analysis dictionary, the algorithm optimizes $Y$ by considering each column individually, which results in a set of optimization problems corresponding to each signal in $Y$ respectively. For the $i$-th signal, the problem for optimizing $Y_{:,i}$ and the co-support
Algorithm 8 Analysis K-SVD

**Input:** training samples $Z$, subspace dimension $r$

**Output:** $\Omega^*$

**Initialization:**
- Initialize the iteration counter $k = 1$ and the dictionary $\Omega^{(k)}$. Perform the following steps.

**Main Iterations:**

1. **Analysis pursuit:**
   - For each training signal $Z_{:,i}$, use BG or OBG to detect the corresponding co-support $\Lambda_i$, by solving the problem (2.41).

2. **Dictionary update:**
   - For each row $j = 1, 2, ..., p$ in $\Omega^{(k)}$, update it as follows
     - (a) Determine the index set of the signals whose co-supports include the index $j$, i.e. $J = \{i : j \in \Lambda_i, i = 1, 2, ..., n\}$.
     - (b) Compute the SVD of $Z_{:,J}$, i.e. $Z_{:,J} = U \Sigma V^T$, update the $j$-th row of $\Omega^{(k)}_{j,:}$ to be the transpose of the singular vector in $U$ corresponding to the smallest singular value in $\Sigma$, i.e $\Omega^{(k)}_{j,:} \leftarrow \Omega^{(k+1)}_{j,:}$.

3. If the stopping criterion is satisfied, $\Omega^* = \Omega^{(k+1)}$ and quit the iteration. Otherwise, increase the iteration counter $k = k + 1$ and go back to step 1.

---

$\Lambda_i$ can be written as

$$\{Y^*_{:,i}, \Lambda^*_i\} = \arg \min_{\{Y_{:,i}, \Lambda_i\}} \|Z_{:,i} - Y_{:,i}\|_2^2$$

subject to

$$\Omega_{\Lambda_i,:} Y_{:,i} = 0$$

$$\text{rank}(\Omega_{\Lambda_i,:}) = m - r,$$

which can be solved using the analysis pursuit algorithm BG or OBG [92].

For the optimization of the dictionary, the atoms in $\Omega$ are updated sequentially using SVD, which is similar to the K-SVD algorithm [5] based on the synthesis model. Generally, each atom is updated to be the vector that is most orthogonal to all the signals associated to it based on the co-supports obtained in the first phase, and thus the update of the $j$-th atom $\Omega_{j,:}$ is affected only by the estimated signals in $Z$ that are orthogonal to it. Let the set $J$ collect the indices of the signals whose co-supports include the index $j$, i.e. $J = \{i : j \in \Lambda_i, i =$
Chapter 2. Literature Review on Sparse Representation and Dictionary Learning

1, 2, ..., n}, and then the atom $\Omega_{\cdot j}$ is updated by solving

$$\{\Omega_{\cdot j}\} = \arg \min_{\Omega_{\cdot j}} \| \Omega_{\cdot j} : Z_{\cdot , J} \|_2^2 \quad \text{s.t.} \quad \| \Omega_{\cdot j} \|_2 = 1,$$

(2.42)

where $Z_{\cdot , J}$ denotes the submatrix of $Z$ containing the columns indexed by the set $J$. The solution is given by singular vector of $Z_{\cdot , J}$ which corresponds to its smallest singular value, which can be obtained by computing the SVD of $Z_{\cdot , J}$. In order to avoid degenerated atoms in the learned dictionary, after updating all the atoms, the atoms that are too close to another atom or orthogonal to too few signals are replaced by new randomly generated atoms [92].

**LOST**

The Learning Overcomplete Sparsifying Transforms (LOST) algorithm [88] minimizes the so-called sparsifying error which is defined in the transform or analysis domain rather than the original signal domain as in the formulation of Analysis K-SVD [92]. In order to avoid all-zero atoms, LOST assumes the learned dictionary has a full column rank, and it also restricts the correlations between the atoms to eliminate similar or repeated rows, which results in the formulation as follows

$$\{\Omega^*, X^*\} = \arg \min_{\{\Omega, X\}} \| \Omega Y - X \|_F^2 - \lambda \log \det (\Omega^T \Omega) + \eta \sum_{j \neq k} |\langle \Omega_{\cdot j} : , \Omega_{\cdot k} : \rangle|^q$$

s.t. $\| X_{\cdot i} \|_0 \leq p - l, \forall 1 \leq i \leq n$

$$\| \Omega_{\cdot j} \|_2 = 1, \forall 1 \leq j \leq p.$$  

(2.43)

Note that the penalty term $\log \det (\Omega^T \Omega)$ aims to enforce the full column rank constraint on $\Omega$, and the term $\sum_{j \neq k} |\langle \Omega_{\cdot j} : , \Omega_{\cdot k} : \rangle|^q$ is to limit the correlations between two distinct rows. The optimization algorithm to solve problem (2.43) alternates between a sparse coding step for updating $X$ with a fixed $\Omega$, and a transform update step where only $\Omega$ is updated.

The sparse coding step updates $X$ only for a given $\Omega$ by solving the problem as follows

$$\{X^*\} = \arg \min_{X} \| \Omega Y - X \|_F^2 \quad \text{s.t.} \quad \| X_{\cdot i} \|_0 \leq p - l, \forall 1 \leq i \leq n.$$  

(2.44)

The solution $X^*$ can be calculated exactly by thresholding each column of the product matrix
\( \Omega Y \) to retain the \( p - l \) largest entries (in magnitude) in each column.

The transform update step is equivalent to optimizing the following problem with respect to \( \Omega \)

\[
\{ \Omega^* \} = \arg \min_{\Omega} \| \Omega Y - X \|_F^2 - \lambda \log \det (\Omega^T \Omega) + \eta \sum_{i \neq k} |\langle \Omega_{j,.}, \Omega_{k,.} \rangle|^{q} \\
\text{s.t. } \| \Omega_{j,.} \|_2 = 1, \forall 1 \leq j \leq p.
\] (2.45)

The above problem can be solved by the standard conjugate gradient algorithm with post-normalization of the rows of \( \Omega \). The convergence of LOST is demonstrated in [88], and its computational cost per iteration is lower than that of Analysis K-SVD [92].

**Transform K-SVD**

Transform K-SVD proposed recently in [39] combines the sparsifying error formulation of LOST [88] with the dictionary update approach of Analysis K-SVD [92]. It models the ADL problem as follows

\[
\{ \Omega^*, X^* \} = \arg \min_{\{ \Omega, X \}} \| \Omega Y - X \|_F^2 \\
\text{s.t. } \| X_{i,.} \|_0 \leq p - l, \forall 1 \leq i \leq n \\
\left\{ \max_{k \neq j} |\Omega_{k,.}, \Omega_{j,.}^T| \right\} \leq 1 - \delta, \forall 1 \leq j, k \leq p \\
\| \Omega_{i,.} \|_2^2 = 1, \forall 1 \leq j \leq p.
\] (2.46)

Compared with the formulation of LOST, Transform K-SVD also minimizes the sparsifying error term, but the penalty term enforcing a full column rank is removed and the term restricting the correlation between atoms is converted to the additional constraint \( \left\{ \max_{k \neq j} |\Omega_{k,.}, \Omega_{j,.}^T| \right\} \leq 1 - \delta, \forall 1 \leq j, k \leq p, \) with \( 0 < \delta < 1 \). The optimization of (2.46) also follows the two-step scheme of LOST, i.e \( X \) and \( \Omega \) are updated alternatively. The step for updating \( X \) with a given \( \Omega \) follows the same procedure as the sparse coding step in LOST.

In the \( \Omega \) update step, Transform K-SVD aims to find the optimal solution of the following
problem

\[
\{\Omega^*\} = \arg\min_{\Omega} \|\Omega Y - X\|^2_F
\]

\[\text{s.t. } \left\{ \max_{k \neq j}\left|\Omega_{k,.:} \Omega_{j,.:}^T\right| \right\} \leq 1 - \delta, \forall 1 \leq j, k \leq p \]

\[(2.47)\]

This problem is addressed by optimizing each row of \(\Omega\) sequentially. The update of the \(j\)-th atom \(\Omega_{j,:}\) is achieved by the same approach as in Analysis K-SVD \[92\], i.e. solving the problem \[(2.42)\] using the SVD of the signals \(Y_{i,:}\), where the matrix \(Y_{i,:}\) consists of the signals whose co-supports have the index \(j\) based on the \(X\) obtained in the first step. The correlation constraint is also addressed using random replacements as in Analysis K-SVD for avoiding similar atoms.

**(NL)AOL/(NA)AOL**

The analysis operator learning (AOL) algorithm \[107\] employs the \(\ell_1\) relaxation version \[(2.39)\] as its objective function, and restricts the dictionary as belonging to a Uniform Normalized Tight Frame (UNTF) for excluding undesired solutions. In fact, the UNTF is in the intersection of two manifolds, Uniform Normalized (UN) frames manifold and Tight Frames (TF) manifolds, and thus it can be represented as

\[
\mathcal{C} = \{\Omega \in \mathbb{R}^{p \times m} : \Omega^T \Omega = \mathbf{I}, \forall i, \Omega_{i,:} = c\},
\]

\[(2.48)\]

where \(c\) is a constant real number. It should be noted that the rank of the matrix \(\Omega \in \mathcal{C}\) can be computed via \(\text{rank}(\Omega) = \text{rank}(\Omega^T \Omega) = m\), and therefore the full column rank constraint is also guaranteed by the UNTF constraint.

Two versions of the AOL algorithm have been proposed for the noiseless and noisy training signals respectively, which are referred to as NoiseLess Analysis Operator Learning ((NL)AOL) and Noise Aware Analysis Operator Learning ((NA)AOL). Given a set of clean training signals \(Y\), the formulation of (NL)AOL can be written as

\[
\{\Omega^*\} = \arg\min_{\Omega \in \mathcal{C}} \|\Omega Y\|_1,
\]

\[(2.49)\]
which can be addressed with sub-gradient iterations followed by projections onto the UN and TF manifolds \[107\].

However, when only noisy training signals \( \mathbf{Z} = \mathbf{Y} + \mathbf{V} \) are available, (NA)AOL needs to be used instead of (NL)AOL. The (NA)AOL algorithm is able to learn an analysis dictionary with the noisy training signals \( \mathbf{Z} \) and also recovers the clean signals \( \mathbf{Y} \), which can be formulated as follows

\[
\{ \Omega^*, \mathbf{Y}^* \} = \arg \min_{\{\Omega \in \mathcal{C}, \mathbf{Y} \}} \| \Omega \mathbf{Y} \|_1 + \frac{\lambda}{2} \| \mathbf{Z} - \mathbf{Y} \|_F^2.
\] (2.50)

This problem can be solved by updating \( \Omega \) and \( \mathbf{Y} \) alternatively with the other fixed. For the update of \( \Omega \), it is equivalent to the optimization of (NL)AOL and thus the same projected sub-gradient algorithm can be employed. As to the update of \( \mathbf{Y} \), it leads to a convex optimization problem which can be solved by \( \ell_1 \) minimization algorithms.

**GOAL**

The GeOmetric Analysis operator Learning (GOAL) algorithm \[56\] employs \( \ell_p \)-norm as the co-sparsity measurement and minimizes both the squared empirical mean and the empirical variance of the co-sparsity, which are different from the algorithms reviewed above. This co-sparsity measurement leads to the promoting function as follows

\[
F_q(\mathbf{X}) = \frac{1}{2n} \sum_{i=1}^{n} \left( \frac{1}{q} \| \mathbf{X}_{:,i} \|_q^q \right)^2.
\] (2.51)

where \( \mathbf{X} = \Omega \mathbf{Y} \) is the analysis representations of the given training signals \( \mathbf{Y} \), and \( \| \cdot \|_q \) denotes the canonical \( \ell_p \)-norm defined as \( \| \mathbf{X}_{:,i} \|_q = (\sum_{j=1}^{p} | \mathbf{X}_{j,i} |^q)^{\frac{1}{q}} \) with \( 0 \leq q \leq 1 \) (\( q \) is used in the definition of the \( \ell_p \)-norm, instead of \( p \), to avoid confusion with the number of atoms in \( \Omega \) represented by \( p \)). The global minimizer of the function (2.51), however, is the trivial solution \( \Omega = \mathbf{0} \). To avoid this, the GOAL algorithm imposes three constraints: (1) The rows of \( \Omega \) have unit \( \ell_2 \)-norm; (2) \( \Omega \) has full column rank; and (3) \( \Omega \) does not have linear dependent rows. These constraints motivate to consider the transpose of the learned analysis dictionary \( \Omega \).
to be an element of the oblique manifold $\text{OB}(m, p)$, which is defined as

$$\text{OB}(m, p) = \{ W \in \mathbb{R}^{m \times p} : \text{rank}(W) = m, \, \text{diag}(W^T W) = I \}. \quad (2.52)$$

Considering $W = \Omega^T \in \text{OB}(m, p)$ and the property of this manifold (see Lemma 1 in [56]), the unit $\ell_2$-norm and full column rank constraints result in the penalty function

$$h(\Omega) = -\frac{1}{m \log(m)} \log \left( \frac{1}{p} \Omega^T \Omega \right). \quad (2.53)$$

The third constraint can be enforced with the logarithmic barrier function of the inner-products between the distinct rows of $\Omega$, that is

$$r(\Omega) = -\sum_{1 \leq i \leq j \leq p, \, i \neq j} \log(1 - \langle \Omega_{i,:}, \Omega_{j,:} \rangle^2). \quad (2.54)$$

Combining the co-sparsity promoting function (2.51) with the penalty functions (2.53), (2.54), the transposed analysis dictionary $W$ can be estimated by optimizing the following regularized problem

$$\{ W^* \} = \arg \min_{W \in \text{OB}(m, p)} F_q(W^T Y) + \kappa h(W^T) + \mu r(W^T), \quad (2.55)$$

and the analysis dictionary can be obtained via $\hat{\Omega} = W^T$. The conjugate gradient method on manifolds [62] is applied for the optimization of the cost function (2.55).

**Comments on the ADL algorithms**

In this thesis, we focus on the analysis model based dictionary learning and signal reconstruction problems. Therefore, we compare the aforementioned ADL algorithms for a deeper understanding. Generally, the ADL algorithms vary in terms of the co-sparsity measurements, the constraints applied to the dictionary $\Omega$ and the optimization methods, as summarized in Table 2.2.

As presented in Table 2.2, the Analysis K-SVD [92], LOST [88] and Transform K-SVD [39] algorithms all employ the $\ell_0$-norm as the measurement of co-sparsity, while (NL)AOL/(NA)AOL [107] and GOAL [56] use $\ell_1$-norm and a $\ell_p$-norm based function, respectively. Some constraints are applied to $\Omega$ for avoiding trivial solutions and encouraging desired properties. A
common constraint used in ADL algorithms is on the $\ell_2$-norms of the rows of $\Omega$, for example, the unit $\ell_2$-norm constraint is utilized explicitly in Analysis K-SVD, LOST and Transform K-SVD, and also embedded in a regularizer of the objective function of GOAL. Similarly, the UNTF constraint in (NL)AOL/(NA)AOL also guarantees that the $\ell_2$-norms of the rows in $\Omega$ are a specific constant. The constraint that $\Omega$ has full column rank is also commonly considered in some algorithms. In particular, this constraint is enforced by regularization terms both in LOST and GOAL, and it is also guaranteed implicitly via the UNTF constraint used in (NL)AOL/(NA)AOL. Different optimization methods were developed to address the formulations of the algorithms, as shown in the last column of Table 2.2. In this thesis, inspired by SimCO, a SDL algorithm as reviewed in Section 2.1.2, a novel ADL algorithm is proposed.

Another point not shown in Table 2.2 but needs to be noted is the approaches employed in the algorithms to avoid similar atoms. As mentioned before, in Analysis K-SVD and Transform K-SVD the similar rows in $\Omega$ are replaced by randomly generated vectors in a post-step after updating all atoms. However, these replacement atoms may degenerate the performance of the learned dictionary due to the random generation fashion. The LOST and GOAL algorithms address the similar atoms problem by applying regularization terms in their objective functions, but this cannot restrict the correlations between atoms directly. In our work, we will propose an alternative solution to this problem based on the idea of rotating the correlated atoms, as

### Table 2.2: Comparison of different ADL algorithms

<table>
<thead>
<tr>
<th>ADL algorithm</th>
<th>Formulation</th>
<th>Co-sparsity Measurement</th>
<th>Constraints applied to $\Omega$</th>
<th>Optimization Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis K-SVD</td>
<td>$\ell_0$-norm</td>
<td>All rows have unit $\ell_2$-norm.</td>
<td>(1) BG/OBG; (2) SVD.</td>
<td></td>
</tr>
<tr>
<td>LOST</td>
<td>$\ell_0$-norm</td>
<td>All rows have unit $\ell_2$-norm; (2) $\Omega$ has full column rank.</td>
<td>(1) Sparse coding; (2) Conjugate gradient.</td>
<td></td>
</tr>
<tr>
<td>Transform K-SVD</td>
<td>$\ell_1$-norm</td>
<td>All rows have unit $\ell_2$-norm.</td>
<td>(1) Sparse coding; (2) SVD.</td>
<td></td>
</tr>
<tr>
<td>(NL)AOL/(NA)AOL</td>
<td>$\ell_1$-norm</td>
<td>$\Omega$ belongs to UNTF.</td>
<td>Projected sub-gradient. $\ell_1$ minimization.</td>
<td></td>
</tr>
<tr>
<td>GOAL</td>
<td>$\ell_p$-norm based function</td>
<td>(1) All rows have unit $\ell_2$-norm; (2) $\Omega$ has full column rank; (3) $\Omega$ does not have linear dependent rows.</td>
<td>Conjugate gradient method on manifolds.</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2. Literature Review on Sparse Representation and Dictionary Learning

will be presented in Chapter 3.

2.3 Applications of Sparse Models

As mentioned in Chapter 1, the sparse models have been successfully applied to a wide range of signal processing problems, such as image denoising [42], [92], [71], [38], [58], blind source separation [16], [110], [106], [1], compressed sensing [74], [24], image compression [19], inpainting [71], [4], super-resolution [108], decomposition [45], recognition [103], classification [70], [109] and more. The research on the applications of the sparse models is too wide to be covered in this section, and thus we mainly review four of them, i.e. additive noise removal, multiplicative noise removal, blind source separation and compressed sensing. A more comprehensive review for the applications of the sparse models can be found in [43], [47], [84], and [102]. Among these applications, additive and multiplicative noise removal will be used in our work to demonstrate the performance of the algorithms proposed in Chapters 3 and 4, respectively.

Additive Noise Removal

Additive noise is the most popular noise model used in image denoising, where the noisy image is formulated as the sum of noise and the original clean image. The noise is usually assumed to satisfy the Gaussian distribution with a zero mean and a known deviation. Using the sparse models, some methods have been proposed to address the additive noise removal problem [42], [30]. The underlying assumption of these methods is that the clean image is sparse with respect to a dictionary while the noise is not. In other words, a sparsity prior based on the sparse synthesis or analysis model is used to reconstruct the image.

A typical sparse synthesis model based additive noise removal method is the K-SVD denoising algorithm proposed in [42]. This algorithm applies the K-SVD algorithm to learn a synthesis dictionary from a set of training samples, and then the OMP algorithm [100] is employed to find the sparse representations of the noisy image patches with respect to the learned dictionary. The denoised image is obtained by multiplying the learned dictionary with the sparse coefficients. It has been shown that this method outperforms the existing algorithms [42]. The
experimental results in [42] also indicate that the denoising results using an overcomplete Discrete Cosine Dictionary (DCT) are not as good as those obtained with the learned dictionary, which demonstrates the superiority of the learned dictionary compared with the pre-defined dictionary. Examples of the denoised images and the Peak Signal to Noise Ratios (PSNR) are presented in Fig. 2.3. Note that the K-SVD denoising algorithm is only tested on greyscale images in [42]. A direct application of this algorithm to color images is denoising each patch as a long concatenated RGB vector, however, the denoised images obtained via this simple method tend to show artifacts in terms of color saturation [71]. This is because the OMP algorithm used in the K-SVD denoising algorithm does not guarantee that the average color of the noisy patch is maintained in the reconstructed one [71]. By modifying the inner-product metric of the OMP step in the K-SVD denoising algorithm, an extended denoising method has been proposed in [71] to avoid the color artifacts.

