PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

Efficient Algorithms for Convolutional Dictionary Learning via Accelerated Proximal Gradient

TESIS PARA OPTAR AL GRADO ACADÉMICO DE MAGÍSTER EN PROCESAMIENTO DE SEÑALES E IMÁGENES DIGITALES

AUTOR

Gustavo Manuel Silva Obregón

ASESOR

Paul Antonio Rodríguez Valderrama

Febrero, 2019

To God, for blessing me everyday of my life.

To my parents, my siblings and all my family, for their unconditional love.

Jesus looked at them and said, "With man this is impossible, but with God all things are possible".
(Matthew 19,26)

Abstract

Convolutional sparse representations and convolutional dictionary learning are mathematical models that consist in representing a whole signal or image as a sum of convolutions between dictionary filters and coefficient maps. Unlike the patch-based counterparts, these convolutional forms are receiving an increase attention in multiple image processing tasks, since they do not present the usual patch-wise drawbacks such as redundancy, multi-evaluations and non-translational invariant. Particularly, the convolutional dictionary learning (CDL) problem is addressed as an alternating minimization between coefficient update and dictionary update stages. A wide number of different algorithms based on FISTA (Fast Iterative Shrinkage-Thresholding Algorithm), ADMM (Alternating Direction Method of Multipliers) and ADMM consensus frameworks have been proposed to efficiently solve the most expensive steps of the CDL problem in the frequency domain. However, the use of the existing methods on large sets of images is computationally restricted by the dictionary update stage.

The present thesis report is strategically organized in three parts. On the first part, we introduce the general topic of the CDL problem and the state-of-the-art methods used to deal with each stage. On the second part, we propose our first computationally efficient method to solve the entire CDL problem using the Accelerated Proximal Gradient (APG) framework in both updates. Additionally, a novel update model reminiscent of the Block Gauss-Seidel (BGS) method is incorporated to reduce the number of estimated components during the coefficient update. On the final part, we propose another alternative method to address the dictionary update stage based on APG consensus approach. This last method considers particular strategies of the ADMM consensus and our first APG framework to develop a less complex solution decoupled across the training images. In general, due to the lower number of operations, our first approach is a better serial option while our last approach has as advantage its independent and highly parallelizable structure.

Finally, in our first set of experimental results, which is composed of serial implementations, we show that our first APG approach provides significant speedup with respect to the standard methods by a factor of $1.6 \sim 5.3$. A complementary improvement by a factor of 2 is achieved by using the reminiscent BGS model. On the other hand, we also report that the second APG approach is the fastest method compared to the state-of-the-art consensus algorithm implemented in serial and parallel. Both proposed methods maintain comparable performance as the other ones in terms of reconstruction metrics, such as PSNR, SSIM and sparsity, in denoising and inpainting tasks.

Keywords

Convolutional Sparse Representation, Convolutional Dictionary Learning, APG, Consensus

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sparse representations (SR) [1],[2] are widely used techniques, yielding effective results in the field of signal/image processing and computer vision. Although the patch-based structure performs very well in a broad range of applications such as object recognition, machine learning, etc. [1], the resulting representations are usually multi-valued and not optimized for the whole image due to the independent estimation among overlapping image patches. In particular, the convolutional scheme of SR [3],[4] overcomes the aforementioned issues by modeling an entire image as convolutions with dictionaries instead of modeling individual patches as linear operations. The most frequent formulation of this convolutional approach is an extension of the Convotutional Basis Pursuit Denoising (CBPDN), also known as Convolution Dictionary Learning (CDL) problem

$$\underset{\{\mathbf{x}_{k,m}\}\{\mathbf{d}_m\}}{\arg\min} \frac{1}{2}\sum_k \Big\| \sum_m \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k \Big\|_2^2 + \lambda \sum_k \sum_m \Big\| \mathbf{x}_{k,m} \Big\|_1 \qquad \text{s.t.} \quad \|\mathbf{d}_m\|_2 = 1 \ \ \forall m \ , \tag{1}$$

where $\{\mathbf{d}_m\}$ represents the set of M, $L_1 \times L_2$ dictionary filters, $\{\mathbf{x}_{k,m}\}$ the K sets of M coefficient maps (each one with $N_1 \times N_2$ samples), $\{s_k\}$ the K training images of size $N_1 \times N_2$, λ the regularization parameter and $*$ denotes the convolution operator. The norm constraint is required to avoid scaling ambiguities between coefficient maps and dictionary filters.

CDL is a non-convex problem when being jointly evaluated in both variables $\{\mathbf{x}_{k,m}\}$ and $\{\mathbf{d}_m\}$; however, by fixing either one, it can be recast as an alternating optimization [5] of two convex problems: coefficient update (sparse coding or SC) and dictionary update (dictionary learning or DL). Among the methods that efficiently handle these updates, works such as [4] and [6] have shown that the ADMM model [7, Ch. 3] computed in the frequency domain provides superior runtime performance in comparison to earlier spatial domain based methods [8],[9]. Moreover, the ADMM consensus approach [10] is considered as one of the fastest methods implemented in parallel due to its separable structure.

Most of these aforementioned ADMM-based algorithms have basically been proposed to deal with the most computationally demanding linear systems of the CDL problem in the frequency domain. Although these systems have closed-form solutions that can directly be tackled via either matrix inversion techniques or conjugate gradient methods [11], they can also be computationally expensive if the training set size becomes larger.

Considering that this stream of research aims to process large sets of images in real world applications, any significant improvement in the processing time would have a compelling effect. In the present report, for solving the CDL problem, we present two efficient algorithms based on the Accelerated Proximal Gradient framework [12, Ch. 4] that substantially outperform the existing serial and parallel ADMM implementations.

- In our first approach published in [13], we propose a novel algorithm that has two complementary contributions with respect to the standard CDL approaches. We first extend the use of an efficient APG-based solution partially computed in the frequency domain, previously introduced in [14] only for the sparse coding problem (9) to both CDL updates. We also describe an update model inspired on the Block Gauss-Seidel method [15]. It enables the computation of partial sets of coefficient maps during each sparse coding stage.
- In our second approach published in [16], we introduce an APG consensus based algorithm that enables to have a decoupled solution of the dictionary update across the training images. This method is particularly characterized by combining the parallelizable structure of the consensus model, the low complexity of the APG solution, as well as the convenience of computing most of the steps in the frequency domain.

Although we are working on the common CDL framework, in which the estimated filters are not separable, the APG approach is easy to extrapolate to a separable filter learning framework. In a related work published in [17], we have proved this last statement by proposing an efficient separable CDL algorithm based on our first APG approach plus a rank-1 constraint.

The rest of this work is organized as follows: Chapter 2 reviews the existing methods for the CDL problem and some technical details. In Chapter 3 and Chapter 4, we present a thorough description of our proposed methods with their corresponding experiments. In Chapter 5, we give our final remarks.

Chapter 2

Convolutional Sparse Representation

2.1 Preliminaries

2.1.1 Alternating Direction Method of Multiplier

ADMM [7, Ch. 3] is a distributed optimization algorithm that strategically blends the decomposability of dual ascent method and the good convergence properties of Augmented Lagrangian method (ALM) [7]. This algorithm can be applied to solve problems of the form

$$\underset{\{\mathbf{x}\},\{\mathbf{y}\}}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{y}) \qquad s.t. \quad A\mathbf{x} + B\mathbf{y} = c, \tag{2}$$

where the functions f and g must be convex, and the two primal variables, x and y, are related by a linear equation. An ALM formulation of (2), which is not separable (joint minimization), implies to estimate the primal variables simultaneously; however, an ADMM formulation, which is like a single Gauss-Seidel pass over x and y, performs the estimation in sequential fashion. Considering the ADMM iterations in scaled form, the derivation is given by

$$\begin{aligned}
\mathbf{x}^{(i+1)} &= \underset{\{\mathbf{x}\}}{\text{minimize}} \quad f(\mathbf{x}) + \frac{\rho}{2}\left\|A\mathbf{x} + B\mathbf{y}^{(i)} - c + \mathbf{u}^{(i)}\right\|_2^2 & (3)\\
\mathbf{y}^{(i+1)} &= \underset{\{\mathbf{y}\}}{\text{minimize}} \quad g(\mathbf{y}) + \frac{\rho}{2}\left\|A\mathbf{x}^{(i+1)} + B\mathbf{y} - c + \mathbf{u}^{(i)}\right\|_2^2 & (4)\\
\mathbf{u}^{(i+1)} &= \mathbf{u}^{(i)} + A\mathbf{x}^{(i+1)} + B\mathbf{y}^{(i+1)} - c, & (5)
\end{aligned}$$

where $\rho > 0$ is the penalty parameter, a.k.a. Augmented Lagrangian parameter, for the step size. In general, ADMM achieves a relatively accurate solution in a few tens of iterations, but requires many iterations for a highly accurate solution. Its convergence rate [7] is $O(1/\varepsilon)$ and its behavior is more like a first-order method (gradient descent, accelerated gradient descent, etc.) than a second-order method (Newton's method, conjugate gradient descent, etc.).

