UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Cássio Fraga Dantas

Learning Structured Dictionaries

Aprendizado de Dicionários Estruturados

Campinas

2016

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Cássio Fraga Dantas

Learning Structured Dictionaries

Aprendizado de Dicionários Estruturados

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Telecomunicações e Telemática.

Supervisor: Prof. Dr. Renato da Rocha Lopes

Este exemplar corresponde à versão final da tese defendida pelo aluno Cássio Fraga Dantas, e orientada pelo Prof. Dr. Renato da Rocha Lopes

---

Campinas

2016

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

D235L

Dantas, Cássio Fraga, 1989-
Learning structured dictionaries / Cássio Fraga Dantas. – Campinas, SP : [s.n.], 2016.

Orientador: Renato da Rocha Lopes.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento digital de sinais. 2. Aprendizado de máquina. 3. Processamento de imagem. 4. Esparsidade. I. Lopes, Renato da Rocha,1972-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Aprendizado de dicionários estruturados
Palavras-chave em inglês:
Digital signal processing
Machine learning
Image processing
Sparsity
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Renato da Rocha Lopes [Orientador]
Lisandro Lovisolo
Leonardo Tomazeli Duarte
Data de defesa: 22-09-2016
Programa de Pós-Graduação: Engenharia Elétrica

Comissão Julgadora – Dissertação de Mestrado

Candidato: Cássio Fraga Dantas RA 080978
Data de defesa: 22 de setembro de 2016

Título da dissertação: "Learning Structured Dictionaries"
"Aprendizado de Dicionários Estruturados"

Prof. Dr. Renato da Rocha Lopes (Presidente, FEEC - Unicamp)
Prof. Dr. Lisandro Lovisolo (UERJ)
Prof. Dr. Leonardo Tomazeli Duarte (FCA - Unicamp)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

To my parents: Taiz and Ruy.

Acknowledgements

Allow me to write this section in Portuguese, it would sound too artificial otherwise.

Primeiramente, agradeço imensamente a meus pais, Taiz e Ruy, pelo apoio incondicional. Ao lado deles, duas pessoas fundamentais em quem sempre pude me espelhar: meus irmãos Danilo e Laiz. Amo vocês. Obrigado a meu primo-irmão Felipe pela parceria de uma vida inteira e sua mãe, tia Tânia, que tantas horas gastou estudando conosco durante nossa infância.

Deixo aqui uma menção especial à minha companheira Thalita. É muito bom ter você ao meu lado para dividir o peso do cotidiano e as alegrias da vida. Você deixou esse caminho mais fácil e agradável de percorrer. Obrigado.

Agradeço a todos os professores que passaram pela minha vida, vocês plantaram sementes de curiosidade em mim. Agradeço especialmente ao meu orientador Renato Lopes, cada aula ou simples conversa de cafezinho foi um grande aprendizado para mim. Agradeço também aos membros da banca pelas valiosas sugestões e cuidadosas revisões deste documento. Por último, mas não menos importante, obrigado aos amigos do DSPCOM (e adjacências) por transformarem a rotina em algo prazeroso e enriquecedor. Sem dúvidas, um ambiente de trabalho e uma etapa da minha caminhada que vou lembrar com nostalgia.

"Distress not yourself if you cannot at first understand the deeper mysteries of Spaceland. By degrees they will dawn upon you"
(Edwin A. Abbott, Flatland - A Romance of Many Dimensions)

Abstract

Sparse representation of signals using learned overcomplete dictionaries has proven to be a powerful tool with applications in denoising, restoration, compression, reconstruction, and more. The modern learned dictionaries stand in opposition to classic transforms usually called analytical dictionaries (e.g DCT, wavelets). While the latter permit to capture the global structure of a signal and allow a fast implementation, the former often perform better in applications as the learning process allows them to adapt to the considered class of signals. The drawback is a much higher computational load, since learned dictionaries are a priori represented by unstructured matrices and, therefore, their use is limited to signals of relatively small size.

In this work, we introduce some degree of structure on the trained dictionary in order to obtain complexity gains on operations involving the dictionary matrix. To achieve this goal, two different approaches are introduced. First, we propose to use the Displacement Structure concept to quantify the dictionary's structure. Alternatively, we may constrain the dictionary to be a sum of Kronecker products of smaller sub-dictionaries. In addition to reduced matrix-vector multiplication costs, the obtained dictionaries also require less training data and storage space than their unstructured counterpart. Some image denoising experiments are provided to validate the proposed techniques.

Keywords: Dictionary Learning; Structured Matrices; Displacement Structure; Toeplitz-like; Hankel-like; Separable dictionaries; Kronecker product; Nuclear Norm; ADMM; Proximal algorithms; K-SVD; Image denoising.

Resumo

A representação esparsa de sinais usando dicionários sobrecompletos tem se mostrado uma ferramenta poderosa com aplicações em remoção de ruído, restauração, compressão, reconstrução e outras. Classicamente isto é feito com dicionários analíticos, como DCT ou wavelets, que permitem capturar a estrutura global de um sinal e dão lugar a implementações rápidas. Em contrapartida, pode-se utilizar um dicionário dito aprendido, que se adapta melhor à classe de sinais em questão. A desvantagem é uma carga computacional mais elevada, visto que os dicionários aprendidos são a princípio representados por matrizes não estruturadas. Isto restringe sua aplicação a sinais de tamanho relativamente reduzido.

Nesta dissertação, propomos a introdução de uma certo grau de estrutura nos dicionário aprendidos, de forma reduzir a complexidade em operações envolvendo a matriz dicionário. Duas abordagens distintas são introduzidas para atingir esse objetivo. Primeiro, propomos a utilização do conceito de *Displacement Structure* para quantificar o nível de estrutura do dicionário. Segundo, nós restringimos o aprendizado a dicionários representados como uma soma de produtos de Kronecker de sub-dicionários menores. Além do reduzido custo no produto matriz-vetor, os dicionários obtidos também requerem uma menor quantidade de amostras no treinamento e um menor espaço de armazenamento se comparado a dicionários sem estrutura. Alguns experimentos de remoção de ruído em imagem são realizados de forma a validar as técnicas propostas.

Palavras-chaves: Aprendizado de dicionários; Matrizes estruturadas; Toeplitz; Hankel; Dicionários Separáveis; Produto de Kronecker; Norma nuclear; ADMM; Algoritmos proximais; K-SVD; Processamento de imagens.

List of Figures

List of Tables

Contents

1 Introduction

Signal transforms are a ubiquitous tool in signal processing. They change the signal's basis of representation, and their relevance lies in the fact that the choice of a suitable representation domain may unveil meaningful characteristics of a signal. A transform may, for instance, provide a more compact representation for a given signal, by concentrating a significant part of its energy in few coefficients. Indeed, this concept of compaction corresponds to the signal *sparsity* on that specific basis.

Sparse signals have revealed appealing properties on many applications and attracted growing attention on the past few decades (BARANIUK *et al.*, 2010). Besides having direct impact in terms of compression, sparsity also leads to efficient denoising capabilities. Additionally, the sparsity assumption enables the solution of ill-posed inverse problems, such as under-determined linear systems. As an example, it allows the solution of under-determined source separation problems where the number of measured signals (number of sensors available) is smaller than the actual number of sources (BOFILL; ZIBULEVSKY, 2001).

The search for representation domains that simultaneously provide sparse representations to a collection of data samples can be interpreted as a *variable selection* problem (GUYON; ELISSEEFF, 2003) concerned with finding a relatively small number of most representative variables in the dataset. This framework is particularly important for improving the *interpretability* of the models, since it unveils underlying processes generating the data.

Interestingly, a given transform might be capable of sparsifying a certain class of signals, while not being as well adapted to other signal classes. Particularly, the discrete Fourier transform is efficient at approximating uniformly smooth signals. However, a large variety of signals – the ones containing discontinuities, for instance – will not result in sparse representations in the Fourier domain.

Optimizing compaction was a major driving force for the continued development of more efficient representations. During the 1970's and 1980's, a new and very appealing source of compaction was brought to light: the data itself. The focus was on a set of statistical tools developed during the first half of the century, known as the Karhunen-Loève Transform (KLT) (JAIN, 1989), or Principal Component Analysis (PCA) (JOLLIFFE, 2002). This technique fits a low-dimensional subspace to the data so as to minimize the ℓ_2 approximation error. Specifically, given the data covariance matrix $\mathbf{\Sigma}$ (either known or empirical), the KLT basis vectors are the first K eigenvectors of the eigenvalue decomposition of $\mathbf{\Sigma}$.

By the second half of the 1990's a conceptual change of a different sort was gradually taking place. The drop of the completeness and orthogonality assumptions initiated an extension of the classic *transform* concept to the *dictionary* mindset. In their seminal work (MALLAT; ZHANG, 1993), the authors proposed a novel sparse signal expansion scheme based on the selection of a small subset of functions from a general over-complete dictionary of functions, allowing for even better representation efficiency.

We now present a simple example that illustrates the pursuit of a representation basis in which the input data can be sparsely represented. Consider a set of two-dimensional points distributed on the plane as in Figure 1. The application of an orthogonal transform corresponds basically to a rotation (and/or a reflection) on the representation basis. By picking a proper rotation angle, the input data can be sparsely represented on the new basis. In this context, sparse means that each data point can be represented as a function of one single basis vector instead of a combination of both vectors.

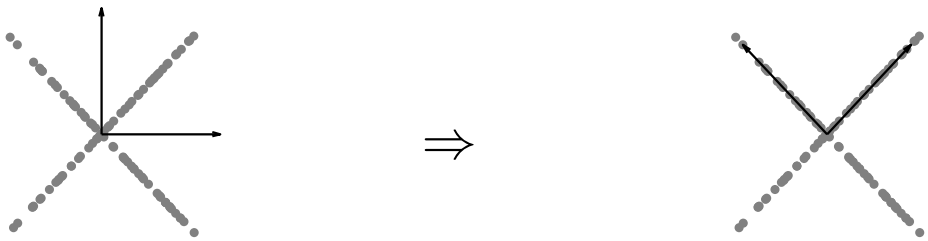

Figure 1 – By rotating the basis vectors, via an orthogonal transformation, the data becomes sparse on the new representation domain.

Note that, in some cases, it might be useful to have non-orthogonal basis vectors (Figure 2 shows an example). Furthermore, since the example is two-dimensional, only two linearly independent vectors are necessary to form a basis and, consequently, to be able to represent any point in the plane. Nevertheless, in Figure 3 we show a situation where using more than two vectors allows for sparse representation of more data: in this case, with three-element dictionary, all the observations may be described using a single coefficient.

In summary, a different representation frame (i.e. a different set of representation vectors) might be better suited for each given input dataset. There are two main strategies for picking a frame, called dictionary, that leads to a sparse representation of the data. The classic one consists on analytically deriving a construction formula, which leads to pre-defined "over-the-shelf" transforms, such as the already mentioned Fourier transform or the Discrete Cosine transform. We refer to those as analytic dictionaries. The second and more contemporary strategy is to adopt a learning procedure (TOSIC; FROSSARD, 2011), where the dictionary is constructed by a training process over the

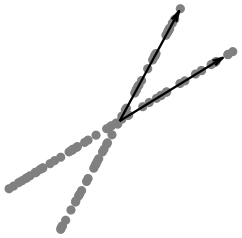

Figure 2 – Non-orthogonal frame

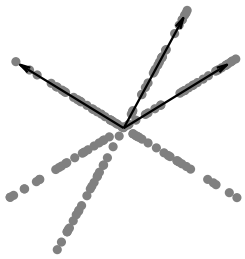

Figure 3 – Overcomplete frame

database. This strategy, called Dictionary Learning, has been a subject of increasing interest in recent years and is at the core of this thesis.

1.1 Motivation

Analytic dictionaries such as the DFT usually allow for a wide theoretical analysis of their properties as well as algorithms for fast implementations based on the operator's underlying structure (e.g the FFT algorithm). The learned dictionaries, on the other hand, are considerably more flexible as the learning process allows adapting to different classes of signals. The drawback is a considerably higher computational load, since the learned dictionaries are a priori represented by unstructured over-complete matrices.

Part of the complexity of using an over-complete dictionary originates in the determination of the sparse representation, which seeks the sparsest solution to an underdetermined linear system. This is known as the *sparse coding problem*, whose exact solution demands exponential time. The iterative algorithms designed to sub-optimally solve this problem heavily rely on matrix-vector multiplications between the dictionary matrix (or its transpose) and each data vector. For a structure-less $(n \times m)$-matrix, such operations cost $\mathcal{O}(nm)$ flops, which becomes prohibitive as the dimensionality grows.

To obtain computationally efficient dictionaries, some of the most recent works in the field employ parametric models in the training process, which produce structured dictionaries. The idea is to find a good compromise between the computational efficiency of the analytic dictionaries and the flexibility of the learned ones.

1.2 Objectives

The first goal of this master thesis is to provide a brief introduction to the emerging field of dictionary learning for sparse representations. More specifically, this thesis reviews different ways to induce some degree of structure on the trained dictionary.

Finally, we propose a novel paradigm for approaching this problem as well as specifying a suitable training algorithm. To that end, we use a concept called Displacement Structure, introduced in 1979 by (KAILATH *et al.*, 1979), which provides a method for measuring the degree of structure of a matrix. Integrating this concept to the dictionary learning domain has not been previously proposed on the literature and constitutes the core of this project.

We also propose a different structure class where the dictionary consists in a sum of Kronecker products. This structure is unprecedented in the literature and can be seen as a generalization of an existing and widespread structure: the separable dictionaries.

The main motivation for introducing structure is the complexity reduction on operations involving the dictionary matrix. Therefore, a large part of the analysis will rely on complexity comparisons between different types of structure. The performance of such dictionaries – in the form of its signal representation capabilities – will also be evaluated. Intuitively, a certain adaptivity loss is expected as the degree of structure increases, given the increasing restrictions on the matrix content. This complexity-performance trade-off is an important indicator that will be further assessed in this document.

Notation

Throughout this document, matrices are represented by bold uppercase letters (e.g. $\mathbf{X}$); vectors by bold lowercase letters (e.g. $\mathbf{x}$). Then, $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ is a matrix with columns given by vectors $\mathbf{x}_1, \dots, \mathbf{x}_N$. The element on the i-th row and j-th column of a matrix $\mathbf{X}$ is denoted $x_{i,j}$. The elements of a vector are also referenced using standard-type letters but with a single index, e.g. $\mathbf{x} = [x_1, \dots, x_n]^T$. We denote vertical concatenation by a semicolon, e.g $\mathbf{x}_2 = [\mathbf{x}; \mathbf{x}]$ where the result $\mathbf{x}_2$ has twice the dimension of $\mathbf{x}$.

The l_0 pseudo-norm (loosely called l_0-norm) which counts the total number of non-zero elements in a vector is denoted by $\|\cdot\|_0$. The l_p-norm for any real number $p \geq 1$ is denoted by $\|\cdot\|_p$ and defined as follows for an input vector $\mathbf{x} \in \mathbb{R}^n$:

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

The Frobenius matrix norm is denoted by $\|\cdot\|_F$ and is defined for an input matrix $\mathbf{X} \in \mathbb{C}^{n \times m}$:

$$\|\mathbf{X}\|_F := \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |x_{ij}|^2} \;=\; \sqrt{\mathrm{Tr}\left(\mathbf{X}^H \mathbf{X}\right)}$$

where $\mathbf{X}^H$ denotes the conjugate transpose of the matrix $\mathbf{X}$.

The nuclear norm, denoted $\|\cdot\|_*$, is defined as the sum of the singular values of a matrix. For an input matrix $\mathbf{X} \in \mathbb{C}^{n\times m}$, it gives:

$$\|\mathbf{X}\|_* := \sum_{i=1}^{\min\{m,n\}} \sigma_i \;=\; \mathrm{Tr}\left(\sqrt{\mathbf{X}^H\mathbf{X}}\right)$$

where singular values are denoted by σ_i.

We denote by $\mathbf{J}$ the so-called *reflection matrix*, which is a square matrix containing ones on its anti-diagonal and zeros elsewhere. When applied to a vector, it will reverse the ordering of the vector elements. When multiplied to another matrix, say $\mathbf{X}$, it will reverse the row ordering of $\mathbf{X}$ if multiplied by the left, or reverse the column ordering if multiplied by the right.

Part I

Fundamentals

2 Dictionary Learning

In the context of sparse representations, a problem that has attracted increasing attention is the choice of a dictionary to efficiently sparsify the training dataset. In order to properly formulate the representation problem, let n be the dimension of each data sample and N the number of samples. Let $\mathbf{Y} \in \mathbb{R}^{n \times N}$ be a training dataset matrix containing the N training samples arranged in its columns and $\mathbf{D} \in \mathbb{R}^{n \times m}$ (with $m \geq n$) be a – possibly over-complete – dictionary. The sparse representation problem is then to find an $(m \times N)$ sparse matrix X such that

$$\mathbf{Y} \approx \mathbf{DX} \quad , \tag{2.1}$$

where each data sample in $\mathbf{Y}$ is approximated as a linear combination of only a few columns of $\mathbf{D}$, referred to as *atoms*. This matrix formulation is illustrated in Figure 4.

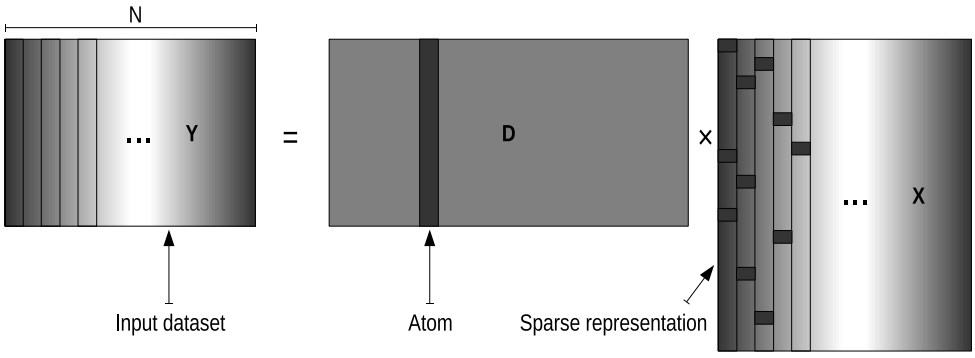

Figure 4 – Dictionary learning problem: matrix formulation.

The first dictionaries to be used were the existing transforms – such as the Fourier, wavelet, STFT, and Gabor transforms, see e.g., (MALLAT; ZHANG, 1993), (CHEN *et al.*, 1998). The dictionaries of this sort are characterized by an analytic formulation, and an important advantage of this approach is that the resulting dictionary usually features a fast implementation which does not involve explicit multiplication by the dictionary matrix. On the other hand, the dictionary can only be as successful as its underlying model, and indeed, these models tend to be over-simplistic compared to the complexity of natural phenomena. For example, not all signals can be sparsely approximated as a sum of sinusoids, particularly the ones containing discontinuities. In such cases, a Fourier dictionary would not lead to satisfactory results.

Data-driven techniques, on the other hand, can be used to learn dictionaries that are better adapted to specific instances of the data, replacing the use of generic models. As with machine learning techniques (BISHOP, 2006), the idea is that the structure

of complex natural phenomena can be more accurately extracted from the data itself than by using a prior mathematical description.

This new paradigm gives rise to the modern dictionary learning problem that goes beyond the classic sparse representation problem by jointly estimating the coefficient matrix $\mathbf{X}$ and the dictionary $\mathbf{D}$ with the goal to minimize the representation error, leading to the following optimization problem

$$\langle \mathbf{D}, \mathbf{X} \rangle = \underset{\mathbf{D},\mathbf{X}}{\operatorname{argmin}} \|\mathbf{DX} - \mathbf{Y}\|_F^2 \quad \text{subject to} \quad \forall i \ \|\mathbf{x}_i\|_0 \leq t \ , \quad \forall j \ \|\mathbf{d}_j\|_2 = 1, \quad , \tag{2.2}$$

where the ℓ_0 pseudo norm, which counts the number of non-zero elements of a vector, has been used as a sparsity inducing function. The dictionary columns are usually restricted to unit Euclidean norm in order to avoid a scale ambiguity problem.

Since there are two variables to simultaneously estimate from the data, most of the existing training methods adopt an alternating optimization strategy with two steps (RUBINSTEIN *et al.*, 2010a)

Sparse coding: For a fixed dictionary ($\mathbf{D}$), find the best sparse approximation ($\mathbf{X}$) for the input data. This is the previously mentioned sparse representation problem. Two different but closely related formulations can be used. The first one, called the t-sparse problem, is a constrained optimization problem of the form

$$\min_{\mathbf{X}} \quad \|\mathbf{DX} - \mathbf{Y}\|_F^2 \quad \text{s.t.} \quad \forall i \ \|\mathbf{x}_i\|_0 \leq t \quad , \tag{2.3}$$

where the number of non-zero coefficients is constrained to be below a certain value. The other formulation is an ℓ_0-regularized optimization problem defined as follows

$$\min_{\mathbf{X}} \quad \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \eta \sum_i \|\mathbf{x}_i\|_0 \quad . \tag{2.4}$$

The two problems are related in that, for each sample $\mathbf{y}_i$, there exists η and t such that the solution $\mathbf{x}_i$ of the ℓ_0-regularized problem is the same as that to the t-sparse (BLUMENSATH; DAVIES, 2008). The higher the η, the smaller the corresponding t.

In both cases, the resulting optimization problem is combinatorial and highly non-convex, and its optimal solution would require exponential time. However, several sub-optimal algorithms have been proposed in the literature, following two main directions:

1. Greedy approaches such as the Matching Pursuit (MALLAT; ZHANG, 1993) and the Orthogonal Matching Pursuit (PATI *et al.*, 1993).

2. Convex relaxation approaches that replace the ℓ_0 by the ℓ_1-norm. For example, FISTA (BECK; TEBOULLE, 2009) and iterative thresholding (DAUBECHIES *et al.*, 2004) algorithms.

In this thesis, we use existing sparse-coding algorithms. Hence, this particular step will not be further developed on the rest of this document.

Dictionary update: For a fixed sparse representation matrix ($\mathbf{X}$), optimize the dictionary atoms for better data approximation:

$$\min_{\mathbf{D}} \|\mathbf{DX} - \mathbf{Y}\|_F^2 \quad . \tag{2.5}$$

Each of the several existing Dictionary learning algorithms proposes a different approach to solve this problem. Some of these techniques will be revised in Sections 2.1 and 2.2.