![Figure 2.3: The original image, noisy image and denoised images using overcomplete DCT and K-SVD. (a) Original image. (b) Noisy image (24.63 dB). (c) Overcomplete DCT (31.77 dB). (d) K-SVD (32.27 dB).](image)

In the denoising methods based on the analysis model, the image is reconstructed using a leaned analysis dictionary. In particular, the analysis dictionary is usually learned via the ADL algorithms mentioned in Section 2.2.2. The reconstruction task can be accomplished by the analysis pursuit algorithms as reviewed in Section 2.2.1. For example, the denoising method based on Analysis K-SVD [92] utilizes the OBG algorithm to reconstruct the image and the method using the learning algorithm (NA)AOL [107] restores the image via the regularized version of the relaxation based analysis pursuit, i.e. equation (2.36). Some other reconstruction formulations have also been proposed for better adapting to a specific dictionary learning algorithm. For example, the denoising method in [56] is based on the dictionary learned via the GOAL algorithm [56], and its reconstruction formulation employs a sparsity promoting
regularizer which is consistent with the sparsity measurement used for dictionary learning in GOAL. The denoising method based on GOAL [56] performs slightly better than the K-SVD [92] denoising method in most cases. Actually, the additive noise removal task has become a baseline to compare the performance of different dictionary learning algorithms. For this reason, additive noise removal is also considered in our proposed ADL algorithm which will be presented in Chapter 3.

Multiplicative Noise Removal

Multiplicative noise, also known as speckle noise, is often observed in Synthetic Aperture Radar (SAR) and Sonar (SAS) images, due to the effect of interference introduced in their acquisition processes [79]. Compared to additive Gaussian noise often assumed in traditional image denoising, removing speckle noise is deemed to be more difficult for two reasons. First, the noise is multiplied with (rather than added to) the original image, which usually degrades the images more severely as compared with additive noise [38]. Second, the study of the statistical properties of speckle noise indicates that Gamma distribution is more suitable for modeling such noise [79], [38], [52], [11] instead of the widely used Gaussian distribution in conventional image denoising, and thus the data fidelity term derived from the noise model is not quadratic, raising difficulties for optimization. To give a visual comparison of the additive Gaussian noise and multiplicative Gamma noise, Fig. 2.4 shows an original image, the images and histograms of the noise, and the noisy images corrupted by these two types of noise.

The sparsity prior has also been employed to address this more challenging denoising task successfully. Most sparsity based methods [38], [58], [55] remove the multiplicative noise in the logarithm transform domain where the multiplicative noise model is converted to an additive one. Among these methods, Duran, Fadili and Nikolova (DFN) [38] adopted the sparsity prior by considering the sparsity of the image in the curvelet transformed domain and restoring the frame coefficients via a Total Variation (TV) regularized formulation in the log-domain. As dictionaries learned from the related data have the potential to fit the data better than pre-defined dictionaries, dictionary learning techniques have also been utilized to model the sparsity prior [58], [55]. The methods proposed in [55] and [58] both introduce dictionary learning to the multiplicative noise removal problem, but with different frameworks. These two methods are
referred to as MNR-DL-TV-1 (Multiplicative Noise Removal via Dictionary Learning and Total Variation) [55] and MNR-DL-TV-2 [58], respectively. In these two methods, the dictionary is learned by the K-SVD algorithm [5]. The MNR-DL-TV-1 method performs noise reduction in two stages, where the image is first denoised using the learned dictionary, and a model based on an $\ell_2$ data fidelity term and TV regularization is then applied to further improve the denoising result. In contrast, the MNR-DL-TV-2 method formulates the image reconstruction task as an optimization problem containing two regularizers: a learned dictionary based term and a TV term respectively. Recently, another dictionary learning and TV based algorithm has also been proposed in [66], however, this algorithm formulates the noise removal task in the original image domain rather than the log-domain. The Iterative method using Nonlocal Sparse Model (Iter-NSM) [105] also employs a dictionary learned with K-SVD [5], and the learned dictionary is used with respect to a nonlocal sparse model where similar image patches are assumed to be represented by the same atoms in the dictionary. Note that the dictionaries in these dictionary learning based algorithms [55], [58], [66] are all based on the sparse synthesis model. As mentioned before, the dictionaries learned based on the analysis model have been shown effective for removing additive noise [92], [107], [56], however, their applications to multiplicative noise removal have not been investigated. We attempt to fill this research gap and consider to apply analysis dictionary learning to multiplicative noise removal, as will be presented in Chapter 4.
Blind Source Separation

Blind Source Separation (BSS) is another field where the sparse models have been used successfully. BSS is a very popular technique to analyze multichannel data in practice where a set of observations is available. In this context, the observed data is usually modeled as the linear combination of sources to be retrieved, which can be formulated as follows

\[ \tilde{X} = \tilde{A}S \] \hfill (2.56)

where each row of \( \tilde{X} \in \mathbb{R}^{r \times N} \) and \( S \in \mathbb{R}^{s \times N} \) denote a mixture and a source signal respectively, and \( \tilde{A} \in \mathbb{R}^{r \times s} \) is the mixing matrix modeling the linear combinations of the sources in \( S \). The task of BSS is to recover the sources \( S \) from the mixtures \( \tilde{X} \) without knowing the mixing matrix \( \tilde{A} \).

Since the BSS problem does not have a unique solution in general, some assumptions need to be imposed to distinguish the sources to separate them. A widely used assumption is the statistical independence of the sources, which leads to a family of Independent Component Analysis (ICA) methods [60]. However, these methods do not work for the under-determined case where the number of sources is greater than that of the mixtures. In the past decade, the assumption that the sources are sparse in a specific domain paves a new path to the BSS problem, especially for the under-determined case. The approach proposed in [16] is restricted to two observations. The sparsity prior of the sources in the orthogonal Fourier transform domain is employed to recover the sources with a pre-estimated mixing matrix. Specifically, the recovery of the sources is formulated as a constrained \( \ell_1 \)-norm minimization problem which is close to the formulation of the BP algorithm [27]. The method in [110] addresses a more general under-determined case. The sources are assumed to be sparsely represented over a selected overcomplete dictionary, and the mixing matrix and the source signals are estimated jointly using a probabilistic framework. More recently, the dictionary learning techniques have been introduced to BSS for a better representation of the data. The approach presented in [106] learns a dictionary based on the synthesis model and then employs this dictionary to address the BSS problem by solving a sparse coding problem. The experimental results with speech signal show that the separation performance using the learned dictionary is more robust in noisy cases as compared with fixed dictionaries. The idea of learning dictionaries while
separating the source signals was introduced in [1], where the K-SVD denoising method in [42] was extended to the BSS problem. In this method, a local dictionary is adaptively learned for each source along with separation.

Compressed Sensing

Apart from the applications in fundamental signal processing problems as mentioned above, sparse models also play an important role in the data acquisition process. In particular, they are the essential premise for the recently developed Compressed Sensing (CS) theory, which is a novel paradigm to sample and recover signals [36]. Conventional approaches to sampling signals follow Shannon-Nyquist sampling theorem, however, the CS theory asserts that one can recover them from far fewer samples or measurements if the unknown signals are approximately sparse [24].

For a given sensing matrix \( M \in \mathbb{R}^{q \times m} \) with \( q \ll m \), the sampling of a sparse signal \( y \in \mathbb{R}^m \) via \( M \) can be expressed as [36]

\[
z = My \quad \text{with} \quad \|y\|_0 = s,
\]

where \( z \in \mathbb{R}^q \) denotes the compressed measurements of the unknown signal \( y \). The measurement process is not adaptive, which means that the sensing matrix \( M \) is fixed and does not depend on the signal \( y \). Even though the number \( q \) of the available measurements is much smaller than the dimension \( m \) of the signal \( y \), it has been shown that one can almost always recover the signal \( y \) exactly by solving the following convex optimization problem [23]

\[
\{y^*\} = \arg \min_y \|y\|_1 \quad \text{s.t.} \quad z = My,
\]

where the measurement matrix \( M \) is known. This is in the same form of the optimization in BP algorithm [27], i.e. equation (2.5). Actually, the signal recovery problem in CS can be regarded as the sparse coding problem based on the synthesis model, by considering the sensing matrix \( M \) and the unknown signal \( y \) as an overcomplete dictionary and the sparse coefficients respectively. As a result, other sparse coding algorithms as reviewed in Section 2.1.1 can also be used as solvers for recovering \( y \) from its measurements \( z \).
Nevertheless, the unknown signal $y$ may not be sparse in its original domain, and it is necessary to transform the signal to another domain to enforce the sparsity assumption in CS. This is where the dictionaries in the sparse models are useful. Assuming that the unknown signal $y$ is sparse with respect to a synthesis dictionary $D \in \mathbb{R}^{m \times d}$, i.e. $y = Da$ with $\|a\|_0 = s$, and then the CS model (2.57) can be rewritten as [9]

$$z = My = MDa \quad \text{with} \quad \|a\|_0 = s.$$  \hfill (2.59)

It should be noted that the sparse coefficients $a$ can be approximated by solving the synthesis model based sparse coding problem with the given dictionary being $\tilde{D} = MD$, and thus the recovered signal $y$ is obtained by $\hat{y} = Da$.

Recently compressed sensing based on the analysis model has also been considered [74] by assuming that the signal of interest satisfies the sparse analysis model. It is referred to as analysis-based compressed sensing [74], which can be described as follows

$$z = My \quad \text{with} \quad \begin{cases} x = \Omega y \\ \|x\|_0 = p - l. \end{cases}$$ \hfill (2.60)

Given $M$ and $\Omega$, the signal $y$ can be recovered via analysis pursuit algorithms as reviewed in Section [2.2.1].

### 2.4 Summary

In this chapter, the background of the sparse synthesis model and analysis model are presented respectively. For each model, the description and the mathematical formulation are introduced first, and then the existing algorithms with respect to the model are reviewed. In particular, for the synthesis model, the sparse coding algorithms for representing a signal sparsely with a given dictionary and the learning algorithms for obtaining data adaptive dictionaries are reviewed. Similarly, the analysis pursuit algorithms and dictionary learning algorithms based on the analysis model are introduced. Finally, the applications of the sparse models are provided. In the next chapter, we study the analysis model based dictionary learning and present two new algorithms to this problem.
Chapter 3

Analysis SimCO Algorithms for Analysis Dictionary Learning

In this chapter, we consider the dictionary learning problem for the sparse analysis model. A novel algorithm is proposed by adapting the SimCO algorithm, based on the sparse synthesis model, to the sparse analysis model. This algorithm assumes that the analysis dictionary contains unit $\ell_2$-norm atoms and learns the dictionary by optimization on manifolds. This framework allows multiple dictionary atoms to be updated simultaneously in each iteration. However, similar to several existing analysis dictionary learning algorithms, dictionaries learned by the proposed algorithm may contain similar atoms, leading to a degenerate (coherent) dictionary. To address this problem, we also consider restricting the coherence of the learned dictionary and propose Incoherent Analysis SimCO by introducing an atom decorrelation step following the update of the dictionary. We demonstrate the competitive performance of the proposed algorithms using experiments with synthetic data and image denoising as compared with existing algorithms.

3.1 Introduction

As mentioned in Chapter 2, how to learn an appropriate analysis dictionary from a set of training samples is an important challenge for the analysis model based representation. Several algorithms have been proposed to address this problem, such as the algorithms reviewed in
Chapter 2: Analysis K-SVD [92], LOST [88], Transform K-SVD [39], (NL)AOL/(NA)AOL [107] and GOAL [56]. However, there are some limitations in the existing methods. For example, the computational complexity of Analysis K-SVD is quite high due to the involvement of the analysis pursuit problem [92]. The coefficients of the penalty terms play an important role in the performance of LOST, but selecting proper coefficients is a practical challenge [88]. The (NL)AOL/(NA)AOL algorithms cannot recover random dictionaries well since the UNTF constraint in their formulation limits the possible dictionaries to be learned.

In this chapter, we propose two new algorithms which can partly address the limitations of the ADL algorithms mentioned above. Firstly, we adapt the synthesis model based SimCO algorithm [30] to the analysis model and develop a new ADL algorithm which is referred to as the Analysis SimCO algorithm. In SimCO, the optimization method on manifolds is applied to update multiple dictionary atoms simultaneously, leading to a better performance compared with K-SVD [5] where the atoms are updated one-by-one. Thus, we adapt the framework of SimCO to the ADL problem to enable the simultaneous update of multiple atoms via the optimization on manifolds. This dictionary update method is different from the methods used by the existing algorithms. Analysis K-SVD and Transform K-SVD only allow one atom to be updated in each iteration. In LOST, the dictionary is updated as a whole matrix by the standard conjugate gradient method. Compared with the AOL algorithms, the updated dictionary in our proposed method is more general without projection onto the UNTF set. Notice that the GOAL algorithm also employs an optimization method on manifolds, however, the objective function of our proposed algorithm is different from that of GOAL due to the different co-sparsity measure based on $\ell_0$-norm, and the fewer penalty terms used. Besides, our proposed algorithm employs the gradient descent method on manifolds rather than the conjugate gradient method as in GOAL.

Secondly, we propose the Incoherent Analysis SimCO algorithm to avoid similar atoms appearing in the dictionaries learned by Analysis SimCO. In the Incoherent Analysis SimCO algorithm, a constraint restricting the correlations of two distinct atoms of the dictionary is considered and an atom decorrelation step is applied to enforce this constraint by rotating the highly-correlated atom pairs. In this way, the correlation of any two distinct atoms can be restricted to be below a given threshold explicitly. Compared with the methods used in existing ADL algorithms to avoid similar atoms, the decorrelation step applied in the Incoherent
3.2 Problem Formulation and Optimization Framework

Analysis SimCO algorithm has some advantages. For example, this method avoids the coefficient selection problem of LOST since the constraint is tackled directly rather than applied as a penalty term of the objective function. Besides, the new atoms obtained by the decorrelation step are more likely to be closer to the atoms replaced than the atoms that are generated randomly in Analysis K-SVD and Transform K-SVD.

3.2 Problem Formulation and Optimization Framework

Given a set of training signals $Y \in \mathbb{R}^{m \times n}$, the ADL problem can be written as \[89\]

$$
\{\Omega^*, X^*\} = \arg \min_{\{\Omega, X\}} \|X - \Omega Y\|_F^2
$$

s.t.  \quad \|X_{:,i}\|_0 = p - l, \; \forall i. 

(3.1)

This is a general formulation without any additional constraint on $\Omega$ apart from the co-sparsity constraints $\|X_{:,i}\|_0 = p - l, \; \forall i$. However, this formulation has ambiguities caused by scaling. In one case, when the training data $Y$ admits exact sparse representations, there exists a dictionary $\Omega$ with which the analysis representations of $Y$, i.e. $X = \Omega Y$, satisfy the co-sparsity constraints. If the dictionary $\Omega$ is scaled by multiplying a scalar $c \in \mathbb{R}$, the corresponding representations $c \cdot X = c \cdot \Omega Y$ will also satisfy the constraints. Thus, the problem (3.1) has infinite optimal solutions $c \cdot \Omega$ and $c \cdot X$. This may introduce difficulty in optimization. In the other case, if the data $Y$ admits approximation representations and $\|X - \Omega Y\|_F^2 = \delta$, the value of the cost function with scaled $X$ and $\Omega$, i.e. $\|X - \Omega Y\|_F^2 = c^2 \cdot \delta$, can be arbitrarily small. In other words, the cost function is unbounded from below, which makes it impossible to find an optimal solution. In addition, (3.1) has trivial solutions where $\Omega$ contains all-zero rows.

In order to avoid these problems, we apply the unit $\ell_2$-norm constraints on the rows of $\Omega$, leading to the following formulation of the ADL problem

$$
\{\Omega^*, X^*\} = \arg \min_{\{\Omega, X\}} \|X - \Omega Y\|_F^2
$$

s.t.  \quad \|X_{:,i}\|_0 = p - l, \; \forall i

$$
\|\Omega_{j, :}\|_2 = 1, \; \forall j.
$$

(3.2)
Chapter 3. Analysis SimCO Algorithms for Analysis Dictionary Learning

The unit $\ell_2$-norm constraints on the rows of $\Omega$ are able to eliminate the scaling ambiguity mentioned above. Besides, the trivial solutions where $\Omega$ has zero rows can be excluded. The formulation (3.2) is different from that of Analysis K-SVD [92] which minimizes the error in the signal domain. It also differs from the objective function of LOST [88] where the penalty terms as described earlier in Chapter 2 are included.

The problem (3.2) can be addressed by an optimization framework alternating between two stages: analysis sparse coding and dictionary update. Given a dictionary $\Omega$, the first stage finds $X$ satisfying the co-sparsity constraints $\|X_{:,i}\|_0 = p - l$, $\forall i$. In the dictionary update stage, $\Omega$ is updated assuming known and fixed $X$ obtained in the first stage.

Here we attempt to update the dictionary using a similar method as in SimCO [30] and refer to our proposed algorithm as Analysis SimCO. The flow diagram of Analysis SimCO is presented in Fig. 3.1 and the optimization framework is summarized in Algorithm 2. In SimCO, the use of the term “simultaneous” comes from the following two facts: (1) multiple dictionary atoms are updated simultaneously, and (2) their corresponding coefficients are also updated simultaneously with these atoms. In the analysis case, we borrow the term “SimCO” mainly because in the proposed algorithm the dictionary atoms are updated simultaneously.

![Figure 3.1: The flow diagram of Analysis SimCO.](image)

A common problem with the popular analysis dictionary learning algorithms, such as Analysis K-SVD [92], is that the learned dictionary $\Omega$ may contain similar atoms. Such a dictionary
3.3. The Analysis SimCO Algorithm

**Algorithm 9** Optimization Framework of Analysis SimCO

**Input:** \( Y, p, l \)

**Output:** \( \Omega^* \)

**Initialization:**

Initialize the iteration counter \( k = 1 \) and the analysis dictionary \( \Omega^{(k)} \). Perform the following steps.

**Main Iterations:**

1. Analysis sparse coding: Compute the representations \( X^{(k)} \) with the fixed dictionary \( \Omega^{(k)} \) and the training signals in \( Y \).
2. Dictionary update: Update the dictionary \( \Omega^{(k+1)} \leftarrow \Omega^{(k)} \).
3. If the stopping criterion is satisfied, \( \Omega^* = \Omega^{(k+1)} \) and quit the iteration. Otherwise, increase the iteration counter \( k = k + 1 \) and go back to step 1.

is regarded as a degenerate solution [89], [39]. This issue is also observed in the dictionary learned from (3.2) with the Analysis SimCO algorithm, as will be shown in Section 3.5. Thus, we develop an extended version of Analysis SimCO to avoid this kind of degenerate dictionary, which will be presented in Section 3.4 in detail.

### 3.3 The Analysis SimCO Algorithm

As the dictionary update stage in our algorithm is based on optimization on matrix manifolds, we begin this section with a brief introduction to the optimization on matrix manifolds to make this section self-contained. The details of the analysis sparse coding and dictionary update are then presented respectively, followed by the convergence and computational complexity analysis of our proposed algorithm.

#### 3.3.1 Optimization on Matrix Manifolds

The Stiefel manifold \( \text{St}(p, m)(p \leq m) \) is defined as \( \text{St}(p, m) := \{ U \in \mathbb{R}^{m \times p} : U^T U = I \} \) [3, pp. 26]. For \( p = 1 \), the Stiefel manifold \( \text{St}(p, m) \) reduces to the unit sphere, i.e. \( S = \{ u \in \mathbb{R}^m : u^T u = 1 \} \). At each point \( u \in S \), there exists a tangent space \( T_uS \) which consists of all vectors orthogonal to \( u \) in \( \mathbb{R}^m \), i.e. \( T_uS = \{ v \in \mathbb{R}^m : u^T v = 0 \} \). The vectors in \( T_uS \) are tangent vectors to \( S \) at the point \( u \). The tangent space \( T_uS \) can be regarded as a vector space approximation of the manifold \( S \) at the point \( u \) [3, pp. 34].
Before dealing with the optimization problem on manifolds, we consider a more general class of problems, i.e. the unconstrained optimization problem, from which the optimization methods on matrix manifolds can be adapted,

$$\min_{u} f(u),$$

where $u \in \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}$ is a differentiable function. This problem can be addressed by the standard line search method. In the $k$-th iteration, the standard line search method selects a descent direction $p$ along which the current point $u_k$ is moved to a new point $u_{k+1}$ leading to a smaller or equal objective function value, i.e.

$$u_{k+1} = u_k + \alpha \cdot p$$

with $f(u_{k+1}) \leq f(u_k)$. Here $\alpha$ is the scalar step size which can be selected carefully to guarantee the reduction of the cost function \cite{78}. In order to determine the search direction $p$, the value and the derivatives of the objective function can be used. The most obvious choice is the steepest descent direction $p_k = -\nabla f(u_k)$ along which the objective function value decreases most rapidly among all the directions \cite[pp. 20]{78}.