2.1.2 Proximal Gradient Descent

Proximal gradient descent (PGD) and its accelerated version (APG) [12, Ch. 4] are first-order methods that iteratively minimize differentiable or non-differentiable convex optimization problems via generalized form of projection (a gradient descent followed by a proximal operator). Let an objective function that can be expressed as the sum of two function:

$$\underset{\{\mathbf{x}\}}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{x}), \tag{6}$$

where f and g are convex, but f must also be differentiable in $\mathbb{R}^n$. From these assumptions, their convergence rates can be identical to corresponding gradient descent counterpart even when optimizing non-differentiable objective functions. The convergence rates of PGD and APG are $O(1/\varepsilon)$ and $O(1/\varepsilon^2)$, respectively. The main difference among them is that APG algorithm employs an additional linear combination among the previous estimations in order to accelerate the convergence. The corresponding proximal gradient method is defined as

$$\mathbf{x}^{i+1} = \text{prox}_{\alpha^i g}(\mathbf{x}^i - \alpha^i \nabla f(\mathbf{x}^i)), \tag{7}$$

where α is a step size ($0 < \alpha \leq 1/L$) and L is the constant Lipschitz that guarantee convergence. If L is not known, the step size can be estimated by a line search in each descent step. The proximal operator [12] of a convex function g is of the form

$$\text{prox}_g(\mathbf{z}) = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + g(\mathbf{x}). \tag{8}$$

2.2 Coefficient Update (sparse coding or SC)

Considering a linear system of equations $Dx = s$, the sparse coding and its convolutional form (CSC) are inverse problems that intend to find an sparse representation x (few non-zero elements) of a certain observed data s from the pre-trained dictionary filters D. Particularly, the standard CBPDN extension of the CSC problem is posed as

$$\underset{\{\mathbf{x}_m\}}{\arg\min} \frac{1}{2} \|\sum_m \mathbf{d}_m * \mathbf{x}_m - \mathbf{s}\|_2^2 + \lambda \sum_m \|\mathbf{x}_m\|_1 \ . \tag{9}$$

For mathematical convenience, we can define D_m as a linear operator such that $\mathbf{d}_m * \mathbf{x}_m = \mathbf{D}_m \mathbf{x}_m$. Using this linear operator, the notation of the problem (9) can be simplified by

$$\underset{\{X\}}{\arg\min} \frac{1}{2} \|\mathbf{D}\mathbf{x} - s\|_2^2 + \lambda \|\mathbf{x}\|_1 \ , \tag{10}$$

where $\mathbf{D} = (\mathbf{D}_1 \ \mathbf{D}_2 \ \cdots)$ and $\mathbf{x} = (\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots)^T$.

2.2.1 ADMM with Sherman-Morrison method

ADMM with Sherman Morrison method [4] is a mixed approach between the spatial and the frequency domain solutions, being the latter applied to efficiently solve the ℓ_2 fidelity term sub-problem using Sherman-Morrison formula. The optimization problem (10) can be expressed in ADMM form by adding a constraint term, in which the primary variable $\mathbf{x}$ must be equal to the auxiliary variable $\mathbf{y}$.

$$\underset{\{\mathbf{x}\},\{\mathbf{y}\}}{\arg\min}\ \frac{1}{2}\|\mathbf{D}\mathbf{x}-\mathbf{s}\|_2^2+\lambda\sum_m\|\mathbf{y}\|_1 \qquad \text{s.t.} \qquad \mathbf{x}-\mathbf{y}=0, \tag{11}$$

The associated ADMM updates are

$$\mathbf{x}^{(i+1)} = \underset{\{\mathbf{x}\}}{\arg\min}\frac{1}{2}\|\mathbf{D}\mathbf{x}-\mathbf{s}\|_2^2+\frac{\rho}{2}\left\|\mathbf{x}-\mathbf{y}^{(i)}+\mathbf{u}^{(i)}\right\|_2^2 \tag{12}$$

$$\mathbf{y}^{(i+1)} = \underset{\{\mathbf{y}\}}{\arg\min}\ \lambda\|\mathbf{y}\|_1+\frac{\rho}{2}\left\|\mathbf{x}^{(i+1)}-\mathbf{y}+\mathbf{u}^{(i)}\right\|_2^2 \tag{13}$$

$$\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)}+\mathbf{x}^{(i+1)}-\mathbf{y}^{(i+1)}. \tag{14}$$

The auxiliary variable update (13), which has a closed-form solution, is solved via soft thresholding function $\mathcal{S}_{\lambda/\rho}$.

$$\mathbf{y}^{(i+1)}=\mathcal{S}_{\lambda/\rho}\left(\mathbf{x}^{(i+1)}+\mathbf{u}^{(i)}\right), \tag{15}$$

where $\mathcal{S}_\gamma(x)=sign(x)\odot max(0,|x|-\gamma)$. Switching the spatial domain problem (12) to the frequency domain by means of the convolution theorem [18], the resulting minimizer is

$$(\hat{\mathbf{D}}^H\hat{\mathbf{D}}+\rho I)\hat{\mathbf{x}}=\hat{D}^H\hat{\mathbf{s}}+\rho(\hat{\mathbf{y}}-\hat{\mathbf{u}}), \tag{16}$$

where the hat symbol $(\hat{\cdot})$ is used to denote variables in the frequency domain.

As the matrix $\hat{\mathbf{D}}$ has a block structure of M concatenated $N\times M$ diagonal matrices, the operation $\hat{\mathbf{D}}^H\hat{\mathbf{D}}$, which has products among zeros, results in a large matrix of size $MN\times MN$. However, [14] noted that it is only necessary to solve N independent linear system of $M\times M$ since the resulting elements of products between zeros are not part of the final solution. Each independent system consists of a single sum between a rank-one tern and diagonal term, which inversion can easily be performed by applying the Sherman-Morrison formula on its rearranged form of non-zero elements.

2.2.2 Fast Iterative Shrinkage-Thresholding Algorithm

Iterative Shrinkage-Thresholding Algorithm (ISTA) and its accelerated variant Fast ISTA (FISTA) [19],[4] are particular PGD algorithms that have been used as an alternative to the ALM methods for the CSC problem. Due to the associated convolution operations of both methods, their computational complexity is high. However, research [4], which main topic is the frequency domain ADMM-based method for the CDL updates, explored the idea of reducing the complexity of the classic FISTA approach by computing the gradient in the frequency domain instead the spatial domain. As proposed in [4], the gradient of the ℓ_2 fidelity term can be rewritten in the frequency domain as

$$\nabla F(\frac{1}{2}\left\|\hat{\mathbf{X}}\hat{\mathbf{d}}-\hat{\mathbf{s}}\right\|_2^2)=\hat{\mathbf{X}}^H(\hat{\mathbf{X}}\hat{\mathbf{d}}-\hat{\mathbf{s}})\ , \tag{17}$$

where $\hat{\mathbf{X}}$, $\hat{\mathbf{d}}$ and $\hat{\mathbf{s}}$ denote the DFT transform of $\mathbf{X}$, $\mathbf{d}$ and $\mathbf{s}$, respectively. The steps of the classic FISTA is depicted in the next algorithm 2.1.

Algorithm 2.1: FISTA applied to (9)

```
Inputs: $\lambda$ (regularization parameter), $L$ (Lipschitz constant)
Step 0: Set $\mathbf{Y}_1 = \mathbf{X}_0$ (initial guess), $\delta_1 = 1$
Step $i$ : ($i \geq 1$) Compute
  1: $\mathbf{X}^i = \text{shrink}(\mathbf{X}^{i-1} + \frac{1}{L}\nabla F(X)|_{X=Y^i}, \frac{\lambda}{L})$
  2:     $\delta^{i+1} = (1 + \sqrt{1+4(\delta^{i+1})^2})$
  3:     $\mathbf{Y}^{i+1} = \mathbf{X}^{i+1} + \frac{\delta^i - 1}{\delta^{i+1}}(\mathbf{X}_k^{i+1} - \mathbf{X}^i)$
```

A brief summary of complexity of the reported CSC methods is presented in Table 2.1.

	Operations	Complexity per operation	**Dominant Term/**$(\mathbf{N_1} \cdot \mathbf{N_2} \cdot \mathbf{M} \cdot \mathbf{K})$
ADMM+SM	$FFT + IFFT$	$2 \cdot log(N_1 \cdot N_2) \cdot N_1 \cdot N_2 \cdot M \cdot K$	$2 \cdot log(N_1 \cdot N_2) + 10 \cdot \tau$
	$Inv.\ Problem\ (SM)$	$\tau \cdot (10 \cdot N_1 \cdot N_2 \cdot M \cdot K)$	
FISTA	$\nabla F(x)$	$(4 \cdot L_1 \cdot L_2 + 1) \cdot N_1 \cdot N_2 \cdot M \cdot K$	$6 \cdot L_1 \cdot L2 + 2$
	Step Size	$(2 \cdot L_1 \cdot L_2 + 1) \cdot N_1 \cdot N_2 \cdot M \cdot K$	
DFT-FISTA	$FFT + IFFT$	$2 \cdot log(N_1 \cdot N_2) \cdot N_1 \cdot N_2 \cdot M \cdot K$	$2 \cdot log(N_1 \cdot N_2) + 2 \cdot L_1 \cdot L_2 + 5 \cdot \tau + 1$
	$\nabla F(x)$	$\tau \cdot (3 \cdot N_1 \cdot N_2 \cdot M \cdot K)$	
	Step Size	$(2 \cdot L_1 \cdot L_2 + 1) \cdot N_1 \cdot N_2 \cdot M \cdot K$	

Table 2.1: Analysis of the computational complexity on the reported CSC methods. SM denotes the Sherman-Morrison algorithm, and τ is a constant used to represent the cost of complex operations ($2 \leq \tau \leq 4$).