The alternating optimization process does not necessarily find the global optimum solution of the considered problem (TOSIC; FROSSARD, 2011). Its high non-convexity will generally lead to local minima or even saddle points solutions.

2.1 Unconstrained Dictionary Learning

The following dictionary learning algorithms employ the mentioned alternating minimization strategy. The sparse coding step can be performed by any standard technique and is not detailed here.

2.1.1 Method of Optimal Directions (MOD)

The Method of Optimal Directions (MOD) was introduced by Engan *et al.* (1999), and was one of the first proposed methods for dictionary learning. The dictionary update step uses the well-known analytic solution for the linear least squares (LS) problem

$$\mathbf{D} = \mathbf{YX}^{\dagger} \quad , \tag{2.6}$$

where † denotes the Moore-Penrose pseudo-inverse ($\mathbf{X}^{\dagger} = \mathbf{X}^T(\mathbf{XX}^T)^{-1}$).

Despite its conceptual simplicity, the method suffers from the relatively high complexity of the required matrix inversion. Several subsequent works have thus focused on reducing this complexity.

2.1.2 K-SVD

In 2006, Aharon, Elad and Bruckstein introduced the K-SVD algorithm (AHARON *et al.*, 2006). The main contribution of the K-SVD is that the dictionary update, rather

than using a matrix inversion, is performed atom-by-atom in a simple and efficient process. Further acceleration is provided by updating both the current atom and its associated sparse coefficients simultaneously. The K-SVD algorithm takes its name from the Singular-Value-Decomposition (SVD) process that forms the core of the atom update step.

The dictionary optimization step updates one column (atom) at a time, fixing all columns in except one, $\mathbf{d}_k$, and finding a new column $\mathbf{d}_k$ and new values for its corresponding coefficients in the matrix $\mathbf{X}$ that best reduce the mean squared error. This is markedly different from all previous methods, which freeze $\mathbf{X}$ while finding a better $\mathbf{D}$.

Fixing $\mathbf{X}$ and all columns of $\mathbf{D}$ except $\mathbf{d}_k$, and denoting $\mathbf{x}_T^k$ the k-th row of $\mathbf{X}$, which contains the coefficients related to the atom $\mathbf{d}_k$, the penalty term in the objective function (2.5) can be rewritten as:

$$\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 = \left\|\mathbf{Y} - \sum_{j=1}^{m} \mathbf{d}_j \mathbf{x}_T^j\right\|_F^2 = \left\|\left(\mathbf{Y} - \sum_{j\neq k} \mathbf{d}_j \mathbf{x}_T^j\right) - \mathbf{d}_k \mathbf{x}_T^k\right\|_F^2 = \|\mathbf{E}_k - \mathbf{d}_k \mathbf{x}_T^k\|_F^2 \quad . \tag{2.7}$$

where the product $\mathbf{DX}$ has been decomposed as the sum of m rank-1 matrices. All terms but the one associated with the k-th atom are grouped and $\mathbf{E}_k$ stands for the error for each of the N samples when the k-th atom is removed.

The minimization of (2.7) for $\mathbf{d}_k$ and $\mathbf{x}_T^k$ corresponds to a rank-1 approximation of $\mathbf{E}_k$, which is optimally solved by the SVD[1]. To avoid introduction of new non-zeros in $\mathbf{X}$, the update process is performed using only the examples whose current representations use the atom $\mathbf{d}_k$. Currently, the K-SVD turns out to be the most widespread dictionary learning technique.

The learning techniques mentioned so far lead to non-structured dictionaries which are relatively costly to apply; therefore these methods remain limited for signals of relatively small size. Thus, the most recent contributions to the field employ parametric models in the training process, which produce structured dictionaries.

2.2 Structured Dictionaries

Although still in its early stages, some research has focused on imposing a certain degree of structure to the learned dictionary. The idea is to preserve the best of both scenarios: the computational efficiency of the analytic dictionaries and the flexibility of the learned ones. Clearly, any kind of structure imposition corresponds to a restriction

[1] This result is known as the Eckart-Young theorem (ECKART; YOUNG, 1936).

on the search space, naturally leading to less adapted dictionaries. The challenge is ultimately to better handle this compromise, providing a higher computational reduction by paying as little performance penalty as possible.

Countless types of dictionary structures may be imagined. Learning a dictionary as a union of orthonormal bases was proposed in (LESAGE *et al.*, 2005). This type of structure allows efficient sparse-coding via block coordinate relaxation (BCR). However, this model turns out to be overly restrictive. Another proposition is the *signature dictionary* (AHARON; ELAD, 2008), in which the dictionary is described by a compact image called signature. Each of its $\sqrt{n} \times \sqrt{n}$ sub-blocks constitute an atom of the dictionary. The reduced number of parameters (only one per atom) also makes this model more restrictive than other structures.

More recently, in (MAGOAROU; GRIBONVAL, 2015) the authors impose the dictionary to be the product of several sparse matrices. The total complexity in this case is determined by the sum of non-zero values on the factor matrices. Still on the sparsity line, (RUBINSTEIN *et al.*, 2010b) proposes to learn a dictionary in the form $\mathbf{D} = \mathbf{\Phi A}$, where $\mathbf{\Phi}$ is a pre-specified *base dictionary* that has a fast implementation (e.g. DCT) and $\mathbf{A}$ is a column sparse matrix. In other words, the atoms in this dictionary are sparse linear combinations of atoms from the *base dictionary*. In (SULAM *et al.*, 2016) this idea is taken further, replacing the fixed base dictionary by an adaptable multi-scale one (*cropped* Wavelets are used). This brings more flexibility to the model, while keeping its scalability. All these techniques obtain promising complexity-performance compromises, but their approach is intrinsically different from the one proposed in this project.

A straightforward approach for complexity reduction is introducing a low-rank restriction on the dictionary. This approach has recently been applied for face recognition (MA *et al.*, 2012; LI *et al.*, 2013; ZHANG *et al.*, 2013; ZHENG *et al.*, 2016). Alternatively, an algorithm for training separable dictionaries, which are formed as the Kronecker product of several sub-dictionaries, has been proposed in (HAWE *et al.*, 2013). In this case, the direct and transpose operators can be obtained directly from the sub-dictionaries, which have much smaller dimensions, thus leading to considerable complexity savings. As one contribution of this thesis, we propose a structure class where the dictionary consists in a sum of Kronecker products. This structure has the aforementioned separable dictionaries as a special case.

The main contribution of this thesis is the use of the Displacement Structure theory, introduced in (KAILATH *et al.*, 1979), to impose some structure to the dictionary. This theory will be reviewed in Section 3, but it essentially provides a measure of the structure level in a given matrix. Unfortunately, this measure is difficult to use for optimization, since it is based on the rank of a certain matrix. To circumvent this problem, we propose a relaxation of the displacement operator, based on the nuclear norm

(RECHT *et al.*, 2010). As we will show, the resulting optimization problem is tractable, and leads to dictionaries with good performance-complexity tradeoff. To the best of our knowledge, this is the first time that the Displacement Structure was used for a design task.

3 Displacement Structure

An $(n \times m)$-matrix is said to be structured when its entries have a formulaic relationship that allows the matrix to be specified by a number of latent parameters significantly lesser than $n \times m$. The number of such latent parameters is inversely proportional to the degree of structure. Generally, such matrices allow for low-complexity implementations of matrix-vector products (GOHBERG; OLSHEVSKY, 1994). Certain families of structured matrices are commonly used in the literature, for instance: Toeplitz, Hankel, Cauchy and Vandermonde. Table 1 describes succinctly these four families. The parameter vectors denoted $\mathbf{t}, \mathbf{h}, \mathbf{v}$ and $\mathbf{s}$ contain all the information necessary to completely determine these matrices content.

Table 1 – Four widespread families of structured matrices

Toeplitz	$\mathbf{T}(i,j) = t_{i-j}$
Hankel	$\mathbf{H}(i,j) = h_{i+j}$
Vandermonde	$\mathbf{V}(i,j) = v_i^{j-1}$
Cauchy	$\mathbf{C}(i,j) = \frac{1}{s_i - t_j}$

General matrices, on the other hand, may not be as neatly structured, but still may have some structure. According to the displacement structure theory, this can be measured by an appropriate linear operator called *displacement operator*. If a matrix $\mathbf{M}$ is structured, this operator should turn $\mathbf{M}$ into a low-rank matrix.

Definition 1. *For fixed matrices* $\mathbf{A}$ *and* $\mathbf{B}$*, the* Stein-type displacement operator*, denoted* $\Delta_{\mathbf{A},\mathbf{B}}$ *is defined by*

$$\Delta_{\mathbf{A},\mathbf{B}}(\mathbf{M}) = \mathbf{M} - \mathbf{AMB} \quad . \tag{3.1}$$

The matrices $\mathbf{A}$ and $\mathbf{B}$ are known as the *displacings* or *operator matrices*, and they determine the type of structure that the displacement operator will identify. The image $\Delta_{\mathbf{A},\mathbf{B}}(\mathbf{M})$ is called *displacement*.

Definition 2. *The* displacement rank *of* $\mathbf{M}$ *with respect to the operator* $\Delta_{\mathbf{A},\mathbf{B}}$ *is the rank of the displacement matrix* $\Delta_{\mathbf{A},\mathbf{B}}(\mathbf{M})$.

Let us introduce the family of unit φ-variant matrices $\mathbb{Z}_{n,\varphi} \in \mathbb{R}^{n\times n}$, where $\varphi \in \mathbb{R}$:

$$\mathbf{Z}_{n,\varphi} = \begin{bmatrix} 0 & 0 & \dots & \varphi \\ 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 1 & 0 \end{bmatrix} . \tag{3.2}$$

For $\varphi = 0$, we obtain the lower-shift matrix $\mathbf{Z}_{n,0}$.

Since Toeplitz and Hankel matrices consist on repetitions of a set of coefficients along the diagonals and anti-diagonals respectively, it is possible to obtain a low-rank matrix by subtracting the given matrix by a shifted version of it. Consider, for instance, a general $(n \times n)$-Toeplitz matrix:

$$\mathbf{T} = \begin{bmatrix} t_0 & t_{-1} & \dots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \dots & t_1 & t_0 \end{bmatrix} . \tag{3.3}$$

Then, the following holds:

$$\Delta_{\mathbf{Z_{n,0}},\mathbf{Z_{n,0}^T}}(\mathbf{T}) = \begin{bmatrix} t_0 & t_{-1} & \dots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \dots & t_1 & t_0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & t_0 & \dots & t_{1-n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & t_{n-1} & t_1 & t_0 \end{bmatrix} = \begin{bmatrix} t_0 & t_{-1} & \dots & t_{1-n} \\ t_1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ t_{n-1} & 0 & \dots & 0 \end{bmatrix} . \tag{3.4}$$

The same result holds for rectangular Toeplitz matrices. A rectangular $(n\times m)$-Toeplitz matrix, with $m > n$, is defined as a cropped version of the square $(m\times m)$-Toeplitz matrix. Figure 5 illustrates the displacement operation for rectangular matrices using a color map.

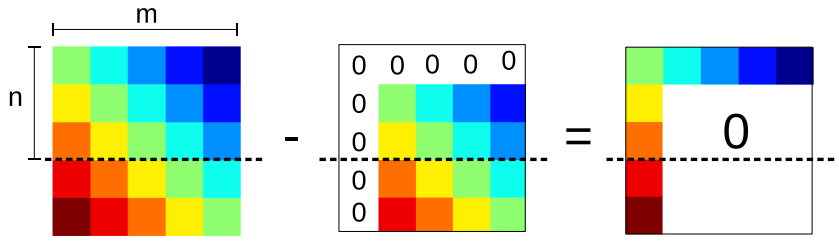

Figure 5 – Displacement operation on a rectangular Toeplitz matrix, with $m = 5$ and $n = 3$.

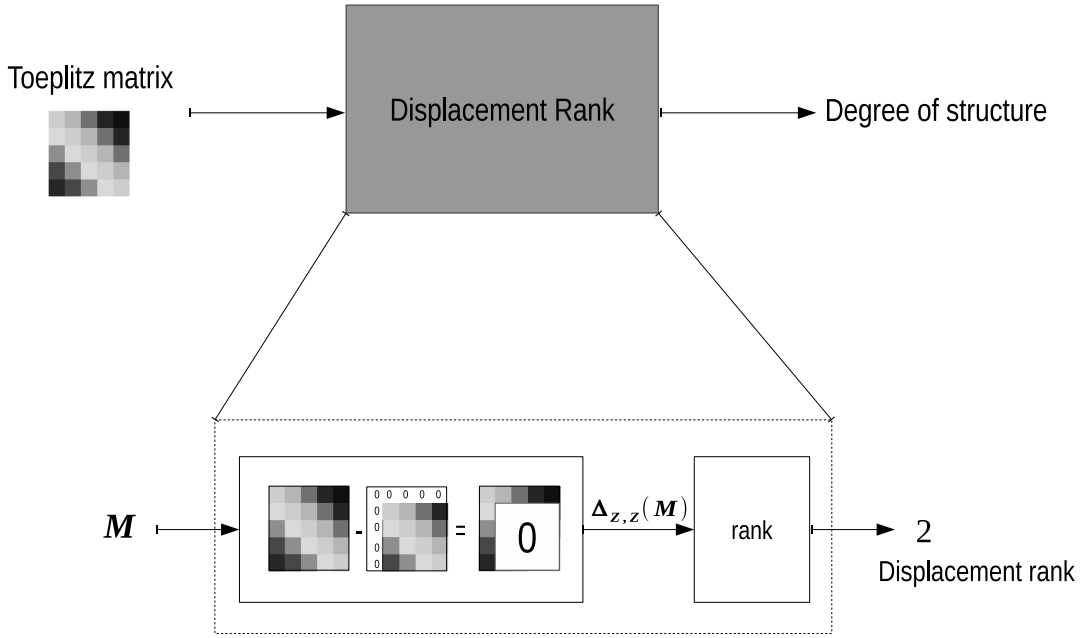

Figure 6 – Block diagram of the displacement structure approach in the Toeplitz case.

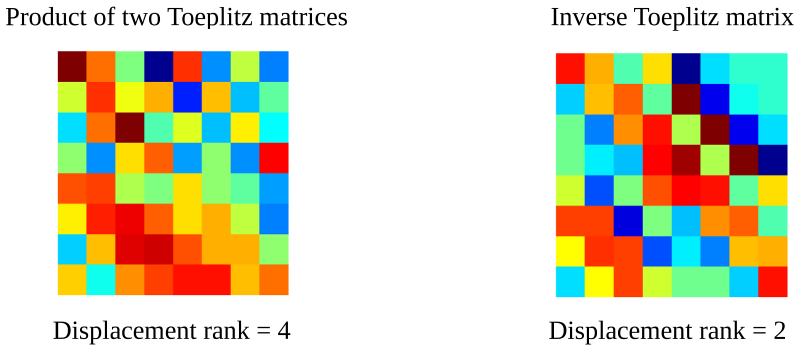

Figure 7 – Toeplitz-like matrices also lead to low displacement ranks.

We have seen that, with respect to the operator $\Delta_{\mathbf{Z_{n,0}},\mathbf{Z^T_{m,0}}}$, a $(n\times m)$-Toeplitz matrix has a displacement rank at most equal to 2, no matter the values of n and m. It can be verified that the same result holds for any operator of the form $\Delta_{\mathbf{Z_{n,\varphi}},\mathbf{Z^T_{m,\psi}}}$ or $\Delta_{\mathbf{Z^T_{n,\varphi}},\mathbf{Z_{m,\psi}}}$ for any $\varphi, \psi \in \mathbb{R}$. Figure 6 summarizes the displacement structure methodology for the Toeplitz case.

Likewise, a Hankel matrix, which is a mirrored version of a Toeplitz matrix, has a displacement rank at most equal to 2 for any operators of the form $\Delta_{\mathbf{Z_{n,\varphi}},\mathbf{Z_{m,\psi}}}$ or $\Delta_{\mathbf{Z^T_{n,\varphi}},\mathbf{Z^T_{m,\psi}}}$ for any $\varphi, \psi \in \mathbb{R}$.

However, even if the matrix does not exactly fit in one of the mentioned families, the displacement approach is still capable of identifying a degree of structure. For example, a product of several Toeplitz matrices – and even some inverse Toeplitz – will still lead to a low displacement rank, although showing no visually remarkable structure, as shown in Figure 7. We call such matrices Toeplitz-like.

As a general rule, the lowest the displacement rank, the more structured is the input matrix with respect that specific displacement operator.

For Vandermonde-like and Cauchy-like matrices we will use diagonal matrices $\mathcal{D}(\mathbf{x})$ (containing the vector $\mathbf{x}$ on its diagonal and zeros elsewhere) as an operator matrix. Table 2, extracted from (MOUILLERON, 2011) relates the operator matrices to the structure families they allow to identify:

Table 2 – Structure families and associated operator matrices

Structure type	**A**	**B**
Toeplitz-like	$Z_{n,\varphi}$	$Z_{m,\psi}^T$
	$Z_{n,\varphi}^T$	$Z_{m,\psi}$
Hankel-like	$Z_{n,\varphi}$	$Z_{m,\psi}$
	$Z_{n,\varphi}^T$	$Z_{m,\psi}^T$
Vandermonde-like	$\mathcal{D}(\mathbf{v})$	$Z_{m,\psi}^T$
Vandermonde-transpose-like	$Z_{m,\psi}$	$\mathcal{D}(\mathbf{v})$
Cauchy-like	$\mathcal{D}(\mathbf{s})$	$\mathcal{D}(\mathbf{t})$

In this thesis we will focus on Toeplitz-like and Hankel-like structures only. The reason for not using the other families is that they demand the choice of extra parameters which are the vectors in the diagonal matrices. Arbitrarily fixing these vectors beforehand would represent a too strong restriction on the matrix form.

Part II

Contributions

4 Displacement-Structured Dictionaries

In this section, we show how the displacement rank can be used to obtain a structured dictionary that can be employed with low computational complexity.

As in the the classic Dictionary Learning problem, we seek a sparse representation $\mathbf{X} \in \mathbb{R}^{m \times N}$ of the data matrix $\mathbf{Y} \in \mathbb{R}^{n \times N}$ on an learned over-complete basis $\mathbf{D} \in \mathbb{R}^{n \times m}$. As seen before, mathematically speaking this task corresponds to the problem in Equation (2.2). It is the third term in equation (4.1) that carries our contribution. We introduce an extra regularization term that enforces the minimization of the dictionary's displacement rank:

$$\min_{\mathbf{D},\mathbf{X}} \quad \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \eta \sum_{i=1}^{N} \|\mathbf{x}_i\|_0 + \lambda \operatorname{rank}(\mathbf{D} - \mathbf{ADB}) \quad , \tag{4.1}$$

where the parameter $\lambda \in \mathbb{R}^+$ controls the rank penalty.

Following the literature (e.g. (ENGAN *et al.*, 1999; AHARON *et al.*, 2006; SADEGHI *et al.*, 2014; MAIRAL *et al.*, 2009)), a two-step alternating optimization technique is adopted to solve this problem. This strategy, as explained in Chapter 2, alternates minimizations on the variables $\mathbf{D}$ (*dictionary update*) and $\mathbf{X}$ (*sparse coding*).

On the sparse coding step, any existing algorithm could be used. In this thesis we use the OMP algorithm (PATI *et al.*, 1993), which is a suboptimal greedy approach to solve the sparse approximation problem.

On the dictionary update step, since the rank term is not convex, we replace it by the nuclear norm, which is defined as the sum of the singular values of a matrix and is denoted $\|\cdot\|_*$. The nuclear norm is a convex relaxation of the rank and, as explained in (RECHT *et al.*, 2010), can be regarded as the matrix-space analogous of the l_1-norm relaxation for inducing sparsity on a vector (the rank function being analogous to the non-convex l_0-norm). This results in the following optimization problem:

$$\text{Dictionary update:} \quad \min_{\mathbf{D}} \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \lambda \|\mathbf{D} - \mathbf{ADB}\|_* \quad . \tag{4.2}$$

Although convex, the nuclear norm function is not differentiable (RECHT *et al.*, 2010). For this reason, the solution of the above optimization problem is not straightforward. One cannot, for instance, apply a basic gradient descent method. We now propose a method to circumvent this problem. The proposed method is an adaptation of existing optimization techniques for nuclear-norm regularized problems (MA *et al.*, 2012; LI *et al.*, 2013) in which the nuclear norm acts directly on the optimization variable. In our case, on the other hand, the nuclear norm is applied to a function of the optimization variable, namely $\mathbf{D} - \mathbf{ADB}$.

4.1 Optimization Framework

In this section we concentrate on solving the optimization problem on the dictionary update step. Although there are numerous works on the literature involving low-rank minimization via nuclear norm regularization (MA *et al.*, 2012; LI *et al.*, 2013; ZHANG *et al.*, 2013; ZHENG *et al.*, 2016; TOH; YUN, 2010), to the best of our knowledge, there is no previous work proposing an optimization framework for the displacement rank minimization. Due to the non-differentiability of the nuclear norm, we will appeal to a proximal algorithm (PARIKH *et al.*, 2014) associated with the alternating method of multipliers (ADMM) (BOYD *et al.*, 2011) to solve this problem. Let us first introduce some useful concepts.