Now we consider the optimization problem where the variable $u$ is restricted on the manifold $S$, i.e.

$$\min_{u \in S} f(u).$$

Analogous line search methods on manifolds have been developed by generalizing the standard line search methods for the unconstrained optimization problem (3.3). Specifically, in the $k$-th iteration, the search direction $q_k$ should be chosen as a tangent vector to $S$ at $u_k$, i.e. $q_k \in T_{u_k}S$. Thus the search direction $q_k$ is the projection of the search direction $p_k$ of the unconstrained optimization methods to the tangent space $T_{u_k}S$ \cite[pp. 49]{3}, that is

$$q_k = (I - u_k u_k^T)p_k.$$  

The new point $u_{k+1}$ obtained by moving $u_k$ in the direction of $q_k$ should stay on $S$. As a
result, the line search path (3.4) is replaced by a curve on \( S \) [3, pp. 103], i.e.

\[
\mathbf{u}_{k+1} = \mathbf{u}_k \cos(\alpha \| \mathbf{q}_k \|_2) + \frac{\mathbf{q}_k}{\| \mathbf{q}_k \|_2} \sin(\alpha \| \mathbf{q}_k \|_2).
\]  
(3.7)

### 3.3.2 Analysis Sparse Coding Stage

The purpose of the analysis sparse coding stage is to get the sparse representations \( \mathbf{X} \) of the training signals in \( \mathbf{Y} \) based on a given dictionary \( \mathbf{\Omega} \). Unlike the corresponding problem of the synthesis model, here the exact representations \( \mathbf{X} \) can be calculated directly by simply multiplying the signals in \( \mathbf{Y} \) by the dictionary \( \mathbf{\Omega} \), that is

\[
\mathbf{X} = \mathbf{\Omega} \mathbf{Y}.
\]  
(3.8)

Since the initial dictionary is an arbitrary one, the representations obtained in this way may not satisfy the co-sparsity constraints on \( \mathbf{X} \) in (3.2). A hard thresholding operation is therefore applied to enforce the co-sparsity

\[
\hat{\mathbf{X}} = HT_l(\mathbf{X}),
\]  
(3.9)

where \( HT_l(\mathbf{X}) \) is the non-linear operator that sets the smallest \( l \) elements (in magnitude) of each column of \( \mathbf{X} \) to zero. The representations \( \hat{\mathbf{X}} \) obtained via equation (3.9) are the best approximation of the exact representations \( \mathbf{X} \) in terms of the error in Frobenius norm among all the matrices satisfying the co-sparsity constraints.

### 3.3.3 Dictionary Update Stage

The dictionary update stage aims at optimizing the following problem (by fixing \( \mathbf{X} \) in (3.2))

\[
\arg \min_{\mathbf{\Omega}} f(\mathbf{\Omega}) = \| \mathbf{X} - \mathbf{\Omega} \mathbf{Y} \|_F^2 \quad \text{s.t.} \quad \| \mathbf{\Omega}_{j,.} \|_2 = 1, \forall j.
\]  
(3.10)

The cost function can be rewritten as a function of the rows of \( \mathbf{\Omega} \). Besides, the constraint that \( \mathbf{\Omega} \) only contains unit \( \ell_2 \)-norm rows restricts the transposes of the rows of \( \mathbf{\Omega} \) to lie on the unit
sphere $\mathcal{S}$, i.e. $\Omega_{j,:}^T \in \mathcal{S}, \forall j$. Thus the problem (3.10) can be rewritten as

$$\arg \min_{\Omega} f(\Omega) = \sum_{j=1}^{p} \|X_{j,:} - \Omega_{j,:} Y\|_2^2 \quad \text{s.t.} \quad \Omega_{j,:}^T \in \mathcal{S}, \forall j.$$  

(3.11)

As a result, the “line” search methods on manifolds can be utilized in this stage. Here we use the first order optimization procedures as in SimCO [30], i.e. the gradient descent line search method. We explain below the key points of this method including search direction, line search path, and step size respectively. The dictionary update stage is summarized in Algorithm 10.

**Algorithm 10 Dictionary Update Stage**

**Input:** $\Omega^{(k)}$, $X^{(k)}$, $Y$  
**Output:** $\Omega^{(k+1)}$  
**Main Steps:**

1. Calculate the search direction, based on equations (3.12) and (3.13).
2. Find a proper step size $\alpha$ using golden section search.
3. Update the dictionary $\Omega^{(k+1)} \leftarrow \Omega^{(k)}$, based on equation (3.14).

**Search direction**

We use the steepest descent direction as the search direction, i.e. the negative gradient of the objective function with respect to $\Omega$ as follows

$$H = -\nabla f(\Omega)$$

$$= -\frac{\partial \|X - \Omega Y\|_F^2}{\partial \Omega}$$

$$= 2XY^T - 2\Omega YY^T.$$  

(3.12)

**Line search path**

The search direction of the $j$-th row of $\Omega$, i.e. the projection of each row of $H$ onto the tangent space of $\mathcal{S}$, is [3] pp. 49]

$$h_j = H_{j,:}(I - \Omega_{j,:}^T \Omega_{j,:}).$$  

(3.13)
According to equation (3.7), the line search path for the $j$-th row of $\Omega$ can be written as

$$
\Omega_{j,:}(\alpha) = \begin{cases} 
\Omega_{j,:} & \text{if } \|\bar{h}_j\|_2 = 0, \\
\Omega_{j,:} \cos(\alpha \|\bar{h}_j\|_2) + (\bar{h}_j / \|\bar{h}_j\|_2) \sin(\alpha \|\bar{h}_j\|_2) & \text{otherwise,}
\end{cases}
$$

(3.14)

where $\alpha$ is the step size.

### Step size

To find a proper step size $\alpha$, we apply the golden section search method [30]. This method consists of two stages. In the first stage, it finds a range which contains a local minimum and within which the objective function is unimodal. In the second stage, the golden section ratio is used to successively narrow the range until the minimizer is located and thus $\alpha$ is determined.

### 3.3.4 Convergence

Our proposed algorithm alternates between the analysis sparse coding stage and the dictionary update stage. For a fixed dictionary $\hat{\Omega}$, $\hat{X}$ obtained in the analysis sparse coding stage is the optimal solution under the constraint of co-sparsity. Thus, the cost function can only decrease in this stage. In the dictionary update stage, since the update of $\hat{\Omega}$ is along a descent direction and the step size is chosen to guarantee that the updated $\hat{\Omega}$ will not increase the cost function. Thus, the cost function is decreasing monotonically in our proposed algorithm. In addition, the cost function of our formulation (3.2) is lower bounded by zero, i.e. $\|X - \Omega Y\|_F^2 \geq 0$. According to the monotone convergence theorem [97], given the cost function decreases monotonically and is lower bounded, the algorithm must converge. The convergence will also be demonstrated experimentally in Section 3.5.1.

### 3.3.5 Computational Complexity

The time complexity of the Analysis SimCO algorithm can be analyzed as follows. The time complexity of the sparse coding stage is dominated by the calculation of $\Omega Y$, at $O(pmn)$, in terms of the analysis in [88]. In the dictionary update stage, the calculation of $H$ is the dominant part. Computing the product $XY^T$ requires $O(pmn)$ operations. The time complexity of
\( \Omega Y Y^T \) is \( O(pm^2) \) with pre-computed \( YY^T \). As a result, the dictionary update stage requires \( O(pmnm) \) operations with the usual case \( n > m \). The total time complexity of each iteration of the Analysis SimCO algorithm thus scales as \( O(pmnm) \).

The computational complexity of Analysis SimCO, similar to those of LOST \([88]\), (NL)AOL \([107]\), and Transform K-SVD \([39]\), shows a reduction compared with those of Analysis K-SVD and (NA)AOL. The complexity of Analysis K-SVD is \( O(pm^2n) \) using BG or \( O(pm^3n) \) using OBG, and (NA)AOL requires \( O(pmnk) \) operations with \( k \) being the number of dictionary update per iteration. The running time of these algorithms in practice will be given in Section 3.5.

### 3.4 Incoherent Analysis SimCO

As mentioned in Section 3.2, dictionaries learned by the existing ADL algorithms may contain similar atoms, which can degrade the representation performance for signal recovery. To address this problem, several methods have been proposed. For example, in Analysis K-SVD and Transform K-SVD, the similar atoms are replaced by randomly generated atoms, as mentioned in Chapter 2. In LOST \([88]\) and GOAL \([56]\), a penalty term is used in the objective function to restrict the correlations between atoms. As will be observed in the experiments of Section 3.5, Analysis SimCO has the same issue, where some of the atoms in the learned dictionary may appear similar. Here, we present an alternative solution to this problem based on \([69]\).

The method in \([69]\) was developed to mitigate the correlations between atoms learned by a synthesis model. Here we adapt this method to our model and optimization problem.

In the context of the sparse synthesis model, the coherence of the dictionary has been defined as a measure of the similarities between the atoms \([99]\). We extend this definition for an analysis dictionary \( \Omega \) and define the coherence \( \mu(\Omega) \) in a row-wise way as

\[
\mu(\Omega) = \max_{\forall i,j,i \neq j} \left| \frac{\Omega_{i,:} \cdot \Omega_{j,:}}{\|\Omega_{i,:}\|_2 \cdot \|\Omega_{j,:}\|_2} \right|. \tag{3.15}
\]

With the unit \( \ell_2 \)-norm constraints on the rows of \( \Omega \), the coherence \( \mu(\Omega) \) can be simplified as

\[
\mu(\Omega) = \max_{\forall i,j,i \neq j} \left| \langle \Omega_{i,:}, \Omega_{j,:) \rangle \right|. \tag{3.16}
\]
The coherence $\mu(\Omega)$ reflects the maximum correlation of two distinct atoms in $\Omega$. If $\mu(\Omega)$ is close to 1, it means that there are very similar rows in $\Omega$, which is the case we attempt to avoid. Thus, we add a coherence constraint $\mu(\Omega) \leq \mu_0$ to the formulation (3.2), i.e.

$$\{\Omega^*, X^*\} = \arg \min_{\Omega, X} \|X - \Omega Y\|_F^2$$

s.t. $\|X_{:,i}\|_0 = p - l, \forall i$

$$\|\Omega_{:,i}\|_2 = 1, \forall j$$

$$\mu(\Omega) \leq \mu_0,$$

where $\mu_0$ is the coherence limit for the learned dictionary $\Omega$.

To enforce the coherence constraint, we add an extra step in the dictionary update stage, aiming to find the closest dictionary $\hat{\Omega}$ to $\Omega$ in Frobenius norm, with the coherence of the dictionary $\hat{\Omega}$ bounded by a threshold $\mu_0$, that is

$$\arg \min_{\hat{\Omega}} \|\hat{\Omega} - \Omega\|_F^2$$

s.t. $\|\hat{\Omega}_{:,i}\|_2 = 1, \forall i$

$$\mu(\hat{\Omega}) \leq \mu_0.$$

Here the unit $\ell_2$-norm constraints for the atoms in the dictionary are also applied to ensure that the transposes of the atoms in the output dictionary are still on the manifold. This problem is addressed by applying the decorrelation method [69] in a row-wise fashion, as presented in Algorithm [11]. The general idea is to determine the atom pairs whose correlations are greater than $\mu_0$, via a labeling process (from line 5 to line 9 of Algorithm [11]), and decorrelate these atom pairs, via a decorrelation process (from line 10 to line 20). This method keeps alternating between the two processes until the coherence of the estimated dictionary $\hat{\Omega}$ reaches the threshold $\mu_0$. Although this is a heuristic algorithm, it typically involves only a few loops to output an incoherent dictionary. The convergence and the effectiveness of this algorithm will be numerically demonstrated in Section 3.5.

In the labeling process, the atoms of $\Omega$ are labeled as either the atom pairs to be decorrelated or atoms that do not need to be modified. An index-pair set $F$ is used to store the index pairs of the atom pairs labeled to be decorrelated and an index set $E$ is employed to save the indices.
Algorithm 11 Atom Decorrelation Step

1: Input: Ω, μ₀
2: Output: ̂Ω
3: Initialization:
   ̂Ω = Ω, θ = 1/2 arccos μ₀, c₁ = cos θ, c₂ = sin θ
4: while μ( ̂Ω) > μ₀ do
5:   E = {1, 2, ..., p}  // line 5-9: labeling process
6:   F = ∅
7:   while μ( ̂Ω_E) > μ₀ do
8:      (i, j) = arg max_{i,j∈E,i≠j} | ̂Ω_i, ̂Ω_j^T |
9:      F ← F ∪ {(i, j)}
10:     E ← E \ {i, j}  // line 10-20: decorrelation process
11:   end while
12:   for ∀ (i, j) ∈ F do
13:      b₁ = ( ̂Ω_i, + ̂Ω_j,)/∥ ̂Ω_i, + ̂Ω_j, ∥
14:      b₂ = ( ̂Ω_i, − ̂Ω_j,)/∥ ̂Ω_i, − ̂Ω_j, ∥
15:      if < ̂Ω_i, ̂Ω_j, > > 0 then
16:         ̂Ω_i,: = c₁b₁ + c₂b₂
17:         ̂Ω_j,:) = c₁b₁ − c₂b₂
18:      else
19:         ̂Ω_i,:) = c₁b₂ + c₂b₁
20:         ̂Ω_j,:) = −c₁b₂ + c₂b₁
21:      end if
22:   end for
23:   return ̂Ω

of the remaining atoms. ̂Ω_E represents the submatrix of ̂Ω only containing the rows indexed by the set E. In each iteration, the correlations of any two distinct rows belonging to ̂Ω_E are calculated to determine the most correlated pair. The indices of these two atoms will be saved, as an index-pair, into the set F, i.e. F ← F ∪ {(i, j)}, indicating that these two atoms are labeled as an atom pair to be decorrelated in the following decorrelation process. Their indices will be removed from E to avoid being detected again, i.e. E ← E \ {i, j}.

In the decorrelation process, the atom pairs indexed by the members of F are decorrelated successively. The decorrelation of each atom pair is achieved by rotating the two atoms symmetrically with respect to their mean so that their correlation reaches μ₀ [69]. The rotated atoms are determined based on the orthonormal basis {b₁, b₂} developed using the atoms to be decorrelated (lines 11 and 12) and the angle θ determined by the coherence limit μ₀, i.e. θ = 1/2 arccos μ₀ [69]. Specifically, each atom pair {Ω_i,:), Ω_j,:) is updated as lines 14, 15 or
3.4. Incoherent Analysis SimCO

lines 17, 18, leading to that \( \hat{\Omega}_{i, i}^{T} = \hat{\Omega}_{j, j}^{T} = c_1^2 + c_2^2 = 1 \) with \( c_1 = \cos \theta \) and \( c_2 = \sin \theta \) as given in line 3. Therefore, the unit \( \ell_2 \)-norm constraints in (3.18) are satisfied and the transposes of the atoms in the incoherent dictionary obtained by Algorithm 11 are still on the manifold.

In order to address the problem (3.17), the atom decorrelation step is inserted after the dictionary update stage in the loop of Analysis SimCO (Algorithm 9), as summarized in Algorithm 12. We referred to this extended version of Analysis SimCO as Incoherent Analysis SimCO. Actually, Analysis SimCO can be regarded as the special case of Incoherent Analysis SimCO if \( \mu_0 = 1 \).

**Algorithm 12 Incoherent Analysis SimCO**

**Input:** \( Y, p, l, \mu_0 \)

**Output:** \( \Omega^{\star} \)

**Initialization:**

- Initialize the iteration counter \( k = 1 \) and the analysis dictionary \( \Omega^{(k)} \).
- Perform the following steps.

**Main Iterations:**

1. Analysis sparse coding: Compute the representations \( X^{(k)} \) with the fixed dictionary \( \Omega^{(k)} \) and the training signals in \( Y \), based on equations (3.8) and (3.9).
2. Dictionary update: Update the dictionary \( \Omega^{(k+1)} \leftarrow \Omega^{(k)} \), using Algorithm 10.
3. Atom decorrelation: Decorrelate the atoms \( \hat{\Omega}^{(k+1)} \leftarrow \hat{\Omega}^{(k+1)} \), using Algorithm 11.
4. If the stopping criterion is satisfied, \( \Omega^{\star} = \hat{\Omega}^{(k+1)} \), quit the iteration. Otherwise, increase the iteration counter \( k = k + 1 \) and go back to step 1.

It is worth noting that other alternative methods could also be used to promote incoherent dictionaries. As mentioned earlier, Analysis K-SVD replaces the similar atoms with vectors generated in a random way. Compared with this method, the decorrelation step applied in Incoherent Analysis SimCO can better preserve the information in the dictionary atoms since the new atoms are generated by rotating the existing atoms to be replaced. The method in Incoherent K-SVD (IK-SVD) [2], proposed for the synthesis model, can also be used to decorrelate the atoms of the analysis dictionary, which can be achieved by minimizing \( \| \Omega \Omega^T - I \|_F^2 \) after the update of the dictionary. However, this method cannot directly control the degree of the coherence of the dictionary. For comparison, we modify the Incoherent Analysis SimCO algorithm by replacing the decorrelation step (i.e. step 3 in Algorithm 12) with these two decorrelation methods, which we refer to as Analysis SimCO-Random (ASimCO-Random) and Analysis
SimCO-IKSVD (ASimCO-IKSVD) respectively.

3.5 Simulation Results

In this section we present two categories of experiments to demonstrate the performance of our proposed algorithms. The first category contains experiments with synthetic data, and the second one provides image denoising results using the dictionaries learned with different ADL algorithms.

3.5.1 Experiments with Synthetic Data

Now we test the ADL algorithms with synthetic data. First of all, the approach to generating the synthetic data sets and the performance metrics employed are introduced. Second, we test our proposed algorithms with different initial dictionaries, showing their convergence and robustness to initializations. Third, the effect of the atom decorrelation step of the Incoherent Analysis SimCO algorithm is demonstrated. Fourth, experiments with different parameters are conducted to provide a more comprehensive comparison between our proposed algorithms and other ADL algorithms.

Synthetic data generation and performance metrics

A set of synthetic data consists of a reference analysis dictionary $\Omega \in \mathbb{R}^{p \times m}$ and a set of signals in $Y \in \mathbb{R}^{m \times n}$ that is sparse with respect to $\Omega$ with co-sparsity $l$. The reference dictionary $\Omega$ is generated as detailed in the settings of the experiments. The generation of the signals in $Y$ is based on the fact that the sparse analysis model can be used as a generative model with a given dictionary [74]. For generating each signal, $l$ rows of $\Omega$ are selected randomly and a basis for the null space of these $l$ rows is determined. Multiplying this basis by a random vector gives a vector which can be one member of the signal set, i.e. one column of $Y$. In the following experiments, $Y$ and its noisy version will both be used as the training samples. The noisy training signals are obtained by adding Gaussian noise with zero mean and standard deviation 0.04 to $Y$, set as in [92]. Learning with the original signals $Y$ is referred to as the noiseless case and learning with the noisy signals as the noisy case.
An advantage of using synthetic data is that the reference dictionary which can sparsify the signals exactly is available, and therefore the quality of a learned dictionary can be evaluated by comparing it with the reference dictionary. We use the recovery rate of the atoms to measure the performance of the algorithms for recovering the reference dictionary, following the experiments in [92]. An atom $\Omega_{j,:}$ of the reference dictionary is regarded as recovered if

$$\min_i (1 - |\hat{\Omega}_{i,:}^T \Omega_{j,:}|) < \tau,$$

(3.19)

where $\hat{\Omega}_{i,:}$ are the atoms of the learned dictionary and $\tau$ is the threshold value to determine whether the atoms are recovered. The value of $\tau$ is typically set as 0.01 [92].