2.3 Dictionary update (dictionary learning or DL)

The dictionary learning problem partially addressed in the frequency domain requires solving a constrained convolutional variant of the Method of Optimal Directions (MOD) [20], namely:

$$\underset{\{\mathbf{d}_m\}}{\arg\min} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \right\|_2^2 \quad \text{s.t} \quad \mathbf{d}_m \in C_{PN} \ . \tag{18}$$

C_{PN} is the constraint set for an adequate spatial support and normalized dictionary filters given by

$$C_{PN} = \{\mathbf{x} \in \mathbb{R}^N : (I - PP^T)\mathbf{x} = 0, \|\mathbf{x}\|_2 = 1\}, \tag{19}$$

where P represents the zero-padding projection operator. Denoting the indicator function of the set C_{PN} as $\iota_{C_{PN}}$, (18) can be rewritten in unconstrained form as

$$\underset{\{\mathbf{d}_m\}}{\arg\min} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \right\|_2^2 + \sum_m \iota_{C_{PN}}(\mathbf{d}_m) \ . \tag{20}$$

2.3.1 ADMM with iterated inversion techniques

This ADMM algorithm [4] is an extension of [14] for the DL update, in which the expensive linear system associated to ℓ_2 fidelity term sub-problem can be approached via a recursive variant of Sherman-Morrison formula or conjugate gradient method. The problem (20) is rewritten in suitable ADMM form using an auxiliary variable g_m

$$\underset{\{\mathbf{d}_m\}\{\mathbf{g}_m\}}{\arg\min} \frac{1}{2}\sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \right\|_2^2 + \sum_m \iota_{C_{PN}}(\mathbf{g}_m) \quad \text{s.t} \quad \mathbf{d}_m - \mathbf{g}_m = 0 \ \ \forall m \ , \tag{21}$$

where corresponding ADMM updates are

$$\mathbf{d}_m^{(i+1)} = \underset{\{\mathbf{d}_m\}}{\arg\min} \frac{1}{2}\sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_m - \mathbf{s}_k \right\|_2^2 + \frac{\beta}{2}\sum_m \left\| \mathbf{d}_m - \mathbf{g}_m^{(i)} + \mathbf{h}_m^{(i)} \right\|_2^2 \tag{22}$$

$$\mathbf{g}_m^{(i+1)} = \underset{\{\mathbf{g}_m\}}{\arg\min} \sum_m \iota_{C_{PN}}(\mathbf{g}_m) + \frac{\beta}{2}\sum_m \left\| \mathbf{d}_m^{(i+1)} - \mathbf{g}_m + \mathbf{h}_m^{(i)} \right\|_2^2 \tag{23}$$

$$\mathbf{h}_m^{(i+1)} = \mathbf{h}_m^{(i)} + \mathbf{d}_m^{(i+1)} - \mathbf{g}_m^{(i+1)} \ . \tag{24}$$

Like its counterpart for SC update, the estimation of the auxiliary variable is obtained by a proximal operator and the estimation of the primary variable is performed in the frequency domain. However, the inversion of the resulting frequency domain system, which consists of a sum between K rank-one terms and a diagonal term, is much more complicated and expensive. Due to this fact, the authors of [4] proposed two different alternative for the inverse problem.

I **Iterated Sherman-Morrison (ISM):** Due to the particular structure of the aforementioned linear systems, it is feasible to iteratively apply the Sherman-Morrison formula in order to get a solution. According to [4], this approach is often computational effective for small to moderate training set size since its complexity is given by a quadratic term corresponding to the training set size.

II **Conjugate Gradient (CG):** Iterative approach that is used to solve (22) without having to explicitly construct the matrix $\hat{X}^H\hat{X} + \beta I$ of the system. In comparison to ISM, this method depend of certain relative residual value (tolerance) to obtain a more accurate solution in the DL update. Experimental results of [4] have shown that a relative residual tolerance equals to 10^{-3} or higher is sufficient to reliably converge in the CDL algorithm as the ISM method. This method is recommended when the training set size is large.

2.3.2 ADMM consensus

The consensus approach [7, Ch.7] consists in defining a constraint of equality to decouple an optimization problem through independent local variables. This framework was first proposed for the DL problem by [6], but then [21] introduced a more complete ADMM consensus approach to improve the convergence rate by finding the best coupling variables that are passed between the SC and DL problems. As proposed in [21], the DL problem (20) can be expressed in ADMM consensus form as

$$\underset{\{\mathbf{d}_{k,m}\}\{\mathbf{g}_m\}}{\arg\min} \frac{1}{2}\sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_{k,m} - \mathbf{s}_k \right\|_2^2 + \sum_m \iota_{C_{PN}}(\mathbf{g}_m) \quad \text{s.t} \quad \mathbf{d}_{0,m} = \mathbf{d}_{2,m} = \ldots = \mathbf{g}_m \ \ \forall m \ , \tag{25}$$

where $\{\mathbf{d}_{k,m}\}$ are the M local dictionary filters for each training image and $\{\mathbf{g}_m\}$ is the global consensus variable of M filters. The associated ADMM updates are

$$\begin{aligned}
\mathbf{d}_{k,m}^{(i+1)} &= \operatorname*{arg\,min}_{\{\mathbf{d}_{k,m}\}} \frac{1}{2}\sum_k \Big\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_{k,m} - \mathbf{s}_k \Big\|_2^2 + \frac{\beta}{2}\sum_k\sum_m \Big\| \mathbf{d}_{k,m} - \mathbf{g}_m^{(i)} + \mathbf{h}_{k,m}^{(i)} \Big\|_2^2 && (26)\\
\mathbf{g}_m^{(i+1)} &= \operatorname*{arg\,min}_{\{\mathbf{g}_m\}} \sum_m \iota_{C_{PN}}(\mathbf{g}_m) + \frac{\beta}{2}\sum_k\sum_m \Big\| \mathbf{d}_{k,m}^{(i+1)} - \mathbf{g}_m + \mathbf{h}_{k,m}^{(i)} \Big\|_2^2 && (27)\\
\mathbf{h}_{k,m}^{(i+1)} &= \mathbf{h}_{k,m}^{(i)} + \mathbf{d}_{k,m}^{(i+1)} - \mathbf{g}_m^{(i+1)} \ . && (28)
\end{aligned}$$

The $\{\mathbf{d}_{k,m}\}$ update (26), which is the most computationally demanding step, consists of K independent linear systems that can be efficiently solved via the DFT domain Sherman Morrison method used in [14] (described in Section 2.2.1). The $\{\mathbf{g}_m\}$ update (27) has a closed-form solution given by

$$\mathbf{g}_m^{(i+1)} = \mathrm{prox}_{\iota_{C_{\mathrm{PN}}}}\Big(\frac{1}{K}\sum_k (\mathbf{d}_{k,m}^{(i+1)} + \mathbf{h}_{k,m}^{(i)})\Big) \ . \qquad (29)$$

It is easy to note that the ADMM consensus approach has as a benefit its independent separable structure, which is highly parallelizable. However, separability [10] has recently been exploited by a parallel algorithm running in multiple cores. This parallel implementation is the fastest algorithm compared to the existing ones.

A brief summary of complexity of the reported DL methods is presented in Table 2.2.

	Operations	Complexity per operation	**Dominant Term/$(\mathbf{N_1 \cdot N_2 \cdot M \cdot K})$**
ADMM + ISM	$FFT + IFFT$	$2 \cdot log(N_1 \cdot N_2) \cdot N_1 \cdot N_2 \cdot M$	$\frac{\tau \cdot (9+5 \cdot K)}{2}$
	$Complex\ Inv.\ Problems\ (ISM)$	$(\tau \cdot \frac{(9+5 \cdot K)}{2}) \cdot N_1 \cdot N_2 \cdot M \cdot K$	
ADMM consensus	$FFT + IFFT$	$2 \cdot log(N_1 \cdot N_2) \cdot N_1 \cdot N_2 \cdot M \cdot K$	$2 \cdot log(N_1 \cdot N_2) + 10 \cdot \tau$
	$Inv.\ Problem\ (SM)$	$\tau \cdot (10 \cdot N_1 \cdot N_2 \cdot M \cdot K)$	

Table 2.2: Analysis of the computational complexity of the reported DL methods. SM denotes the Sherman-Morrison algorithm, ISM the iterated version, and τ is a constant used to represent the cost of complex operations ($2 \leq \tau \leq 4$).

2.4 Separable filter Learning

Separable filter learning is a relatively new research branch in which is trying to address optimization problems via separable filter formulations instead of non-separable ones. The convenience of using separable filters was firstly approached for Convolutional Neural Network (CNN) applications in [22]. As the synthesis structure of CNNs is formed of a large amount of non-separable filters, this aforementioned work proposed to approximate the non-separable filters as a linear combination of a smaller number of separable ones. They reported that this approximation considerably reduced the execution time with no loss in performance for classification and denoising tasks.

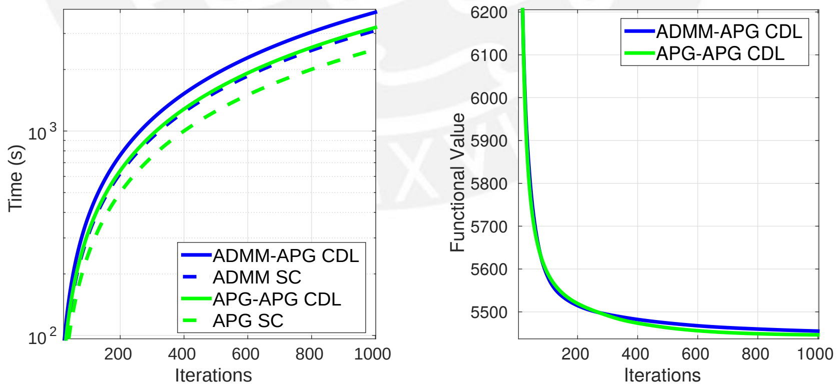
Other obvious application is on the Convolution Sparse Coding topic, [23] proposed a FISTA-based algorithm to perform the minimization through convolutions with separable filters. The corresponding experiments also shown that a small combination of separable filters provides better results in terms of runtime performance than a large set of non-separable filters.