Definition 3. *The* proximal operator *(PARIKH* et al.*, 2014) associated with a convex function h is*

$$\text{prox}_{\lambda h}(x) = \underset{u}{\text{argmin}} \left(h(u) + \frac{1}{2\lambda}\|u - x\|_2^2 \right) \quad . \tag{4.3}$$

Some proximal operators are well known, including the singular value soft-thresholding operation (CAI *et al.*, 2010), which is associated with the nuclear norm

$$h(\mathbf{X}) = \|\mathbf{X}\|_* \;\Rightarrow\; \text{prox}_{\lambda\|\cdot\|_*}(\mathbf{X}) = \mathbf{U}\,\text{shrink}(\mathbf{\Sigma}, \lambda)\mathbf{V}^\dagger \quad , \tag{4.4}$$

where λ is any non-negative real constant, $\mathbf{U\Sigma V}^\dagger$ is the singular value decomposition of the matrix $\mathbf{X}$ and $\text{shrink}(\mathbf{\Sigma}, \lambda)$ is the soft-thresholding or shrinkage operator on each element of the diagonal singular value matrix $\mathbf{\Sigma}$ defined as

$$\text{shrink}(x, \lambda) = \text{sign}(x)(|x| - \lambda) \quad . \tag{4.5}$$

4.1.1 Augmented Lagrangian and Alternating Direction Method of Multipliers

We now show how the proximal operator can be applied in the context of the optimization problem (4.2). The nuclear norm proximal operator, as presented in Equation (4.4), is only defined when the nuclear norm applies directly to the optimization variable. To meet this condition, we reformulate (4.2) as the following equality-constrained problem

$$\begin{aligned} \min_{\mathbf{D},\mathbf{\Delta}} \quad & \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \lambda\|\mathbf{\Delta}\|_* \\ \text{s.t.} \quad & \mathbf{\Delta} = \mathbf{D} - \mathbf{ADB} \qquad . \end{aligned} \tag{4.6}$$

The introduction of the auxiliary variable $\mathbf{\Delta}$ decouples the two terms in the cost function. Each of them now applies exclusively to one of the optimization variables $\mathbf{D}$

or $\mathbf{\Delta}$. This new formulation may seem more complicated than the original unconstrained one, but it is preferable when each of the decoupled problems can be easily solved. This is the case here, since the first term on the cost function is differentiable, and hence can be optimized via a simple gradient method. The nuclear norm term, which now applies directly to an optimization variable, although not differentiable, has a known proximal operator.

The equality constraint in problem (4.6) can be handled iteratively with the augmented Lagrangian approach [1] (BERTSEKAS, 1982). The problem's augmented Lagrangian function consists of the classic Lagrangian with an additional quadratic term that penalizes the non-fulfillment of the equality constraint (the so-called augmentation):

$$\begin{aligned}\mathcal{L}(\mathbf{D}, \mathbf{\Delta}, \mathbf{\Lambda}) =& \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \lambda\|\mathbf{\Delta}\|_* \\ &- \mathrm{Tr}(\mathbf{\Lambda}^T(\mathbf{\Delta} - \mathbf{D} + \mathbf{ADB})) + \frac{\mu}{2}\|\mathbf{\Delta} - \mathbf{D} + \mathbf{ADB}\|_F^2 \quad ,\end{aligned} \tag{4.7}$$

where $\mathbf{\Lambda}$ is a matrix of Lagrangian multipliers related to the equality constraint and μ is a positive penalization coefficient.

For a sufficiently large μ, the solution of the original problem coincides with finding a saddle point of the Lagrangian $\mathcal{L}(\mathbf{D}, \mathbf{\Delta}, \mathbf{\Lambda})$ (NOCEDAL; WRIGHT, 2006). This can be done by the iterative Augmented Lagrange Method (ALM) shown in Algorithm 4.1. As soon as the variation on the Lagrangian multiplier becomes sufficiently small, the algorithm stops.

Algorithm 4.1 Augmented Lagrange method (ALM)

Initialize $\mathbf{D}$, $\mathbf{\Delta}$, $\mathbf{\Lambda}$
while stopping criterion is not met **do**
  $(\mathbf{D}_{k+1}, \mathbf{\Delta}_{k+1}) = \mathrm{argmin}_{\mathbf{D},\mathbf{\Delta}}\, \mathcal{L}(\mathbf{D}, \mathbf{\Delta}, \mathbf{\Lambda}_k)$
  $\mathbf{\Lambda}_{k+1} = \mathbf{\Lambda}_k - \mu(\mathbf{\Delta}_{k+1} - \mathbf{D}_{k+1} + \mathbf{AD}_{k+1}\mathbf{B})$
end while

The minimization of the augmented Lagrangian for a fixed μ and $\mathbf{\Lambda}$ that appears on the ALM loop can be shown to be equivalent to the unconstrained joint minimization problem

$$\min_{\mathbf{D},\mathbf{\Delta}} \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \lambda\|\mathbf{\Delta}\|_* + \frac{\mu}{2}\left\|\mathbf{\Delta} - \mathbf{D} + \mathbf{ADB} - \frac{\mathbf{\Lambda}}{\mu}\right\|_F^2 \quad . \tag{4.8}$$

Once again, we can resort to an alternate minimization strategy over the variables $\mathbf{D}$ and $\mathbf{\Delta}$. This idea is exploited by the alternating direction method of multipliers

[1] The equality constrained problem (4.6) cannot be analytically solved with the classic method of Lagrange multipliers since the Lagrangian is not differentiable. For this reason we call upon an iterative penalty method which adds a quadratic penalization term.

(ADMM) (BOYD *et al.*, 2011) which, instead of alternating between $\mathbf{D}$ and $\mathbf{\Delta}$ until convergence, performs a single iteration over each variable before updating the multiplier matrix $\mathbf{\Lambda}$. It can be seen as a variant of the augmented Lagrangian scheme that uses partial updates of the involved variables.

The resulting ADMM approach is described in Algorithm 4.2, where we denote $\mathbf{Z} = \mathbf{\Lambda}/\mu$ for simplicity.

Algorithm 4.2 Alternating direction method of multipliers (ADMM)

Initialize $\mathbf{D}$, $\mathbf{\Delta}$, $\mathbf{Z}$
while stopping criterion is not met **do**
$\mathbf{D}_{k+1} = \text{argmin}_{\mathbf{D}} \|\mathbf{DX}-\mathbf{Y}\|_F^2 + \frac{\mu}{2}\|\mathbf{\Delta}_k - \mathbf{D} + \mathbf{ADB} - \mathbf{Z}_k\|_F^2$
$\mathbf{\Delta}_{k+1} = \text{argmin}_{\mathbf{\Delta}} \lambda\|\mathbf{\Delta}\|_* + \frac{\mu}{2}\|\mathbf{\Delta} - \mathbf{D}_{k+1} + \mathbf{AD}_{k+1}\mathbf{B} - \mathbf{Z}_k\|_F^2$
$\mathbf{Z}_{k+1} = \mathbf{Z}_k + (\mathbf{D}_{k+1} - \mathbf{AD}_{k+1}\mathbf{B} - \mathbf{\Delta}_{k+1})$
end while

Although in Algorithm 4.2 we denote the updates in $\mathbf{D}$ and $\mathbf{\Delta}$ as the solution of a corresponding minimization problem, we have seen that only a partial solution is required. Since the first minimization problem in the ADMM algorithm can be approached by a regular gradient descent, a single gradient step can be performed instead of iterating until convergence. Denoting

$$\mathrm{J}_1(\mathbf{D}) = \|\mathbf{DX}-\mathbf{Y}\|_F^2 \qquad \text{and} \qquad \mathrm{J}_2(\mathbf{D}) = \frac{1}{2}\|\mathbf{\Delta}_k - \mathbf{D} + \mathbf{ADB} - \mathbf{Z}_k\|_F^2 \quad , \tag{4.9}$$

the partial update with respect to $\mathbf{D}$ becomes

$$\mathbf{D}_{k+1} = \mathbf{D}_k - \gamma\,(\,\nabla \mathrm{J}_1(\mathbf{D}_k) + \mu \nabla \mathrm{J}_2(\mathbf{D}_k)\,) \quad , \tag{4.10}$$

where γ is the stepsize and the gradients (derived in Appendix B) are given by:

$$\begin{aligned} \nabla \mathrm{J}_1(\mathbf{D}_k) &= 2(\mathbf{D}_k\mathbf{X} - \mathbf{Y})\mathbf{X}^T \\ \nabla \mathrm{J}_2(\mathbf{D}_k) &= \mathbf{D}_k - \mathbf{AD}_k\mathbf{B} - \mathbf{\Delta}_k + \mathbf{Z}_k \\ &\quad - \mathbf{A}^T(\mathbf{D}_k - \mathbf{AD}_k\mathbf{B} - \mathbf{\Delta}_k + \mathbf{Z}_k)\mathbf{B}^T \quad . \end{aligned} \tag{4.11}$$

Regarding the second minimization problem in the ADMM algorithm, the exact solution, by definition of the proximal operator, is given by:

$$\mathbf{\Delta}_{k+1} = \text{prox}_{\frac{\lambda}{\mu}\|\cdot\|_*}(\mathbf{D}_{k+1} - \mathbf{AD}_{k+1}\mathbf{B} + \mathbf{Z}_k) \quad . \tag{4.12}$$

4.1.2 Ensuring column normalization

The described optimization approach, despite effectively promoting the reduction of the displacement rank, will generally not lead to a column-normalized dictionary. Therefore, a column normalization must be performed as a post-processing step.

Although it seems harmless, this operation does not preserve the displacement rank. In short, it will revert the displacement rank minimization previously performed. To solve this issue, one should view the final dictionary as a product of a low displacement rank matrix with a diagonal normalization matrix $\mathbf{N}$:

$$\mathbf{D} = \mathbf{D}_{\text{structured}}\mathbf{N} \quad . \tag{4.13}$$

This means that a regular column normalization is performed at the final stage, so that only the dictionary content prior to the normalization, $\mathbf{D}_{\text{structured}}$, is kept in the optimization procedure.

Both terms should be stored separately in order to allow fast implementations based on the dictionary structure, since it is the matrix $\mathbf{D}_{\text{structured}}$ that has a low displacement rank and can benefit from the fast implementations derived in Chapter 6. This entails an additional step when multiplying the dictionary by a vector, which is the product between the diagonal matrix $\mathbf{N}$ and the vector itself. In terms of complexity, this corresponds to an overhead of only m operations.

4.2 Proposed Algorithm

Algorithm 4.3 shows a high-level view of the alternating minimization strategy adopted for learning both the sparse representation matrix and a low displacement rank dictionary. An initial dictionary $\mathbf{D}_{\text{init}}$ must be set – usually, a simple analytic dictionary is chosen, such as the DCT for image processing applications.

Algorithm 4.3 Training algorithm overview

Input: Data matrix $\mathbf{Y} \in \mathbb{R}^{n \times N}$
Operator matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{m \times m}$
Output: Dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$
Sparse representation matrix $\mathbf{X} \in \mathbb{R}^{m \times N}$

$\mathbf{D}_0 = \mathbf{D}_{\text{init}}$
for $j \leftarrow 0$ to $N_{iter} - 1$ **do**
  ▷ Sparse coding via OMP
  $\mathbf{X}_{j+1} = \text{argmin}_{\mathbf{X}} \quad \|\mathbf{D}_j\mathbf{X} - \mathbf{Y}\|_F^2 + \eta \sum_i \|\mathbf{x}_i\|_0$
  ▷ Dictionary update via Algorithm 4.4
  $\mathbf{D}_{\text{structured},j+1} = \text{argmin}_{\mathbf{D}} \, \|\mathbf{D}\mathbf{X}_{j+1} - \mathbf{Y}\|_F^2 + \lambda\|\mathbf{D} - \mathbf{A}\mathbf{D}\mathbf{B}\|_*$
  ▷ Normalize dictionary columns as in Eq. 4.13
  $\mathbf{D}_{j+1} = \mathbf{D}_{\text{structured},j+1}\mathbf{N}$
end for
return $\mathbf{D}_{N_{iter}}$, $\mathbf{X}_{N_{iter}}$

The resulting dictionary update step is described in Algorithm 4.4, where we denote γ the gradient step-size and *tol* the convergence tolerance. The initialization of

the matrix $\mathbf{Z}$ may be random. Good convergence behaviour has been observed by setting, on the first execution of the dictionary update (i.e. $j=0$ on Alg. 4.3), $\mathbf{Z}$ as the all zeros matrix.

Algorithm 4.4 Dictionary Update

Input: Data matrix $\mathbf{Y} \in \mathbb{R}^{n\times N}$
Sparse representation matrix $\mathbf{X} \in \mathbb{R}^{m\times N}$
Current dictionary $\mathbf{D}_{\text{current}} \in \mathbb{R}^{n\times m}$
Converge tolerance (tol) and step-size (γ)
Output: Updated dictionary $\mathbf{D}_{\text{updated}} \in \mathbb{R}^{n\times m}$

Initialize $\mathbf{D}_0 = \mathbf{D}_{\text{current}}$, $\mathbf{Z}_0$
while not *converged* **do**
  $\mathbf{D}_{k+1} = \mathbf{D}_k - \gamma(\nabla_{\mathbf{D}} J_1 + \mu \nabla_{\mathbf{D}} J_2)$ ▷ by using Eq. (4.11)
  $\mathbf{\Delta}_{k+1} = \text{prox}_{\frac{\lambda}{\mu}\|.\|_*}(\mathbf{D}_{k+1} - \mathbf{A}\mathbf{D}_{k+1}\mathbf{B} + \mathbf{Z}_k)$
  $\mathbf{Z}_{k+1} = \mathbf{Z}_k + (\mathbf{D}_{k+1} - \mathbf{A}\mathbf{D}_{k+1}\mathbf{B} - \mathbf{\Delta}_{k+1})$
  ▷ Stopping criterion
  if $\|\mathbf{Z}_{k+1} - \mathbf{Z}_k\|_F^2 < tol$ **then**
    $converged = true$
  end if
end while
return $\mathbf{D}_{\text{updated}} = \mathbf{D}_{k+1}$

4.3 Expected benefits

A structured dictionary under the displacement rank criterion provides computational cost reductions in different aspects of their manipulation:

- *Storage cost:* Only the low-rank transformed version of the dictionary $\mathbf{\Delta}_{\mathbf{A},\mathbf{B}}(\mathbf{D})$ needs to be stored, since all operations with the dictionary can be derived from it. For a $(n\times m)$-dictionary with displacement rank α, the matrix $\mathbf{\Delta}_{\mathbf{A},\mathbf{B}}(\mathbf{D})$ which rank is α can be represented as the product of two smaller matrices of sizes $(m\times\alpha)$ and $(\alpha\times n)$. Therefore, the total storage cost will be $\alpha(m+n)$ instead of mn.

- *Multiplication cost:* As will be shown in Chapter 6, the multiplication of a displacement structured dictionary by a vector can be performed in $\mathcal{O}((\alpha n+m)\log n)$ compared to $\mathcal{O}(mn)$ for unstructured dictionaries.

- *Sample complexity:* A dictionary with displacement structure can be determined by smaller set of parameters than its actual number of elements. Combining this with the fact that the dictionary is estimated from a training database, leads to an advantage in terms of statistical significance. For a given amount of training data, there are fewer parameters to estimate. In other words, the sample complexity

is reduced (GRIBONVAL *et al.*, 2015). Better estimation accuracy is expected or, alternatively, less training data is required for the same accuracy when compared to the unstructured counterpart.

5 Dictionaries as Sums of Kronecker Products

The displacement structure framework presented in Chapter 4 is only one of the many possible types of structure for a matrix – in our specific case, the dictionary matrix. An alternative structure class arises from the concept of Kronecker product (here denoted by $\otimes$).

The Kronecker product of two matrices $\mathbf{B} \in \mathbb{R}^{n_1 \times m_1}$ and $\mathbf{C} \in \mathbb{R}^{n_2 \times m_2}$ is an $(n_1 n_2 \times m_1 m_2)$-matrix given by

$$\mathbf{B} \otimes \mathbf{C} = \begin{bmatrix} b_{1,1}\mathbf{C} & b_{1,2}\mathbf{C} & \dots & b_{1,m_1}\mathbf{C} \\ b_{2,1}\mathbf{C} & b_{2,2}\mathbf{C} & \dots & b_{2,m_1}\mathbf{C} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n_1,1}\mathbf{C} & b_{n_1,2}\mathbf{C} & \dots & {}_{n_1,m_1}\mathbf{C} \end{bmatrix} . \tag{5.1}$$

The resulting matrix can be considered structured in the sense that all its $n_1 \times n_2 \times m_1 \times m_2$ elements are completely determined by a smaller set of parameters, which are actually the elements of the matrices $\mathbf{B}$ and $\mathbf{C}$, with a total of $n_1 \times m_1 + n_2 \times m_2$.

The Kronecker product is closely related to multi-dimensional signal processing, in the sense that it generates multi-dimensional operators from one-dimensional ones (MALLAT, 2008). In the case of two-dimensional signals, such as images, given two one-dimensional operator matrices $\mathbf{B}$ and $\mathbf{C}$, then $\mathbf{B} \otimes \mathbf{C}$ is a two-dimensional operator where each of the composing operators acts separately in one signal dimension. In other words, as will be seen in Chapter 6, applying the composite operator $\mathbf{B} \otimes \mathbf{C}$ in a vectorized version of an image is equivalent to applying the operator $\mathbf{B}$ in each of the image rows and the operator $\mathbf{C}$ in each of the image columns (see Equation (6.6)). Such operators are commonly called *separable operators*. A widespread example is the 2D-DCT, which is defined as a composition (via Kronecker product) of two 1D-DCTs.

The separable structure is an appealing possibility for dictionaries, specially for applications involving multi-dimensional signals. This particular structure is explored in (HAWE *et al.*, 2013), where the authors propose an algorithm for learning a dictionary that can be represented as the Kronecker product of two sub-dictionaries, i.e. $\mathbf{D} = \mathbf{B} \otimes \mathbf{C}$.

In this chapter, we propose a broader structure class from which the separable structure is a special case. It consists in a sum of α separable dictionaries, where the number of components, α, serves as a fine tuner for the complexity-adaptability tradeoff:

$$\mathbf{D} = \sum_{r=1}^{\alpha} \mathbf{B}^{(r)} \otimes \mathbf{C}^{(r)} \quad . \tag{5.2}$$

To design the dictionary, we use a mathematical result (LOAN; PITSIANIS, 1993) that establishes a relation between a separable matrix and a rank-1 matrix. This result has not been previously used for separable dictionary design. As is well-known and as discussed in Section 5.2, optimization problems involving a rank constraint are hard to solve, but good solutions can be obtained by relaxing this constraint using the nuclear norm (RECHT *et al.*, 2010). The proposed technique is hereby named SuKro (Sum of Kronekers).

With respect to the Separable Dictionary Learning (SeDiL) technique proposed in (HAWE *et al.*, 2013) for learning separable dictionaries, the algorithm proposed in the following, besides presenting a more general structure form, also introduces a different formulation and optimization strategy.

5.1 Theoretical Background

We begin by introducing a useful result that provides a way of transforming a Kronecker product into a rank-1 matrix (LOAN; PITSIANIS, 1993). Consider a matrix $\mathbf{D} \in \mathbb{R}^{n_1 n_2 \times m_1 m_2}$ which is the Kronecker product of two sub-matrices $\mathbf{B} \in \mathbb{R}^{n_1 \times m_1}$ and $\mathbf{C} \in \mathbb{R}^{n_2 \times m_2}$. In other words,

$$\mathbf{D} = \mathbf{B} \otimes \mathbf{C} \quad . \tag{5.3}$$

We define a *rearrangement operator*, denoted $\mathcal{R}(\cdot)$, that reorganizes the elements $d_{i,j}$ of $\mathbf{D}$ in such a way that the elements $\tilde{d}_{i,j}$ of the rearranged matrix $\mathcal{R}(\mathbf{D}) \in \mathbb{R}^{m_1 n_1 \times m_2 n_2}$ are given by

$$\begin{aligned} &\tilde{d}_{i_1+(j_1-1)n_1, i_2+(j_2-1)n_2} = d_{i_2+(i_1-1)n_2, j_2+(j_1-1)m_2} \\ &\text{with} \quad \begin{cases} i_1 \in \{1, 2, \dots, n_1\}, & j_1 \in \{1, 2, \dots, m_1\} \\ i_2 \in \{1, 2, \dots, n_2\}, & j_2 \in \{1, 2, \dots, m_2\} \end{cases} . \end{aligned} \tag{5.4}$$

As shown in (LOAN; PITSIANIS, 1993), this rearrangement maps a matrix $\mathbf{D}$ of the form in (5.3) into a rank-1 matrix, which can be written as an outer product of the vectorized versions of $\mathbf{B}$ and $\mathbf{C}$:

$$\mathcal{R}(\mathbf{D}) = \operatorname{vec}(\mathbf{B}) \operatorname{vec}(\mathbf{C})^{\mathrm{T}} \ . \tag{5.5}$$

Example. *We illustrate this result with a simple example, with* $n_1 = n_2 = m_1 = m_2 = 2$. *Consider the* (2×2) *matrices*

$$\mathbf{B} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}. \tag{5.6}$$

The resulting Kronecker product is given by

$$\mathbf{D} = \mathbf{B} \otimes \mathbf{C} = \left[\begin{array}{c:c} b_{1,1}\mathbf{C} & b_{1,2}\mathbf{C} \\ \hdashline b_{2,1}\mathbf{C} & b_{2,2}\mathbf{C} \end{array}\right] = \left[\begin{array}{cc:cc} 1a & 1c & 3a & 3c \\ 1b & 1d & 3b & 3d \\ \hdashline 2a & 2c & 4a & 4c \\ 2b & 2d & 4b & 4d \end{array}\right] \tag{5.7}$$

and its rearranged version becomes

$$\mathcal{R}(\mathbf{D}) = \begin{bmatrix} 1a & 1b & 1c & 1d \\ 2a & 2b & 2c & 2d \\ 3a & 3b & 3c & 3d \\ 4a & 4b & 4c & 4d \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \begin{bmatrix} a & b & c & d \end{bmatrix}, \tag{5.8}$$

which is a rank-1 matrix.

Now, let us consider a sum of α Kronecker products

$$\mathbf{D} = \sum_{r=1}^{\alpha} \mathbf{B}^{(r)} \otimes \mathbf{C}^{(r)} = \sum_{r=1}^{\alpha} \mathbf{D}^{(r)} \ . \tag{5.9}$$

After rearrangement, we obtain a rank-α matrix, since each term $\mathbf{D}^{(r)}$ leads to a rank-1 matrix [1]. In other words,

$$\mathcal{R}(\mathbf{D}) = \sum_{r=1}^{\alpha} \mathcal{R}(\mathbf{D}^{(r)}) = \sum_{r=1}^{\alpha} \mathrm{vec}\Big(\mathbf{B}^{(r)}\Big)\, \mathrm{vec}\Big(\mathbf{C}^{(r)}\Big)^{\mathrm{T}} . \tag{5.10}$$

Therefore, using (5.10), we can introduce a low-rank regularization term to the original optimization problem in order to learn a dictionary as a sum of Kronecker products:

$$\begin{aligned} &\min_{\mathbf{D},\mathbf{X}} \ \|\mathbf{D}\mathbf{X} - \mathbf{Y}\|_F^2 + \lambda \,\mathrm{rank}(\mathcal{R}(\mathbf{D})) \\ &\text{s.t.} \quad \forall i \ \|\mathbf{x}_i\|_0 \leq t \ , \quad \forall j \ \|\mathbf{d}_j\|_2 = 1 \ \ , \end{aligned} \tag{5.11}$$

where the parameter $\lambda \in \mathbb{R}^+$ controls the rank penalty.