Another way for evaluating a learned dictionary is to consider the average co-sparsity of the original signals in $Y$ with respect to this dictionary since the final goal of ADL is to acquire a dictionary with which the analysis representations of the signals are sparse. We introduce an operator $\|x\|_0$ counting the number of the elements of $x \in \mathbb{R}^p$, which are below the threshold $\epsilon$, i.e.

$$\|x\|_0 = \text{card}\{i : |x_i| < \epsilon, i = 1, 2, \ldots, p\},$$

(3.20)

where $x_i$ denotes the $i$-th element of $x$ and $\epsilon > 0$. The threshold value $\epsilon$ should be close to zero and it is set as $\epsilon = 0.001$ throughout our experiments. The co-sparsity of a signal can be obtained by applying this operator to the product of the learned dictionary and this signal. The average co-sparsity of all signals are used as the second metric to evaluate the learned dictionaries.

Convergence of the proposed algorithms

Different initial dictionaries are used to demonstrate the convergence of our proposed algorithms. The reference dictionaries were generated with the random variables satisfying the i.i.d. Gaussian distribution with zero mean and unit variance and then the rows of the dictionaries were normalized. The size of the reference dictionaries was $50 \times 25$ (i.e. $p = 50$, $m = 25$). The number of training signals was 50000 (i.e. $n = 50000$) and their co-sparsity was 21 (i.e. $l = 21$) set as in [92]. Analysis SimCO and Incoherent Analysis SimCO were applied to learn analysis dictionaries respectively. The co-sparsity parameters of these two algorithms
were both set as the reference co-sparsity. The coherence limit of Incoherent Analysis SimCO
was set as $\mu_0 = 0.6$, based on our empirical tests.

Three types of matrices were used as initial dictionaries, following the experiments of [89].
The first type is the random matrix consisting of i.i.d. zero mean and unit variance Gaussian
elements. The other two types are vertical concatenations of two matrices. One type is the
vertical concatenations of two $25 \times 25$ 2D DCT matrices (defined as the Kronecker product
of two $5 \times 5$ 1D DCT matrix), and the other is composed of two $25 \times 25$ identity matrices.
We have used 100 independent runs to test the proposed algorithms, and the change of the
objective function in (3.2) shows similar patterns in different runs. Fig. 3.2 shows the object-
ive function value averaged from ten independent tests of the proposed algorithms over the
iterations in noiseless case and noisy case. The objective function decreases monotonically for
all the initializations in both the noiseless case and noisy case. Though the dictionaries were
initialized in different ways, the algorithms converge to a similar final value. This indicates
that our proposed algorithms can converge robustly with different initializations.

Figure 3.2: Objective function value with different initializations in the noiseless case (top) and
the noisy case (bottom). Left column: Analysis SimCO. Right column: Incoherent Analysis
SimCO.
Effect of the atom decorrelation step

Now we compare the correlations of the atoms in the dictionaries learned by Analysis SimCO and Incoherent Analysis SimCO to show the effect of the atom decorrelation step. The initial dictionaries were set as random Gaussian matrices with normalized rows. Other settings were the same as those in the experiments of Fig. 3.2.

In order to observe the correlations of the atoms of a learned dictionary $\Omega$, we define its mutual correlation matrix $M$ as follows

$$ M(\Omega) = \text{abs}(I - \Omega\Omega^T), $$ (3.21)

where operator $\text{abs}(\cdot)$ takes the element-wise absolute value of a matrix. The non-diagonal elements of $M(\Omega)$ represent the correlations between atoms of $\Omega$ and thus the coherence of $\Omega$ is the maximum value of all the elements of $M$, i.e. $\mu(\Omega) = \max(M(\Omega))$. The histograms of the mutual correlation matrices of the dictionaries learned by Analysis SimCO and Incoherent Analysis SimCO in one test are presented in Fig. 3.3 in both the noiseless case and noisy case. In the mutual correlation matrix obtained by Analysis SimCO, there are some elements close to 1, which means that highly-correlated atoms exist in the learned dictionary. These highly-correlated atoms disappear in the dictionary learned by Incoherent Analysis SimCO, as shown in the right plot of Fig. 3.3. This demonstrates that the atom decorrelation step can effectively avoid the highly-correlated atoms in the learned dictionary.

The recovery rate and average co-sparsity averaged from ten independent tests are shown in Fig. 3.4. It can be seen that the recovery rate is higher in both the noiseless case and noisy case, when the Incoherent Analysis SimCO algorithm is applied. The average co-sparsity in the noisy case also increases due to the atom decorrelation step. In the noiseless case, the average co-sparsity obtained by Incoherent Analysis SimCO is lower than that obtained by Analysis SimCO. This is because some atoms which can sparsify the training signals with high co-sparsity are replaced because of their high correlation. Even though the dictionaries learned by Analysis SimCO can reach higher average co-sparsity, Incoherent Analysis SimCO can learn the dictionaries without highly-correlated atoms.
Chapter 3. Analysis SimCO Algorithms for Analysis Dictionary Learning

Figure 3.3: The histograms of the elements in the mutual correlation matrices of the dictionaries learned in the noiseless case (top) and the noisy case (bottom). Left column: Analysis SimCO. Right column: Incoherent Analysis SimCO.

Figure 3.4: Recovery Rate and Average Co-sparsity over iterations in the noiseless case (top) and the noisy case (bottom).
3.5. Simulation Results

Simulations with different parameters

Our proposed algorithms were compared to seven baseline algorithms: ASimCO-Random, ASimCO-IKSVD, Analysis K-SVD [92], LOST [88], AOL [107], Transform K-SVD [39] and GOAL [56].

The algorithms were tested with different parameters, i.e. co-sparsity \( l \), the number of training signals \( n \) and the number of atoms \( p \). In each test, one parameter was changed while the others were fixed, as shown in Table 3.1. These parameters are selected empirically to show the trends of the learning results of the algorithms in terms of recovery rate and average co-sparsity. The reference dictionaries were generated with random variables satisfying i.i.d. Gaussian distribution with zero mean and unit variance and their rows are normalized. The initial dictionaries used in all the algorithms were also generated in the same way.

Table 3.1: Parameters used in the comparison of different ADL algorithms with synthetic data.

<table>
<thead>
<tr>
<th></th>
<th>Fixed parameters</th>
<th>Varying parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( p = 50, m = 25, n = 50000 )</td>
<td>( l \in {4, 8, 12, 16, 20, 24} )</td>
</tr>
<tr>
<td>2</td>
<td>( p = 50, m = 25, l = 18 )</td>
<td>( n \in {0.5, 1, 2, 4, 6, 8} \times 10^4 )</td>
</tr>
<tr>
<td>3</td>
<td>( m = 25, l = 18, n = 50000 )</td>
<td>( p \in {30, 40, 50, 60, 70, 80} )</td>
</tr>
</tbody>
</table>

Analysis SimCO and Incoherent Analysis SimCO were applied for 2500 iterations. For Incoherent Analysis SimCO, the coherence limit was \( \mu_0 = 0.6 \). The parameters of Analysis K-SVD were set as the experiments with synthetic data in [92]. We found that the LOST algorithm fails to recover any atom of the reference dictionary if the parameters as in the original paper [88] are used. This may be because the experiments with synthetic data scale differently from the experiments with image patches in [88]. Extensive experiments were conducted to find good parameters of LOST for the experiments with synthetic data. The coefficients of the penalty terms in the objective function were chosen as 50 and the index parameter in the correlation penalty term was 20. The step size and the iteration number of the inner gradient conjugate algorithm were \( 10^{-4} \) and 30 respectively. The number of iterations for LOST was fixed to 1000. For the AOL algorithms, its noiseless version (NL)AOL and noise-aware ver-

---

1The code of GOAL was downloaded from http://www.gol.ei.tum.de/index.php?id=25&type=98.
(NA)AOL were applied to the noiseless case and the noisy case respectively. The iteration numbers of (NL)AOL and (NA)AOL were 50000 and 10 respectively, according to the settings in [107]. The coefficient of the objective function of (NA)AOL was \( \lambda = 0.3 \). Other parameters of these two algorithms were the same as suggested in [107]. The parameters of Transform K-SVD were set at their default values as in [39]. The parameters of GOAL were set as in the original code\(^2\). The threshold used to replace similar rows in ASimCO-Random is also set as \( \mu_0 \) to be consistent with the coherence limit of Incoherent Analysis SimCO. The parameters for the decorrelation method in ASimCO-IKSVD are set as recommended in [2].

The recovery rate and average co-sparsity averaged from five independent tests with different \( l, n \) and \( p \) are presented in Figs. 3.5, 3.6 and 3.7 respectively. Abbreviations are used in the legends because of space limitation (IN-ASimCO, ASimCO, AKSVD and TKSVD are short for Incoherent Analysis SimCO, Analysis SimCO, Analysis K-SVD and Transform K-SVD respectively).

![Figure 3.5](image)

Figure 3.5: Recovery Rate (left) and Average Co-sparsity (right) with different co-sparsities \( (l \in \{4, 8, 12, 16, 20, 24\}) \) in the noiseless case (top) and the noisy case (bottom).

\(^2\)We should note that, in GOAL the values of the parameters set in the code downloaded are different from those presented in the original paper. Therefore, we tested both sets of parameters, the values of the parameters set in the code were used in our experiments as we observed that they usually lead to better results.
3.5. Simulation Results

Figure 3.6: Recovery Rate (left) and Average Co-sparsity (right) with different numbers of training samples ($n \in \{0.5, 1, 2, 4, 6, 8\} \times 10^4$) in the noiseless case (top) and the noisy case (bottom).

In general, our proposed algorithms, Analysis K-SVD, LOST and Transform K-SVD show similar trends over the varying parameters. This may result from the same measurements used for co-sparsity, i.e. $\ell_0$-norm, and their similar optimization procedure which alternates between the update of the analysis representation and the update of the dictionary. For these five algorithms, better dictionaries can be learned with larger co-sparsities (cf. Fig. 3.5) and more training samples (cf. Fig. 3.6). With the increase of the number of atoms, the recovery rates obtained by these algorithms decrease (cf. Fig. 3.7). The results of Incoherent Analysis SimCO are similar to the results of Analysis K-SVD and Transform K-SVD, which are better than the results of LOST. The recovery rates of the dictionaries obtained by Incoherent Analysis SimCO are higher than Analysis SimCO in all cases due to the restriction of the coherence of the learned dictionary. The average co-sparsities obtained by Analysis SimCO are closer to the reference co-sparsities than those obtained by Incoherent Analysis SimCO in the noiseless case, but the Incoherent Analysis SimCO algorithm shows advantage for the average co-sparsity in the noisy case. The results of (NL)AOL, (NA)AOL and GOAL appear to be quite different from
Chapter 3. Analysis SimCO Algorithms for Analysis Dictionary Learning

The other methods compared. This might be due to the \( \ell_1 \)-norm or \( \ell_p \)-norm \((0 \leq p \leq 1)\) used to estimate the co-sparsity of the coefficients, as opposed to the \( \ell_0 \)-norm used in the other algorithms. The relatively limited performances of (NL)AOL and (NA)AOL may result from the application of the UNTF constraint to the learned dictionaries, that the reference dictionaries do not satisfy. The results of Incoherent Analysis SimCO and ASimCO-Random are very similar to each other, and they both outperform the ASimCO-IKSVD algorithm.

The time (in seconds) of one test with different parameters is presented in Table 3.2\(^3\). From Table 3.2 we can see that our proposed algorithms are faster than Analysis K-SVD, LOST and (NL)AOL, but slower than Transform K-SVD and GOAL. It seems that Transform K-SVD is the best choice to learn dictionaries with synthetic data considering its good performance and efficient computation. However, for the application to image denoising, our proposed algorithm outperforms Transform K-SVD, which will be presented in the next two subsections. The running time of (NA)AOL changes substantially in different cases, as the iteration number changes.

---

\(^3\)All algorithms were implemented in Matlab R2012a and performed with an Intel Core i5 CPU at 3.30GHz and 8GB memory.
of the sub-gradient step changes with different initializations and sometimes this step requires longer time to converge. (NA)AOL seems to be faster than our proposed algorithms as shown in Table 3.2. It should be noted that the running time presented here is the time for one test, however, the conditions for terminating the iterations in the algorithms compared are different. For each iteration, our proposed algorithms are faster than (NA)AOL which is consistent with the analysis of the computational complexities in Section 3.3.5.

3.5.2 Experiments for Image Denoising

We apply the learned dictionaries to image denoising which has become a common application for demonstrating ADL algorithms [92], [39], [88]. In this section, the image denoising framework, the performance evaluation index, and the parameter selection are introduced first. After that, the denoising results for face images and natural images are presented.

Image denoising framework

The image denoising framework employed in our experiments consists of dictionary learning and image recovering, which are both based on small image patches [5], [92]. To denoise a large image of size $\sqrt{N} \times \sqrt{N}$, small image patches of size $\sqrt{m} \times \sqrt{m}$ with $m < N$ are used as the training signals to learn an analysis dictionary $\Omega \in \mathbb{R}^{p \times m}$. These training patches are extracted from the image to be denoised or from other clean images. In the image recovering process, the noisy image is also handled as overlapping patches of the same size. Specifically, $\sqrt{m} \times \sqrt{m}$ patches extracted from the noisy image are reshaped as column vectors which are concatenated as a matrix $Z \in \mathbb{R}^{m \times n}$, where $n$ is the number of patches. The recovering operation is directly applied to $Z$ using the learned dictionary $\Omega$, resulting in a noiseless estimation $Y \in \mathbb{R}^{m \times n}$. Overlapping patches are used to mitigate the blockiness artifacts caused by this patch-based framework. The denoised image can be obtained by reshaping the columns of $Y$ as image patches and averaging these overlapping patches.

The key idea of estimating $Y$ is to solve an optimization problem where the learned analysis dictionary $\Omega$ serves in the regularization term reflecting the co-sparsity prior of $Y$, that is

$$\arg\min_{Y} \| \Omega Y \|_1 + \frac{\lambda}{2} \| Z - Y \|_F^2,$$  (3.22)
Table 3.2: Time of one test with different parameters (in seconds).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASimCO</td>
<td>1373</td>
<td>1232</td>
<td>1323</td>
<td>1290</td>
<td>1329</td>
<td>1488</td>
</tr>
<tr>
<td>IN-ASimCO</td>
<td>1411</td>
<td>1264</td>
<td>1227</td>
<td>1219</td>
<td>1233</td>
<td>1064</td>
</tr>
<tr>
<td>ASimCO-Random</td>
<td>1234</td>
<td>1136</td>
<td>1022</td>
<td>1051</td>
<td>1027</td>
<td>1025</td>
</tr>
<tr>
<td>ASimCO-IKSVD</td>
<td>1188</td>
<td>1054</td>
<td>936</td>
<td>946</td>
<td>981</td>
<td>1002</td>
</tr>
<tr>
<td>AKSVD</td>
<td>1376</td>
<td>2940</td>
<td>4697</td>
<td>6532</td>
<td>8677</td>
<td>11037</td>
</tr>
<tr>
<td>LOST</td>
<td>1523</td>
<td>1500</td>
<td>1559</td>
<td>1716</td>
<td>1521</td>
<td>1448</td>
</tr>
<tr>
<td>TKSVD</td>
<td>209</td>
<td>222</td>
<td>232</td>
<td>242</td>
<td>252</td>
<td>260</td>
</tr>
<tr>
<td>(NL)AOL</td>
<td>4196</td>
<td>4489</td>
<td>4108</td>
<td>4255</td>
<td>4430</td>
<td>4167</td>
</tr>
<tr>
<td>(NA)AOL</td>
<td>13</td>
<td>13</td>
<td>346</td>
<td>347</td>
<td>15</td>
<td>348</td>
</tr>
<tr>
<td>GOAL</td>
<td>28</td>
<td>34</td>
<td>31</td>
<td>34</td>
<td>28</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASimCO</td>
<td>87</td>
<td>263</td>
<td>531</td>
<td>1055</td>
<td>1580</td>
<td>2201</td>
</tr>
<tr>
<td>IN-ASimCO</td>
<td>105</td>
<td>222</td>
<td>504</td>
<td>862</td>
<td>1506</td>
<td>1989</td>
</tr>
<tr>
<td>ASimCO-Random</td>
<td>135</td>
<td>268</td>
<td>537</td>
<td>993</td>
<td>1458</td>
<td>1874</td>
</tr>
<tr>
<td>ASimCO-IKSVD</td>
<td>121</td>
<td>240</td>
<td>463</td>
<td>839</td>
<td>1214</td>
<td>1704</td>
</tr>
<tr>
<td>AKSVD</td>
<td>807</td>
<td>1557</td>
<td>3075</td>
<td>6164</td>
<td>9243</td>
<td>12090</td>
</tr>
<tr>
<td>LOST</td>
<td>354</td>
<td>449</td>
<td>721</td>
<td>1305</td>
<td>2039</td>
<td>2953</td>
</tr>
<tr>
<td>TKSVD</td>
<td>132</td>
<td>146</td>
<td>175</td>
<td>219</td>
<td>272</td>
<td>323</td>
</tr>
<tr>
<td>(NL)AOL</td>
<td>563</td>
<td>1017</td>
<td>1775</td>
<td>3356</td>
<td>5459</td>
<td>7131</td>
</tr>
<tr>
<td>(NA)AOL</td>
<td>59</td>
<td>52</td>
<td>57</td>
<td>71</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>GOAL</td>
<td>5</td>
<td>13</td>
<td>18</td>
<td>29</td>
<td>14</td>
<td>57</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASimCO</td>
<td>1120</td>
<td>1264</td>
<td>1491</td>
<td>1530</td>
<td>1726</td>
<td>1787</td>
</tr>
<tr>
<td>IN-ASimCO</td>
<td>766</td>
<td>874</td>
<td>1344</td>
<td>1360</td>
<td>1454</td>
<td>1528</td>
</tr>
<tr>
<td>ASimCO-Random</td>
<td>908</td>
<td>1000</td>
<td>1106</td>
<td>1231</td>
<td>1356</td>
<td>1518</td>
</tr>
<tr>
<td>ASimCO-IKSVD</td>
<td>799</td>
<td>884</td>
<td>1001</td>
<td>1069</td>
<td>1170</td>
<td>1386</td>
</tr>
<tr>
<td>AKSVD</td>
<td>6700</td>
<td>7069</td>
<td>7561</td>
<td>8081</td>
<td>8697</td>
<td>9127</td>
</tr>
<tr>
<td>LOST</td>
<td>1168</td>
<td>1229</td>
<td>1552</td>
<td>2138</td>
<td>2068</td>
<td>2305</td>
</tr>
<tr>
<td>TKSVD</td>
<td>162</td>
<td>202</td>
<td>244</td>
<td>283</td>
<td>309</td>
<td>338</td>
</tr>
<tr>
<td>(NL)AOL</td>
<td>2610</td>
<td>3349</td>
<td>4690</td>
<td>5255</td>
<td>6134</td>
<td>6893</td>
</tr>
<tr>
<td>(NA)AOL</td>
<td>36</td>
<td>53</td>
<td>68</td>
<td>101</td>
<td>21</td>
<td>95</td>
</tr>
<tr>
<td>GOAL</td>
<td>23</td>
<td>31</td>
<td>32</td>
<td>45</td>
<td>48</td>
<td>60</td>
</tr>
</tbody>
</table>
3.5. Simulation Results

where $\lambda$ is the Lagrangian multiplier to balance the data fidelity term $\|Z - Y\|^2_2$ and the regularization term $\|\Omega Y\|_1$. The Alternating Direction Method of Multipliers (ADMM) [17], [107] is applied to tackle this problem.

It should be noted that the methods used for image recovery in the experiments of LOST [88], Analysis K-SVD [107], Transform K-SVD [39] and GOAL [56] are different, which makes it difficult to evaluate the dictionaries learned by different algorithms consistently. To make a fair comparison, the same image recovering method, formulated as (3.22), is used in our experiments. This method is selected because of its high computational efficiency.