Chapter 3

Proposed Method 1 : Partial Update APG approach

3.1 Frequency domain APG

An accelerated proximal gradient (APG) algorithm is mainly composed by four steps: a gradient estimation, a step size estimation, a proximal operator, and a Nesterov's accelerated gradient calculation. Most of them could be computationally demanding if they are addressed in the spatial domain. Due to this fact, we propose to perform most of these steps in the frequency domain, keeping only the proximal operator in the spatial domain, to avoid unnecessary convolutions or transformations between both domains.



(a) Runtime performance (b) Convergence performance

Figure 3.1: Comparison of the CDL algorithm using APG and ADMM frameworks for solving the SC update. Note: These preliminary experiments were simulated using a training set of 40 images. CDL = convolutional dictionary learning and SC = sparse coding

It worth nothing that we have heuristically observed that the most suitable way to handle both updates of the problem (1) is using the APG approach in both cases. In the Figure 3.1, we show evidences that this strategy delivers better runtime performance (1.25 times faster) with similar convergence rate in compassion to an ADMM-APG combination. On the other hand, all details of our proposed APG method for solving the DL problem (20) are explained in Section 3.1.1 as well as our extension of the FISTA algorithm for the SC update in Section 3.1.2.

3.1.1 Dictionary update

To perform the gradient estimation of the ℓ_2 fidelity term (20), labeled ∇F, in the frequency domain, we define a linear operator of $\mathbf{X}_{k,m}$ such that $\mathbf{x}_{k,m} * \mathbf{d}_m = \mathbf{X}_{k,m}\mathbf{d}_m$. We also denote the variables $\mathbf{X}_{k,m}$, $\mathbf{d}_m$ and $\mathbf{s}_k$ in the frequency domain as $\hat{\mathbf{X}}_{k,m}$ (diagonal matrix), $\hat{\mathbf{d}}_m$ and $\hat{\mathbf{s}}_k$ (column vectors), respectively. Accordingly, the fidelity term can be arranged as follows:

$$\frac{1}{2}\sum_k \left\| \sum_m \hat{\mathbf{X}}_{k,m}\hat{\mathbf{d}}_m - \hat{\mathbf{s}}_k \right\|_2^2 = \frac{1}{2}\sum_k \left\| \hat{\mathbf{X}}_k\hat{\mathbf{D}} - \hat{\mathbf{s}}_k \right\|_2^2 = \frac{1}{2}\left\| \hat{\mathbf{X}}\hat{\mathbf{D}} - \hat{\mathbf{S}} \right\|_2^2 , \tag{30}$$

where

$$\hat{\mathbf{X}}_k = (\hat{\mathbf{X}}_{k,1}\ \hat{\mathbf{X}}_{k,2}\ \cdots), \quad \hat{\mathbf{X}} = \begin{pmatrix} \hat{\mathbf{X}}_1 \\ \hat{\mathbf{X}}_2 \\ \vdots \end{pmatrix} \quad \hat{\mathbf{D}} = \begin{pmatrix} \hat{\mathbf{d}}_1 \\ \hat{\mathbf{d}}_2 \\ \vdots \end{pmatrix} \quad \text{and} \quad \hat{\mathbf{S}} = \begin{pmatrix} \hat{\mathbf{s}}_1 \\ \hat{\mathbf{s}}_2 \\ \vdots \end{pmatrix}$$

The inexact line search as back-tracking [24],[25] is a customary option since the exact counterpart [26],[27] could be computationally prohibitive. However, in the frequency domain, an exact line search can be effectively perform via (31).

$$\underset{\{\alpha\}}{\arg\min}\ \frac{1}{2}\left\| \hat{\mathbf{X}}(\hat{\mathbf{D}} - \alpha\nabla F(\hat{\mathbf{D}})) - \hat{\mathbf{S}} \right\|_2^2 \tag{31}$$

The derivation of this exact line search problem results in a single step size given by

$$([\hat{\mathbf{X}}\nabla F(\hat{\mathbf{D}})]^T[\hat{\mathbf{X}}\nabla F(\hat{\mathbf{D}})])\alpha = [\nabla F(\hat{\mathbf{D}})^T \nabla F(\hat{\mathbf{D}})] \ . \tag{32}$$

Algorithm 3.1: DL algorithm using Frequency domain APG

```
1: Compute gradient in the frequency domain
       $\nabla F(\hat{\mathbf{G}}^i) = \hat{\mathbf{X}}^H(\hat{\mathbf{X}}\hat{\mathbf{G}}^i - \hat{\mathbf{S}})$
2: Compute step size in the frequency domain
       $\alpha = \|\nabla F(\hat{\mathbf{G}}^i)\|_2^2 / \|\hat{\mathbf{X}}\nabla F(\hat{\mathbf{G}}^i)\|_2^2$
3: Compute dictionary
       $\mathbf{H}^{i+1} = IFFT2\{\hat{\mathbf{G}}^i - \alpha \cdot \nabla F(\hat{\mathbf{G}}^i)\}$
       $\mathbf{D}^{i+1} = \ \mathrm{prox}_{\iota_{C_{PN}}}(\mathbf{H}^{i+1})$
       $\hat{\mathbf{D}}^{i+1} = \ FFT2\{\mathbf{D}^{i+1}\}$
4: Compute auxiliary dictionary $\hat{\mathbf{G}}^{i+1}$ (Nesterov accelerated method) in the frequency domain
       $\gamma^{i+1} = (1 + \sqrt{1 + 4(\gamma^{i+1})^2})$
       $\hat{\mathbf{G}}^{i+1} = \hat{\mathbf{D}}^{i+1} + \frac{\gamma^i - 1}{\gamma^{i+1}}(\hat{\mathbf{D}}^{i+1} - \hat{\mathbf{D}}^i)$
5: Compute normalization of auxiliary dictionary
       $\hat{\mathbf{G}}^{i+1} = \sqrt{N} \cdot \hat{\mathbf{G}}^{i+1} / \|\hat{\mathbf{G}}^{i+1}\|_2$
```

The last step of our algorithm is a normalization of the auxiliary dictionary which is required to avoid the scaling ambiguities when it is passed to the other sub-problem. Using Parseval's theorem [18], we extend this normalization to the frequency domain as $\|\hat{\mathbf{D}}\|_2/\sqrt{N} = 1$, where N is the number of pixels.

3.1.2 Coefficient update

This coefficient update is an extension of the FISTA approach (more generally known as APG approach) presented in Section 2.2.2, in which now most steps of the solution are posed in the frequency domain. Similar to the previous APG approach for the DL update (Section 3.1.1), we define the linear operator D_m, such that $\mathbf{d}_m * \mathbf{x}_{k,m} = \mathbf{D}_m \mathbf{x}_{k,m}$, and denote $\mathbf{D}_m$, $\mathbf{x}_{k,m}$ and $\mathbf{s}_k$ in the frequency domain as $\hat{\mathbf{D}}_m$, $\hat{\mathbf{x}}_{k,m}$ and $\hat{\mathbf{s}}_k$. The fidelity term of the full SC problem (9) is rearranged as

$$\frac{1}{2}\sum_k \left\| \sum_m \hat{\mathbf{D}}_m \hat{\mathbf{x}}_{k,m} - \hat{\mathbf{s}}_k \right\|_2^2 = \frac{1}{2}\sum_k \left\| \hat{\mathbf{D}}\hat{\mathbf{X}}_k - \hat{\mathbf{s}}_k \right\|_2^2, \tag{33}$$

where $\hat{\mathbf{D}} = (\hat{\mathbf{D}}_1 \ \hat{\mathbf{D}}_2 \ \cdots)$ and $\hat{\mathbf{X}}_k = (\hat{\mathbf{x}}_{k,1} \ \hat{\mathbf{x}}_{k,2} \ \cdots)^T$. For computational convenience justified in the previous Section 3.1.1, we also compute the step size in the frequency domain from the exact line search, namely:

$$\underset{\{\alpha_k\}}{\arg\min} \frac{1}{2}\left\| \hat{\mathbf{D}}(\hat{\mathbf{X}}_k - \alpha_k \nabla F(\hat{\mathbf{X}}_k)) - \hat{\mathbf{s}}_k \right\|_2^2. \tag{34}$$

The completed mathematical description of our APG approach is presented in the algorithm 3.1 and algorithm 3.2.

Algorithm 3.2: SC algorithm using Frequency domain APG

```
1: Compute gradient in the frequency domain
     $\nabla F_k(\hat{\mathbf{Y}}_k^{i+1}) = \hat{\mathbf{G}}^H(\hat{\mathbf{G}}\hat{\mathbf{Y}}_k^i - \hat{\mathbf{s}}_k)$
2: Compute step size in the frequency domain
     $\alpha_k = \|\nabla F_k(\hat{\mathbf{Y}}_k^{i+1})\|_2^2 / \|\hat{\mathbf{G}}\nabla F_k(\hat{\mathbf{Y}}_k^{i+1})\|_2^2$
3: Compute Coef. Maps
     $\mathbf{U}_k^{i+1} = IFFT2\{\hat{\mathbf{Y}}_k^i - \alpha_k \cdot \nabla F_k(\hat{\mathbf{Y}}_k^{i+1})\}$
     $\mathbf{X}_k^{i+1} = \text{Shrink}_{\lambda\alpha_k}(\mathbf{U}_k^{i+1})$
     $\hat{\mathbf{X}}_k^{i+1} = FFT2\{\mathbf{X}_k^{i+1}\}$
4: Compute auxiliary coef. map $\hat{\mathbf{Y}}_k^{i+1}$ (Nesterov accelerated method) in the frequency domain
     $\delta^{i+1} = (1 + \sqrt{1 + 4(\delta^{i+1})^2})$
     $\hat{\mathbf{Y}}_k^{i+1} = \hat{\mathbf{X}}_k^{i+1} + \frac{\delta^i - 1}{\delta^{i+1}}(\hat{\mathbf{X}}_k^{i+1} - \hat{\mathbf{X}}_k^i)$
```

3.2 Partial update model

Given an efficient dictionary update implementation (as proposed in Section 3.1), we noted that the coefficient update becomes the dominant part of the whole CDL problem. With this in mind, we explore a new update model inspired by BGS method [15] (a.k.a. Alternating Optimization [5]) which raises the optimization problem for a given function $f(x)$ as

$$\mathbf{x}_r^{i+1} = \arg\min_{\mathbf{y}\in x_r} f(\mathbf{x}_1^{i+1},\ \cdots,\ \mathbf{x}_{r-1}^{i+1},\ \mathbf{y},\ \mathbf{x}_{r+1}^{i+1},\ \cdots,\ \mathbf{x}_R^{i+1})\ . \tag{35}$$

From this approach (35), the minimization is established for a single partition of the interest variable while keeping the other partitions fixed.