Note that we do not explicitly impose the structure defined in Equation (5.9). Instead, we try to limit the rank of the rearranged matrix $\mathcal{R}(\mathbf{D})$ through a rank penalization on the cost function. This strategy relies on the assumption that the dictionaries obtained by solving the minimization problem (5.11) will have a low-rank rearranged matrix $\mathcal{R}(\mathbf{D})$. This assumption is empirically confirmed with the optimization algorithm

[1] We assume here that the vectors $\mathrm{vec}\big(\mathbf{B}^{(r)}\big)$ are linearly independent (and the same for the vectors $\mathrm{vec}\big(\mathbf{C}^{(r)}\big)$). Otherwise, the summation in equation 5.10 would lead to matrix of rank smaller than α.

presented in Section 5.2. Therefore, even though the structure (5.9) is not explicitly imposed via an equality constraint, we still converge to solutions that actually have the proposed structure.

Interestingly, despite coming from a completely different premise, the obtained optimization problem is closely related to the one obtained in Chapter 4. Both contain an additional regularization term on the rank of a transformed version of the dictionary. This allows us to employ some of the optimization tools already detailed in Chapter 4 and, for this reason, we provide a briefer description of the optimization procedure that follows the same structure of Chapter 4.

5.2 Optimization framework

As done in Chapter 4, we solve the problem in (5.11) by alternately minimizing on the variables $\mathbf{D}$ and $\mathbf{X}$, as typically done in the literature (ENGAN *et al.*, 1999; AHARON *et al.*, 2006; MAIRAL *et al.*, 2009). The minimization on $\mathbf{X}$ is called the *sparse coding* step and the minimization on $\mathbf{D}$ is the *dictionary update* step. We use the existing Orthogonal Matching Pursuit (OMP) algorithm (PATI *et al.*, 1993) for sub-optimally solving the NP-hard *sparse coding* problem.

The dictionary update step, in its turn, has been modified by the addition of the rank regularization term. Given the non-convexity of the rank operator, we use the nuclear norm (denoted $\|\cdot\|_*$) as its convex relaxation (RECHT *et al.*, 2010), which yields

$$\text{Dict. update:} \quad \min_{\mathbf{D}} \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \lambda \|\mathcal{R}(\mathbf{D})\|_* \quad . \tag{5.12}$$

The above problem cannot be addressed by a regular gradient descent, since the nuclear norm operator is not differentiable. However, the following variable introduction turns it into an approachable equality constrained problem:

$$\begin{aligned} &\min_{\mathbf{D},\widetilde{\mathbf{D}}} \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \lambda \|\widetilde{\mathbf{D}}\|_* \\ &\text{s.t.} \quad \widetilde{\mathbf{D}} = \mathcal{R}(\mathbf{D}) \quad . \end{aligned} \tag{5.13}$$

As before, the motivation for introducing a new variable in (5.13) is that, in this new problem, the nuclear norm applies directly to an optimization variable, which enables the use of the nuclear norm proximal operator as defined in Equation (4.4).

The Augmented Lagrangian Multipliers (ALM) method can be employed to solve such problem (BERTSEKAS, 1982). It replaces the constrained optimization problem (5.13) by a series of unconstrained problems of the form

$$\min_{\mathbf{D},\widetilde{\mathbf{D}}} \|\mathbf{DX} - \mathbf{Y}\|_F^2 + \lambda \|\widetilde{\mathbf{D}}\|_* + \frac{\mu}{2} \left\| \widetilde{\mathbf{D}} - \mathcal{R}(\mathbf{D}) - \mathbf{Z} \right\|_F^2 \quad , \tag{5.14}$$

where $\mu\mathbf{Z}$ is the Lagrangian multiplier matrix and μ is a positive scalar controlling the quadratic penalization on the non-fulfilment of the equality constraint. This unconstrained problem is repeatedly solved as the the Lagrangian multiplier gets updated, until convergence.

More precisely, we use a variant of the standard Augmented Lagrangian Method known as the Alternating Direction Method of Multipliers (ADMM) (BOYD *et al.*, 2011) that performs partial updates on the minimization variables $\mathbf{D}$ and $\widetilde{\mathbf{D}}$ before updating the Lagrangian multiplier, as shown in Algorithm 5.1.

Algorithm 5.1 Alternating direction method of multipliers

Initialize $\mathbf{D}_0$, $\widetilde{\mathbf{D}}_0$, $\mathbf{Z}_0$
while stopping criterion is not met **do**
$\mathbf{D}_{k+1} = \text{argmin}_{\mathbf{D}} \|\mathbf{DX}-\mathbf{Y}\|_F^2 + \frac{\mu}{2}\left\|\widetilde{\mathbf{D}}_k - \mathcal{R}(\mathbf{D}) - \mathbf{Z}_k\right\|_F^2$
$\widetilde{\mathbf{D}}_{k+1} = \text{argmin}_{\widetilde{\mathbf{D}}} \lambda\|\widetilde{\mathbf{D}}\|_* + \frac{\mu}{2}\left\|\widetilde{\mathbf{D}} - \mathcal{R}(\mathbf{D}_{k+1}) - \mathbf{Z}_k\right\|_F^2$
$\mathbf{Z}_{k+1} = \mathbf{Z}_k - \left(\widetilde{\mathbf{D}}_{k+1} - \mathcal{R}(\mathbf{D}_{k+1})\right)$
end while

In Algorithm 5.1 we loosely denote the updates in $\mathbf{D}$ and $\widetilde{\mathbf{D}}$ as the solution of a corresponding minimization problem (first and second steps on the ADMM loop), but we have seen that only a partial solution is required.

The partial update with respect to the variable $\mathbf{D}$ (first step in Alg. 5.1) corresponds to a single gradient step

$$\mathbf{D}_{k+1} = \mathbf{D}_k - \gamma\nabla \mathrm{J}(\mathbf{D}_k) \quad , \tag{5.15}$$

where $\mathrm{J}(\mathbf{D}_k) = \|\mathbf{D}_k\mathbf{X}-\mathbf{Y}\|_F^2 + \frac{\mu}{2}\left\|\widetilde{\mathbf{D}}_k - \mathcal{R}(\mathbf{D_k}) - \mathbf{Z}_k\right\|_F^2$ and γ is the stepsize.

In order to calculate $\nabla \mathrm{J}(\mathbf{D}_k)$ we use the fact that the Frobenius norm is indifferent to the elements ordering on a matrix. So, by applying the inverse of the rearrangement operation $\mathcal{R}$ denoted $\mathcal{R}^{-1}$, the second term in J can be rewritten in an equivalent way as $\left\|\mathcal{R}^{-1}(\widetilde{\mathbf{D}}_k) - \mathbf{D}_k - \mathcal{R}^{-1}(\mathbf{Z}_k)\right\|_F^2$.

The gradient is therefore given by:

$$\nabla \mathrm{J}(\mathbf{D}_k) = 2(\mathbf{D}_k\mathbf{X}-\mathbf{Y})\mathbf{X}^T + \mu\Big(\mathbf{D}_k - \mathcal{R}^{-1}(\widetilde{\mathbf{D}}_k - \mathbf{Z}_k)\Big) \quad . \tag{5.16}$$

The partial update with respect to the variable $\widetilde{\mathbf{D}}$ (second step in Alg. 5.1) is the proximal operator associated with the nuclear norm (denoted $\text{prox}_{\frac{\lambda}{\mu}\|.\|_*}$). It consists on the singular value soft-thresholding operation, see (CAI *et al.*, 2010) for details.

$$\widetilde{\mathbf{D}}_{k+1} = \text{prox}_{\frac{\lambda}{\mu}\|.\|_*}\left(\mathcal{R}(\mathbf{D}_{k+1}) + \mathbf{Z}_k\right) \quad . \tag{5.17}$$

The variation on the multiplier matrix $\mathbf{Z}$ was used as a convergence criterion:

$$\|\mathbf{Z}_{k+1} - \mathbf{Z}_k\|_F < tol \quad .$$

5.3 Proposed Algorithm

The resulting dictionary update step is described in Algorithm 5.2. The initialization of $\mathbf{D}_0$, $\widetilde{\mathbf{D}}_0$ and $\mathbf{Z}_0$ may be random, the algorithm consistently converged in all tested cases. As we mentioned before, the algorithm does not require the dictionary structure to be imposed beforehand. Instead, the structure is gradually induced during the optimization process.

Algorithm 5.2 Dictionary Update

Input: Data matrix $\mathbf{Y} \in \mathbb{R}^{n \times N}$
Sparse representation matrix $\mathbf{X} \in \mathbb{R}^{m \times N}$
Output: Dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$

Initialize $\mathbf{D}_0$, $\widetilde{\mathbf{D}}_0$, $\mathbf{Z}_0$, tol, γ
while not *converged* **do**
$\mathbf{D}_{k+1} = \mathbf{D}_k - \gamma \left[2(\mathbf{D}_k\mathbf{X} - \mathbf{Y})\mathbf{X}^T + \mu\Big(\mathbf{D}_k - \mathcal{R}^{-1}(\widetilde{\mathbf{D}}_k - \mathbf{Z}_k)\Big)\right]$
$\widetilde{\mathbf{D}}_{k+1} = \text{prox}_{\frac{\lambda}{\mu}\|\cdot\|_*}\left(\mathcal{R}(\mathbf{D}_{k+1}) + \mathbf{Z}_k\right)$
$\mathbf{Z}_{k+1} = \mathbf{Z}_k - \Big(\widetilde{\mathbf{D}}_{k+1} - \mathcal{R}(\mathbf{D}_{k+1})\Big)$
▷ Stopping criterion
if $\|\mathbf{Z}_{k+1} - \mathbf{Z}_k\|_F^2 < tol$ **then**
$converged = true$
end if
end while
return D

Algorithm 5.3 shows a high-level view of the alternating minimization strategy adopted for learning both the sparse representation matrix and a dictionary as a sum separable terms.

Algorithm 5.3 SuKro algorithm overview

Input: Data matrix $\mathbf{Y} \in \mathbb{R}^{n \times N}$
Output: Dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$
Sparse representation matrix $\mathbf{X} \in \mathbb{R}^{m \times N}$

for $j \leftarrow 0$ to $N_{iter} - 1$ **do**
▷ Sparsecoding via OMP
$\mathbf{X}_{j+1} = \text{argmin}_{\mathbf{X}} \quad \|\mathbf{D}_j\mathbf{X} - \mathbf{Y}\|_F^2 + \eta \sum_i \|\mathbf{x}_i\|_0$
▷ Dictionary update via Algorithm 5.2
$\mathbf{D}_{j+1} = \text{argmin}_{\mathbf{D}} \|\mathbf{D}\mathbf{X}_{j+1} - \mathbf{Y}\|_F^2 + \lambda\|\mathcal{R}(\mathbf{D})\|_*$
Normalize columns of $\mathbf{D}_{j+1}$
end for
return $\mathbf{D}_{N_{iter}}$, $\mathbf{X}_{N_{iter}}$

6 Complexity analysis

Complexity savings are expected when operating with structured matrices. In the following sections, we derive expressions for the computational complexity of the matrix-vector multiplication when the matrix has some type of structure. We treat in more details the structures proposed in Chapters 4 and 5 respectively in Sections 6.1 and 6.2.

6.1 Low displacement rank dictionaries

Operating with low-rank matrices leads to complexity savings because the total number of parameters required to represent such matrices is actually smaller than its total number of elements. In fact, a $(n \times m)$-matrix with rank α can be represented as a product of two smaller matrices of sizes $(n \times \alpha)$ and $(\alpha \times m)$ (MARSAGLIA; STYAN, 1974).

Now, consider a matrix $\mathbf{M}$ with displacement rank α with respect to the operator matrices $\mathbf{A}$ and $\mathbf{B}$. Then, the displacement matrix, which has rank α, admits the following low-rank representation:

$$\Delta_{\mathbf{A},\mathbf{B}}(\mathbf{M}) = \mathbf{G}\mathbf{H}^T = \sum_{k=1}^{\alpha} \mathbf{g}_k \mathbf{h}_k^T \quad , \tag{6.1}$$

where the pair $(\mathbf{G}, \mathbf{H})$, of sizes $(n \times \alpha)$ and $(m \times \alpha)$, is called a *generator of length* α.

The fact that a transformed version of the matrix is low-rank enforces the intuition that the matrix itself may benefit from complexity gains. However, it is not evident how to exploit such property. To make it possible, one needs to be able to recover the original matrix from its generators.

Extensive studies on the invertibility of the displacement operator are available on the literature (PAN; WANG, 2003). In the following we present two cases where the invertibility holds and where we have explicit recovery formulas.

1. Let $\mathbf{M}$ be a Toeplitz-like $(n \times m)$-matrix, which has a generator of length α with respect to the operator $\Delta_{\mathbf{Z}_{n,0},\mathbf{Z}_{m,0}^T}$. Then

$$\mathbf{M} = \sum_{k=1}^{\alpha} \mathbf{L}(\mathbf{g}_k)\bar{\mathbf{U}}(\mathbf{h}_k^T) \quad , \tag{6.2}$$

   where $\mathbf{L}(\mathbf{g})$ denotes a $(n \times n)$ lower-triangular Toeplitz matrix whose first column is $\mathbf{g}$, and $\bar{\mathbf{U}}(\mathbf{h}^T)$ denotes a truncated $(n \times m)$ upper-triangular Toeplitz matrix whose

first row is $\mathbf{h}^T$ – it contains the first n rows of the complete $(m \times m)$ upper-triangular matrix $\mathbf{U}(\mathbf{h}^T)$. The following example illustrates the matrices $\mathbf{L}$, $\mathbf{U}$ and $\bar{\mathbf{U}}$ for clarity.

Example. *Let $n = 3$, $m = 5$ and the vectors*

$$\boldsymbol{g} = [1, 2, 3]^T, \qquad \boldsymbol{h} = [1, 2, 3, 4, 5]^T.$$

The matrices $\mathbf{L}(\mathbf{g})$, $\mathbf{U}(\mathbf{h}^T)$ and $\bar{\mathbf{U}}(\mathbf{h}^T)$ are given by

$$\mathbf{L}(\mathbf{g}) = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}, \quad \mathbf{U}(\mathbf{h}^T) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \bar{\mathbf{U}}(\mathbf{h}^T) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 2 & 3 \end{bmatrix}.$$

The corresponding Toeplitz-like matrix $\mathbf{M}$ is

$$\mathbf{M} = \mathbf{L}(\mathbf{g})\bar{\mathbf{U}}(\mathbf{h}^T) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 8 & 11 & 14 \\ 3 & 8 & 14 & 20 & 26 \end{bmatrix}.$$

2. Let $\mathbf{M}$ be a Hankel-like $(n \times m)$-matrix, which has a generator of length α with respect to the operator $\Delta_{\mathbf{Z}_{n,0},\mathbf{Z}_{m,0}}$. Then

$$\mathbf{M} = \left(\sum_{k=1}^{\alpha} \mathbf{L}(\mathbf{g}_k) \bar{\mathbf{U}}((\mathbf{J}\mathbf{h}_k)^T) \right) \mathbf{J} \quad , \tag{6.3}$$

where $\mathbf{L}$ and $\bar{\mathbf{U}}$ are defined as before, and $\mathbf{J}$ is a reflection matrix having the appropriate size.

It is important to note that the recovery formula is not unique. Other recovery formulas are available at (PAN; WANG, 2003) and lead to similar results in terms of complexity.

We have intentionally factored out the rightmost term $\mathbf{J}$ in (6.3). On a matrix-vector multiplication, this particular term can be multiplied by the vector beforehand, and the problem can be reinterpreted as the multiplication of a mirrored version of the Hankel-like matrix $\mathbf{M}$ – which is Toeplitz-like – by a reversed version of the original vector. We can take advantage of the fact that both lower and upper triangular Toeplitz matrices have known fast implementations for matrix-vector multiplication (KAILATH; SAYED, 1999). The result can be achieved by performing a product of two polynomial or, equivalently, the convolution of two vectors, both of which can be calculated by using the FFT algorithm in sub-quadratic time $\mathcal{O}(n \log n)$ for a square $(n \times n)$ matrix.

Lower/Upper triangular Toeplitz matrix-vector product: Let $\mathbf{g}, \mathbf{v} \in \mathbb{R}^n$ and $\mathbf{h}$, $\mathbf{w} \in \mathbb{R}^m$. Introduce the polynomials

$$\begin{aligned} g(x) &= \textstyle\sum_{i=1}^{n} g_i x^{i-1} & v(x) &= \textstyle\sum_{i=1}^{n} v_i x^{i-1}, \\ \tilde{h}(x) &= \textstyle\sum_{i=1}^{m} h_{m+1-i} x^{i-1} & w(x) &= \textstyle\sum_{i=1}^{m} w_i x^{i-1} \quad . \end{aligned} \tag{6.4}$$

Then the following holds (KAILATH; SAYED, 1999):

- the ith entry of the vector $\mathbf{L}(\mathbf{g})\mathbf{v}$ is the coefficient of x^{i-1} in the polynomial product $u(x)v(x)$, or, equivalently, the ith output of the convolution $\mathbf{u} * \mathbf{v}$.
- the ith entry of the vector $\mathbf{U}(\mathbf{h}^T)\mathbf{w}$ is the coefficient of x^{m-2+i} in the polynomial product $\tilde{h}(x)w(x)$, or, equivalently, the $(m-1+i)$th output of the convolution $(\mathbf{Jh}) * \mathbf{w}$, where $\mathbf{Jh}$ is simply the vector $\mathbf{h}$ in reverse ordering. The product $\bar{\mathbf{U}}(\mathbf{h}^T)\mathbf{w}$ is merely a truncation of the previous result.

The convolution (or the polynomial product) can be performed by taking the FFT of both vectors (or coefficient vectors), performing a simple point-wise product between the transformed vectors, and then taking the inverse transform (IFFT). Based on this, the complexity of multiplying a structured $(n \times m)$-matrix $\mathbf{M}$ (or its transpose $\mathbf{M}^T$) that admits the representation (6.2) or (6.3) by an m-dimensional vector can be shown to be of the order $\mathcal{O}((\alpha n + m)\log n)$.

The arithmetic complexity of the matrix-vector operation, in total number of real multiplication and real additions, is given by

$$T_{\text{Disp}} = 4[n(2\alpha+1)+m]\log(2n) + 4m(\alpha-2) - 4n(3\alpha+2) + 6(2\alpha + \frac{m}{n} + 1) \quad . \tag{6.5}$$

Refer to Appendix A for a detailed derivation of this result.

The developed framework for matrix-vector multiplication leading to the computational complexity in Equation (6.5) is only valid for the Toeplitz-like and Hankel-like cases. Naturally, different structure types would lead to different reconstruction formulas than Equations (6.2) or (6.3) and, consequently, to different algorithms for fast implementation.

6.2 Sum of Kronecker products

When it comes to a matrix-vector multiplication, the separable structure can be exploited by using the following Kronecker product property:

$$(\mathbf{B} \otimes \mathbf{C})\mathbf{x} = \text{vec}(\mathbf{C}\,\text{unvec}(\mathbf{x})\mathbf{B}^{\mathrm{T}}) \quad . \tag{6.6}$$

The right-hand side expression contains a product of three matrices with sizes $(n_2 \times m_2)$, $(m_2 \times m_1)$ and $(m_1 \times n_1)$ respectively. If no particular structure is imposed to the sub-matrices $\mathbf{B}$ and $\mathbf{C}$, we obtain a complexity (in total number of multiplications and additions) of

$$2m_1n_2(n_1 + m_2) \quad . \tag{6.7}$$

For instance, if we assume $n_1 = n_2 = \sqrt{n}$ and $m_1 = m_2 = \sqrt{m}$ the complexity becomes:

$$2(\sqrt{m}n + m\sqrt{n}) \quad , \tag{6.8}$$

which is a considerable reduction when compared to the $2mn$ operations in the case of a unstructured matrix.

For a matrix with α separable terms in the form of eq. (5.2), the total complexity becomes:

$$2\alpha(\sqrt{m}n + m\sqrt{n}) \quad . \tag{6.9}$$

6.3 Other matrix-vector multiplication complexities – comparison

1. *Explicit Dictionaries:* If the dictionary is represented by an explicit structure-less $(n \times m)$-matrix, then its total complexity is

$$T_{Explicit} = 2mn \quad . \tag{6.10}$$

2. *Unstructured Low-rank Dictionaries:* Assuming that the dictionary can be represented by a $(n \times m)$-matrix of rank β, its complexity is

$$T_{Low-rank} = 2\beta(m + n) \quad . \tag{6.11}$$

3. *Separable Dictionaries:* A dictionary is called separable when it is the Kronecker product of two or more sub-dictionaries (MALLAT, 2008). Consider two sub-dictionaries $\mathbf{\Phi}_0, \mathbf{\Phi}_1 \in \mathbb{R}^{\sqrt{n} \times \sqrt{m}}$, we can construct a dictionary $\mathbf{\Phi} \in \mathbb{R}^{n \times m}$ in the form $\mathbf{\Phi} = \mathbf{\Phi}_0 \otimes \mathbf{\Phi}_1$. It is worth noting that the dictionary transpose is separable as well and given by $\mathbf{\Phi}^T = \mathbf{\Phi}_0^T \otimes \mathbf{\Phi}_1^T$. One such dictionary leads to efficient direct and transpose operator, with a total complexity of

$$T_{\Phi} = 2n\sqrt{m} + 2m\sqrt{n} \quad , \tag{6.12}$$

assuming $\mathbf{\Phi}_0$ and $\mathbf{\Phi}_1$ have no particular structure and are applied via explicit matrix multiplication. This result has been derived in Section 6.2 with more details. The 2D-DCT dictionary is an example of separable dictionary as well as the ones learned via the SeDiL algorithm (HAWE *et al.*, 2013).