**Denoising performance evaluation and parameter selection**

The images to be denoised were artificially corrupted by additive white Gaussian noise with the standard deviation being either $\sigma = 12.8$ or $\sigma = 45$, choosing empirically to represent the case of a relatively low or high level of noise respectively. Peak Signal to Noise Ratio (PSNR) was used to measure the denoising performance. For an $N$-pixel noise-free image $y \in \mathbb{R}^N$, the PSNR in decibels (dB) of its denoised version $\hat{y} \in \mathbb{R}^N$ is defined as

$$\text{PSNR} = 10 \log_{10} \frac{255^2 N}{\|\hat{y} - y\|^2_2},$$

where $\|\hat{y} - y\|^2_2$ is the mean squared error between the original image and its denoised version.

Throughout our experiments, we followed the same set up as in [107], [39] and fixed the size of the image patches to $8 \times 8$, i.e. $m = 64$. The overlap of the patches was set as $7$. The size of the learned dictionaries was $128 \times 64$, i.e. $p = 128$. In the image recovering process, the proper selection of the Lagrangian multiplier $\lambda$ is related to the noise level. In general, $\lambda$ needs to be smaller when the noise level is higher. Herein a set of different $\lambda$’s was tested, and only the results of $\lambda \in \{0.002, 0.01, 0.05, 0.1, 0.3, 0.5\}$ are presented to show the trends of the denoising results.

We still compare our proposed algorithms with the baseline algorithms as employed in the experiments for the synthetic data. The parameters about co-sparsity were set as follows. For Analysis SimCO and Incoherent Analysis SimCO, the co-sparsities were set as either $l = 40$ or $l = 80$. The corresponding parameters of the baseline algorithms were set based on the value of
Chapter 3. Analysis SimCO Algorithms for Analysis Dictionary Learning

In order to ensure the equal co-sparsity. For Analysis K-SVD, only the $l = 40$ case was tested since $l$ cannot be greater than the signal dimension $m$ in its parameter settings [92]. There is no parameter related to the co-sparsity $l$ in (NA)AOL and GOAL. Other parameters were set as those employed in their original papers except for GOAL whose parameters are the same as in the experiments for the synthetic data. The coherence limit of Incoherent Analysis SimCO and the correlation threshold of ASimCO-IKDVD were both set as 0.2, which was lower than the value used in the experiments with synthetic data, since we found that, in general, the image dictionaries learned have a relatively lower coherence, as compared with that in the synthetic case. The same initial dictionaries, generated with i.i.d. Gaussian distribution with zero mean and unit variance, were used for different algorithms.

**Face image denoising**

Now we denoise face images using the learned analysis dictionaries, following the experiments in [107]. The face images are centred and cropped [65] and can be modeled as piecewise smooth signals approximately. The original face and the noisy face images are shown in Fig. 3.8. Two types of training data were tested: Type I consists of the patches extracted from 13 other clean face images [107]; Type II includes the patches extracted from the face image to be denoised. 16384 patches were randomly selected as training data in both cases [107].

![Figure 3.8: Face images.](image)

(a) (b) (c)

Figure 3.8: Face images. (a) Original face. (b) Noisy face with $\sigma = 12.8$ (PSNR = 26.00 dB). (c) Noisy face with $\sigma = 45$ (PSNR = 15.13 dB).

The PSNR (in dB) values of the denoised face images averaged from five independent tests with varying $\lambda$ are presented in Fig. 3.9 ($\sigma = 12.8$) and Fig. 3.10 ($\sigma = 45$). In each of these two figures, the top and bottom sub-figures show the results using the Type I and Type
3.5. Simulation Results

II training data, respectively. The left and right sub-figures present the denoising results with $l = 40$ and $l = 80$, obtained by our proposed algorithms, LOST and Transform K-SVD. The results of Analysis K-SVD are only shown in the left sub-figures. The results of (NA)AOL and GOAL, which are not related to the co-sparsity setting, are plotted without modifications in both the left and right sub-figures. The best denoising results obtained via different algorithms with varying $\lambda$, i.e. the peak PSNR values of the lines in Fig. 3.9 and Fig. 3.10, are summarized in Table 3.3.

![Graph](image.png)

Figure 3.9: Face image denoising ($\sigma = 12.8$). Top row: training patches are extracted from 13 other clean face images, with the co-sparsity $l = 40$ (left) and $l = 80$ (right). Bottom row: training patches are extracted from the face image to be denoised, with the co-sparsity $l = 40$ (left) and $l = 80$ (right).

Fig. 3.9 and Fig. 3.10 reveal that the denoising results change considerably with various $\lambda$. The best $\lambda$ of the set tested in the case $\sigma = 45$ is smaller than that of the case $\sigma = 12.8$ due to the increase of the noise level. Some common features of the ADL algorithms can also be observed from Table 3.3. The dictionaries learned with higher co-sparsity ($l = 80$) perform better in general. In terms of the types of training data, the image patches from the noisy face image itself (Type II training data) seem to be more suitable to be the training data. As shown in Table 3.3, the performance of GOAL is competitive, compared with other baseline algorithms. For the lower noise level with Type I training data, it obtains the best result. However, our
Figure 3.10: Face image denoising ($\sigma = 45$). Top row: training patches are extracted from 13 other clean faces, with the co-sparsity $l = 40$ (left) and $l = 80$ (right). Bottom row: training patches are extracted from the face image to be denoised, with the co-sparsity $l = 40$ (left) and $l = 80$ (right).

Table 3.3: The best face image denoising results (PSNR in decibels).

<table>
<thead>
<tr>
<th>Noise level</th>
<th>$\sigma = 12.8$</th>
<th>$\sigma = 45$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data type</td>
<td>Type I</td>
<td>Type II</td>
</tr>
<tr>
<td>co-sparsity $l$</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>ASimCO</td>
<td>33.46</td>
<td>34.22</td>
</tr>
<tr>
<td>IN-ASimCO</td>
<td>33.63</td>
<td>34.08</td>
</tr>
<tr>
<td>ASimCO-Random</td>
<td>34.06</td>
<td>34.06</td>
</tr>
<tr>
<td>ASimCO-IKSVD</td>
<td>31.78</td>
<td>33.07</td>
</tr>
<tr>
<td>AKSVD</td>
<td>28.89</td>
<td>---</td>
</tr>
<tr>
<td>LOST</td>
<td>30.90</td>
<td>32.14</td>
</tr>
<tr>
<td>TKSVD</td>
<td>29.16</td>
<td>30.58</td>
</tr>
<tr>
<td>(NA)AOL</td>
<td>33.08</td>
<td>33.27</td>
</tr>
<tr>
<td>GOAL</td>
<td>34.47</td>
<td>33.43</td>
</tr>
</tbody>
</table>
proposed algorithms can obtain better results in other cases. Incoherent Analysis SimCO is able to get higher PSNR than Analysis SimCO in some cases. The Incoherent Analysis SimCO outperforms ASimCO-Random and ASimCO-IKSVD in all cases, except for the cases when Type I training data is applied with co-sparsity being 40.

**Natural image denoising**

Now we examine the denoising of the natural images shown in Fig. 3.11. The size of the images is 256 × 256. Similar to the denoising of face images, two types of training data are tested: Type I contains image patches extracted from 5 other clean images shown in Fig. 3.12; Type II includes the patches of the image to be denoised. The number of training patches is 20000 [92]. The results (PSNR in dB) averaged from the denoised versions of the four test images are plotted in Fig. 3.13 (σ = 12.8) and Fig. 3.14 (σ = 45), and the peak results of each curve are listed in Table 3.4.

![Figure 3.11](image1.png): Test images for natural image denoising.

![Figure 3.12](image2.png): Training images used for learning analysis dictionaries.

According to Fig. 3.13 and Fig. 3.14, the best λ is bigger for the lower noise level, which is consistent with the objective function (3.22). Table 3.4 indicates that the patches extracted from other clean natural images are preferred to the patches of the image to be denoised. In the lower noise level case, for Type I training data (NA)AOL and GOAL gives slightly better results than our proposed algorithms and for Type II training data ASimCO-Random performs
better. Our proposed algorithms outperform the baseline algorithms in other cases. The results obtained by Analysis SimCO and Incoherent Analysis SimCO are similar to each other.

![Graphs showing PSNR comparison for different methods and sparsity levels](image)

Figure 3.13: Natural image denoising ($\sigma = 12.8$). Top row: training patches are extracted from the images in Fig. 3.12 with the co-sparsity $l = 40$ (left) and $l = 80$ (right). Bottom row: training patches are extracted from the natural image to be denoised, with the co-sparsity $l = 40$ (left) and $l = 80$ (right).

### Table 3.4: The best natural image denoising results (PSNR in decibels).

<table>
<thead>
<tr>
<th>Noise level</th>
<th>$\sigma = 12.8$</th>
<th>$\sigma = 45$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data type</td>
<td>Type I</td>
<td>Type II</td>
</tr>
<tr>
<td>co-sparsity $l$</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>ASimCO</td>
<td>31.45</td>
<td>31.17</td>
</tr>
<tr>
<td>IN-ASimCO</td>
<td>31.40</td>
<td>30.91</td>
</tr>
<tr>
<td>ASimCO-Random</td>
<td>31.29</td>
<td>31.25</td>
</tr>
<tr>
<td>ASimCO-IKSVD</td>
<td>30.22</td>
<td>31.26</td>
</tr>
<tr>
<td>AKSVD</td>
<td>28.12</td>
<td>—</td>
</tr>
<tr>
<td>LOST</td>
<td>29.47</td>
<td>31.05</td>
</tr>
<tr>
<td>TKSVD</td>
<td>29.59</td>
<td>30.17</td>
</tr>
<tr>
<td>(NA)AOL</td>
<td>31.47</td>
<td>29.57</td>
</tr>
<tr>
<td>GOAL</td>
<td>31.53</td>
<td>29.56</td>
</tr>
</tbody>
</table>

Besides the analysis dictionary learning based denoising methods as presented above, we have also run some experiments using state-of-the-art algorithms that are specifically designed for
3.6 Summary

In this chapter we have proposed an analysis dictionary learning algorithm: Analysis SimCO. The dictionary learning process is formulated as an optimization problem with the co-sparsity and unit $\ell_2$-norm constraints on the atoms of the dictionary. This algorithm iteratively solves this problem by hard thresholding and the gradient descent method on manifolds. We have also
presented an extension of Analysis SimCO: Incoherent Analysis SimCO, by incorporating an atom decorrelation step after the dictionary update step. Extensive experiments on synthetic data, face and natural image data have confirmed the competitive performance of our proposed algorithms. The applications of learned analysis dictionaries in other signal processing tasks merit more study, which we leave for future work.
Chapter 4

Multiplicative Noise Removal Based on Sparse Analysis Model

The problem of multiplicative noise removal from corrupted images has recently been considered under the framework of regularization based optimization approaches, where the regularizations are typically defined on sparse dictionaries and/or Total Variation (TV). This framework was demonstrated to be effective and promising. However, the sparse regularizers used so far are based overwhelmingly on the synthesis model, and the TV based regularizer may induce the stair-casing effect in the reconstructed image with piecewise constant areas.

In this chapter, we propose two new methods for this problem based on the sparse analysis model. The first algorithm is referred to as Removing Speckle Noise by Analysis Dictionary Learning (RSN-ADL). The formulation of this algorithm consists of a data fidelity term derived from the distribution of the noise and a regularization term using a learned analysis dictionary. Then the Alternating Direction Method of Multipliers (ADMM) is applied to address the optimization of this model.

The RSN-ADL algorithm is able to preserve details while reducing multiplicative noise, however, the smooth regions cannot be well-recovered. To overcome this, we extend the reconstruction model of RSN-ADL by introducing an enhanced TV regularizer. This regularizer employs a parameter to control the smoothness constraint defined on the pixel-wise differences. To address the optimization problem of the proposed model, we adapt the ADMM framework,
and present a new method where a relaxation technique is developed to allow the variables to be updated flexibly with either image patches or the whole image, as required by the learned dictionary and the enhanced TV regularizer, respectively. Experimental results demonstrate the improved performance of the proposed method as compared with several recent baseline methods, especially for a relatively high noise level.

To sum up, the main contributions in this chapter are the following four points. (1) A regularization term based on a learned analysis dictionary is introduced to remove multiplicative noise, as will be presented in the formulations of RSN-ADL and MNR-ADL-SR. (2) The optimization method to address the formulation of RSN-ADL is developed using the framework of ADMM. (3) A smoothness regularizer is introduced in the formulation of MNR-ADL-SR to further smooth the images. (4) Optimization methods corresponding to the formulation of MNR-ADL-SR are proposed by adapting ADMM and employing a relaxation technique.

4.1 Introduction

Mathematically, the observed image \( w \in \mathbb{R}^N \) (reshaped from a \( \sqrt{N} \times \sqrt{N} \) image), contaminated by the multiplicative noise \( u \in \mathbb{R}^N \), can be represented as \cite{11, 58}

\[
w = g \circ u,
\]

where \( g \in \mathbb{R}^N \) denotes the image to be restored. The symbol \( \circ \) denotes the Hadamard product (i.e. entry-wise product) of two matrices/vectors. The aim of despeckling is to estimate \( g \) from the observed image \( w \). In this paper, we focus on Gamma distributed multiplicative noise \( u \) whose entries are assumed to be independent and identically distributed (i.i.d.) with probability density function given by \cite{38, 11, 58}

\[
f_u(u) = \frac{L^L}{\Gamma(L)} u^{L-1} e^{-Lu},
\]

where \( L \) is a positive integer implying the noise level and \( \Gamma(\cdot) \) is the classical Gamma function defined by \( \Gamma(L) = (L - 1)! \). A smaller \( L \) indicates stronger noise.

Classical methods for removing multiplicative noise are spatial filtering \cite{64, 49, 63} and
4.1. Introduction

wavelet domain filtering \cite{104,6}. More recently, regularization based approaches where the image reconstruction task is formulated as an optimization problem with regularizers have attracted much attention \cite{11,58,8,95,59}. A popular regularizer employed in these approaches is Total Variation (TV) which was proposed originally for reducing additive Gaussian noise \cite{94}. The TV-based methods were then used for multiplicative noise in the original image domain as in equation (4.1) or in the log-domain by applying the logarithm transform. Typical examples performed in the original domain are the first TV-based multiplicative noise removal method proposed in \cite{93} and the method of Aubert-Aujol (AA) \cite{8}. The method in \cite{93} minimizes the TV of the image to be recovered with the constraints involving the mean and variation of the noise, but this method is not effective for removing Gamma distributed noise as the noise considered in its restoration model is assumed to follow Gaussian distribution. The AA method \cite{8} exploits a Bayesian maximum a posteriori (MAP) estimate, yielding an image restoration model consisting of a data fidelity term based on the prior distribution of the multiplicative noise and a TV regularization term. However, the quality of the image restored by AA may be limited by the local solutions obtained from the optimization of a non-convex model.

Another class of denoising methods based on TV regularizer considers the image restoration in the log-domain \cite{11,58,95,59}, aiming to simplify the multiplicative noise model as an additive model which is easier to deal with. In general, the reconstruction models employed in these methods commonly consist of a data fidelity term and regularization terms reflecting prior information of the image. However, the formulations of these terms and optimization approaches may differ substantially. In \cite{95}, Shi and Osher (SO) considered both the data fidelity and TV terms of AA \cite{8} in the log-domain to overcome the non-convex optimization issue in AA. Multiplicative Image Denoising with the Augmented Lagrangian (MIDAL) algorithm \cite{11} uses the same model as in SO but applies a different optimization framework based on variable splitting and augmented Lagrangian for better numerical efficiency. Apart from the data fidelity term and the TV regularization as in the reconstruction model of SO \cite{95} and MIDAL \cite{11}, the method presented in \cite{59} also incorporates a quadratic data fitting term to apply the TV term in a more efficient manner, but it tends to be outperformed by MIDAL \cite{11}. The algorithm proposed in \cite{67} introduces a different quadratic term to reflect the statistical distribution of the noise. This model has been proven to be strictly convex under a mild condition and it can lead to better results than the method presented in \cite{59}. 
Although the TV regularization proves to be effective for reducing multiplicative noise, the smoothly varying regions in the original image are usually recovered as piecewise constant areas, which is also well known as the stair-casing effect [38]. An approach to avoid this issue is to use alternative regularization terms instead of the TV term. The method proposed in [68] uses nonlocal functionals based regularizations to develop denoising models in the log-domain and the original domain, respectively. A strategy to choose the regularization parameters has also been developed. The SAR Block-Matching 3D (SAR-BM3D) algorithm [81] combines the nonlocal filtering technique and the wavelet-domain shrinkage method, and it can be considered as one of the most promising despeckling algorithms.

In recent years, the sparsity prior was shown to be helpful for the reconstruction of the images with multiplicative Gamma noise. Typical methods are DFN [38], MNR-DL-TV1 [55], MNR-DL-TV2 [58] and Iter-NSM [105], as reviewed in Chapter 2. However, it should be noted that the learned dictionaries employed in these methods are based on the sparse synthesis model [5]. Dictionary learning based on the sparse analysis model was also shown to be effective in the reduction of additive Gaussian noise [92], [34], but little work has been done for studying its potential in removing multiplicative noise.

In this chapter, we propose two new multiplicative noise removal algorithms based on the sparse analysis model. The first algorithm is RSN-ADL, as mentioned before, where the image reconstruction model contains a data fidelity term and a sparse analysis model based regularization term. Specifically, the data fidelity term is formed using the maximum likelihood estimate, and the sparse analysis model based regularizer is constructed with an analysis dictionary learned from image patches via the Analysis SimCO algorithm [33], [34] presented in Chapter 3. In order to address the optimization problem of the proposed model, ADMM [17] is applied to decompose it as a sequence of sub-problems which are easier to solve. In particular, one auxiliary variable is introduced to split the variables by reformulating our model as a constrained optimization problem, and then the augmented Lagrangian function is constructed to covert the constrained problem as an unconstrained one. The variables are then updated in an alternating manner.

To further improve the reconstruction quality of the smooth regions, the model of RSN-ADL is extended by introducing a smoothness regularizer, resulting in a new algorithm referred to
as Multiplicative Noise Removal using Analysis Dictionary Learning and a Smoothness Regularizer (MNR-ADL-SR). The joint employment of the sparse analysis model and smoothness regularizers in MNR-ADL-SR aims to exploit the benefit of both priors. Since dictionaries are usually well adapted to textures but not good at smooth areas [58], the introduction of the smoothness regularizer has the potential to overcome this issue. The employment of the two regularizers in the restoration formulation of MNR-ADL-SR, however, renders the optimization task non-trivial, especially since the two regularizers are defined in different forms of the image. In particular, the dictionary is learned with image patches instead of the whole image in order to reduce the computational complexity. As a result, the sparse analysis model based regularizer is represented with image patches. The smoothness regularizer, however, is defined with the pixel-wise differences in the whole image. In order to address the optimization problem of the presented model, we propose a new method based on the framework of ADMM [17]. Similar to the optimization for the RSN-ADL model, the optimization problem for MNR-ADL-SR can also be converted to a sequence of sub-problems by introducing auxiliary variables and developing the augmented Lagrangian function. However, in the sub-problem related to the smoothness regularizer, there exist two variables in different forms. To address this issue, an approximation technique is applied to relax the original sub-problem as a problem with a unified variable.

4.2 Removing Speckle Noise by Analysis Dictionary Learning

In this section, we introduce the formulation of our proposed RSN-ADL algorithm and the optimization method for this model.