Adapting this model to the CDL problem, the coefficient update (9) can be written as

$$\mathbf{x}_{k,m}^{(i+1,r)} = \arg\min_{\{\mathbf{x}_{k,m}\}} \frac{1}{2}\sum_{k=1}^{P_r}\left\|\sum_m \mathbf{d}_m * \mathbf{x}_{k,m} - \mathbf{s}_k^{(r)}\right\|_2^2 + \lambda\sum_{k=1}^{P_r}\sum_m \|\mathbf{x}_{k,m}\|_1 \tag{36}$$

where the dataset $\{s_k\}$ is divided into R partitions ($\mathbf{s}_k = \{\mathbf{s}_k^{(1)},\ \mathbf{s}_k^{(2)}, \ldots, \mathbf{s}_k^{(R)}\}$) and P_r represents the partition size. A single partition of coefficient maps $\{\mathbf{x}_{k,m}^{(i,r)}\}$ is estimated in each outer-loop of the CDL problem.

The complete variable of coefficient maps $\{\mathbf{x}_{k,m}\}$ is composed from the current estimated partition and its previous values of the other partitions.

$$\mathbf{x}_{k,m}^{(i+1)} = [\mathbf{x}_{k,m}^{(i,1)},\ \ldots\ ,\ \mathbf{x}_{k,m}^{(i+1,r)},\ \ldots\ ,\ \mathbf{x}_{k,m}^{(i,R)}] \tag{37}$$

This complete set of coefficient maps is used to estimate the current dictionary given by (20), reproduced here for convenience:

$$\mathbf{d}_m^{(i+1)} = \arg\min_{\{\mathbf{d}_m\}} \frac{1}{2}\sum_k\left\|\sum_m \mathbf{x}_{k,m}^{(i+1)} * \mathbf{d}_m^{(i)} - \mathbf{s}_k\right\|_2^2 + \sum_m \iota_{C_{PN}}(\mathbf{d}_m^{(i)}).$$

At the end of each outer-loop the variable r is reassigned as $\{(r+1)\ mod\ R\}$ to periodically switch from the first to the last partition.

As this model is a generic structure, graph presented in Figure 3.2 for illustrative purposes, it could be applied to any CDL framework to solve each subproblem. However, we choose to merge this model with our APG-based solution since in the Section 3.3, we will show that our APG algorithm is computationally more efficient than the other approaches.

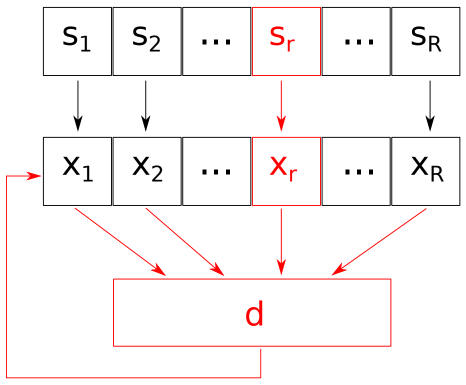


Figure 3.2: Partial Update model of the CDL problem.

3.3 Experimental Results

Two distinct set of experiments were carried out on a desktop computer equipped with an Intel i7-7700K CPU (4.20 GHz, 8MB Cache, 32GB RAM).

• **Learning stage:** The convergence and computational runtime of the CDL algorithms listed in Section 3.3.1 are compared. The training sets used for this first experiment consist of 5, 10, 20 and 40 gray-scale images of size 256×256 pixels, cropped and rescaled from a set of images obtained from the MIRFFLICKR-IM dataset [28]. Furthermore, for each training set, 32 dictionary filters of size 12×12 were learned using a sparsity parameter λ = 0.2 and 1000 iterations.

• **Testing stage:** As second set of experiments, the performance of the learned dictionaries from 40 training images is evaluated in terms of PSNR, SSIM, and sparsity metrics[1] for the denoising and inpainting CSC tasks presented in [14] and [29], respectively. Both test algorithms (denoising and inpainting) correspond to the ADMM-based MATLAB code of the SPORCO library available on [30]. For the simulations, we used eight standard test images that were corrupted with AWGN $\sigma = 0.2$ for the denoising task and dropped 20 % of pixels for the inpainting task.

Since the test algorithms have an adjustable parameter λ, in order to ensure a fair comparison, a search grid over $\lambda \in$ [0.001 - 0.95] was used to find the optimal value that provides the best PSNR for each learned dictionary.

3.3.1 Details of the evaluated methods

We compared the following CDL algorithms:

○ **Iterated Sherman-Morrison (ISM)** and **Conjugate Gradient (CG):** The CDL algorithms as proposed in [4].

○ **ADMM Consensus (ADMM-C):** The CDL algorithm proposed in [21].

○ **PU-APG:** Our proposed APG-based algorithm with partial update structure (code available on [31]). It is worth noting that PU-APG solution with a single partition is equivalent to the APG-based method proposed in Section 3.1 without any partial update model.

3.3.2 Simulations of the learning stage

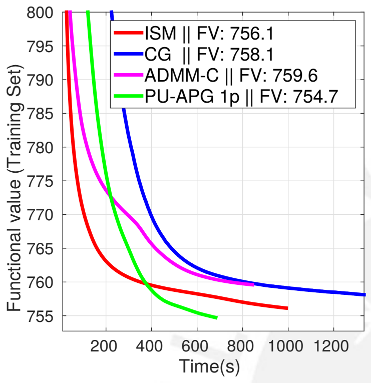
Since in each outer-loop, the PU-APG algorithm updates a single partition of the coefficient maps, this directly affects the l_1-term when computing the functional value (FV) of the training set, making it an unsuitable convergence reference. A fair comparison of the functional value would be with the validation set whereas more than one partition is used.

We show in Figure 3.3, the performance of the existing CDL methods (ISM, CG and ADMM-C) along with our proposed method in terms of functional values of (1) with respect to the learning runtime, using 5, 10, and 20 training images. We observe that all methods converge to similar values with distinct run-times and behaviors. Our proposed method outperforms the rest by achieving the same

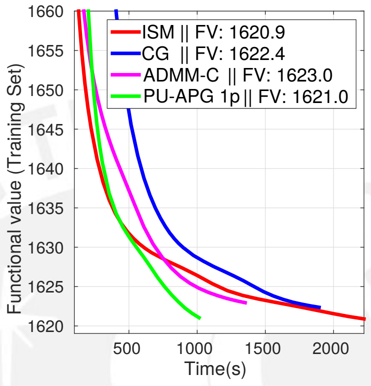
[1]Sparsity measure (L0 %) is defined as $100 \cdot \|x\|_0/N$, where x is the coefficient maps and N is the number of pixel in a test image.

functional value in less time, but for small training images, ISM has a relatively better convergence at the beginning.

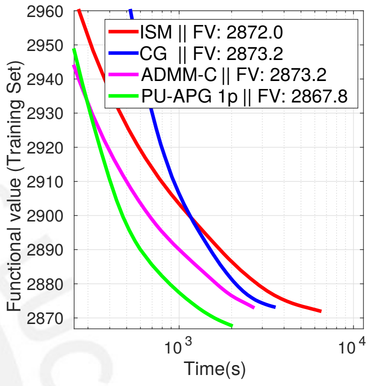
We report in Figure 3.4 the same comparisons as in the Figure 3.3, but for a larger training set size (40 training images). It also includes the progress of the functional value in the validation set (5 validation images) in order to observe the generalization of the estimated dictionaries with respect to the training runtime. We note that while the number of training images becomes larger our APG algorithm consistently outperforms the other ones in terms of runtime and functional value. Furthermore, it can be seen in Figure 3.4.b that our method provides a similar generalization as the other ones.



(a) 5 training images

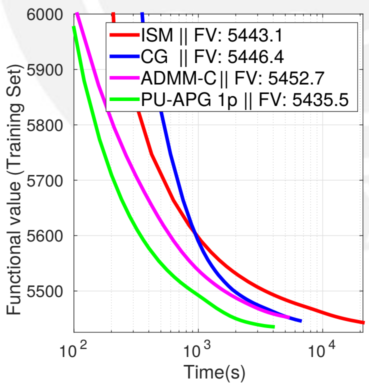


(b) 10 training images

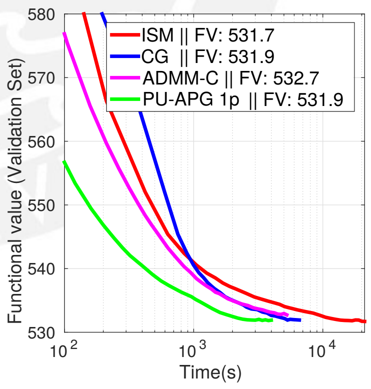


(c) 20 training images

Figure 3.3: Comparison of the functional value decay with respect to execution time on the learning process of 5, 10 and 20 training images.