4. *Sparse Dictionaries:* The complexity of the matrix-vector product is proportional to the number of non-zero entries in the matrix. Denoting $\text{nnz}(\mathbf{D})$ the number of non-zero entries in a sparse matrix $\mathbf{D}$, the complexity is given by

$$T_{sparse} = 2\,\text{nnz}(\mathbf{D}) \quad . \tag{6.13}$$

So, for a $(n \times m)$-matrix containing sm non-zero entries instead of the nm of a dense matrix, with $s \ll n$, the matrix-vector product complexity becomes[1] $T_{sparse} = 2sm$, and s may be seen as the average column sparsity.

Now, if the dictionary is a product of several sparse matrices, say J factors, $\mathbf{D} = \prod_{j=1}^{J} \mathbf{S}_j$ as is the case in (MAGOAROU; GRIBONVAL, 2016), the complexity becomes

$$T_{sparse-factors} = 2\sum_{j=1}^{J} \text{nnz}(\mathbf{S}_j) \quad . \tag{6.14}$$

It is also important to mention the complexity related to the Sparse K-SVD dictionaries proposed in (RUBINSTEIN *et al.*, 2010b), which are given by a product of a base dictionary and a sparse matrix, $\mathbf{D} = \mathbf{\Phi A}$. Since the adopted base dictionary is a 2-D overcomplete DCT [2], which is separable, and assuming a column sparsity p for the matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$, the matrix-vector operator complexity becomes [3]

$$T_{S-KSVD} = T_{\Phi} + 2pm \quad . \tag{6.15}$$

To conclude, we illustrate in Figure 8 how some of the mentioned structures scale as the data dimensionality grows. We have supposed an over-completeness factor of four on the dictionary (i.e. $m = 4n$), which is the same ratio m/n used on the simulations reported in Chapter 7. As we can see, the structures leading to sub-quadratic complexities for matrix-vector multiplication present much better scalability properties.

Note that if we consider an overcompleteness factor proportional to n, as is usually the case, for example $m/n = \sqrt{n}$, the advantage of our structured matrices is even more pronounced as shown in Figure 9.

---

[1] In (RUBINSTEIN *et al.*, 2010b), the authors include a multiplicative factor $\alpha = 7$ to simulate the several implementation issues related to sparse matrices operations (see (IM; YELICK, 2000) for a in-depth analysis). However, to simplify the analysis, we will not take such practical complications into account, as done in (MAGOAROU; GRIBONVAL, 2016).

[2] The 1-D $n \times m$ overcomplete DCT dictionary, as defined in (RUBINSTEIN *et al.*, 2010b), is a cropped and renormalized version of the orthogonal $m \times m$ DCT dictionary matrix. The 2-D ODCT is the Kronecker product of two 1-D ODCT dictionaries of size $\sqrt{n} \times \sqrt{m}$.

[3] Note that the complexity T_{Φ} of a general separable matrix is used. If the regular 2D-DCT was used as a base dictionary, then this complexity would be smaller since both sub-matrices $\mathbf{\Phi}_0$ and $\mathbf{\Phi}_1$ would be 1D-DCTs, which have fast implementations. Such advantage is lost when an overcomplete version of the DCT is used, since the DCT matrix is truncated and renormalized, losing its original structure.

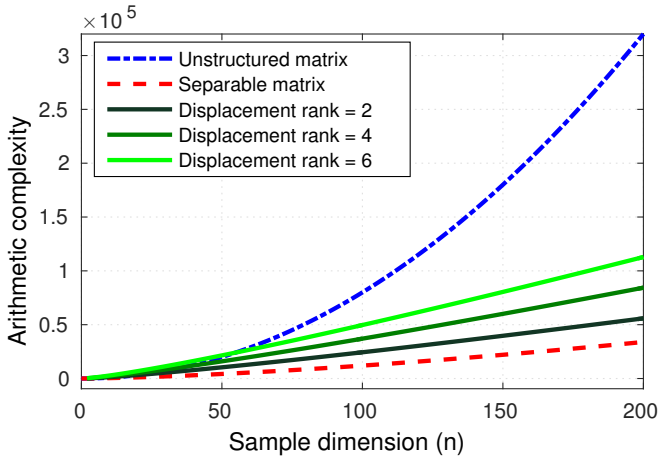

Figure 8 – Matrix-vector multiplication complexity (in total number of real multiplications and additions) assuming $m = 4n$.

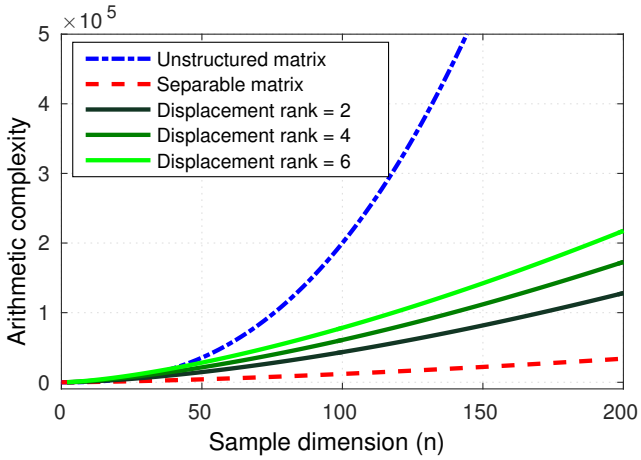

Figure 9 – Matrix-vector multiplication complexity (in total number of real multiplications and additions) assuming an overcompleteness factor of $\sqrt{n}$.

7 Results and Discussion

To illustrate the performance of the two proposed algorithms on real data, we have chosen an image denoising application, which is commonly used on the literature for evaluating dictionary learning algorithms. We have also established comparisons with some related techniques.

7.1 Simulation Set-up

We reproduce the same simulation set-up used in (ELAD; AHARON, 2006). The chosen grey-scale images (barbara, boats, house, lena and peppers) are corrupted with additive white Gaussian noise (AWGN) with different standard deviations σ.

The training data is composed by (8×8)-pixel patches extracted from the noisy image. According to the number of training data used, uniformly spaced – potentially overlapping – patches are extracted. The patches are then vectorized (by stacking its columns) leading to 64-dimensional training samples, which are used on the dictionary learning process. Figure 10 illustrates how a training sample is extracted from the noisy image.

The obtained dictionary $\mathbf{D}$ and sparse representation matrix $\mathbf{X}$ provide a re-

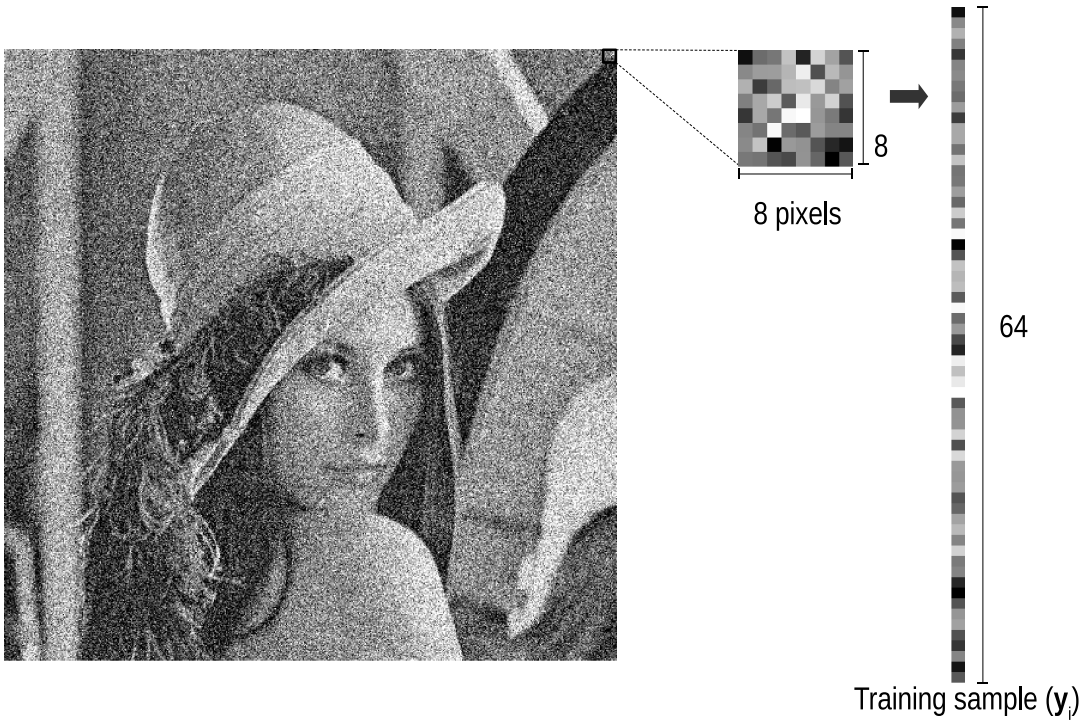

Figure 10 – Training data: patch extraction from the noisy image.

construction of the input data which is expected to reject part of the noise. The recovered image is constructed by averaging the overlapping pixel values on the reconstructed patches.

To quantify the reconstruction quality we use the peak signal-to-noise ratio (PSNR) between the original image and the recovered one, computed by

$$\text{PSNR} = 10 \log \left(\frac{255^2 N_{pixel}}{\sum_{i=1}^{N_{pixel}} (y_i - \hat{y}_i)^2} \right) \quad , \tag{7.1}$$

where 255 is the maximum pixel value, N_{pixel} is the total number of pixels on the input image (256×256 or 512×512), y_i and $\hat{y}_i$ are respectively the i-th pixel value on the input and reconstructed image.

The optimization parameters were empirically set to $\mu = 10^7$ and $\gamma = 6 \times 10^{-9}$. The convergence tolerance was set to $tol = \|\mathbf{D}\|_F \times 10^{-4}$ at all iterations but the last[1], when $tol = \|\mathbf{D}\|_F \times 10^{-7}$. The dictionary has been initialized with the 2-D overcomplete DCT dictionary[2] and $N_{iter} = 100$ iterations were used.

The reported results are averaged over 10 experiments with different noise realizations. Table 3 summarizes the simulation parameters (unless explicitly stated otherwise).

Table 3 – Simulation parameters

Signal dimension (n)	64
Number of atoms (m)	256
Training samples (N)	40000
Step-size (γ)	6×10^{-9}
Lagrangian penalty (μ)	10^7
Convergence tolerance (tol)	$\|\mathbf{D}\|_F \times 10^{-4}$
Iterations (N_{iter})	100
Dictionary initialization ($\mathbf{D}_{init}$)	ODCT

For the SuKro simulations we have used $n_1 = n_2 = \sqrt{n}$ and $m_1 = m_2 = \sqrt{m}$ as the sub-matrices dimensions.

The adopted simulation set-up supposes an application where only a noisy version of an image is available and it is desired to improve its quality. Applications ranging

[1] Due to column normalization we always have that $\|\mathbf{D}\|_F = \sqrt{m}$. The tolerance is smaller at the last iteration in order to guarantee a more accurate final result (the same accuracy is unnecessary in intermediate iterations).

[2] The 1-D $n \times m$ overcomplete DCT dictionary, as defined in (RUBINSTEIN *et al.*, 2010b), is a cropped and renormalized version of the orthogonal $m \times m$ DCT dictionary matrix. The 2-D ODCT is the Kronecker product of two 1-D ODCT dictionaries of size $\sqrt{n} \times \sqrt{m}$.

from ordinary digital camera image denoising to bio-medical, seismic and astronomical image treatment fit this scenario. All the training data is extracted from this one available noisy image. A dictionary specific for that image is trained, and then used on the denoising process.

Another possibility would be to use a database of noiseless images of a certain type (eg. astronomical, seismic, bio-medical) as training data for the dictionary learning. The obtained dictionary would be a kind of *general dictionary* that could be used to denoise images other images of that same type. The term *general* is employed in opposition to the *image-specific* dictionary obtained in the first scenario. Actually, this second scenario is more suited to the use of structured dictionaries, since the motivation for obtaining a more economic dictionary lies on the assumption that, after trained, it will be repeatedly used on a certain application. However, since the experiments performed here have the primary goal to justify the relevance of the proposed algorithm and provide a maximum of comparisons with similar existing methods, we have decided to use a more typical simulation set-up.

7.2 Benchmarks

We have chosen two techniques as the main benchmarks for the proposed algorithms: the K-SVD (AHARON *et al.*, 2006) method, which is a state-of-the-art algorithm for learning unstructured dictionaries, and the ODCT analytic dictionary. These two benchmarks represent opposite extremes on the complexity-flexibility tradeoff. The former representing the – flexible but complex – learned dictionaries and the latter representing the – fast but rigid – analytic counterpart. The structured dictionaries are expected to be in-between in terms of complexity and flexibility.

We also provide comparisons with other structured dictionary learning techniques: the SeDiL (Separable Dictionary Learning)(HAWE *et al.*, 2013) algorithm for learning separable dictionaries, the FAμST (Flexible Approximate Multi-layer Sparse Transforms) (MAGOAROU; GRIBONVAL, 2016) dictionaries, which are the product of sparse matrices, and the Sparse K-SVD (RUBINSTEIN *et al.*, 2010b) algorithm that constrains the dictionary to be a product of a fast base dictionary with a sparse matrix. In all cases, we use the implementation provided by the authors. In the following, we present a brief description and some details on the parameter setting for each of these methods.

7.2.1 Separable Dicionary Learning (SeDiL)

The SeDiL algorithm, proposed in (HAWE *et al.*, 2013), learns a separable dictionary, i.e. $\mathbf{D} = \mathbf{A} \otimes \mathbf{B}$. The ODCT analytic dictionary is an example of the separable structure. Other examples include the classic two dimensional orthogonal DCT and

Fourier transforms, as well as some Wavelet dictionaries (MALLAT, 2008). This structure can also be seen as a special case of the structure we propose in Chapter 5, when a single separable term is used.

In our simulations, to allow for a fair comparison, the training patches are extracted from the noisy image itself, which ensures that the same simulation set-up is used for all techniques. This is in contrast to the original work, which extracted the training data from noiseless images. In addition, we use the OMP algorithm for the sparse coding step while the SeDiL technique originally used the FISTA (BECK; TEBOULLE, 2009) algorithm. We initialize the dictionary as an ODCT and maintain the default value of all remaining simulation parameters.

As will be seen on the simulation results, the SeDiL algorithm performance is considerably reduced on this new configuration. It may even underperform the ODCT occasionally.

7.2.2 Flexible Approximate Multi-layer Sparse Transforms (FAμST)

In the FAμST technique, proposed in (MAGOAROU; GRIBONVAL, 2016), the dictionary ($\mathbf{D}$) is constrained to be the product of a small number (J) of sparse matrices ($\mathbf{S}_j$),

$$\mathbf{D} = \prod_{j=1}^{J} \mathbf{S}_j \quad . \tag{7.2}$$

In order to completely specify this structure, several parameters need to be set. Besides the number of sparse factors (J), it is necessary to specify the size[3] as well as the number of non-zero elements in each factor $\mathbf{S}_j$.

The algorithm takes a pre-established dictionary (trained by the K-SVD algorithm, for instance) and finds an approximation in the form of Equation (7.2). The dictionary factorization is performed in a hierarchical way, i.e. by computing successive two-factor factorizations. The dictionary is first decomposed into two matrices $\mathbf{D} = \mathbf{T}_1\mathbf{S}_1$ with $\mathbf{S_1}$ sparse and $\mathbf{T}_1$ containing fewer non-zero elements than $\mathbf{D}$. Then, the process is repeated with $\mathbf{T}_1$, yielding $\mathbf{T}_1 = \mathbf{T}_2\mathbf{S}_2$ with $\mathbf{T_2}$ sparser than $\mathbf{T_1}$, and so on, until there are J factors. The resulting decomposition is thus $\mathbf{D} = \mathbf{T}_{J-1}\mathbf{S}_{J-1} \dots \mathbf{S}_1$, with $\mathbf{T}_{J-1}$ potentially less sparse than the other terms, depending on the parameter setting.

The sparsity of the factors $\mathbf{S}_j$ is controlled by the parameter s, which determines the average column sparsity. The sparsity of the "residual" matrix $\mathbf{T}_\ell$ is chosen to decrease geometrically with the step ℓ and is controlled by two parameters ρ and P. The number of non-zero elements in $\mathbf{T}_\ell$ is given by $P\rho^\ell$, with $\rho < 1$.

[3] As long as the final product has the same size of $\mathbf{D}$, i.e. the number of columns of the rightmost factor $\mathbf{S}_J$ and the number of lines of the leftmost factor $\mathbf{S}_1$ are determined by the dictionary dimensions.

In our simulations, we have used a subset of the configurations tested in (MAGOAROU; GRIBONVAL, 2016). The size of the factors are: $\mathbf{T_{J-1}}, \mathbf{S_{J-1}}, \ldots, \mathbf{S_2} \in \mathbb{R}^{n \times n}$, $\mathbf{S_1} \in \mathbb{R}^{n \times m}$. The number of factors is $J = 4$. When it comes to the sparsity level of the factors, we have tested several configurations, with $s \in \{3, 6, 12, 24\}$, $\rho \in \{0.7, 0.9\}$ and $P = n^2$. The K-SVD dictionary, trained under the conditions described in Section 7.1, is used as an initialization.

7.2.3 Sparse K-SVD (S-KSVD)

In this algorithm, introduced in (RUBINSTEIN *et al.*, 2010b), the resulting atoms are sparse linear combinations of the atoms in a fixed base dictionary. The dictionary takes the form

$$\mathbf{D} = \mathbf{\Phi A} \quad ,$$

where $\mathbf{\Phi}$ is the base dictionary and $\mathbf{A}$ has sparse columns.

Following (RUBINSTEIN *et al.*, 2010b), we use the ODCT as the base dictionary. In this case, we have $\mathbf{\Phi} \in \mathbb{R}^{n \times m}$ and $\mathbf{A} \in \mathbb{R}^{m \times m}$. The column sparsity of $\mathbf{A}$ is controlled by the parameter p. In our simulations, we have tested the following configurations: $p \in \{3, 6, 12, 18, 24, 32\}$.

7.3 Simulation results

7.3.1 Displacement Rank and Number of Separable Terms

In Figures 11 and 12 we show the reconstruction PSNR as a function of the dictionary displacement rank for each of the five tested images under different noise levels (respectively $\sigma = \{20, 50\}$). The various displacement ranks were obtained by sweeping the parameter λ inside the interval $\lambda \in [0, 6000]$.

Two different structure types have been used: Toeplitz-like and Hankel-like. Naturally, the recovered PSNR decreases as the dictionary displacement rank is reduced. This is due to a lack of adaptability implied by the increased level of constraint on the matrix structure. It can be seen as the price to pay for a complexity reduction. Nevertheless, the obtained dictionaries still lead to very competitive results, being close to the KSVD for a large range of displacement ranks and overcoming the ODCT dictionary even for very low displacement ranks. The unconstrained dictionary has some advantage on very textured images ("barbara" for instance) as it has the required flexibility to fit such complex patterns, while in smoother images like "boat", "house" and "peppers" the displacement rank reduction entails a less pronounced performance degradation.

Also note that the proposed technique becomes more competitive as the noise level increases, until a certain point. In such cases, the flexibility of the KSVD dictionary

turns into a disadvantage as it allows for fitting the noise (overfitting), whereas imposing a degree of structure seems to prevent it by acting as a regularization. This argument will be further explored in section 7.3.2.

Most of the presented remarks also apply for the SuKro (sum of Kronecker products) structure, introduced in Chapter 5. In figures 13 and 14 the denoised images PSNR is plotted as a function of the number of separable terms forming the dictionary, with $\sigma = \{20, 50\}$ respectively. The resulting number of separable terms is controlled via the parameter λ, which is swept inside the interval $\lambda \in [100, 2200]$.

Note that even with very few separable terms, SuKro achieves similar results (or even better for higher noise scenarios) when compared to K-SVD, besides consistently outperforming the ODCT dictionary. We also show the performance of the SeDiL dictionary, which has exactly the same structure as a one-term SuKro dictionary. The superiority of the SuKro dictionary is exclusively due to the different problem formulation and learning algorithm. Naturally, as the number of separable terms increases, so does the denoising performance, since the dictionary becomes more flexible. The drawback is the increase on the complexity.

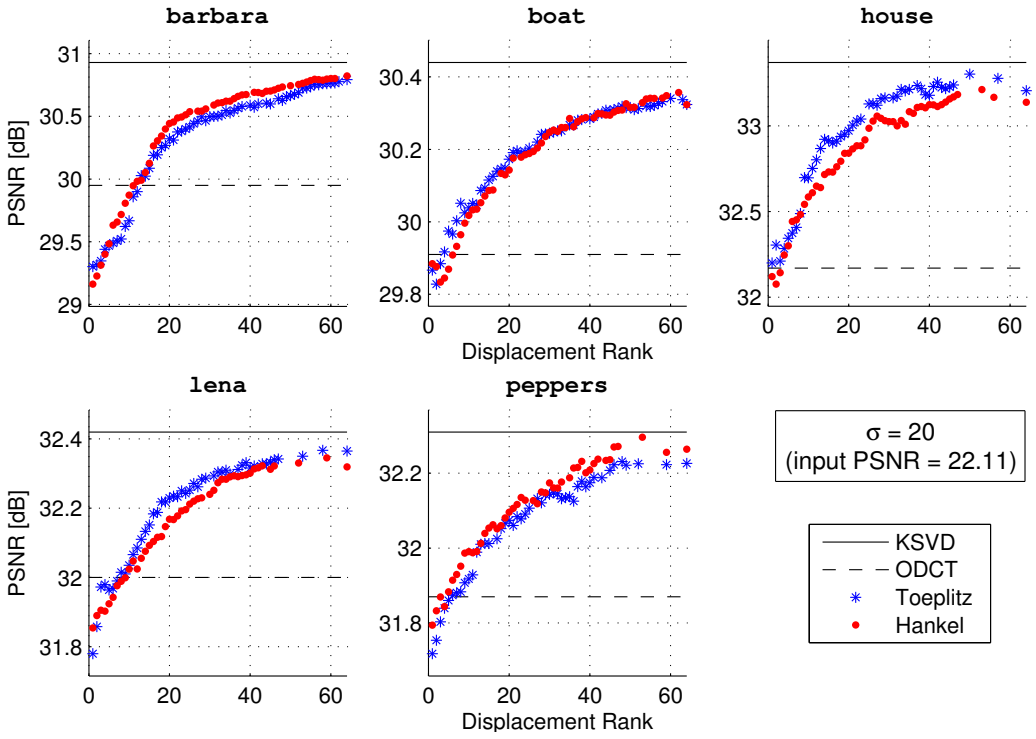

Figure 11 – PSNR vs. Displacement Rank, with $\sigma = 20$. The higher the better.