4.2.1 Problem Formulation

To simplify the problem, the logarithm transform is employed here to convert the multiplicative noise model to an additive one, as in [11], [38], [58]. Taking the (element-wise) logarithms of both sides of (4.1), we have

\[
\log w = \log g + \log u
\]

\[
z = y + v
\]
where \( z, y \) and \( v \) denote the element-wise logarithms of \( w, g \), and \( u \), respectively. Since the function \( u = e^v \) is strictly monotonic and the entries of \( u \) satisfy the i.i.d. Gamma distribution (4.2), the probability density function of the entries in \( v \) is given by [10, pp. 207]

\[
f_v(v) = f_u(e^v) \frac{d(e^v)}{dv} = \frac{L^L}{\Gamma(L)} e^{L(v-e^v)}. \tag{4.4}
\]

Hence, the probability density function of \( v \) is given by

\[
f_v(v) = \prod_{i=1}^{N} \frac{L^L}{\Gamma(L)} e^{L(v_i-e^{v_i})}, \tag{4.5}
\]

where \( v_i \) denote the entries of the vector \( v \) with \( i = 1, 2, ..., N \). As a result, the log-likelihood function can be written as

\[
\log f_{z|y}(z|y) = \log f_v(z - y) = N \log \frac{L^L}{\Gamma(L)} + L \sum_{i=1}^{N} z_i - L \sum_{i=1}^{N} (y_i + e^{z_i} - y_i). \tag{4.6}
\]

The ML estimate for \( y \) can be determined by maximizing (4.6) with respect to \( y \). Furthermore, by omitting the first two terms which do not depend on \( y \) and scaling the last term by the negative constant coefficient \(-L\), the maximization of (4.6) is equivalent to the minimization problem as follows

\[
\hat{y} = \arg \min_y \sum_{i=1}^{N} (y_i + e^{z_i} - y_i). \tag{4.7}
\]

It is easy to check that the optimal solution to the above problem is \( \hat{y} = z \), but it is a trivial solution for the denoising task. This is due to the over-fitting problem which can be regarded as a general issue of maximum likelihood [12, pp. 9]. In order to avoid this problem, the regularization technique is often employed, which involves adding penalty terms based on the prior information of \( y \).

Using the data fidelity term based on the ML estimate (4.7), our proposed restoration formulation utilizes a regularization promoting the sparsity. In particular, the image patches to be recovered are assumed to be sparse with respect to an analysis dictionary. Since adaptive ana-
4.2. Removing Speckle Noise by Analysis Dictionary Learning

Analysis dictionaries usually have the potential to fit signals better than pre-defined dictionaries [92], the analysis dictionary learned via the Analysis SimCO algorithm is applied in our proposed method. The proposed formulation can be written as

\[
Y^* = \arg \min_{Y} \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j}} - Y_{i,j}) + \lambda \|\Omega Y\|_1, \tag{4.8}
\]

where \( \Omega \in \mathbb{R}^{p \times m} \) denotes the learned analysis dictionary with Analysis SimCO (cf. Algorithm 9). In this formulation, the restored image \( y \) is expanded in small patches of size \( \sqrt{m} \times \sqrt{m} \) which composes the matrix \( Y \in \mathbb{R}^{m \times n} \) as column vectors, as the dictionary \( \Omega \) is usually learned from image patches instead of a whole image. Accordingly, the matrix \( Z \in \mathbb{R}^{m \times n} \) is obtained with the observed log-image \( z \) in the same way, where \( n \) denotes the number of image patches. The data fidelity term \( \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j}} - Y_{i,j}) \) is the image patch version of the ML estimate (4.7). The regularization term \( \|\Omega Y\|_1 \) reflects the sparse property of the image patches with respect to the dictionary \( \Omega \). The multiplier \( \lambda \) is used to balance the data fidelity term and the regularizer.

In general, our proposed method consists of two stages: analysis dictionary learning and image recovery. In the dictionary learning stage, an analysis dictionary \( \Omega \) is learned using the Analysis SimCO algorithm [34] as presented in Chapter 3. The goal of the image recovery stage is to restore the clean image from the observed image, which is achieved by addressing the optimization problem (4.8). The restored log-image \( \hat{y} \) can be obtained by reshaping the solution to (4.8), and thus the denoised image \( \hat{g} \) can be obtained by taking the exponential transform of \( \hat{y} \). The RSN-ADL algorithm is summarised as Algorithm 13. The optimization method to address (4.8) (i.e. Step 2(b) of Algorithm 13) will be presented in the next subsection.

4.2.2 Optimization Method

In this subsection, the optimization method to address the problem (4.8) is presented. First, a variable splitting technique is employed to construct a decomposable structure in the objective function across multiple variables. Then the ADMM framework [17] is applied to deal with the constrained optimization problem.

Using the variable splitting technique, the problem (4.8) can be converted to an equivalent
Algorithm 13 Removing Speckle Noise by Analysis Dictionary Learning (RSN-ADL)

**Input:** noisy image $w$, training data $A$, multiplier $\lambda$

**Output:** recovered image $\hat{g}$

**RSN-ADL:**

The RSN-ADL algorithm consists of two stages as follows.

1. **Dictionary Learning Stage:**
   - Using the training data $A$, learn an analysis dictionary $\Omega$ via the Analysis SimCO Algorithm.

2. **Image Recovery Stage:**
   - (a) Apply logarithm transform to the noisy image $w$, i.e. $z = \log(w)$, and generate matrix $Z$ by using the image patches extracted from $z$ as columns.
   - (b) Substitute the learned dictionary $\Omega$ and the transformed image $Z$ into the optimization problem (4.8) and solve it to obtain the solution $Y^\star$.
   - (c) Reconstruct the denoised log-image $\hat{y}$ by reshaping $Y^\star$ to a complete image and take the exponential transform of $\hat{y}$ to obtain the denoised image $\hat{g}$, i.e. $\hat{g} = \exp(\hat{y})$.

The variable $T = \Omega Y$ is introduced to eliminate $Y$ in the regularization term and therefore make the objective function separable with respect to the variables $Y$ and $T$.

ADMM can be viewed as an attempt to combine the decomposable benefit of dual ascent and the superior convergence property of the augmented Lagrangian methods for constrained optimization [17]. The constrained optimization problem (4.9) can be handled with ADMM since the objective function becomes separable across the variables and the decomposed sub-problems are easier to address. Using a dual parameter $B \in \mathbb{R}^{p \times n}$, the augmented Lagrangian function can be formed by adding a penalty term $\langle B, \Omega Y - T \rangle$ and an extra quadratic term.
4.2. Removing Speckle Noise by Analysis Dictionary Learning

related to the constraint \( T = \Omega Y \), that is

\[
L_\gamma(Y, T, B) = \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j}} - Y_{i,j}) + \lambda \|T\|_1 + \gamma \langle B, \Omega Y - T \rangle + \frac{\gamma}{2} \|\Omega Y - T\|_F^2
\]

\[
L_\gamma(Y_{i,j} + e^{Z_{i,j}} - Y_{i,j}) + \lambda \|T\|_1 + \frac{\gamma}{2} \|B + \Omega Y - T\|_F^2 - \frac{\gamma}{2} \|B\|_F^2.
\]

(4.10)

where \( \gamma > 0 \) is the penalty coefficient. ADMM iteratively updates each of the variables \( \{Y, T, B\} \), while keeping the others fixed. We use the scaled form of ADMM \[17\] as it is more concise to express. In the \( t \)-th iteration, it consists of the following steps

\[
Y^{(t+1)} = \arg \min_Y L_\gamma(Y, T^{(t)}, B^{(t)})
\]

(4.11)

\[
T^{(t+1)} = \arg \min_T L_\gamma(Y^{(t+1)}, T, B^{(t)})
\]

(4.12)

\[
B^{(t+1)} = B^{(t)} + (\Omega Y^{(t+1)} - T^{(t+1)}).
\]

(4.13)

In fact, herein ADMM can be interpreted as reducing the \( \ell_1 \) regularized problem to a sequence of sub-problems which are easier to solve \[17\]. The ADMM iterations (4.11) and (4.12) are performed until the change of \( Y^{(t+1)} \) is relatively small compared with \( Y^{(t)} \).

Now we explain the update of variables in (4.11)-(4.13) respectively. Ignoring the terms unrelated to \( Y \), the minimization problem (4.11) can be written as

\[
\arg \min_Y \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j}} - Y_{i,j}) + \frac{\gamma}{2} \|B + \Omega Y - T\|_F^2.
\]

(4.14)

As this objective function is differentiable, the gradient-based methods can be applied. Here we employ the gradient descent method which has a relatively low computational complexity. The step size can be determined by line search methods \[78\], however, a small fixed step size also works well, according to our experiments. Given a step size \( \mu \), the update of \( Y \) can be written as

\[
Y = Y - \mu \nabla_Y.
\]

(4.15)

The symbol \( \nabla_Y \) denotes the gradient of the objective function in (4.14) with respect to \( Y \),
which can be calculated as follows

\[ \nabla_Y = (1 - e^{Z-Y}) + \gamma \Omega^T (B + \Omega Y - T), \]  

(4.16)

where \(1 \in \mathbb{R}^{m \times n}\) is an all-one matrix with the same size as \(Y\) and \(e^{Z-Y}\) denotes the element-wise exponential of \(Z - Y\).

**Algorithm 14** Optimization in the Image Recovery Stage of RSN-ADL

**Input:** \(Z, \Omega, \lambda\)  
**Output:** \(Y^*\)  
**Initialization:**  
Initialize the iteration counter \(t = 1\) and the initial point \(Y^{(t)} = Z\). Perform the following steps.  
**Main Iterations:**

1. Update \(Y^{(t+1)} \leftarrow Y^{(t)}\) by applying the gradient descent method to the problem (4.14), i.e. using equations (4.15) and (4.16).  
2. Update \(T^{(t+1)} \leftarrow T^{(t)}\) by solving the problem (4.17) via soft-thresholding (4.18), (4.19).  
3. Update \(B^{(t+1)} \leftarrow B^{(t)}\) based on equation (4.13).  
4. If the stopping criterion is satisfied, \(Y^* = Y^{(t+1)}\), quit the iteration. Otherwise, increase the iteration counter \(t = t + 1\) and go back to step 1).

For the update of \(T\), the problem (4.12) can be equivalently written as

\[ \arg \min_T \lambda \|T\|_1 + \frac{\gamma}{2} \|B + \Omega Y - T\|_F^2. \]  

(4.17)

Notice that this problem has a closed-form solution given by (17)

\[ T = ST_{\lambda/\gamma}(\Omega Y + B). \]  

(4.18)

The symbol \(ST_{\lambda/\gamma} (\cdot)\) represents the element-wise soft-thresholding operator defined by

\[ ST_{\lambda/\gamma} (\theta) = \begin{cases} \theta - \frac{\lambda}{\gamma} \cdot \text{sgn}(\theta) & \text{if } |\theta| \geq \frac{\lambda}{\gamma} \\ 0 & \text{otherwise}, \end{cases} \]  

(4.19)

where \(\text{sgn}(\theta)\) returns the sign of \(\theta\).
4.3 Sparse Analysis Model Based Multiplicative Noise Removal with Enhanced Regularization

To sum up, the optimization for problem (4.8) in the image recovery stage of RSN-ADL can be summarized as Algorithm [14].

4.3 Sparse Analysis Model Based Multiplicative Noise Removal with Enhanced Regularization

As will be observed Section [4.4], the RSN-ADL algorithm performs well in preserving image details but its capability to recover smooth regions is limited. The MNR-ADL-SR algorithm is proposed to address this issue. Following the framework of RSN-ADL, MNR-ADL-SR also contains the analysis dictionary learning stage and the image recovery stage. However, a different formulation is applied to recover the image, i.e. the formulation in step 2(b) of Algorithm [13] is modified in MNR-ADL-SR. The image restoration formulation model of MNR-ADL-SR and the corresponding optimization method will be presented in the following subsections.

4.3.1 Proposed Image Restoration Formulation

Apart from the data fidelity and sparsity promoting regularization terms in the formulation of RSN-ADL (cf. equation 4.8), the image restoration model of MNR-ADL-SR employs a smoothness regularizer defined with the discrete derivatives of the image. Specifically, the formulation for recovering image in MNR-ADL-SR can be written as follows

$$Y^* = \arg \min_Y \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e Z_{i,j} - Y_{i,j}) + \lambda_1 \|\Omega Y\|_1 + \lambda_2 G_\beta\{R(Y)\}.$$ (4.20)

The data fidelity term \(\sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e Z_{i,j} - Y_{i,j})\) and the first regularization term \(\|\Omega Y\|_1\) are the same as those employed in (4.8). The second regularization term \(G_\beta\{R(Y)\}\) is used to promote the smoothness of the whole image, where the image patch version \(Y\) is reshaped back to the complete image by applying the operator \(R(\cdot)\) and \(G_\beta\{\cdot\}\) is the smoothness promotion function.
For a given image denoted by $S \in \mathbb{R}^{d \times d}$, $G_\beta \{ S \}$ is defined as

$$
G_\beta \{ S \} = \sum_{i=1}^{d} \sum_{j=1}^{d} \left( \sqrt{ (\nabla_h S_{i,j})^2 + (\nabla_v S_{i,j})^2 } \right)^\beta,
$$

(4.21)

where $\nabla_h S_{i,j}$ and $\nabla_v S_{i,j}$ denote the horizontal and vertical differences at pixel $S_{i,j}$. More specifically, they are given by the first-order differences between pixel $S_{i,j}$ and its horizontal and vertical neighbouring pixels respectively, i.e.

$$
\nabla_h S_{i,j} = \begin{cases} 
S_{i+1,j} - S_{i,j} & \text{if } i < d, \\
0 & \text{if } i = d.
\end{cases}
$$

(4.22)

and

$$
\nabla_v S_{i,j} = \begin{cases} 
S_{i,j+1} - S_{i,j} & \text{if } j < d, \\
0 & \text{if } j = d.
\end{cases}
$$

(4.23)

The parameter $\beta$ controls the degree of smoothness. Notice that the smoothness promotion function $G_\beta \{ \cdot \}$ is equivalent to the TV regularizer [94] when $\beta = 1$ in which sense the smoothness regularizer $G_\beta \{ \cdot \}$ can be viewed as a generalization of the TV regularizer.

If the multiplier $\lambda_2$ in the model (4.20) is set as zero, model (4.20) will reduce to (4.8). From this point of view, the restoration model (4.8) can be seen as a special case of (4.20). The sparsity based regularizer $\| \Omega Y \|_1$ can be regarded as a local prior since it is defined with the image patches in $Y \in \mathbb{R}^{m \times n}$ whereas the smoothness regularizer in (4.20) is a global prior which depends on the entire image $R(Y) \in \mathbb{R}^{\sqrt{N} \times \sqrt{N}}$. Thus, the introduction of the smoothness regularizer not only further reinforces the smoothness of the restored image, but also takes the global prior of the image into consideration.

### 4.3.2 Optimization Method

The optimization of problem (4.20) can also be decomposed as a set of sub-problems following the ADMM framework as employed in the image recovery stage of RSN-ADL. However, the decomposition and the alternating update of the variables is more complicated because of the introduction of more regularization terms. Moreover, as the smoothness regularizer is defined in the form of whole image rather than the image patch form as used by the sparsity based
regularizer, two variables in different forms are involved in the sub-problem related to the
smoothness regularizer, which also makes the optimization more difficult. In this section, we
propose a new method to solve the optimization problem in (4.20).

To eliminate $Y$ in the regularization terms and thus decompose the problem, the variables
$T = \Omega Y$ and $M = Y$ are introduced, which leads to the equivalent constrained form of
problem (4.20), that is

$$\arg \min_{\{Y, T, M\}} \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j}-Y_{i,j}}) + \lambda_1 \|T\|_1 + \lambda_2 G_\beta \{R(M)\}$$

(4.24)

s.t. $T = \Omega Y$, $M = Y$.

Hence, the augmented Lagrangian function can be written as

$$L_{\gamma_1, \gamma_2}(Y, T, M, B_1, B_2)$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j}-Y_{i,j}}) + \lambda_1 \|T\|_1 + \lambda_2 G_\beta \{R(M)\}$$

$$+ \gamma_1 \langle B_1, \Omega Y - T \rangle + \frac{\gamma_1}{2} \|\Omega Y - T\|_F^2 + \gamma_2 \langle B_2, Y - M \rangle + \frac{\gamma_2}{2} \|Y - M\|_F^2$$

(4.25)

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j}-Y_{i,j}}) + \lambda_1 \|T\|_1 + \lambda_2 G_\beta \{R(M)\}$$

$$+ \frac{\gamma_1}{2} \|B_1 + \Omega Y - T\|_F^2 - \frac{\gamma_1}{2} \|B_1\|_F^2 + \frac{\gamma_2}{2} \|B_2 + Y - M\|_F^2 - \frac{\gamma_2}{2} \|B_2\|_F^2,$$

where $\gamma_1, \gamma_2 > 0$ are the penalty coefficients. Note that two dual parameters $B_1 \in \mathbb{R}^{p \times n}$ and
$B_2 \in \mathbb{R}^{m \times n}$ are introduced to incorporate the constraints of (4.24) into the objective function.

Following the ADMM framework, the variables are updated alternatively for optimizing (4.25),
that is

$$Y^{(t+1)} = \arg \min_Y L_{\gamma_1, \gamma_2}(Y^{(t+1)}, T^{(t)}, M^{(t)}, B_1^{(t)}, B_2^{(t)})$$

(4.26)

$$T^{(t+1)} = \arg \min_T L_{\gamma_1, \gamma_2}(Y^{(t+1)}, T, B_1^{(t)})$$

(4.27)

$$M^{(t+1)} = \arg \min_M L_{\gamma_1, \gamma_2}(Y^{(t+1)}, M, B_2^{(t)})$$

(4.28)

$$B_1^{(t+1)} = B_1^{(t)} + (\Omega Y^{(t+1)} - T^{(t+1)})$$

(4.29)

$$B_2^{(t+1)} = B_2^{(t)} + (Y^{(t+1)} - M^{(t+1)})$$

(4.30)
where $t$ denotes the iteration number.

The update of $Y$ and $T$ is similar to that in the optimization of RSN-ADL. Specifically, the update of $Y$ involves a differentiable objective function as follows

$$
\arg \min_Y \sum_{i=1}^{m} \sum_{j=1}^{n} (Y_{i,j} + e^{Z_{i,j} - Y_{i,j}}) + \gamma_1 \|B_1 + \Omega Y - T\|_F^2 + \gamma_2 \|B_2 + Y - M\|_F^2,
$$

which can be addressed by the gradient descent method as given in (4.15). The gradient $\nabla Y$ can be calculated as follows

$$
\nabla Y = (1 - e^{Z-Y}) + \gamma_1 \Omega^T (B_2 + \Omega Y - T) + \gamma_2 (B_2 + Y - M).
$$

The update of $T$ is corresponding to the optimization of (4.27), i.e.

$$
\arg \min_T \lambda_1 \|T\|_1 + \gamma_1 \|B_1 + \Omega Y - T\|_F^2.
$$

The solution to the above problem can be also given in a closed-form using the soft-thresholding operator (4.19), that is

$$
T = ST_{\lambda_1/\gamma_1} (\Omega Y + B_1).
$$

Dropping the unrelated terms, the update of $M$ based on (4.28) can be obtained by considering the following problem

$$
\arg \min_M \lambda_2 G_\beta (M_R) + \gamma_2 \|N - M\|_F^2
$$

s.t. $M_R = R(M)$,

where $N = B_2 + Y$. In the objective function, there are two variables $M_R$ and $M$ which are linked via the constraint $M_R = R(M)$. By applying the operator $R(\cdot)$ to the matrices $N$ and $M$ respectively, the quadratic term can be rewritten in terms of $M_R$, i.e.

$$
\|N - M\|_F^2 = \|(R(N) - R(M)) \circ C\|_F^2 = \|(N_R - M_R) \circ C\|_F^2,
$$

where $N_R = R(N)$, and $C$ is a constant matrix depending on the operator $R(\cdot)$. Specifically,
4.3. Sparse Analysis Model Based Multiplicative Noise Removal with Enhanced Regularization

the squares of the elements in $C$ represent the appearance times of the corresponding elements of $N_R - M_R$ in the matrix $N - M$. When the overlap between two neighbouring patches is $\sqrt{m} - 1$, most elements in $C$, except for the elements on the border, are the same. Hence, the quadratic term $\|N - M\|_F^2$ can be approximated as $c\|N_R - M_R\|_F^2$ with $c$ being a scalar. As a result, the problem (4.35) can be relaxed as

$$M_R^* = \arg\min_{M_R} \lambda_2 G_{\beta}(M_R) + \frac{\gamma_2}{2} \|N_R - M_R\|_F^2,$$ (4.37)

and $M$ can be obtained by applying the inverse operator of $R(\cdot)$ to $M_R$.