(a) FV of the training set

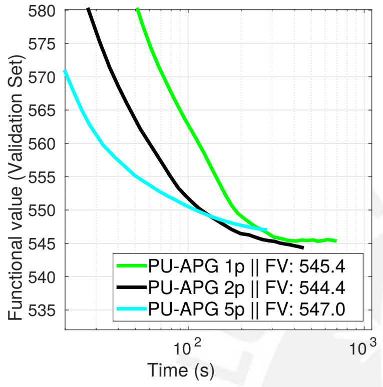


(b) FV of the validation set

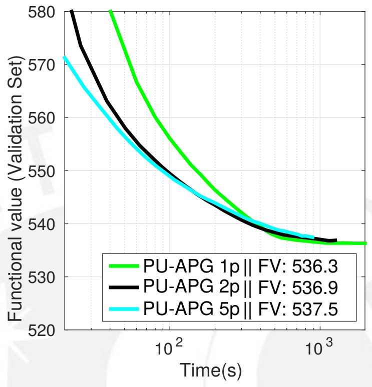
Figure 3.4: Comparison of the value decay of **the training and validation functional** with respect to execution time on the learning process of 40 training images.

In Figure 3.5, we present the comparison of our APG-based algorithm with the partial update structure for 1, 2 and 5 partitions. As can be observed the PU-APG with a single partition achieves a lower point of convergence, which means that its learned dictionaries must be better at generalizing the test set. Although PU-APG with 2 partitions achieves a slightly higher point of convergence, it is reached in less execution time. On the other hand, PU-APG with 5 partitions is faster, but it also needs more iterations to converge while the image set becomes larger.

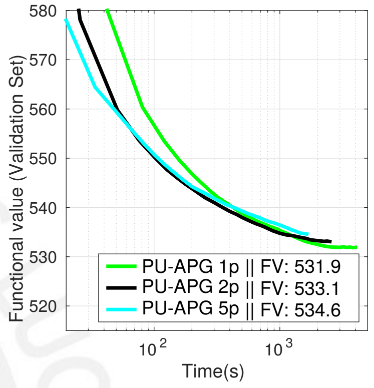
In Figure 3.6, we present the average execution time per iteration of the CDL methods during the learning with respect to the training set size and filter set size. Our proposed method provides better runtime per iteration in comparison to the other ones. As additional information, we report the exact values of runtime during the training in Table 3.1. In this table, we note that our proposed APG algorithm (PU-APG-1P), when using 5 to 40 training images, is about 1.5 to 5.3 times faster than ISM, about 2.5 times faster than the CG, and about 1.5 times faster than ADMM-C. The proposed update model applied in our algorithm provides an additional improvement of 1.6 and 2.5 times in the computational performance using 2 and 5 partitions, respectively.



(a) 5 training images

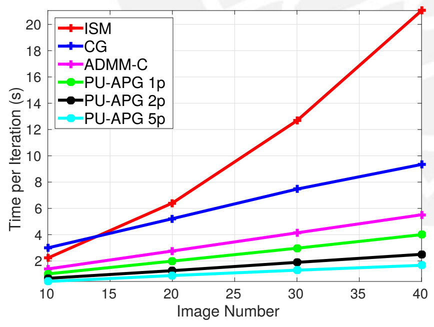


(b) 20 training images

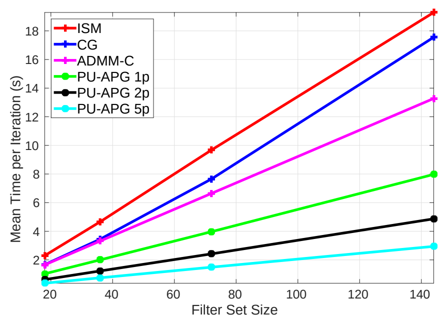


(c) 40 training images

Figure 3.5: Comparison of the value decay of **validation functional** with respect to execution time using our proposed method with 1, 2 and 5 partitions on the learning process of 5, 20 and 40 training images.



(a) 32 fixed dictionary filters



(b) 20 fixed training images

Figure 3.6: Average time per iteration of the CDL methods with respect to (a) training set size and (b) filter set size.

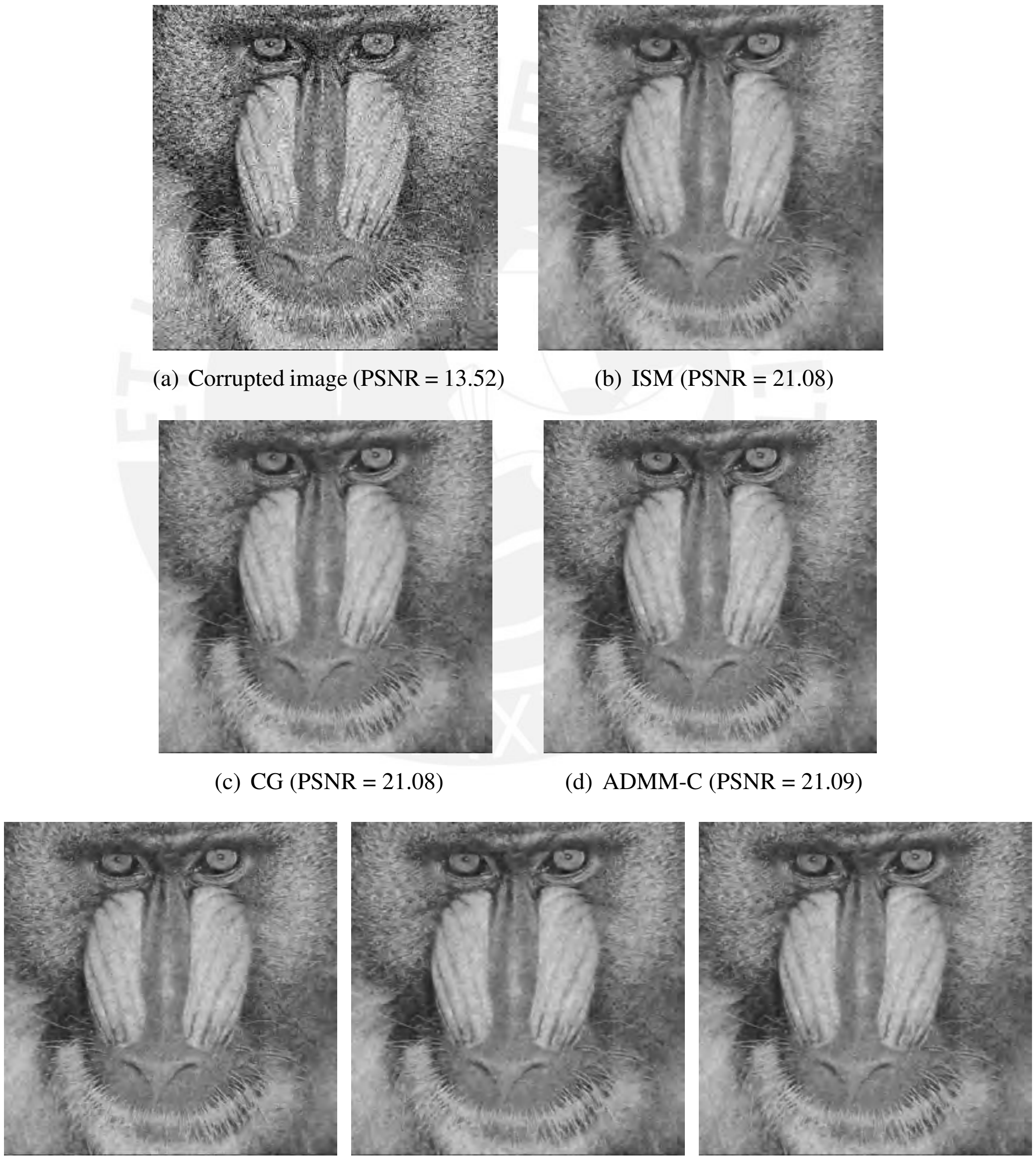
		ISM	**CG**	**ADMM-C**	**PU-APG 1p**	**PU-APG 2p**	**PU-APG 5p**
Training	5	991	1333	843	682	444	274
set size	40	21051	9343	5866	4003	2504	1668
Filter	18	3255	2053	1576	1286	824	526
set size	144	28709	24636	12623	10101	6565	4280

Table 3.1: Execution time (seconds) during the learning process of the evaluated CDL methods.

3.3.3 Simulations of the testing stage

Table 3.2 and Table 3.3 present the performance of the learned dictionaries in terms of PSNR, SSIM and sparsity metrics for the denoising and inpainting task, respectively. In both tables, we observe that the dictionaries learned by our proposed algorithm yield equivalent results as the existing methods, since the maximum differences in terms of PSNR, SSIM and sparsity between the worst metrics of our methods and the best metrics of the other methods are only equal to 0.16 dB, 0.0231 and 1.21%, respectively. These maximum differences are negligible.

A visual representation of the denoising process is presented in Figure 3.7.



(a) Corrupted image (PSNR = 13.52) (b) ISM (PSNR = 21.08)

(c) CG (PSNR = 21.08) (d) ADMM-C (PSNR = 21.09)

(e) PU-APG 1p (PSNR = 21.08) (f) PU-APG 2p (PSNR = 21.08) (g) PU-APG 5p (PSNR = 21.08)

Figure 3.7: Denoising of the corrupted Mandrill image using the learned dictionaries by the reported CDL methods.