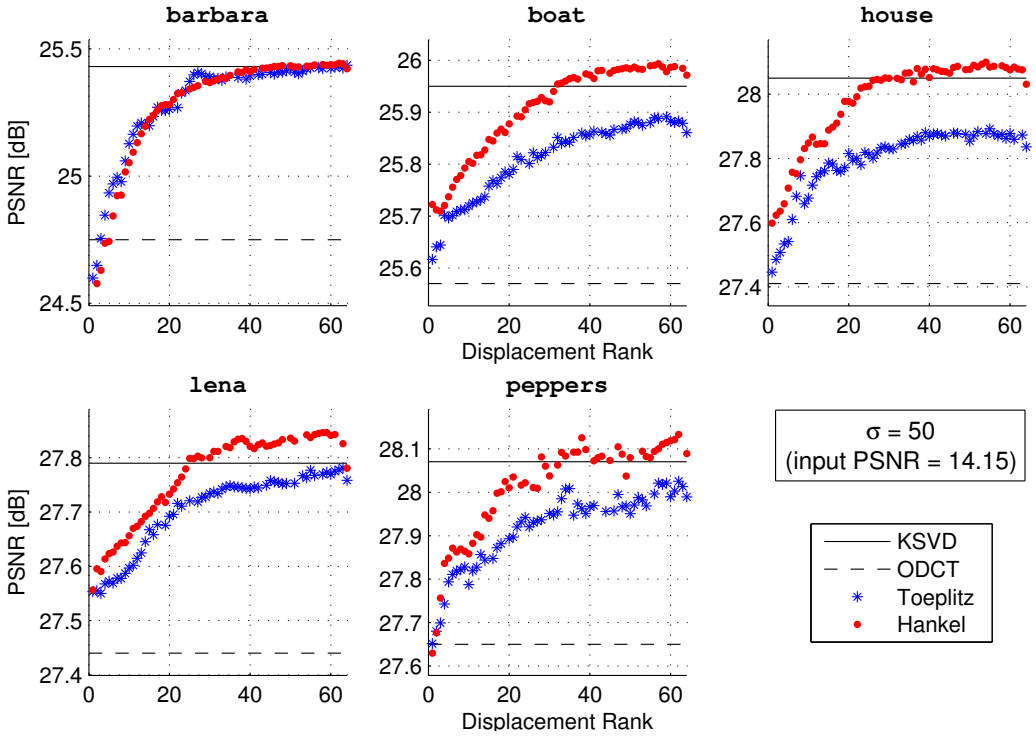

Figure 12 – PSNR vs. Displacement Rank, with $\sigma = 50$. The higher the better.

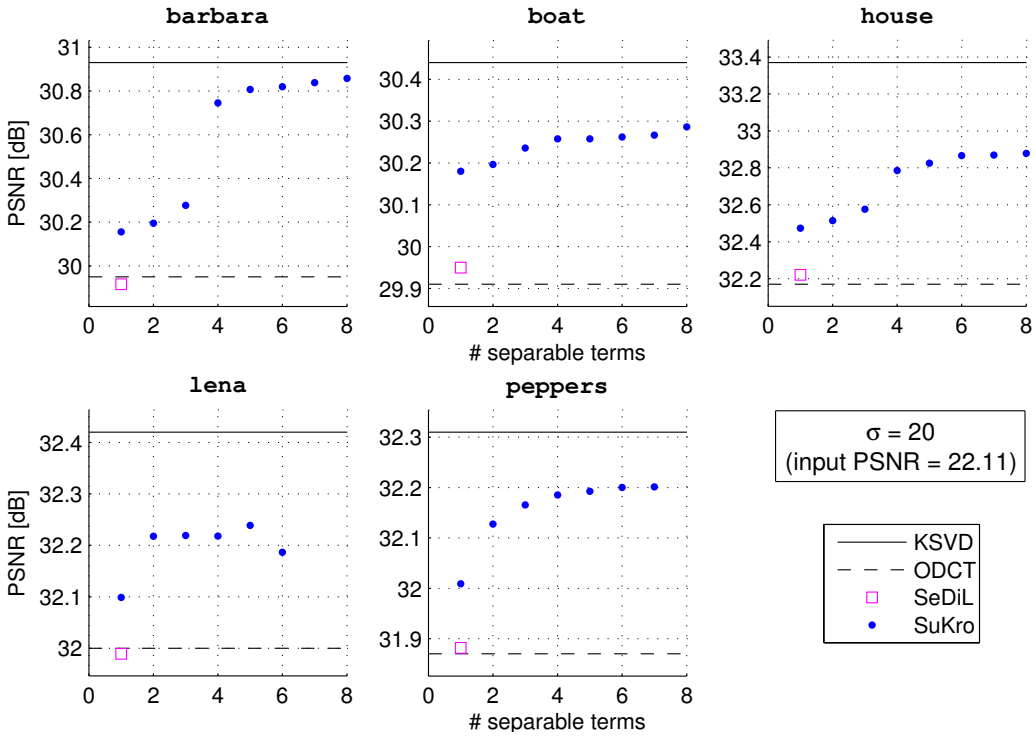

Figure 13 – PSNR vs. rank($\widetilde{\mathbf{D}}$) (i.e. the number of separable terms), with $\sigma = 20$. The higher the better.

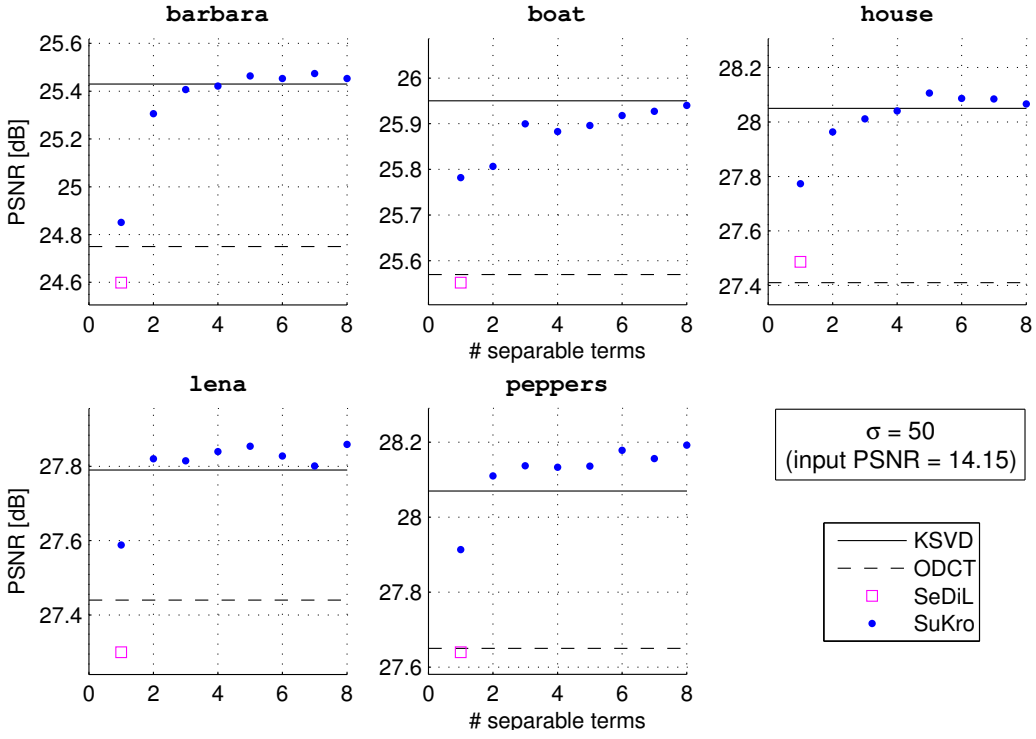

Figure 14 – PSNR vs. rank($\widetilde{\mathbf{D}}$) (i.e. the number of separable terms), with $\sigma = 50$.

7.3.2 Varying Input Noise Power

In Section 7.3.1 we have analyzed the dictionary denoising capabilities as the constraints on the dictionary structure are gradually loosened. In this section we analyze the effect of the input noise power on the dictionary performance.

In Figures 15 and 16 we compare the reconstruction PSNR of the simulated techniques at different input noise levels. The ODCT PSNR results are taken as a reference and subtracted from the results of all other techniques to highlight the differences between them and isolate the impact of the techniques from the input noise. Just as in the previous graphs, higher is better. Besides comparing with the K-SVD and the ODCT dictionaries, we also include in the graph the results of other structured dictionary learning algorithms. We split the results in two graphs to provide a better visualization. The Sparse K-SVD technique results are displayed in Figure 15 along with the proposed low displacement rank dictionaries, while the SeDiL and FAuST techniques are displayed in Figure 16 along with the SuKro results.

The two proposed methods proved to be more robust to noise as its performance degradation is less pronounced as the noise increases when compared to the unconstrained K-SVD dictionary, to the point of nearing its performance in high-noise scenarios (even for the very low displacement rank and number of separable terms displayed). The reason is that, by reducing the flexibility of the dictionary, we end up preventing the learned dictionary from fitting the noise present in the input data. For the same reason, the Sparse K-SVD technique is capable of outperforming the KSVD at noise standard deviations above $\sigma = 50$.

Comparing the five tested structured dictionary techniques, the Sparse KSVD and SuKro techniques obtained the best results, with quite similar performance (except for $\sigma = 20$ where S-KSVD achieved a slightly better result). Low displacement rank and FAuST dictionaries achieved very close results, while SeDiL lagged behind.

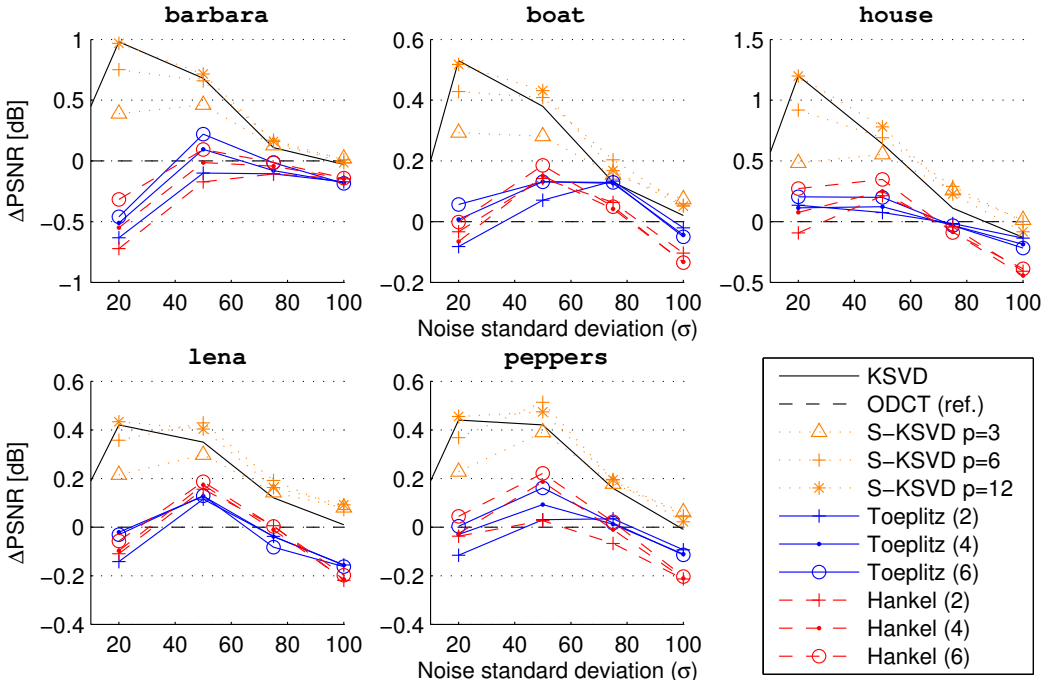

Figure 15 – Performance comparison on varying noise power. The PNSR difference is taken with respect to the ODCT (reference). The Hankel-like and Toeplitz-like dictionaries' displacement ranks are shown in brackets.

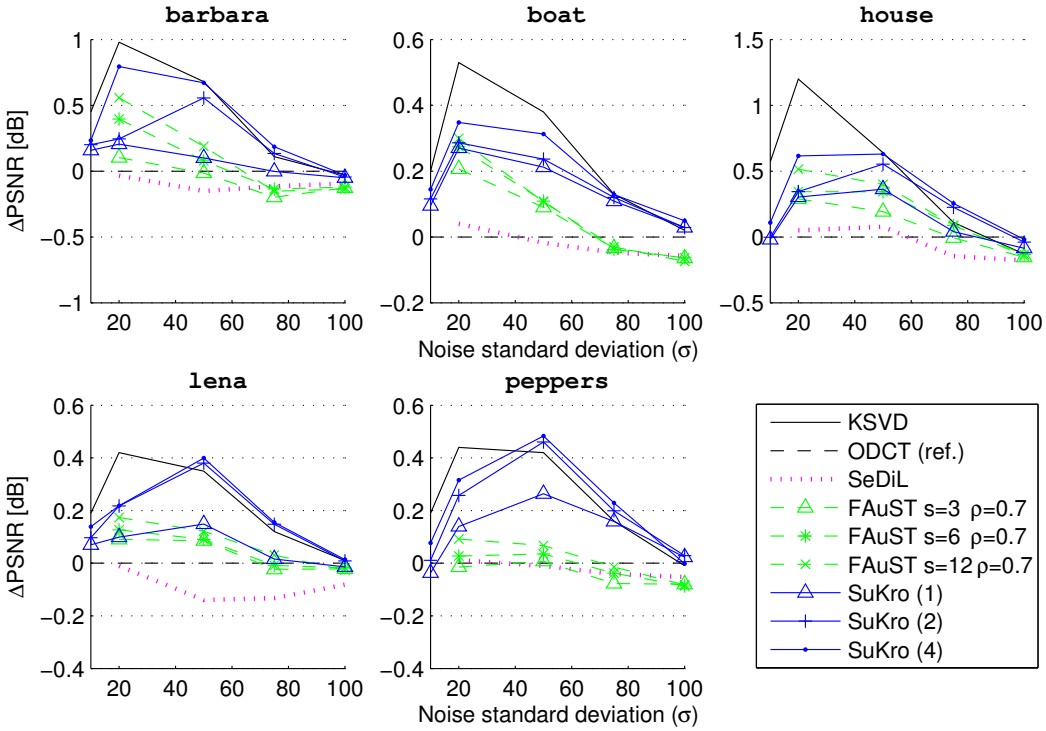

Figure 16 – Performance comparison on varying noise power. The PNSR difference is taken w.r.t. the ODCT (reference). The number of separable terms is shown in brackets.

7.3.3 Varying Training Dataset Size

In this section, we analyze the sample complexity of several dictionary learning methods, that is, the impact of reducing the training dataset size on the quality of the learned dictionary. Structured dictionaries are expected to be more robust to reduced training datasets, due to the decreased number of free parameters to estimate compared to unstructured dictionaries. In Figures 17 and 18 we show the reconstruction results on varying training dataset sizes. As expected, the performance of the structured dictionaries is much less affected by the reduction on the number of training samples. It is important to mention that 40000 training samples were used in all the previously presented results. As seen in Figures 17 and 18, if fewer training samples were used, the results would be even more favourable to our method.

Compared to the other structured dictionaries, the two proposed techniques proved to be more robust to reduced training datasets. Although the S-KSVD and SuKro techniques achieve very close results with large training datasets, as the number of training samples gets smaller, the SuKro dictionaries stand out. Both proposed techniques obtain better results than the FAuST and SeDiL algorithms with smaller training datasets.

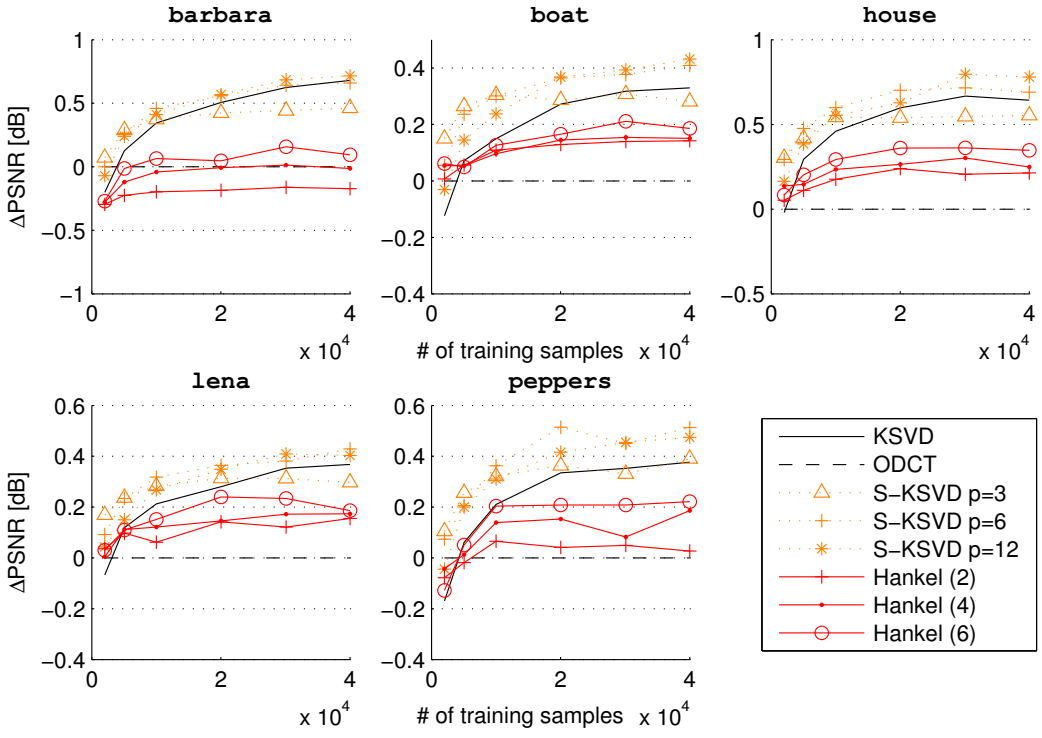

Figure 17 – Robustness to reduced training datasets, with $\sigma = 50$. The PNSR difference is taken with respect to the ODCT (reference).The Hankel-like and Toeplitz-like dictionaries' displacement ranks are shown in brackets.

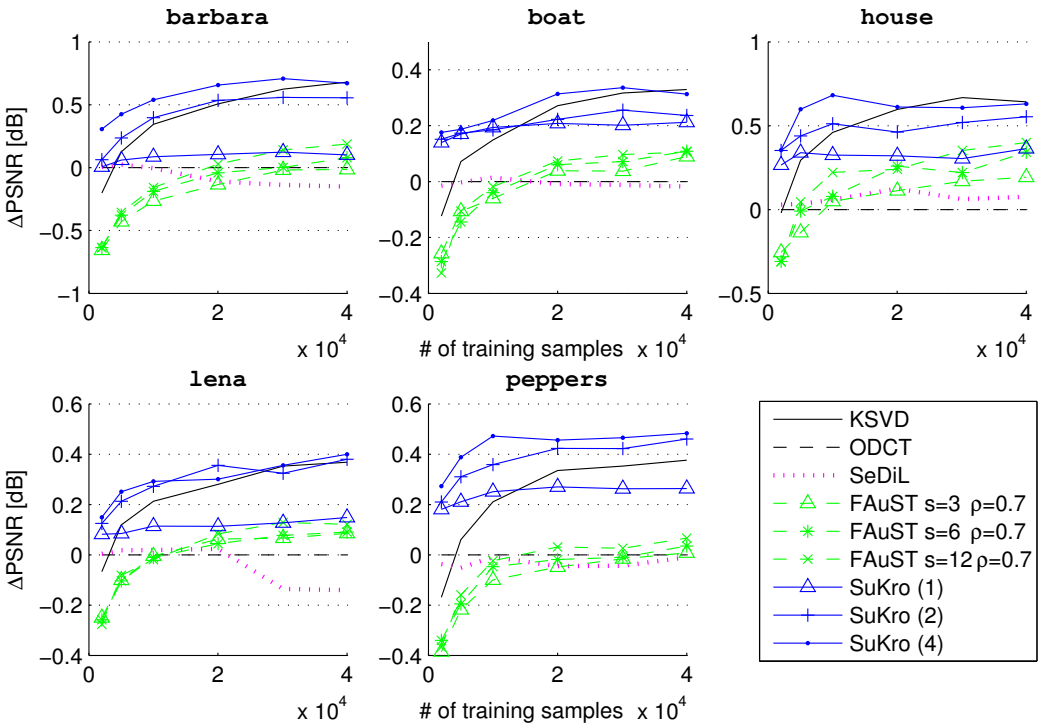

Figure 18 – Robustness to reduced training datasets, $\sigma = 50$. The PNSR difference is taken w.r.t. the ODCT (reference). In brackets, the number of separable terms.

7.3.4 Complexity-performance Trade-off

It is important to clarify that the computational complexity of the training algorithm is not evaluated here. Following the literature, we suppose a scenario where the training process is performed beforehand, in an offline fashion, and the trained dictionary is then to be repeatedly used in a certain application. Therefore, it is the complexity of operating with the trained dictionary (measured through the matrix-vector multiplication cost) that interests us. Indeed, if the whole training process was to be embedded, the training complexity would be critical too. Nevertheless, it is important to mention that the reduced multiplication costs obtained by the structured dictionaries proposed here could also be exploited to accelerate the numerous sparse coding steps performed during the training process. This possibility has not been explored on this thesis.

Substituting the dictionary dimensions in the complexity expressions derived in Chapter 6 we obtain the total number of operations required for operating with it[4]. The main trade-off here is between complexity reduction and reconstruction capabilities. This compromise is illustrated in Figures 19, 20, 21 and 22, where each configuration corresponds to a point on the plane PSNR vs. Complexity. In this graph, the higher and/or the more to the left the points are located, the better compromises they represent. A given point dominates another if it has the same or a smaller x coordinate (complexity) while having the same or a greater y coordinate (PSNR)– supposing, of course, that the points are distinct.