Obviously, the optimization of (4.37) depends on the value of $\beta$. Here, two cases are considered, i.e. $\beta \in \{1, 2\}$. When $\beta = 1$, (4.37) can be written as

$$\arg\min_{M_R} \lambda_2 \|M_R\|_{TV} + \frac{\gamma_2}{2} \|N_R - M_R\|_F^2.$$ (4.38)

This can be viewed as a TV-$\ell_2$ minimization problem which can be addressed by Chambolle’s algorithm [25].

When $\beta = 2$, the problem (4.37) is equivalent to

$$\arg\min_{M_R} \lambda_2 \sum_{i,j} [(\nabla h(M_R)_{i,j})^2 + (\nabla v(M_R)_{i,j})^2] + \frac{\gamma_2}{2} \|N_R - M_R\|_F^2,$$ (4.39)

and it can be addressed by solving the Euler-Lagrange equation numerically [94], [26]. Specifically, the optimal solution can be approached iteratively by the gradient descent step [26] as follows (the detailed derivation is given in Appendix A)

$$M_R^{(k+1)} = M_R^{(k)} + \tau \left[ 2\lambda_2 \left( \frac{\partial M_R^{(k)}}{\partial x} \left( \frac{\partial M_R^{(k)}}{\partial x} \right) + \frac{\partial M_R^{(k)}}{\partial y} \left( \frac{\partial M_R^{(k)}}{\partial y} \right) \right) - \gamma_2 (M_R^{(k)} - N_R) \right],$$ (4.40)

where $\frac{\partial}{\partial x} (\frac{\partial M_R}{\partial x})$ and $\frac{\partial}{\partial y} (\frac{\partial M_R}{\partial y})$ denote the second-order discrete derivatives in the horizontal and vertical directions respectively, $\tau$ is the step size and $k$ represents the iteration number.

The optimization method for the image recovery stage of the MNR-ADL-SR algorithm is summarized as Algorithm 15.
Algorithm 15 Optimization in the Image Recovery Stage of MNR-ADL-SR

Input: $Z$, $Ω$, $λ_1$, $λ_2$, $β$
Output: $Y^*$

Initialization:
Initialize the iteration counter $t = 1$ and the initial point $Y^{(t)} = Z$. Perform the following steps.

Main Iterations:

1. Update $Y^{(t+1)} ← Y^{(t)}$ by applying the gradient descent method to the problem $4.31$, i.e. using equations $4.15$ and $4.32$.
2. Update $T^{(t+1)} ← T^{(t)}$ by solving the problem $4.33$ via soft-thresholding $4.34$, $4.19$.
3. Update $M^{(t+1)} ← M^{(t)}$ by addressing the problem $4.37$ and applying the inverse operator of $R(·)$ to the optimal solution $M^*_{R}$. When $β = 1$, $M^*_{R}$ can be obtained by applying Chambolle’s algorithm to $4.38$. When $β = 2$, $M^*_{R}$ is estimated with the iteration $4.40$.
4. Update $B_1^{(t+1)} ← B_1^{(t)}$ and $B_2^{(t+1)} ← B_2^{(t)}$ based on equations $4.29$ and $4.30$.
5. If the stopping criterion is satisfied, $Y^* = Y^{(t+1)}$, quit the iteration. Otherwise, increase the iteration counter $t = t + 1$ and go back to step 1).

4.4 Experimental Results

In this section, the experiments for synthetic images with multiplicative noise and real SAR images are presented respectively. The MNR-ADL-SR algorithm is tested with $β = 1$ and $β = 2$, which are referred to as MNR-ADL-SR$1$ and MNR-ADL-SR$2$ respectively. Our proposed algorithms are compared with three recent algorithms: DFN [38] (which outperforms the AA [8] and SO [95] algorithms), MIDAL [11], and MNR-DL-TV-2 [58]. These three algorithms were selected as baselines because of the involvement of sparsity or TV regularizer in their formulations and the availability of their code.

For our proposed MNR-ADL-SR$1$, MNR-ADL-SR$2$ and RSN-ADL [32] algorithms, the images in Fig. 4.1 were used as the training data to learn analysis dictionaries. Specifically, the training samples employed to learn the analysis dictionary $Ω$ were the logarithmic transforms of 20000 patches that were extracted randomly from these training images. The size of the

---

1 The codes of DFN and MIDAL were downloaded from [https://fadili.users.greyc.fr/software.html](https://fadili.users.greyc.fr/software.html) and [http://www.lx.it.pt/~bioucas/publications.html](http://www.lx.it.pt/~bioucas/publications.html) respectively. We thank the authors of [58] for sharing their code via email.
4.4. Experimental Results

Figure 4.1: Training images used to learn analysis dictionaries.

The dictionary size is 128 and the co-sparsity for dictionary learning was set as $l = 100$. The Analysis SimCO algorithm was performed with 2000 iterations. These parameters were set empirically to be consistent with the work in [34].

4.4.1 Experiments with Synthetic Images

Experiment Settings

Four test images: “Cameraman”, “Nîmes”, “Fields” and “Peppers” were employed, which are illustrated in Fig. 4.2. These images are commonly used to evaluate the algorithms for removing multiplicative noise [38], [11], [58]. The size of the Cameraman and Peppers images is $256 \times 256$ and the size of Nîmes and Fields is $512 \times 512$. The grey-scales of all the test images are normalized so that they are in the range $[1, 256]$. The synthetic noisy images were generated by multiplying the pixels of the original images by i.i.d. Gamma random variables (cf. equations (4.1) and (4.2)), with different parameters $L \in \{1, 4, 10\}$. The synthetic noisy images are displayed in Fig. 4.3 with the noise level increasing from top to bottom.

Figure 4.2: Test images: Cameraman, Nîmes, Fields and Peppers.
Figure 4.3: Synthetic noisy images. Top row: $L = 10$. Middle row: $L = 4$. Bottom row: $L = 1$.

**Performance Metrics**

The denoising performance is evaluated with three quantities: Peak Signal-to-Noise Ratio (PSNR), Mean Absolute-deviation Error (MAE), and the Mean Structural SIMilarity Index (MSSIM) \[101\]. The PSNR and MAE indices have been widely used for the quality assessment of multiplicative noise removal algorithms \[38\], \[11\], \[58\], due to their simplicity and clear physical meanings. For a clean image $g \in \mathbb{R}^N$, the PSNR of its denoised version $\hat{g} \in \mathbb{R}^N$ is defined as

$$\text{PSNR} = 10 \log_{10} \frac{N|\max(g) - \min(g)|^2}{\|\hat{g} - g\|_2^2} \quad \text{(in dB)}, \quad (4.41)$$

where $\max(\cdot)$ and $\min(\cdot)$ return the maximum value and the minimum value contained within their operands respectively. The MAE is given by

$$\text{MAE} = \frac{1}{N}\|\hat{g} - g\|_1. \quad (4.42)$$
As indicated by the definitions above, both PSNR and MAE can be regarded as the error-based measurements which are determined by the pixel-to-pixel differences between the denoised image and the reference image. They are useful to obtain general performance assessments on the whole image, but they consider little information about the preservation of specific features so that their evaluations are not very consistent with the perceptual quality. In contrast, the MSSIM index stresses the preservation quality of structural information and is able to reflect the human visual perception better \[101\]. The value of MSSIM ranges over the interval \([0, 1]\), with 1 indicating perfect structure similarity. The same set of parameters as originally suggested in \[101\] is employed.

The Selection of the Regularization Parameters

For our proposed algorithms, the selection of the regularization parameters, i.e. \(\lambda\) in RSN-ADL and \(\lambda_1, \lambda_2\) in MNR-ADL-SR, is critical. Taking the Cameraman image as an example, the PSNR results obtained by RSN-ADL, MNR-ADL-SR\(_1\) and MNR-ADL-SR\(_2\) using different regularization parameters are demonstrated in Fig. 4.4. The subfigures from top to bottom display the results with the noise levels \(L = 10, 4, 1\) respectively. From left to right, the subfigures show the results obtained with RSN-ADL, MNR-ADL-SR\(_1\) and MNR-ADL-SR\(_1\). From Fig. 4.4, we can see that the performance of our proposed algorithms varies with the regularization coefficients \(\lambda_1\) and \(\lambda_2\). The PSNR results of RSN-ADL improves when \(\lambda\) increases and drops for a very large \(\lambda\). The favourable \(\lambda\) for the higher noise level is larger than that in the lower noise case. The changing patterns of MNR-ADL-SR\(_1\) and MNR-ADL-SR\(_2\) are similar to each other in general. In the cases of \(L = 10\) and \(L = 4\), when \(\lambda_1\) is set as a relatively small value, the increase of \(\lambda_2\) leads to an improvement in PSNR to some point followed by a reduction in PSNR. When the value of \(\lambda_1\) is relatively large, the PSNR will decrease with the increase of \(\lambda_2\) and the decrease rate of MNR-ADL-SR\(_2\) is slower than that of MNR-ADL-SR\(_1\).

In the \(L = 1\) case, a relatively large \(\lambda_2\) does not result in a degenerated PSNR as when \(L = 10\) and \(L = 4\), especially for MNR-ADL-SR\(_2\).

In our experiments, the regularization coefficients for our proposed RSN-ADL, MNR-ADL-SR\(_1\) and MNR-ADL-SR\(_2\) algorithms were selected empirically. Likewise, the parameters of MNR-DL-TV-2 \[58\] were also determined in this way. The parameters of MIDAL \[11\] and
Figure 4.4: PSNR results for the Cameraman image with different regularization parameters. Left column: RSN-ADL. Middle column: MNR-ADL-SR\(_1\). Right column: MNR-ADL-SR\(_2\). Top row: \(L = 10\). Middle row: \(L = 4\). Bottom row: \(L = 1\).

DFN [38] for the first three test images were set as in their original papers and for the Peppers image the parameters are manually tuned to lead to the best PSNR. The parameters of the algorithms used in our experiments are summarized in Table 4.1.

**Experimental Results**

The algorithms were tested with 30 noise realizations for each case. The denoising results found in one test for the four test images are shown in Figs. 4.6-4.8, Figs. 4.9-4.11, Figs. 4.12-4.14, and Figs. 4.15-4.17 respectively. The average results over the 30 random noise realizations measured in PSNR, MAE and MSSIM are demonstrated in Fig. 4.5, where the bars illustrate the mean results and the error bars display the standard deviations. From the top row to the bottom row, the noise levels are \(L = 10, 4, 1\) respectively.

It can be seen that the RSN-ADL algorithm can preserve image details better than the baseline...
4.4. Experimental Results

Table 4.1: Parameters used in the algorithms for multiplicative noise removal.

<table>
<thead>
<tr>
<th>L</th>
<th>Algorithm</th>
<th>Cameraman</th>
<th>Nîmes</th>
<th>Fields</th>
<th>Peppers</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>MNR-ADL-SR₁</td>
<td>$\lambda_1 = 0.2, \lambda_2 = 0.1$</td>
<td>$\lambda_1 = 0.2, \lambda_2 = 0.01$</td>
<td>$\lambda_1 = 0.4, \lambda_2 = 0.01$</td>
<td>$\lambda_1 = 0.1, \lambda_2 = 0.2$</td>
</tr>
<tr>
<td></td>
<td>MNR-ADL-SR₂</td>
<td>$\lambda_1 = 0.3, \lambda_2 = 0.1$</td>
<td>$\lambda_1 = 0.2, \lambda_2 = 0.2$</td>
<td>$\lambda_1 = 0.4, \lambda_2 = 0.01$</td>
<td>$\lambda_1 = 0.4, \lambda_2 = 0.01$</td>
</tr>
<tr>
<td></td>
<td>RSN-ADL</td>
<td>$\lambda = 0.4$</td>
<td>$\lambda = 0.3$</td>
<td>$\lambda = 0.5$</td>
<td>$\lambda = 0.4$</td>
</tr>
<tr>
<td></td>
<td>MIDAL</td>
<td>$\lambda = 6.1$</td>
<td>$\lambda = 4$</td>
<td>$\lambda = 6.7$</td>
<td>$\lambda = 5.9$</td>
</tr>
<tr>
<td></td>
<td>MNR-DL-TV-2</td>
<td>$\lambda = 2.7$</td>
<td>$\lambda = 17.5$</td>
<td>$\lambda = 3.7$</td>
<td>$\lambda = 4.2$</td>
</tr>
<tr>
<td></td>
<td>DFN</td>
<td>$T = 2.1 \sqrt{\Psi_1(L)}$</td>
<td>$T = 2 \sqrt{\Psi_1(L)}$</td>
<td>$T = 2 \sqrt{\Psi_1(L)}$</td>
<td>$T = 1.8 \sqrt{\Psi_1(L)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\lambda_0 = 1.3, \lambda_1 = 10$</td>
<td>$\lambda_0 = 1.3, \lambda_1 = 10$</td>
<td>$\lambda_0 = 1.3, \lambda_1 = 10$</td>
<td>$\lambda_0 = 0.9, \lambda_1 = 5$</td>
</tr>
<tr>
<td>4</td>
<td>MNR-ADL-SR₁</td>
<td>$\lambda_1 = 0.3, \lambda_2 = 0.2$</td>
<td>$\lambda_1 = 0.4, \lambda_2 = 0.001$</td>
<td>$\lambda_1 = 0.7, \lambda_2 = 0.01$</td>
<td>$\lambda_1 = 0.2, \lambda_2 = 0.3$</td>
</tr>
<tr>
<td></td>
<td>MNR-ADL-SR₂</td>
<td>$\lambda_1 = 0.6, \lambda_2 = 0.1$</td>
<td>$\lambda_1 = 0.3, \lambda_2 = 0.4$</td>
<td>$\lambda_1 = 0.7, \lambda_2 = 0.01$</td>
<td>$\lambda_1 = 0.6, \lambda_2 = 0.01$</td>
</tr>
<tr>
<td></td>
<td>RSN-ADL</td>
<td>$\lambda = 0.7$</td>
<td>$\lambda = 0.5$</td>
<td>$\lambda = 1$</td>
<td>$\lambda = 0.7$</td>
</tr>
<tr>
<td></td>
<td>MIDAL</td>
<td>$\lambda = 4.5$</td>
<td>$\lambda = 2.7$</td>
<td>$\lambda = 4.5$</td>
<td>$\lambda = 4.1$</td>
</tr>
<tr>
<td></td>
<td>MNR-DL-TV-2</td>
<td>$\lambda = 1.2$</td>
<td>$\lambda = 13.5$</td>
<td>$\lambda = 0.7$</td>
<td>$\lambda = 2.2$</td>
</tr>
<tr>
<td></td>
<td>DFN</td>
<td>$T = 2.5 \sqrt{\Psi_1(L)}$</td>
<td>$T = 2 \sqrt{\Psi_1(L)}$</td>
<td>$T = 2 \sqrt{\Psi_1(L)}$</td>
<td>$T = 3.9 \sqrt{\Psi_1(L)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\lambda_0 = 1.8, \lambda_1 = 5.7$</td>
<td>$\lambda_0 = 1.5, \lambda_1 = 10$</td>
<td>$\lambda_0 = 1.3, \lambda_1 = 10$</td>
<td>$\lambda_0 = 1.1, \lambda_1 = 3.5$</td>
</tr>
<tr>
<td>1</td>
<td>MNR-ADL-SR₁</td>
<td>$\lambda_1 = 0.5, \lambda_2 = 0.7$</td>
<td>$\lambda_1 = 1.2, \lambda_2 = 10^{-4}$</td>
<td>$\lambda_1 = 1.8, \lambda_2 = 0.01$</td>
<td>$\lambda_1 = 1.6, \lambda_2 = 0.001$</td>
</tr>
<tr>
<td></td>
<td>MNR-ADL-SR₂</td>
<td>$\lambda_1 = 1.3, \lambda_2 = 0.2$</td>
<td>$\lambda_1 = 1.2, \lambda_2 = 10^{-4}$</td>
<td>$\lambda_1 = 1.8, \lambda_2 = 0.01$</td>
<td>$\lambda_1 = 1.6, \lambda_2 = 0.001$</td>
</tr>
<tr>
<td></td>
<td>RSN-ADL</td>
<td>$\lambda = 1.6$</td>
<td>$\lambda = 1.2$</td>
<td>$\lambda = 3.5$</td>
<td>$\lambda = 1.6$</td>
</tr>
<tr>
<td></td>
<td>MIDAL</td>
<td>$\lambda = 2.7$</td>
<td>$\lambda = 2$</td>
<td>$\lambda = 3.5$</td>
<td>$\lambda = 2.4$</td>
</tr>
<tr>
<td></td>
<td>MNR-DL-TV-2</td>
<td>$\lambda = 0.01$</td>
<td>$\lambda = 3.2$</td>
<td>$\lambda = 0.01$</td>
<td>$\lambda = 0.01$</td>
</tr>
<tr>
<td></td>
<td>DFN</td>
<td>$T = 2.5 \sqrt{\Psi_1(L)}$</td>
<td>$T = 2 \sqrt{\Psi_1(L)}$</td>
<td>$T = 2 \sqrt{\Psi_1(L)}$</td>
<td>$T = 2 \sqrt{\Psi_1(L)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\lambda_0 = 1.8, \lambda_1 = 5.7$</td>
<td>$\lambda_0 = 1.5, \lambda_1 = 10$</td>
<td>$\lambda_0 = 1.2, \lambda_1 = 10$</td>
<td>$\lambda_0 = 1, \lambda_1 = 3.5$</td>
</tr>
</tbody>
</table>

For the images Cameraman and Peppers with noise levels $L = 10$ and $L = 4$, the results of MNR-DL-TV-2 are the best in terms of the metrics. However, our proposed algorithms can
better preserve some fine textures (see the building details in the background of Cameraman in Figs. 4.6 and 4.7). For Nîmes and Fields, our proposed algorithms outperform other baseline algorithms for most cases in terms of PSNR and MAE. The DFN algorithm obtains the best MSSIM values for these two images, but some artifacts are also introduced as can be seen in Figs. 4.11 and 4.13. The denoised images obtained by MIDAL have the stair-casing effect, especially when the noise level is high (see Fig. 4.8). As can be seen in Fig. 4.5 when \( L = 1 \), our proposed algorithms obtain the best results for most cases, which indicates their superiority in removing a high level of multiplicative noise compared with the baseline algorithms.

![Figure 4.5: Denoising results in PSNR, MAE and MSSIM based on 30 noisy realizations for each case. Top row: \( L = 10 \). Middle row: \( L = 4 \). Bottom row: \( L = 1 \). (Note that legends are identical for all plots, but omitted in two figures to retain clarity.)](image)
4.4. Experimental Results

Figure 4.6: Results for Cameraman ($L = 10$). (a) MNR-ADL-SR$_1$ (25.67 dB). (b) MNR-ADL-SR$_2$ (25.52 dB). (c) RSN-ADL (25.36 dB). (d) MIDAL (25.40 dB). (e) MNR-DL-TV-2 (26.62 dB). (f) DFN (26.04 dB).

Figure 4.7: Results for Cameraman ($L = 4$). (a) MNR-ADL-SR$_1$ (23.65 dB). (b) MNR-ADL-SR$_2$ (23.51 dB). (c) RSN-ADL (23.35 dB). (d) MIDAL (23.26 dB). (e) MNR-DL-TV-2 (24.52 dB). (f) DFN (23.02 dB).
Chapter 4. Multiplicative Noise Removal Based on Sparse Analysis Model

Figure 4.8: Results for Cameraman ($L = 1$). (a) MNR-ADL-SR$_1$ (20.97 dB). (b) MNR-ADL-SR$_2$ (20.89 dB). (c) RSN-ADL (20.62 dB). (d) MIDAL (20.86 dB). (e) MNR-DL-TV-2 (19.69 dB). (f) DFN (19.44 dB).