		Mandrill	**Barbara**	**Peppers**	**Boats**	**House**	**Gold hill**	**Monarch**	**Airplane**
PSNR (dB)	ISM	21.08	23.15	25.17	25.36	24.90	24.70	24.95	24.28
	GC	21.08	23.15	25.17	25.35	24.90	24.70	24.94	24.28
	ADMM-C	21.09	23.14	25.15	25.34	24.89	24.69	24.94	24.30
	PU-APG 1p	21.08	23.15	25.17	25.36	24.95	24.71	24.98	24.29
	PU-APG 2p	21.08	23.11	25.17	25.37	24.99	24.71	24.96	24.29
	PU-APG 5p	21.08	23.08	25.15	25.34	24.91	24.70	24.90	24.27
SSIM	ISM	0.5286	0.6091	0.6741	0.6818	0.6553	0.6308	0.7009	0.7162
	GC	0.5282	0.6091	0.6739	0.6816	0.6735	0.6307	0.7255	0.7161
	ADMM-C	0.5293	0.6082	0.6738	0.6805	0.6731	0.6303	0.7255	0.7172
	PU-APG 1p	0.5293	0.6093	0.6745	0.6829	0.6587	0.6310	0.7029	0.7174
	PU-APG 2p	0.5293	0.6084	0.6749	0.6834	0.6589	0.6315	0.7030	0.7177
	PU-APG 5p	0.5280	0.6076	0.6746	0.6825	0.6571	0.6311	0.7024	0.7182
L0 %	ISM	7.46	5.68	1.52	1.58	3.23	1.40	3.29	2.71
	GC	7.48	5.69	1.51	1.60	2.02	1.40	2.09	2.70
	ADMM-C	7.63	5.73	1.51	1.58	2.15	1.43	2.13	2.74
	PU-APG 1p	7.43	5.61	1.49	1.55	3.21	1.40	3.25	2.67
	PU-APG 2p	7.38	5.56	1.46	1.53	3.14	1.38	3.24	2.67
	PU-APG 5p	7.38	5.57	1.48	1.57	3.23	1.36	3.30	2.69

Table 3.2: Denoising results for a standard set of test images, considering dictionary sets of size $12 \times 12 \times 32$ learned using a set of 40 training images.

		Mandrill	**Barbara**	**Peppers**	**Boats**	**House**	**Gold hill**	**Monarch**	**Airplane**
PSNR (dB)	ISM	29,86	40,14	37,33	44,89	43,56	38,94	41,62	42,42
	GC	29,86	40,16	37,34	44,89	43,57	38,94	41,63	42,42
	ADMM-C	29,70	40,24	37,26	44,84	43,45	38,87	41,61	42,41
	PU-APG 1p	29,83	40,22	37,30	44,87	43,63	38,95	41,54	42,49
	PU-APG 2p	29,80	40,18	37,29	44,87	43,50	38,94	41,47	42,44
	PU-APG 5p	29,84	40,13	37,30	44,83	43,64	38,92	41,63	42,44
SSIM	ISM	0.9467	0.9855	0.9400	0.9909	0.9875	0.9743	0.9917	0.9901
	GC	0.9468	0.9856	0.9400	0.9909	0.9876	0.9743	0.9917	0.9901
	ADMM-C	0.9451	0.9855	0.9382	0.9907	0.9874	0.9737	0.9916	0.9900
	PU-APG 1p	0.9459	0.9854	0.9392	0.9908	0.9877	0.9742	0.9915	0.9901
	PU-APG 2p	0.9446	0.9850	0.9391	0.9908	0.9873	0.9742	0.9915	0.9900
	PU-APG 5p	0.9461	0.9840	0.9397	0.9908	0.9876	0.9741	0.9916	0.9900
L0 %	ISM	82.27	86.76	70.45	90.57	91.24	82.63	89.95	90.83
	GC	82.40	86.88	70.60	90.71	91.30	82.75	89.98	90.95
	ADMM-C	81.58	86.10	66.39	89.91	90.46	81.79	89.39	90.15
	PU-APG 1p	81.77	86.04	69.84	89.97	90.42	81.90	89.57	90.26
	PU-APG 2p	80.09	86.32	69.69	90.09	81.09	81.99	89.61	90.28
	PU-APG 5p	81.67	96.95	69.79	90.14	90.63	82.11	89.38	90.40

Table 3.3: Inpainting results for a standard set of test images, considering dictionary sets of size $12 \times 12 \times 32$ learned using a set of 40 training images.

Chapter 4

Proposed Method 2 : APG consensus approach

A general global consensus problem [12, Ch. 5] has the form

$$\underset{\{\mathbf{y}\}}{\text{minimize}} \quad \sum_{j=1}^{N} f_j(\mathbf{y}_j) \;\; \text{s.t} \;\; \mathbf{y}_1 = \mathbf{y}_2 = \cdots = \mathbf{y}_N \ , \tag{38}$$

where the common global variable is not part of this problem as in the case of the ADMM consensus formulation (25). Via the definition of the constraint set

$$C = \{(\mathbf{y}_1, \cdots, \mathbf{y}_N) | \mathbf{y}_1 = \cdots = \mathbf{y}_N\} \ , \tag{39}$$

The optimization problem (38) can be switched to a special consensus form, resulting in the unconstrained problem given by

$$\underset{\{\mathbf{y}\}}{\text{minimize}} \quad \sum_{j=1}^{N} f_j(\mathbf{y}_j) + I_C(\mathbf{y}_1, \cdots, \mathbf{y}_N) \ , \tag{40}$$

where this type of formulation allows performing a proximal-based solution [12]. In comparison to independent linear systems solved via inversion techniques (26), a proximal solution would entail simple independent steps in direction opposite to each gradient.

4.1 Frequency domain APG Consensus

Rewriting the unconstrained DL problem (20) in form of the particular consensus model presented in (40), it becomes

$$\underset{\{\mathbf{d}_{k,m}\}}{\arg\min} \frac{1}{2} \sum_k \left\| \sum_m \mathbf{x}_{k,m} * \mathbf{d}_{k,m} - \mathbf{s}_k \right\|_2^2 + \sum_k \sum_m \iota_{C_{\text{PN}}}(\mathbf{d}_{k,m}) + \sum_m \iota_{C_C}(\mathbf{d}_{1,m}, \mathbf{d}_{2,m}, \cdots, \mathbf{d}_{K,m}) \ , \tag{41}$$

where C_C represents the constraint set of equality among the local dictionaries.

$$C_c = \{(\mathbf{d}_{1,m}, \mathbf{d}_{2,m}, \cdots, \mathbf{d}_{K,m}) | \mathbf{d}_{1,m} = \mathbf{d}_{2,m} = \cdots = \mathbf{d}_{K,m}\} \tag{42}$$

Applying the Proximal Gradient Descent approach on this consensus problem (41), the solution mainly consists of the gradient descent step (44) and proximal step (47). To simplify the notation, we define a linear operator $\mathbf{X}_{k,m}$ such that $\mathbf{x}_{k,m} * \mathbf{d}_{k,m} = \mathbf{X}_{k,m}\mathbf{d}_{k,m}$ and

$$\mathbf{X}_k = (\mathbf{X}_{k,1}\ \mathbf{X}_{k,2}\ \cdots) \ , \qquad \mathbf{D}_k = (\mathbf{d}_{k,1}\ \mathbf{d}_{k,2}\ \cdots)^T ,$$

so that the problem can be expressed as

$$\underset{\{\mathbf{D}_k\}}{\arg\min}\, \frac{1}{2}\sum_k \left\| \mathbf{X}_k\mathbf{D}_k - \mathbf{s}_k \right\|_2^2 + \sum_k \iota_{C_{\text{PN}}}(\mathbf{D}_k) + \iota_{C_{\text{C}}}(\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_K) \ . \tag{43}$$

The corresponding gradient step is

$$\mathbf{H}_k = \mathbf{D}_k - \frac{1}{L_k}\nabla F_k(\mathbf{D}_k) \ , \tag{44}$$

where L_k is the Lipschitz constant to satisfy the convergence. The proximal step of the problem (43) is of form

$$\underset{\{\mathbf{D}_k\}}{\arg\min}\, \frac{1}{2}\sum_k \left\| \mathbf{D}_k - \mathbf{H}_k \right\|_2^2 + \sum_k \iota_{C_{\text{PN}}}(\mathbf{D}_k) + \iota_{C_{\text{C}}}(\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_K) \ ; \tag{45}$$

however, we consider the equivalent formulation

$$\underset{\{\mathbf{G}\}}{\arg\min}\, \frac{1}{2}\sum_k \left\| \mathbf{G} - \mathbf{H}_k \right\|_2^2 + \sum_k \iota_{C_{\text{PN}}}(\mathbf{G}) \ , \tag{46}$$

where $\mathbf{G}$ is the global consensus dictionary. The minimizer of this problem (46) is given by

$$\mathbf{G} = \text{Prox}_{\iota_{C_{\text{PN}}}}\left(\frac{1}{K}\sum_k \mathbf{H}_k\right) \ . \tag{47}$$

Analogous to the Section 3.1.1, the computation of the gradient and other components are performed in the frequency domain by switching the spatial domain variables $\mathbf{X}_k, \mathbf{D}_k$ and $\mathbf{s}_k$ to the frequency domain variables $\hat{\mathbf{X}}_k$ (block with M diagonal matrix), $\hat{\mathbf{D}}_k$ and $\hat{\mathbf{s}}_k$, where $\hat{\mathbf{X}}_k$ denotes the DFT transform of $\mathbf{X}_k$. All the details of our implementation including the gradient derivation are shown in Algorithm 4.1.