The merit of our methodology is providing a range of options on this trade-off curve. The displacement rank and the number of separable terms, respectively on the first and second proposed methods, may be adjusted to provide faster operators at the expense of some performance reduction. In this way, we offer dictionaries that are both less complex than the unconstrained ones (e.g. KSVD) and more flexible than the analytic ones (e.g. ODCT). Naturally, the less complex dictionaries will most probably achieve weaker denoising results.

Other techniques also offer similar tradeoffs. For instance, different complexities can be obtained with the S-KSVD by varying the parameter p (column sparsity of matrix $\mathbf{A}$), and with FAuST by varying both parameters s (column sparsity of the factor matrices $\mathbf{S}_i$) and ρ (related to the sparsity of the remainder matrix T). The SeDiL technique, in turn, does not offer such flexibility.

Note in Figures 21 and 22 that the SuKro dictionaries containing a single separable term, obtain a considerably better performance than the ODCT while having

[4] We recall that the ODCT is a separable dictionary. Note that we use the complexity of a general separable matrix, given in Equation (6.12), for the ODCT. If the regular 2D-DCT was used as a base dictionary, then this complexity would be smaller since both sub-matrices $\mathbf{\Phi}_0$ and $\mathbf{\Phi}_1$ would be 1D-DCTs, which have fast implementations. Such advantage is lost when an overcomplete version of the DCT is used, since the DCT matrix is truncated and renormalized, losing its original structure.

exactly the same computational complexity for matrix-vector multiplication. On this specific application, the S-KSVD technique obtained better results with $\sigma = 20$, while the SuKro and FAuST dictionaries obtained similar results. With a higher noise ($\sigma = 50$), the SuKro technique catches up with S-KSVD, while FAuST lags behind.

As we can see in Figures 19 and 20, although achieving similar results to the FAuST dictionaries at $\sigma = 50$, the displacement rank dictionaries are notably less competitive at $\sigma = 20$. However, we cannot imply that a certain method is superior to another from the presented simulations, since the results may vary with the application or even with the simulation set-up. The low displacement rank method, for instance, is severely disfavored by the relatively small size of the dictionary used (that is, the sample dimensionality $n = 64$ and number of atoms $m = 256$), due to some constant multiplication factors on the matrix-vector complexity expression – see Equation (6.5). Such complexity overhead is supposed to become less significant as the dimensionality grows making the technique much more competitive.

The chosen application may also affect the performance of the different dictionary structure types. As explained in Chapter 5, the separable structure is particularly suited to multidimensional signals such as images (two-dimensional signals). The S-KSVD, in turn, uses as the base dictionary a Discrete Cosine Transform[5], which is widely recognized for being well adapted to natural images. These observations help explaining some superior results obtained by SuKro and S-KSVD with respect to FAuST and low displacement rank dictionaries.

[5] In the proposed techniques the ODCT is used only as an initialization.

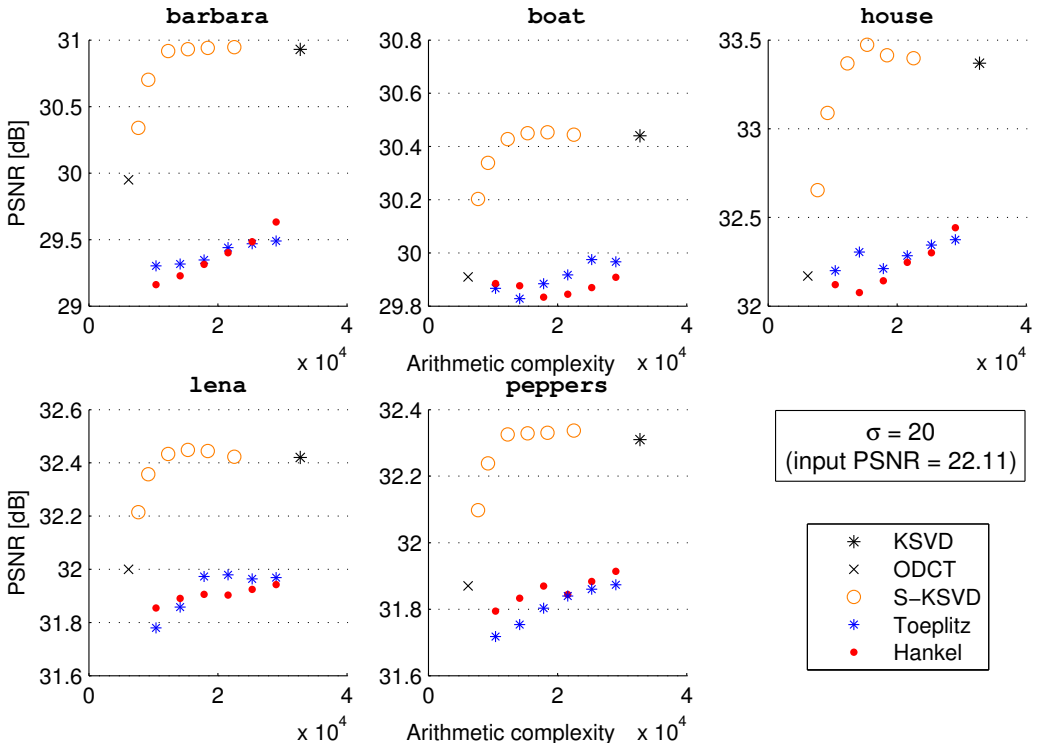

Figure 19 – Complexity-performance (PSNR) compromise, with $\sigma = 20$. Displacement ranks 1 to 6 (left to right) and S-KSVD with $p = \{3, 6, 12, 18, 24, 32\}$ (left to right).

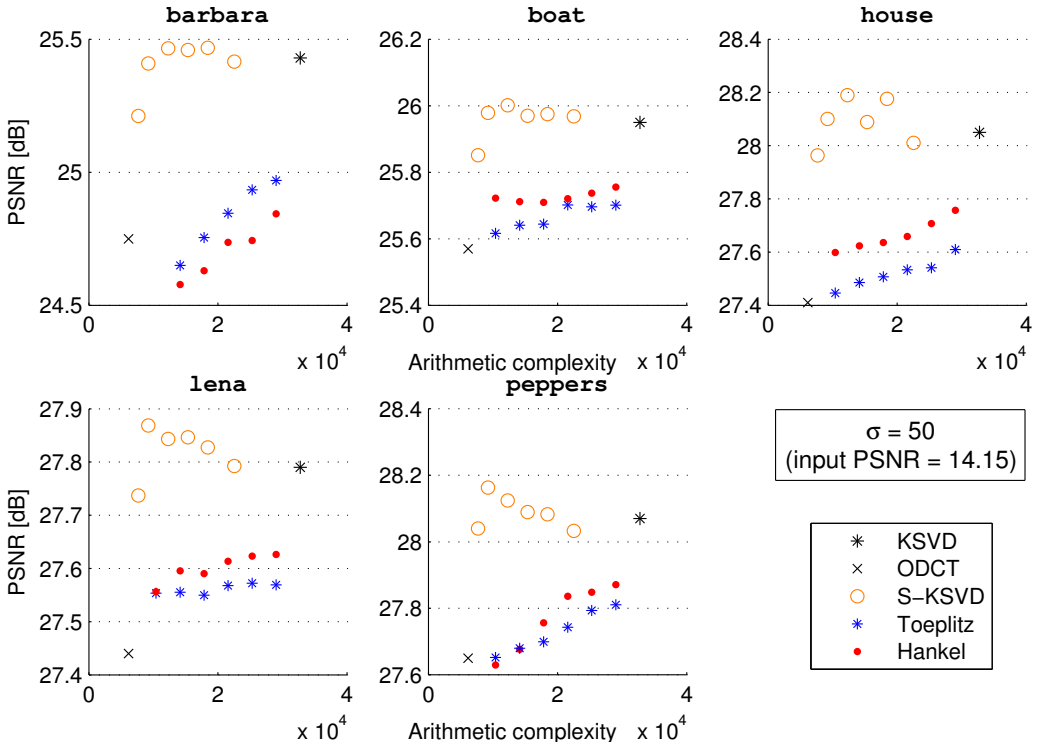

Figure 20 – Complexity-performance (PSNR) compromise, with $\sigma = 50$. Displacement ranks 1 to 6 (left to right) and S-KSVD with $p = \{3, 6, 12, 18, 24, 32\}$ (left to right).

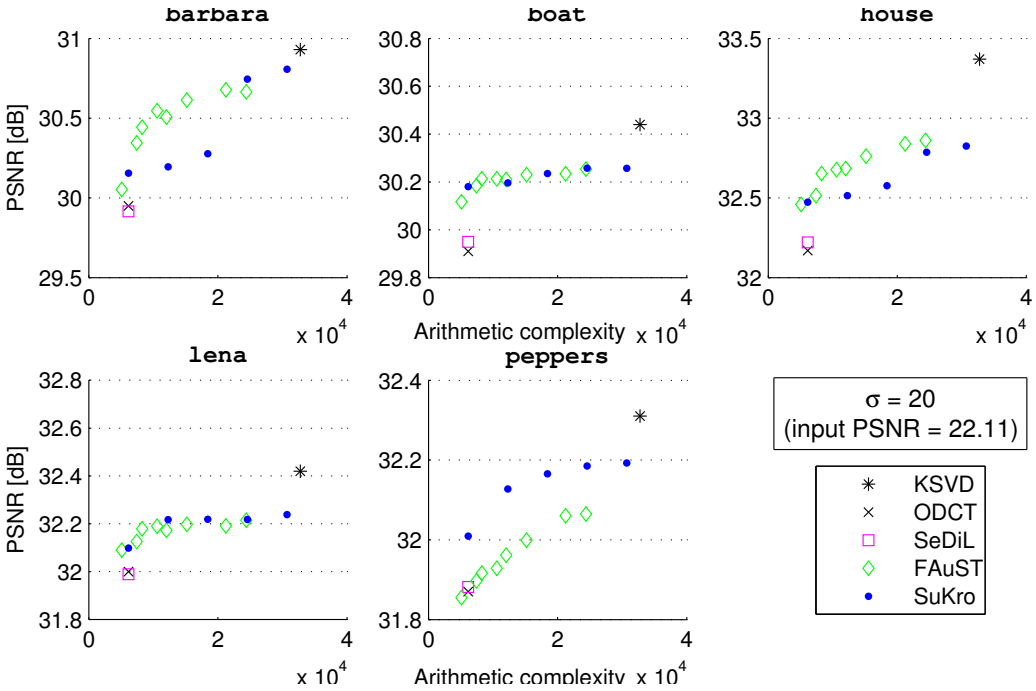

Figure 21 – Complexity-performance (PSNR) compromise, $\sigma = 20$. FAuST with $(s, \rho) = \{(3, 0.7), (6, 0.7), (3, 0.9), (6, 0.9), (12, 0.7), (12, 0.9), (24, 0.7), (24, 0.9)\}$ (left to right), Sukro with 1 to 6 separable terms (left to right) and SeDiL.

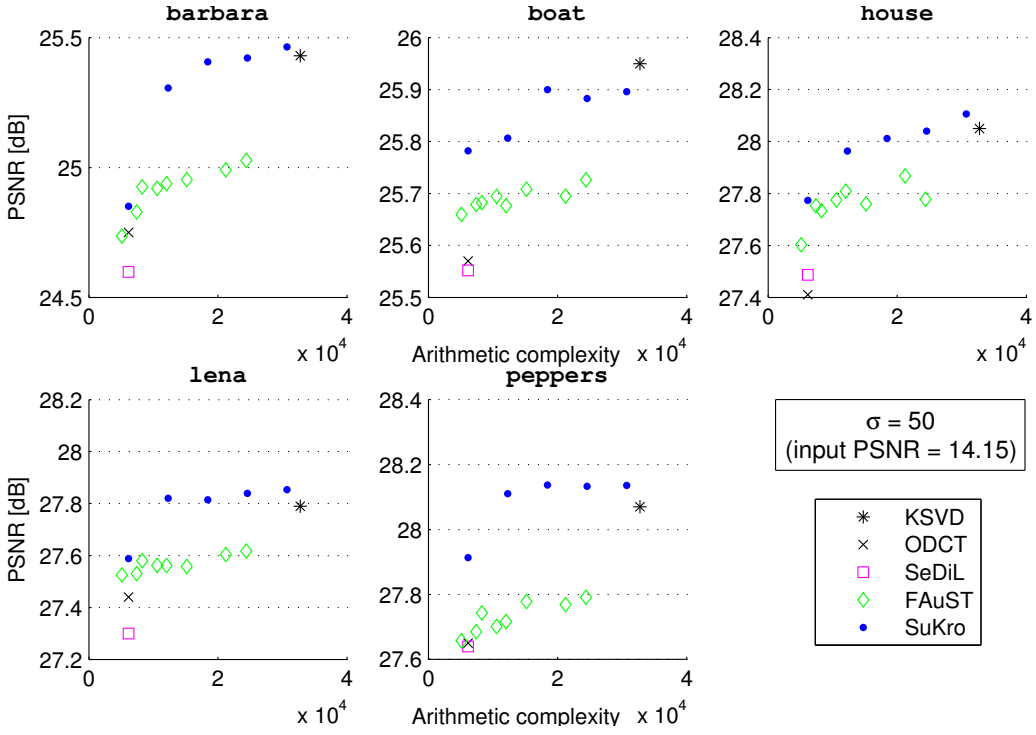

Figure 22 – Complexity-performance (PSNR) compromise, with $\sigma = 50$.

Conclusion

This master degree thesis is the result of studies on dictionary learning for sparse representations. In this project we introduce two algorithms for learning structured dictionaries. In the first one, the concept of displacement structure is used as a structure measure. The second one uses some existing Kronecker product results to learn dictionaries as a sum of separable terms. The two proposed structures give rise to similar optimization problems, both involving a rank minimization term. Some recent optimization concepts had to be called upon to solve such problems. We have used a convex relaxation for the rank operator which is the nuclear norm as well as a proximal algorithm.

Various image denoising experiments were shown as a proof of concept for the proposed methodologies. Competitive results have been obtained, proving the interest of the proposed ideas. In high noise scenarios and in presence of less training data, the structured dictionaries can even surpass the performance of an unstructured one. More importantly, the proposed techniques have the merit of providing a range of options on the complexity-adaptability trade-off. They lead to fast operators while keeping a considerable degree of flexibility and this trade-off can be controlled through the displacement rank (on the first method) and the number of separable terms (on the second method). Another important advantage of the proposed dictionaries is a higher robustness to small training datasets.

The obtained complexity gains may be even more pronounced if the signal dimension is increased along with the dictionary size. The main interest is exactly to improve the scalability properties of the dictionary learning algorithms.

Perspectives

On the displacement rank framework, other structure families are still to be explored, since in this thesis we have only covered the Toeplitz-like and Hankel-like cases. Other possibilities include classic families like Vandermonde and Cauchy as well as any other type of structure for which the displacement operator is known.

In addition, the proposed algorithms are not restricted to image denoising applications and could be applied the wide range of applications in which dictionary learning techniques are being employed. It would be interesting to try and identify applications that are better suited to each of the proposed structure types.

It is even possible to go beyond the dictionary learning context. In a broader sense, the proposed methods are actually matrix factorization techniques. Instead of struc-

tured dictionaries, they could be used to approximate any matrix (or any linear operator, represented by a matrix) by a structured version of it. Furthermore, the motivation need not be restricted to increasing the scalability. After some adaptations, the very same idea could be explored in many other domains where a certain matrix structure is desired. An example is the deconvolution problem, where the sought convolution matrix is expected to be Toeplitz. In other applications, a different structure family (Hankel, Vandermonde, Cauchy) may as well arise.

Bibliography

AHARON, M.; ELAD, M. Sparse and redundant modeling of image content using an image-signature-dictionary. *SIAM Journal on Imaging Sciences*, v. 1, n. 3, p. 228–247, 2008. Available from Internet: <http://dx.doi.org/10.1137/07070156X>. Cited on page 25.

AHARON, M.; ELAD, M.; BRUCKSTEIN, A. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, v. 54, n. 11, p. 4311–4322, Nov 2006. ISSN 1053-587X. Cited on pages 23, 32, 42, and 53.

BARANIUK, R. G.; CANDES, E.; ELAD, M.; MA, Y. (Ed.). *Special issue on applications of sparse representation and compressive sensing*, v. 98, n. 6. [S.l.]: Proceedings of the IEEE, 2010. Cited on page 15.

BECK, A.; TEBOULLE, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, v. 2, n. 1, p. 183–202, 2009. Available from Internet: <http://dx.doi.org/10.1137/080716542>. Cited on pages 23 and 54.

BERTSEKAS, D. P. *Constrained Optimization and Lagrange Multiplier Methods*. [S.l.]: Academic Press, 1982. Cited on pages 34 and 42.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 0387310738. Cited on page 21.

BLUMENSATH, T.; DAVIES, M. E. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, Springer, v. 14, n. 5-6, p. 629–654, 2008. Cited on page 22.

BOFILL, P.; ZIBULEVSKY, M. Underdetermined blind source separation using sparse representations. *Signal Processing*, v. 81, n. 11, p. 2353 – 2362, 2001. ISSN 0165-1684. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0165168401001207>. Cited on page 15.

BOYD, S.; PARIKH, N.; CHU, E.; PELEATO, B.; ECKSTEIN, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, Now Publishers Inc., Hanover, MA, USA, v. 3, n. 1, p. 1–122, Jan 2011. ISSN 1935-8237. Available from Internet: <http://dx.doi.org/10.1561/2200000016>. Cited on pages 33, 35, and 43.

CAI, J.-F.; CANDÈS, E. J.; SHEN, Z. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, SIAM, v. 20, n. 4, p. 1956–1982, 2010. Cited on pages 33 and 43.

CHEN, S. S.; DONOHO, D. L.; SAUNDERS, M. A. Atomic decomposition by basis pursuit. *SIAM JOURNAL ON SCIENTIFIC COMPUTING*, v. 20, p. 33–61, 1998. Cited on page 21.

DAUBECHIES, I.; DEFRISE, M.; MOL, C. D. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, Wiley Subscription Services, Inc., A Wiley Company, v. 57, n. 11, p. 1413–1457, 2004. ISSN 1097-0312. Available from Internet: <http://dx.doi.org/10.1002/cpa.20042>. Cited on page 23.

DUHAMEL, P. Implementation of "split-radix" fft algorithms for complex, real, and real-symmetric data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 34, n. 2, p. 285–295, Apr 1986. ISSN 0096-3518. Cited on page 77.

ECKART, C.; YOUNG, G. The approximation of one matrix by another of lower rank. *Psychometrika*, Springer, v. 1, n. 3, p. 211–218, 1936. Cited on page 24.

ELAD, M.; AHARON, M. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, v. 15, n. 12, p. 3736–3745, Dec 2006. ISSN 1057-7149. Cited on page 51.

ENGAN, K.; AASE, S. O.; HUSOY, J. H. Method of optimal directions for frame design. In: *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*. [S.l.: s.n.], 1999. v. 5, p. 2443–2446 vol.5. ISSN 1520-6149. Cited on pages 23, 32, and 42.

GOHBERG, I.; OLSHEVSKY, V. Complexity of multiplication with vectors for structured matrices. *Linear Algebra and its Applications*, v. 202, p. 163 – 192, 1994. ISSN 0024-3795. Cited on page 27.

GRIBONVAL, R.; JENATTON, R.; BACH, F.; KLEINSTEUBER, M.; SEIBERT, M. Sample complexity of dictionary learning and other matrix factorizations. *IEEE Transactions on Information Theory*, v. 61, n. 6, p. 3469–3486, June 2015. ISSN 0018-9448. Cited on page 38.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *Journal of machine learning research*, v. 3, n. Mar, p. 1157–1182, 2003. Cited on page 15.

HAWE, S.; SEIBERT, M.; KLEINSTEUBER, M. Separable dictionary learning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2013. p. 438–445. Cited on pages 25, 39, 40, 48, and 53.

IM, E.-J.; YELICK, K. A. *Optimizing the performance of sparse matrix-vector multiplication*. Tese (Doutorado) — University of California, Berkeley, 2000. Cited on page 49.

JAIN, A. K. *Fundamentals of Digital Image Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989. ISBN 0-13-336165-9. Cited on page 15.

JOLLIFFE, I. *Principal component analysis*. [S.l.]: Wiley Online Library, 2002. Cited on page 15.

KAILATH, T.; KUNG, S.-Y.; MORF, M. Displacement ranks of matrices and linear equations. *Journal of Mathematical Analysis and Applications*, v. 68, n. 2, p. 395 – 407, 1979. ISSN 0022-247X. Available from Internet: <http://www.sciencedirect.com/science/article/pii/0022247X79901240>. Cited on pages 18 and 25.

KAILATH, T.; SAYED, A. *Fast Reliable Algorithms for Matrices with Structure*. Society for Industrial and Applied Mathematics, 1999. Available from Internet: <http://epubs.siam.org/doi/abs/10.1137/1.9781611971354>. Cited on pages 46 and 47.

LESAGE, S.; GRIBONVAL, R.; BIMBOT, F.; BENAROYA, L. Learning unions of orthonormal bases with thresholded singular value decomposition. In: IEEE. *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*. [S.l.], 2005. v. 5, p. v–293. Cited on page 25.

LI, L.; LI, S.; FU, Y. Discriminative dictionary learning with low-rank regularization for face recognition. In: *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. [S.l.: s.n.], 2013. p. 1–6. Cited on pages 25, 32, and 33.

LOAN, C. F. V.; PITSIANIS, N. Approximation with kronecker products. In: *Linear algebra for large scale and real-time applications*. [S.l.]: Springer, 1993. p. 293–314. Cited on page 40.

MA, L.; WANG, C.; XIAO, B.; ZHOU, W. Sparse representation for face recognition based on discriminative low-rank dictionary learning. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. [S.l.: s.n.], 2012. p. 2586–2593. ISSN 1063-6919. Cited on pages 25, 32, and 33.

MAGOAROU, L. L.; GRIBONVAL, R. Chasing butterflies: In search of efficient dictionaries. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2015. p. 3287–3291. ISSN 1520-6149. Cited on page 25.