Figure 4.9: Results for Nîmes ($L = 10$). (a) MNR-ADL-SR$_1$ (28.14 dB). (b) MNR-ADL-SR$_2$ (28.21 dB). (c) RSN-ADL (28.22 dB). (d) MIDAL (27.93 dB). (e) MNR-DL-TV-2 (28.42 dB). (f) DFN (27.73 dB).
4.4. Experimental Results

Figure 4.10: Results for Nîmes ($L = 4$). (a) MNR-ADL-SR$_1$ (26.05 dB). (b) MNR-ADL-SR$_2$ (26.20 dB). (c) RSN-ADL (26.27 dB). (d) MIDAL (25.82 dB). (e) MNR-DL-TV-2 (25.62 dB). (f) DFN (25.93 dB).

Figure 4.11: Results for Nîmes ($L = 1$). (a) MNR-ADL-SR$_1$ (23.17 dB). (b) MNR-ADL-SR$_2$ (23.65 dB). (c) RSN-ADL (23.61 dB). (d) MIDAL (23.00 dB). (e) MNR-DL-TV-2 (23.00 dB). (f) DFN (22.98 dB).
Chapter 4. Multiplicative Noise Removal Based on Sparse Analysis Model

Figure 4.12: Results for Fields ($L = 10$). (a) MNR-ADL-SR$_1$ (28.82 dB). (b) MNR-ADL-SR$_2$ (28.82 dB). (c) RSN-ADL (28.83 dB). (d) MIDAL (28.55 dB). (e) MNR-DL-TV-2 (28.49 dB). (f) DFN (28.76 dB).

Figure 4.13: Results for Fields ($L = 4$). (a) MNR-ADL-SR$_1$ (27.33 dB). (b) MNR-ADL-SR$_2$ (27.31 dB). (c) RSN-ADL (27.15 dB). (d) MIDAL (27.06 dB). (e) MNR-DL-TV-2 (26.81 dB). (f) DFN (26.93 dB).
4.4. Experimental Results

Figure 4.14: Results for Fields ($L = 1$). (a) MNR-ADL-SR$_1$ (25.15 dB). (b) MNR-ADL-SR$_2$ (25.14 dB). (c) RSN-ADL (24.19 dB). (d) MIDAL (24.41 dB). (e) MNR-DL-TV-2 (22.77 dB). (f) DFN (23.99 dB).

Figure 4.15: Results for Peppers ($L = 10$). (a) MNR-ADL-SR$_1$ (26.87 dB). (b) MNR-ADL-SR$_2$ (26.64 dB). (c) RSN-ADL (26.68 dB). (d) MIDAL (26.69 dB). (e) MNR-DL-TV-2 (27.21 dB). (f) DFN (24.96 dB).
Figure 4.16: Results for Peppers ($L = 4$). (a) MNR-ADL-SR$_1$ (24.60 dB). (b) MNR-ADLSR$_2$ (24.50 dB). (c) RSN-ADL (24.66 dB). (d) MIDAL (24.03 dB). (e) MNR-DL-TV-2 (24.82 dB). (f) DFN (23.47 dB).

Figure 4.17: Results for Peppers ($L = 1$). (a) MNR-ADL-SR$_1$ (21.27 dB). (b) MNR-ADL-SR$_2$ (21.17 dB). (c) RSN-ADL (21.36 dB). (d) MIDAL (20.53 dB). (e) MNR-DL-TV-2 (19.65 dB). (f) DFN (20.10 dB).
4.4. Experimental Results

4.4.2 Experiments with Real SAR Images

In this subsection, the algorithms are applied to removing the speckle noise in real SAR images shown in Fig. 4.18. The sizes of the images are $506 \times 506$ and $350 \times 350$ respectively. Due to the lack of reference clean images, the metrics used in the experiments for synthetic data cannot be applied to assess the despeckling performance here. For the homogeneous areas where the scene variation is supposed to be negligible, as in the regions marked with red rectangles in Fig. 4.18, the Equivalent Number of Look (ENL) is suitable for evaluating the level of smoothness [7]. For a given homogeneous region $\hat{g}_{\text{reg}}$, the ENL can be computed as

$$\text{ENL} = \frac{[E(\hat{g}_{\text{reg}})]^2}{\text{Var}(\hat{g}_{\text{reg}})},$$  \hspace{1cm} (4.43)

where $E(\hat{g}_{\text{reg}})$ and $\text{Var}(\hat{g}_{\text{reg}})$ denote the mean and the variation of the pixel values in region $\hat{g}_{\text{reg}}$. The higher ENL indicates the better smooth effect.

![Figure 4.18: Original SAR images.](https://github.com/zhangyiwei79/Opticks-SAR/tree/master/SAR%20images)

For a relatively fair comparison, the same parameters as used for Cameraman with the noise level $L = 4$ (see Table 4.1) were employed for the real SAR images. The denoised images are displayed in Figs. 4.19 and 4.20. The ENL values of the four regions as marked in Fig. 4.18 are summarized in Table 4.2.

---

2 The test SAR images were downloaded from [https://github.com/zhangyiwei79/Opticks-SAR/tree/master/SAR%20images]
Chapter 4. Multiplicative Noise Removal Based on Sparse Analysis Model

Figure 4.19: Results for the first SAR image. (a) MNR-ADL-SR$_1$. (b) MNR-ADL-SR$_2$. (c) RSN-ADL. (d) MIDAL. (e) MNR-DL-TV-2. (f) DFN.

Figure 4.20: Results for the second SAR image. (a) MNR-ADL-SR$_1$. (b) MNR-ADL-SR$_2$. (c) RSN-ADL. (d) MIDAL. (e) MNR-DL-TV-2. (f) DFN.
4.4. Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Region 1</th>
<th>Region 2</th>
<th>Region 3</th>
<th>Region 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>20</td>
<td>22</td>
<td>17</td>
<td>29</td>
</tr>
<tr>
<td>MNR-ADL-SR₁</td>
<td>5649</td>
<td>1538</td>
<td>4873</td>
<td>124100</td>
</tr>
<tr>
<td>MNR-ADL-SR₂</td>
<td>4579</td>
<td>1376</td>
<td>3595</td>
<td>43902</td>
</tr>
<tr>
<td>RSN-ADL</td>
<td>738</td>
<td>779</td>
<td>514</td>
<td>4483</td>
</tr>
<tr>
<td>MIDAL</td>
<td>12162</td>
<td>2345</td>
<td>26067</td>
<td>15516</td>
</tr>
<tr>
<td>MNR-DL-TV-2</td>
<td>758</td>
<td>688</td>
<td>559</td>
<td>1077</td>
</tr>
<tr>
<td>DFN</td>
<td>111</td>
<td>132</td>
<td>110</td>
<td>106</td>
</tr>
</tbody>
</table>

Table 4.2: ENL for the homogeneous regions in the denoised SAR images

From Figs. 4.19 and 4.20, we can see that all the algorithms are capable of reducing the speckle noise in the SAR images. However, there is still some visible speckle noise in the denoised versions obtained via MNR-DL-TV-2 and DFN (see subfigures (e) and (f) of Figs. 4.19 and 4.20). The homogeneous areas in the results of MIDAL are well-smoothed, which is also demonstrated by the large ENL values in Table 4.2, but some texture details are over-smoothed as shown in the subfigure (d) of Fig. 4.19. Our proposed MNR-ADL-SR₁ and MNR-ADL-SR₂ algorithms seem to maintain a good balance between removing noise and preserving original geometric details. Their reconstructions have clearer visual appearance and higher ENL, compared with the results of RSN-ADL, which demonstrates the effect of the smoothness regularizer. Comparing the images recovered by MNR-ADL-SR₁ and MNR-ADL-SR₂, the results are similar for the first image and MNR-ADL-SR₁ generates a better version for the second one.

4.4.3 Comparison with State-of-the-art Methods

In this subsection, we compare our proposed MNR-ADL-SR₁ method with two state-of-the-art methods, i.e. SAR-BM3D [81] and Iter-NSM [105], using the Cameraman image as an example. The SAR-BM3D method is one of the most effective multiplicative noise removal algorithms and the Iter-NSM algorithm has been shown to be able to achieve state-of-the-art performance. The parameters of SAR-BM3D and Iter-NSM were set as suggested in their original papers. The denoising results for different noise levels are presented in Figs. 4.21, 4.22, and 4.23 respectively.
Figure 4.21: Results for Cameraman ($L = 10$). (a) SAR-BM3D (26.84 dB). (b) Iter-NSM (26.94 dB). (c) MNR-ADL-SR$_1$ (25.67 dB).

Figure 4.22: Results for Cameraman ($L = 4$). (a) SAR-BM3D (25.30 dB). (b) Iter-NSM (25.08 dB). (f) MNR-ADL-SR$_1$ (23.65 dB).

Figure 4.23: Results for Cameraman ($L = 1$). (a) SAR-BM3D (22.23 dB). (b) Iter-NSM (20.71 dB). (c) MNR-ADL-SR$_1$ (20.97 dB).

When $L = 10$ and $L = 4$, the denoised results obtained with MNR-ADL-SR$_1$ is not as good as those of SAR-BM3D and Iter-NSM in terms of PSNR. In the case of $L = 1$, SAR-BM3D obtains the best PSNR and the proposed MNR-ADL-SR$_1$ outperforms Iter-NSM. As our proposed algorithms, SAR-BM3D and Iter-NSM are also based on image patches, however, they
divide similar patches into groups and denoise these groups sequentially. The superiority of these two algorithms may result from this grouping framework. In particular, this framework allows one to reconstruct each patch group using “local” information of the image patches in this group and the denoising process is adapted better to this specific group.

4.5 Summary

In this chapter, we have proposed two sparse analysis model based algorithms for removing multiplicative noise: RSN-ADL and MNR-ADL-SR. Firstly, an analysis dictionary is learned via Analysis SimCO in the dictionary learning stage. Secondly, the learned dictionary is used to recover the image in the next stage. The proposed algorithms both consider the image recovery task in the log-domain and utilize the ML estimate as the data fidelity term. Nevertheless, they are different in terms of the regularization terms used in the image reconstruction formulations. Specifically, RSN-ADL utilizes a sparsity promoting regularizer using the learned dictionary while MNR-ADL-SR extends the formulation of RSN-ADL by incorporating a smoothness regularizer defined with the pixel-wise differences of the image.

We have also presented the optimization methods for the reconstruction formulations of RSN-ADL and MNR-ADL-SR respectively. Generally, a variable splitting technique is applied and the ADMM framework is carefully adapted. Simulation results with synthetic noisy images and real SAR images demonstrate the promising performance of our proposed method, especially for a relatively high noise level.
Chapter 4. Multiplicative Noise Removal Based on Sparse Analysis Model
Chapter 5

Conclusions and Future Research

5.1 Conclusions

In this thesis, the sparse analysis model based dictionary learning and signal reconstruction problems have been investigated. The signal reconstruction work is demonstrated with an application to multiplicative noise removal. The conclusions are presented as follows.

5.1.1 Analysis Dictionary Learning

In the first part of this thesis, the analysis dictionary learning problem has been studied and two algorithms have been proposed: Analysis SimCO and Incoherent Analysis SimCO. The Analysis SimCO algorithm addresses the dictionary learning task by minimizing the representation error of the training data with two constraints. One constraint enforces the co-sparsities of the training data to address the assumption of the analysis model, and the other one ensures that each row of the learned dictionary has unit $\ell_2$-norm to avoid the scaling ambiguity problem and trivial solutions. The optimization of this model is achieved iteratively by updating the representation vectors and the dictionary in an alternative manner. In each iteration, the representation vectors are estimated by multiplying the training signals with the dictionary and applying a hard-thresholding operator to the products, and the atoms of the dictionary are updated simultaneously using a gradient descent method on manifolds where the unit $\ell_2$-norm constraints are imposed implicitly.
Chapter 5. Conclusions and Future Research

It has been found that some similar atoms may appear in the dictionaries learned by Analysis SimCO. To address this issue, an extended version of Analysis SimCO has been proposed, which is referred to as Incoherent Analysis SimCO. A coherence constraint based on the correlations between two distinct atoms is introduced to the formulation for dictionary learning. To enforce this constraint, an atom decorrelation step is introduced after the dictionary update step. In the decorrelation step, the correlated atoms are rotated so that their correlations reach the threshold given by the coherence constraint.

To demonstrate the performance of the proposed algorithm, two categories of experiments have been performed, i.e. recovering ground-truth dictionaries with synthetic data and removing additive noise in images. In the experiments with synthetic data, the performance of Analysis SimCO and Incoherent Analysis SimCO is similar to Analysis K-SVD [92] and Transform K-SVD [39], but better than LOST [88], (NL)AOL/(NA)AOL [107], and GOAL [56]. In the image denoising experiments, the proposed algorithms perform better than these baseline algorithms in most cases, especially when the noise level is relatively high. The effectiveness of the atom decorrelation step in Incoherent Analysis SimCO has also been demonstrated by the increased recovery rates and better denoising results, as compared with Analysis SimCO, and the decorrelation approaches in Analysis K-SVD [92] and Incoherent K-SVD [2]. We have also found that the denoising results obtained with the ADL algorithms are not as good as the algorithms designed specifically for denoising. More sophisticated reconstruction methods are needed to further benefit from the learned analysis dictionary.

5.1.2 Sparse Analysis Model Based Multiplicative Noise Removal

In the second part of this thesis, the RSN-ADL and MNR-ADL-SR algorithms have been proposed to address the multiplicative noise removal problem based on the sparse analysis model. The RSN-ADL algorithm considers the denoising problem in the logarithm domain and develops a formulation consisting of a data fidelity term and a regularization term. The data fidelity term is the ML estimate depending on the probabilistic distribution of the noise and the regularization term is defined based on the analysis dictionary. Specifically, the regularization term is the $\ell_1$-norm of the product of an analysis dictionary and the denoised image patches, which models the sparsity prior of the denoised images. To obtain a proper dictionary, the Analysis
SimCO algorithm is used to train an adaptive dictionary. The optimization for this formulation is addressed by a variable splitting technique and the ADMM framework. Our experiments have shown the promising performance of the proposed algorithm, especially in terms of preserving details in the denoised images. Nevertheless, it has also been found that the capability of RSN-ADL for recovering relatively smooth regions is limited.

Based on the work on RSN-ADL, the MNR-ADL-SR algorithm has been proposed, aiming to further improve the recovery quality for smooth areas. The formulation of MNR-ADL-SR is extended from that of RSN-ADL by introducing an additional smoothness regularizer. This smoothness regularizer can be regarded as an enhanced TV term with a parameter controlling the smoothness constraint defined on the pixel-wise differences. Due to the introduction of the smoothness regularizer, the optimization for this new formulation has become more challenging, as compared with that of the RSN-ADL algorithm. The ADMM framework has also been adapted to address this optimization problem with more auxiliary variables being introduced. In addition, a relaxation method is developed to update the variables related to the denoised image flexibly.

The proposed algorithms have been compared with three baseline sparsity or TV based algorithms, i.e. MIDAL [11], MNR-DL-TV2 [58] and DFN [38] via experiments with synthetic speckled images and real SAR images. It has been found that the proposed algorithms perform better than MIDAL and DFN in general. Compared with the synthesis dictionary learning based denoising algorithm, MNR-DL-TV2, the performance of RSN-ADL and MNR-ADL-SR is slightly worse in some cases, however, they can obtain better results for a relatively high noise level. From the denoised images, it has also been seen that the MNR-ADL-SR algorithm is able to preserve details as well as RSN-ADL, but recover the smooth areas much better than RSN-ADL, which demonstrates the effect of the smoothness regularizer in MNR-ADL-SR. Based on the experiments of two recently proposed MNR methods, it has been found that the proposed algorithms need to be modified further to achieve state-of-the-art performance.

5.2 Future Research

Based on the work in this thesis, several directions merit further study for future research. One direction is to develop online analysis dictionary learning algorithms. Prior work on analy-
sis dictionary learning, including our proposed Analysis SimCO algorithms, focuses on batch learning, where the processing of the entire training set is involved. However, for large training data sets, batch learning techniques can be computationally expensive in both time and memory. Moreover, in real-time applications, data arrives sequentially, which makes batch learning infeasible. Online algorithms where data samples are allowed to be processed sequentially is worth considering, since they are usually more suitable for large data sets and the real-time case. For these reasons, further study to extend the Analysis SimCO algorithms to an online setting needs to be carried out.

In the work on multiplicative noise removal, it has been shown that the coefficients of the regularization terms are critical to the denoising performance. However, these coefficients are selected empirically in our current work. The automating or adaptive selection of these parameters remains an open problem and merits detailed investigation.

In addition, it should be noted that in the proposed RSN-ADL and MNR-ADL-SR algorithms, dictionary learning and image reconstruction are carried out in two separate stages. It would be interesting to fuse dictionary learning and image reconstruction in one formulation so that they can be conducted simultaneously.

A possible direction to improve the proposed MNR algorithms is to incorporate the effective techniques employed in state-of-the-art MNR methods. For example, the grouping idea used in SAR-BM3D [81] and Iter-NSM [105] may be adapted to our proposed framework by classifying the image patches based on their smoothness and then adaptively denoising different groups.

Another possible area of future research is to investigate the applications of the sparse analysis model and analysis dictionary learning in other signal processing tasks where prior knowledge or assumption is required, for instance, blind source separation, inpainting and super-resolution.
Appendix A

Derivation of the Gradient Descent Step in MNR-ADL-SR

In this appendix, the reason that the problem (4.39) can be addressed with the gradient descent iteration described as (4.40) will be explained in detail.

Let the function $m(x, y)$ denote the pixel values of the image $M_R$ for the pixel indices $x, y \in \Omega$. Similarly, the given matrix $N_R$ can be denoted as $n(x, y)$. Using these new notations, the problem (4.39) can be written as the functional minimization problem as follows

$$\arg \min_m \int\int_{\Omega} \left\{ \lambda_2 \left( \frac{\partial m}{\partial x} \right)^2 + \left( \frac{\partial m}{\partial y} \right)^2 \right\} + \frac{\gamma^2}{2} (n - m)^2 \, dx \, dy. \quad (A.1)$$

Define

$$F(m(x, y), m_x, m_y) = \lambda_2 \left( \frac{\partial m}{\partial x} \right)^2 + \left( \frac{\partial m}{\partial y} \right)^2 \, + \frac{\gamma^2}{2} (n - m)^2 \quad (A.2)$$

$$= \lambda_2 (m_x^2 + m_y^2) + \frac{\gamma^2}{2} (n - m)^2,$$

where $m_x$ and $m_y$ represent the partial derivatives $\frac{\partial m}{\partial x}$ and $\frac{\partial m}{\partial y}$ respectively, and the problem
(A.1) can be written as

$$\arg \min_m \int_\Omega F(m(x,y), m_x, m_y) dx \, dy$$  \hspace{1cm} (A.3)$$

The Euler-Lagrange equation associated with this problem is given by [26]

$$\frac{\partial F}{\partial m} - \frac{\partial}{\partial x} \left( \frac{\partial F}{\partial m_x} \right) - \frac{\partial}{\partial y} \left( \frac{\partial F}{\partial m_y} \right) = 0.$$  \hspace{1cm} (A.4)$$

Since

$$\frac{\partial F}{\partial m} = \gamma_2 (m - n),$$  \hspace{1cm} (A.5)$$

$$\frac{\partial}{\partial x} \left( \frac{\partial F}{\partial m_x} \right) = 2\lambda_2 \frac{m_x}{\partial x}$$  \hspace{1cm} (A.6)$$

and

$$\frac{\partial}{\partial y} \left( \frac{\partial F}{\partial m_y} \right) = 2\lambda_2 \frac{m_y}{\partial y},$$  \hspace{1cm} (A.7)$$

the Euler-Lagrange equation (A.4) is equivalent to

$$\gamma_2 (m - n) - 2\lambda_2 \left( \frac{m_x}{\partial x} + \frac{m_y}{\partial y} \right) = 0,$$  \hspace{1cm} (A.8)$$

which can be addressed numerically [26]. In the \( k \)-th iteration, \( m \) is updated according to the following iteration

$$m^{(k+1)} = m^{(k)} + \tau \left[ 2\lambda_2 \left( \frac{m_x^{(k)}}{\partial x} + \frac{m_y^{(k)}}{\partial y} \right) - \gamma_2 (m^{(k)} - n) \right],$$  \hspace{1cm} (A.9)$$

where \( \tau \) denotes the step size. Alternatively, the iteration equation above can be written in matrix form given by equation (4.40) as shown in Section 4.3.2.
Appendix B

List of Publications


Bibliography