MAGOAROU, L. L.; GRIBONVAL, R. Flexible multilayer sparse approximations of matrices and applications. *IEEE Journal of Selected Topics in Signal Processing*, v. 10, n. 4, p. 688–700, June 2016. ISSN 1932-4553. Cited on pages 49, 53, 54, and 55.

MAIRAL, J.; BACH, F.; PONCE, J.; SAPIRO, G. Online dictionary learning for sparse coding. In: ACM. *Proceedings of the 26th Annual International Conference on Machine Learning*. [S.l.], 2009. p. 689–696. Cited on pages 32 and 42.

MALLAT, S. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. 3rd. ed. [S.l.]: Academic Press, 2008. ISBN 0123743702, 9780123743701. Cited on pages 39, 48, and 54.

MALLAT, S. G.; ZHANG, Z. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, IEEE, v. 41, n. 12, p. 3397–3415, 1993. Cited on pages 16, 21, and 22.

MARSAGLIA, G.; STYAN, G. P. Equalities and inequalities for ranks of matrices†. *Linear and multilinear Algebra*, Taylor & Francis, v. 2, n. 3, p. 269–292, 1974. Cited on page 45.

MOUILLERON, C. *Efficient computation with structured matrices and arithmetic expressions*. Tese (Doutorado) — Ecole normale supérieure de lyon - ENS LYON, Nov 2011. Available from Internet: <https://tel.archives-ouvertes.fr/tel-00688388>. Cited on page 30.

NOCEDAL, J.; WRIGHT, S. *Numerical optimization*. [S.l.]: Springer Science & Business Media, 2006. Cited on page 34.

PAN, V. Y.; WANG, X. Inversion of displacement operators. *SIAM Journal on Matrix Analysis and Applications*, v. 24, n. 3, p. 660–677, 2003. Available from Internet: <http://dx.doi.org/10.1137/S089547980238627X>. Cited on pages 45 and 46.

PARIKH, N.; BOYD, S. P. *et al.* Proximal algorithms. *Foundations and Trends in optimization*, v. 1, n. 3, p. 127–239, 2014. Cited on page 33.

PATI, Y. C.; REZAIIFAR, R.; KRISHNAPRASAD, P. S. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*. [S.l.: s.n.], 1993. p. 40–44 vol.1. ISSN 1058-6393. Cited on pages 22, 32, and 42.

PETERSEN, K. B.; PEDERSEN, M. S. *The Matrix Cookbook*. Technical University of Denmark, 2012. Version 20121115. Available from Internet: <http://www2.imm.dtu.dk/pubdb/p.php?3274>. Cited on page 79.

RECHT, B.; FAZEL, M.; PARRILO, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, v. 52, n. 3, p. 471–501, 2010. Available from Internet: <http://dx.doi.org/10.1137/070697835>. Cited on pages 26, 32, 40, and 42.

RUBINSTEIN, R.; BRUCKSTEIN, A. M.; ELAD, M. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, IEEE, v. 98, n. 6, p. 1045–1057, 2010. Cited on page 22.

RUBINSTEIN, R.; ZIBULEVSKY, M.; ELAD, M. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *Signal Processing, IEEE Transactions on*, IEEE, v. 58, n. 3, p. 1553–1564, 2010. Cited on pages 25, 49, 52, 53, and 55.

SADEGHI, M.; BABAIE-ZADEH, M.; JUTTEN, C. Learning overcomplete dictionaries based on atom-by-atom updating. *Signal Processing, IEEE Transactions on*, IEEE, v. 62, n. 4, p. 883–891, 2014. Cited on page 32.

SULAM, J.; OPHIR, B.; ZIBULEVSKY, M.; ELAD, M. Trainlets: Dictionary learning in high dimensions. *IEEE Transactions on Signal Processing*, v. 64, n. 12, p. 3180–3193, June 2016. ISSN 1053-587X. Cited on page 25.

TOH, K.-C.; YUN, S. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization*, 2010. Available from Internet: <http://www.math.nus.edu.sg/~{}mattohkc/papers/mc11.p>. Cited on page 33.

TOSIC, I.; FROSSARD, P. Dictionary learning. *IEEE Signal Processing Magazine*, v. 28, n. 2, p. 27–38, March 2011. ISSN 1053-5888. Cited on pages 16 and 23.

ZHANG, Y.; JIANG, Z.; DAVIS, L. S. Learning structured low-rank representations for image classification. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. [S.l.: s.n.], 2013. p. 676–683. ISSN 1063-6919. Cited on pages 25 and 33.

ZHENG, C.-H.; HOU, Y.-F.; ZHANG, J. Improved sparse representation with low-rank representation for robust face recognition. *Neurocomputing*, p. –, 2016. ISSN 0925-2312. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0925231216003076>. Cited on pages 25 and 33.

Appendix

APPENDIX A – Complexity of Forward and Transpose Operators

In this appendix we derive the complexity of multiplying an $(n \times m)$-matrix $\mathbf{M}$ in the form of equation (6.2) – Toeplitz-like – or equation (6.3) – Hankel-like – by a vector $\mathbf{w} \in \mathbb{R}^m$. Throughout this appendix we denote $\odot$ the element-wise vector multiplication.

Forward Operator

As derived in Section 6.1, the forward operator $\mathbf{Mw}$ leads to the following, for the Toeplitz-like and Hankel-like case respectively:

$$\left(\sum_{k=1}^{\alpha} \mathbf{L}(\mathbf{g}_k) \bar{\mathbf{U}}(\mathbf{h}_k^T) \right) \mathbf{w} \tag{A.1}$$

$$\left(\sum_{k=1}^{\alpha} \mathbf{L}(\mathbf{g}_k) \bar{\mathbf{U}}((\mathbf{J}\mathbf{h}_k)^T) \right) \mathbf{J}\mathbf{w} \tag{A.2}$$

Let us take the Toeplitz-like case (equation (A.1)), since the Hankel-like case is equivalent upon the inversion of the vector $\mathbf{w}$. Denoting $FFT(\mathbf{x}, m)$ the m-point fast Fourier transform of the vector $\mathbf{x}$ (zero-padded if it has fewer than m elements), the forward operator can be implemented by Algorithm A.1.

Considering that the vectors $\hat{\mathbf{g}}_k$ and $\hat{\mathbf{h}}_k$ are pre-calculated and stored in memory and denoting $\Phi(n)$ the arithmetic complexity of an n-point FFT, the total cost of Algorithm A.1 is given by:

$$\Phi(2m) + \alpha\,(2m + \Phi(2m) + \Phi(2n) + 4n) + \Phi(2n) \tag{A.3}$$

However, the complexity of the product $\mathbf{v} = \bar{\mathbf{U}}(\mathbf{h}_k^T)\mathbf{w}$ can be further reduced when m is an integer multiple of n, i.e $m/n \in \mathbb{Z}$. In this case, we can use a block representation for $\bar{\mathbf{U}}$ and $\mathbf{w}$:

$$\bar{\mathbf{U}} = [\bar{\mathbf{U}}_1 \mid \bar{\mathbf{U}}_2 \mid \ ... \ \mid \bar{\mathbf{U}}_{m/n}] \qquad \text{with each } \bar{\mathbf{U}}_i \in \mathbb{R}^{n \times n},$$

$$\mathbf{w} = [\mathbf{w}_1^T \mid \mathbf{w}_2^T \mid \ ... \ \mid \mathbf{w}_{m/n}^T]^T \qquad \text{with each } \mathbf{w}_i \in \mathbb{R}^n.$$

Then $\bar{\mathbf{U}}\mathbf{w} = \sum_{i=1}^{m/n} \bar{\mathbf{U}}_i \mathbf{w}_i$.

The fact that $\bar{\mathbf{U}}(\mathbf{h}_k^T)$ is a truncated upper triangular Toeplitz matrix implies that $\bar{\mathbf{U}}_1$ is upper triangular Toeplitz and all other $\bar{\mathbf{U}}_i$ are Toeplitz. So, it comes to $\frac{m}{n}$

Algorithm A.1 Forward Operator - fast implementation

Input: vector $\mathbf{w} \in \mathbb{R}^m$,
matrices $\mathbf{G}$ and $\mathbf{H}$, s.t. $\Delta_{\mathbf{z}_{n,0},\mathbf{z}_{m,0}^T}(\mathbf{M}) = \mathbf{G}\mathbf{H}^T$
Output: vector $\mathbf{x}_f \in \mathbb{R}^n$, s.t. $\mathbf{x}_f = \mathbf{M}\mathbf{w}$

▷ Pre-calculations
for $k \leftarrow 1$ to α **do**
  $\hat{\mathbf{g}}_k \leftarrow FFT(\mathbf{g}_k, 2n)$
  $\hat{\mathbf{h}}_k \leftarrow FFT(\mathbf{J}\mathbf{h}_k, 2m)$
end for
$\mathbf{x}_f = \mathbf{0}$

▷ Multiplication
$\hat{\mathbf{w}} \leftarrow FFT(\mathbf{w}, 2m)$
for $k \leftarrow 1$ to α **do**
  ▷ Compute $\mathbf{v} = \bar{\mathbf{U}}(\mathbf{h}_k^T)\mathbf{w}$
  $\hat{\mathbf{v}} \leftarrow \hat{\mathbf{h}}_k \odot \hat{\mathbf{w}}$ ▷ Element-wise multiplication
  $\mathbf{v} \leftarrow IFFT(\hat{\mathbf{v}}, 2m)$
  $\mathbf{v} \leftarrow \mathbf{v}(m : m + n + 1)$ ▷ Result of $\bar{\mathbf{U}}(\mathbf{h}_k^T)\mathbf{w}$

  ▷ Compute $\mathbf{x} = \mathbf{L}(\mathbf{g}_k)\mathbf{v}$
  $\hat{\mathbf{v}} \leftarrow FFT(\mathbf{v}, 2n)$
  $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{g}}_k \odot \hat{\mathbf{v}}$

  ▷ Accumulate results
  $\hat{\mathbf{x}}_f \leftarrow \hat{\mathbf{x}}_f + \hat{\mathbf{x}}$
end for
$\mathbf{x}_f \leftarrow IFFT(\hat{\mathbf{x}}_f, 2n)$

return $\mathbf{x}_f(1 : n)$

Toeplitz $(n \times n)$-matrix-vector multiplications (one of them being upper triangular). It turns out that the Toeplitz matrix profits from a similar convolution-like implementation.

Consider $\mathbf{T}$ a $(n \times n)$ Toeplitz matrix determined by the vector $\mathbf{t} = (t_{1-n}, \dots, t_0, \dots, t_{n-1})^T$ as in figure 3.3. We introduce the polynomials

$$t(x) = \sum_{i=1}^{2n-1} t_{i-n} x^{i-1} \qquad w(x) = \sum_{i=1}^{n} w_i x^{i-1}$$

Then the following holds:

- the ith entry of the vector $\mathbf{T}\mathbf{w}$ is the coefficient of x^{n-2+i} in the polynomial product $t(x)w(x)$.
  Or, equivalently, the $(n-1+i)$th output of the convolution $\mathbf{t} * \mathbf{w}$.

This leads to a less complex forward operator described in Algorithm A.2.

Algorithm A.2 Forward Operator - improved implementation

Input: vector $\mathbf{w} \in \mathbb{R}^m$,
matrices $\mathbf{G}$ and $\mathbf{H}$, s.t. $\Delta_{\mathbf{z}_{n,0},\mathbf{z}_{m,0}^T}(\mathbf{M}) = \mathbf{G}\mathbf{H}^T$
Output: vector $\mathbf{x}_f \in \mathbb{R}^n$, s.t. $\mathbf{x}_f = \mathbf{M}\mathbf{w}$

▷ Pre-calculations
for $k \leftarrow 1$ to α **do**
  $\hat{\mathbf{g}}_k \leftarrow FFT(\mathbf{g}_k, 2n)$
  $\hat{\mathbf{h}}_{k,1} \leftarrow FFT(\mathbf{J}\mathbf{h}_k(1:n), 2n)$
  for $l \leftarrow 2, m/n$ **do**
    $\hat{\mathbf{h}}_{k,l} \leftarrow FFT(\mathbf{J}\mathbf{h}_k((l-2)n+2 : ln), 2n)$
  end for
end for
$\mathbf{x}_f = \mathbf{0}$

▷ Multiplication
for $l \leftarrow 1$ to m/n **do**
  $\hat{\mathbf{w}}_l \leftarrow FFT(\mathbf{w}((l-1)n+1 : ln), 2n)$
end for

for $k \leftarrow 1$ to α **do**
  ▷ Compute $\mathbf{v} = \bar{\mathbf{U}}(\mathbf{h}_k^T)\mathbf{w}$
  $\hat{\mathbf{v}} = \mathbf{0}$
  for $l \leftarrow 1$ to m/n **do**
    $\hat{\mathbf{v}} \leftarrow \hat{\mathbf{v}} + \hat{\mathbf{h}}_{k,l} \odot \hat{\mathbf{w}}_l$
  end for
  $\mathbf{v} \leftarrow IFFT(\hat{\mathbf{v}}, 2n)$
  $\mathbf{v} \leftarrow \mathbf{v}(n : 2n-1)$ ▷ Result of $\bar{\mathbf{U}}(\mathbf{h}_k^T)\mathbf{w}$

  ▷ Compute $\mathbf{x} = \mathbf{L}(\mathbf{g}_k)\mathbf{v}$
  $\hat{\mathbf{v}} \leftarrow FFT(\mathbf{v}, 2n)$
  $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{g}}_k \odot \hat{\mathbf{v}}$

  ▷ Accumulate results
  $\hat{\mathbf{x}}_f \leftarrow \hat{\mathbf{x}}_f + \hat{\mathbf{x}}$
end for
$\mathbf{x}_f \leftarrow IFFT(\hat{\mathbf{x}}_f, 2n)$

return $\mathbf{x}_f(1:n)$

Its complexity in terms of the FFT cost is given by:

$$\frac{m}{n}\Phi(2n) + \alpha\left(4m + 2\Phi(2n) + 4n\right) + \Phi(2n) \tag{A.4}$$

The exact arithmetic complexity of the classic split-radix implementation of the FFT in total number of real additions and real multiplications is (DUHAMEL, 1986):

$$\Phi(n) = 2n\log n - 4n + 6 \tag{A.5}$$

To obtain the arithmetic complexity of Algorithm A.2 we substitute A.5 in A.4, which gives:

$$\begin{aligned} 4[n(2\alpha+1)+m]\log(2n) + 4m(\alpha-2) - 4n(3\alpha+2) \\ +6(2\alpha + \frac{m}{n} + 1) \end{aligned} \tag{A.6}$$

Transpose Operator

The implementation of the transpose operator $\mathbf{M}^T\mathbf{v}$, with $\mathbf{M}^T \in \mathbb{R}^{m,n}$ and $\mathbf{v} \in \mathbb{R}^n$ is quite similar from the previous one. Just consider that

$$\mathbf{M}^T = \left(\sum_{k=1}^{\alpha} \bar{\mathbf{U}}(\mathbf{h}_k^T)^T \mathbf{L}(\mathbf{g}_k)^T\right) \tag{A.7}$$

Keeping in mind that $\mathbf{L}(\mathbf{g}_k)^T$ becomes an upper triangular Toeplitz matrix and $\bar{\mathbf{U}}(\mathbf{h}_k^T)^T$ a truncated (in the columns) lower triangular Toeplitz matrix, we can still apply the same multiplication strategy.

The block strategy previously presented becomes:

$$\bar{\mathbf{U}}^T\mathbf{w} = \begin{bmatrix} \bar{\mathbf{U}}_1^T\mathbf{w} \\ \bar{\mathbf{U}}_2^T\mathbf{w} \\ \vdots \\ \bar{\mathbf{U}}_{m/n}^T\mathbf{w} \end{bmatrix} \tag{A.8}$$

with $\mathbf{w} \in \mathbb{R}^n$

Algorithm A.3 shows the resulting implementation. Its total count of real multiplications and real additions is:

$$\begin{aligned} 4[n(2\alpha+1)+m]\log(2n) + 4m(\alpha-2) - 2n(7\alpha+4) \\ +6(2\alpha + \frac{m}{n} + 1) \end{aligned} \tag{A.9}$$

Algorithm A.3 Transpose Operator - improved implementation

Input: vector $\mathbf{v} \in \mathbb{R}^n$,
matrices $\mathbf{G}$ and $\mathbf{H}$, s.t. $\Delta_{\mathbf{Z}_{n,0},\mathbf{Z}_{m,0}^T}(\mathbf{M}) = \mathbf{G}\mathbf{H}^T$
Output: vector $\mathbf{x}_f \in \mathbb{R}^m$, s.t. $\mathbf{x}_f = \mathbf{M}^T\mathbf{v}$

▷ Pre-calculations
for $k \leftarrow 1$ to α **do**
  $\hat{\mathbf{g}}_k \leftarrow FFT(\mathbf{J}\mathbf{g}_k, 2n)$
  $\hat{\mathbf{h}}_{k,1} \leftarrow FFT(\mathbf{h}_k(1:n), 2n)$
  for $l \leftarrow 2$ to m/n **do**
    $\hat{\mathbf{h}}_{k,l} \leftarrow FFT(\mathbf{h}_k((l-2)n+2 : ln), 2n)$
  end for
end for
$\mathbf{x}_f = \mathbf{0}$

▷ Multiplication
$\hat{\mathbf{v}} \leftarrow FFT(\mathbf{v}, 2n)$

for $k \leftarrow 1$ to α **do**
  ▷ Compute $\mathbf{w} = \mathbf{L}(\mathbf{g}_k)^T\mathbf{v}$
  $\hat{\mathbf{w}} \leftarrow \hat{\mathbf{g}}_k \odot \hat{\mathbf{v}}$
  $\mathbf{w} \leftarrow IFFT(\hat{\mathbf{w}}, 2n)$
  $\mathbf{w} \leftarrow \mathbf{w}(n : 2n-1)$

  ▷ Compute $\mathbf{x} = \bar{\mathbf{U}}(\mathbf{h}_k^T)^T\mathbf{w}$
  $\hat{\mathbf{w}} \leftarrow FFT(\mathbf{w}, 2n)$
  for $l \leftarrow 1$ to m/n **do**
    $\hat{\mathbf{x}}_l \leftarrow \hat{\mathbf{x}}_l + \hat{\mathbf{h}}_{k,l} \odot \hat{\mathbf{w}}$
  end for
end for

for $l \leftarrow 1$ to m/n **do**
  $\mathbf{x}_l \leftarrow IFFT(\hat{\mathbf{x}}_l, 2n)$
end for
▷ Construct $\mathbf{x}_f$ by vertical concatenation
$\mathbf{x}_f = [\mathbf{x}_1(1:n)\ ;\ \mathbf{x}_2(n{:}2n{-}1)\ ;\ \ldots\ ;\ \mathbf{x}_{m/n}(n{:}2n{-}1)]$
return $\mathbf{x}_f$

APPENDIX B – Gradients Calculation

We will use the following facts valid for any matrices $\mathbf{M}$, $\mathbf{N}$:

$$\|\mathbf{M}\|_F^2 = \mathrm{Tr}(\mathbf{M}^T\mathbf{M})$$
$$\mathrm{Tr}(\mathbf{M}+\mathbf{N}) = \mathrm{Tr}(\mathbf{M}) + \mathrm{Tr}(\mathbf{N})$$
$$\mathrm{Tr}(\mathbf{M}) = \mathrm{Tr}(\mathbf{M}^T)$$

Allied to the following rules for matrix derivatives (PETERSEN; PEDERSEN, 2012).

$$\frac{\partial}{\partial \mathbf{M}} \mathrm{Tr}(\mathbf{BMC}) = \mathbf{B}^T\mathbf{C}^T \quad , \quad \frac{\partial}{\partial \mathbf{M}} \mathrm{Tr}(\mathbf{BM}^T\mathbf{C}) = \mathbf{CB} \tag{B.1}$$
$$\frac{\partial}{\partial \mathbf{M}} \mathrm{Tr}(\mathbf{A}^T\mathbf{M}^T\mathbf{CMB}) = \mathbf{C}^T\mathbf{MAB}^T + \mathbf{CMBA}^T \tag{B.2}$$

Now we can easily derive the gradients:

$$\nabla_{\mathbf{D}}\|\mathbf{DX}-\mathbf{Y}\|_F^2 = \frac{\partial}{\partial \mathbf{D}} \mathrm{Tr}\left((\mathbf{DX}-\mathbf{Y})^T(\mathbf{DX}-\mathbf{Y})\right) \tag{B.3}$$
$$= \frac{\partial}{\partial \mathbf{D}} \left[\mathrm{Tr}(\mathbf{X}^T\mathbf{D}^T\mathbf{DX}) - 2\,\mathrm{Tr}(\mathbf{X}^T\mathbf{D}^T\mathbf{Y})\right] \tag{B.4}$$
$$= 2(\mathbf{DX}-\mathbf{Y})\mathbf{X}^T \tag{B.5}$$
$$\tag{B.6}$$
$$\nabla_{\mathbf{D}}\|\mathbf{\Delta}-\mathbf{D}+\mathbf{ADB}-\mathbf{Z}\|_F^2 = \frac{\partial}{\partial \mathbf{D}} \mathrm{Tr}\left((\mathbf{\Delta}-\mathbf{D}+\mathbf{ADB}-\mathbf{Z})^T(\mathbf{\Delta}-\mathbf{D}+\mathbf{ADB}-\mathbf{Z})\right) \tag{B.7}$$
$$= \frac{\partial}{\partial \mathbf{D}} \Big[-2\,\mathrm{Tr}((\mathbf{\Delta}^T-\mathbf{Z}^T)\mathbf{D}) + 2\,\mathrm{Tr}((\mathbf{\Delta}^T-\mathbf{Z}^T)\mathbf{ADB}) \tag{B.8}$$
$$+ \mathrm{Tr}(\mathbf{D}^T\mathbf{D}) - 2\,\mathrm{Tr}(\mathbf{D}^T\mathbf{ADB}) - \mathrm{Tr}(\mathbf{B}^T\mathbf{D}^T\mathbf{A}^T\mathbf{ADB})\Big] \tag{B.9}$$
$$= 2\left[\mathbf{D}-\mathbf{ADB}-\mathbf{\Delta}+\mathbf{Z} - \mathbf{A}^T(\mathbf{D}-\mathbf{ADB}-\mathbf{\Delta}+\mathbf{Z})\mathbf{B}^T\right] \tag{B.10}$$