Algorithm 4.1: DL algorithm using Frequency domain APG consensus

```
1: Compute gradient in the frequency domain
     $\nabla F_k(\hat{\mathbf{R}}^{i+1}) = \hat{\mathbf{X}}_k^H(\hat{\mathbf{X}}_k\hat{\mathbf{R}}^i - \hat{\mathbf{s}}_k)$
2: Compute dictionary
     $\mathbf{H}_k^{i+1} = \text{IFFT2}\{\hat{\mathbf{R}}^i - \frac{1}{L_k}\nabla F_k(\hat{\mathbf{R}}^i)\}$
     $\mathbf{G}^{i+1} = \text{prox}_{\iota_{C_{PN}}}(\frac{1}{K}\sum_k \mathbf{H}_k^{i+1})$
     $\hat{\mathbf{G}}^{i+1} = \text{FFT2}\{\mathbf{G}^{i+1}\}$
3: Compute auxiliary dictionary $\hat{\mathbf{R}}^{i+1}$ (Nesterov accelerated method) in the frequency domain
     $\gamma^{i+1} = (1 + \sqrt{1 + 4(\gamma^{i+1})^2})$
     $\hat{\mathbf{R}}^{i+1} = \hat{\mathbf{G}}^{i+1} + \frac{\gamma^i - 1}{\gamma^{i+1}}(\hat{\mathbf{G}}^{i+1} - \hat{\mathbf{G}}^i)$
```

We noted that the step 2 of Algorithm 4.1 can be efficiently implemented, due to the linearity property of the DFT, as

$$\begin{aligned}\mathbf{H}^{i+1} &= \text{IFFT2}\{\ \hat{\mathbf{R}}^i - \frac{1}{K}\sum_k(\ \frac{1}{L_k}\nabla F_k(\hat{\mathbf{R}}^i)\ )\ \} \\ \mathbf{G}^{i+1} &= \text{prox}_{\iota_{C_{PN}}}(\mathbf{H}^{i+1})\ ,\end{aligned}$$

with this consideration, the K inverse DFT operations are reduced to a single one.

4.2 General Remarks

The CDL algorithm is implemented by combining our proposed DL update with the ADMM-based SC update proposed in [14]. It is worth mentioning that all the steps of the SC update as well as the gradient step of the DL update are completely parallelizable in the training image index K, the only synchronization point is on the average of the gradients, along other simple operations.

In order to provide the best serial implementation of our method for environments such as MATLAB and Python, the gradient is calculated from a vectorized form (48) instead of loop-based form.

$$\frac{1}{2}\sum_k \left\|\hat{\mathbf{X}}_k\hat{\mathbf{D}}_k - \hat{\mathbf{s}}_k\right\|_2^2 = \frac{1}{2}\left\|\hat{\mathbf{X}}\hat{\mathbf{D}} - \hat{\mathbf{S}}\right\|_2^2 \tag{48}$$

where

$$\hat{\mathbf{X}} = \begin{pmatrix} \hat{\mathbf{X}}_1 & 0 & \cdots \\ 0 & \hat{\mathbf{X}}_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \quad \hat{\mathbf{D}} = \begin{pmatrix} \hat{\mathbf{D}}_1 \\ \hat{\mathbf{D}}_2 \\ \vdots \end{pmatrix} \quad \hat{\mathbf{S}} = \begin{pmatrix} \hat{\mathbf{s}}_1 \\ \hat{\mathbf{s}}_2 \\ \vdots \end{pmatrix} .$$

The step size ($\frac{1}{L_k}$) used on our algorithm consists of a single fixed Lipschitz constant, which was heuristically selected via a grid search over $L \in [10^1 - 10^3]$. We also used the optimal parameters reported in [10] for the ADMM consensus method.

Table 4.1: Optimal parameters found by grid search, where ρ and β are the penalty parameters of the ADMM-based SC and DL problems, respectively.

		Parameter			Parameter	
Method	K	ρ	L	Method	ρ	β
Proposed APG Consensus	5	5.00	100	ADMM Consensus	3.59	1.29
	10	3.59	50		3.59	1.29
	20	3.59	67		3.59	2.15
	40	3.59	100		3.59	1.08

4.3 Experimental Results

The experimental setup is analogous to the previous experimental framework described in Section 3.3, in which the performance is reported in terms of learning runtime, denoising and inpainting results.

The only difference is that for each training set, 64 dictionary filters of size 8×8 were learned using a sparsity parameter $\lambda = 0.1$ and 1000 iterations.

For the computational performance comparisons, we evaluate our APG consensus implementations against to the ADMM consensus (methods listed in Section 4.3.1) by varying training set size and filter set size. Then, the performance of learned dictionaries (of size $64 \times 8 \times 8$) from 40 training images are evaluated in terms of PSNR, SSIM and sparsity[1] metrics. These experiments were also carried out on a desktop computer equipped with an Intel i7-7700K CPU (4.20 GHz, 8MB Cache, 32GB RAM) and a Nvidia GTX1080 Ti graphic card.

4.3.1 Details of the evaluated methods

We compared the following CDL implementations:

∘ **ADMM-C**: The ADMM consensus algorithm as proposed in [21].

∘ **ADMM-CP1**: The parallel version of the ADMM-C proposed in [10], which consists in a Python code running on multi-core. Only included in our experiments for illustrative purposes.

∘ **ADMM-CV**: The vectorized form of the loop-based ADMM consensus algorithm (ADMM-C).

∘ **ADMM-CP2**: The parallel implementation of the vectorized algorithm ADMM-CV using CUDA-enabled MATLAB code[2].

∘ **APG-C**: Our proposed APG consensus based method, which is also vectorized implementation, and its parallel version **APG-CP**, which consists in a CUDA-enabled MATLAB code[2].

The algorithms listed here are MATLAB codes, with the exception of ADMM-CP1, which is python-based. Our MATLAB library, which can be downloaded from [31], includes our own implementation of ADMM-CV, ADMM-CP2, APG-C and APG-CP, whereas ADMM-C and ADMM-CP1 can be downloaded from [30].

4.3.2 Simulations of the learning stage

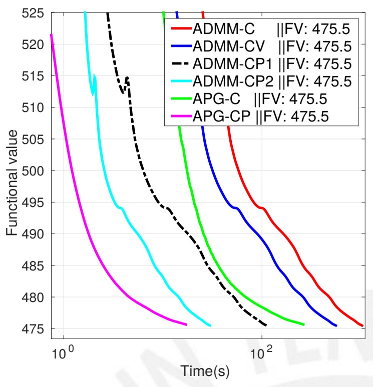
We primarily focus on the execution time of the DL update for this first set of experiments since all the presented CDL methods use the same ADMM-based solution [14] for the SC update.

In Figure 4.1, we illustrate the functional behavior and computational performance comparisons in the learning process of $8 \times 8 \times 64$ from 5 to 40 training images for the state-of-the-art consensus method with respect to our APG consensus method. Average runtime of all implementations for different training set and filter set sizes are shown in Figure 4.2. As can be observed in the Figures 4.1 (a)-(d), the APG and ADMM consensus methods converge to a relatively similar functional value in distinct time periods. Considering only serial implementations, it can be corroborated that a vectorized algorithm as ADMM-CV has better computational behavior than its loop-based counterpart (ADMM-C). However, our proposed algorithm (APG-C) outperforms both of them in terms of runtime by a factor of $2 \sim 6$.

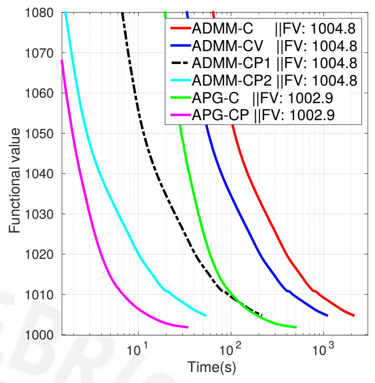
On the other hand, since ADMM-CP1 exploits the separable structure of the consensus approach via a multi-core implementation, we can see in the Figure 4.2 that this algorithm is more competitive than

[2]MATLAB implementation that exploits the convenience of the gpuArrays and high-level GPU operations.

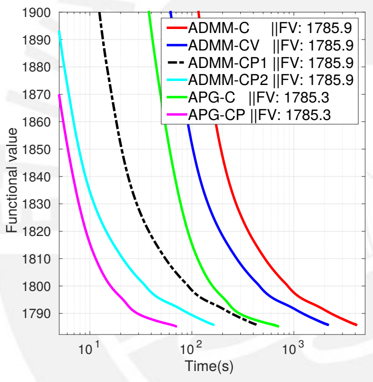
the serial implementations. But it is not faster than the other parallel ADMM consensus implementation that uses GPU (ADMM-CP2). Moreover, the parallel version of our approach (APG-CP), that also uses high-level GPU functions, is overall the fastest method considering the other two parallel algorithms.



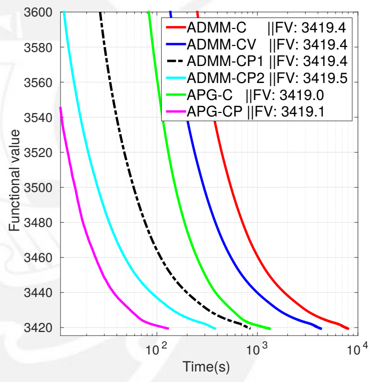
(a) 5 training images



(b) 10 training images

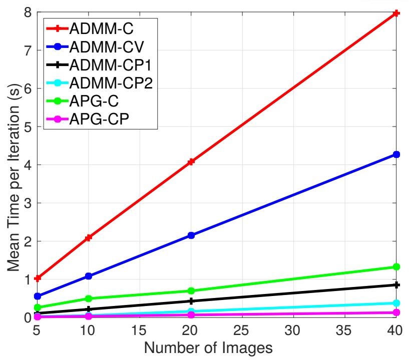


(c) 20 training images

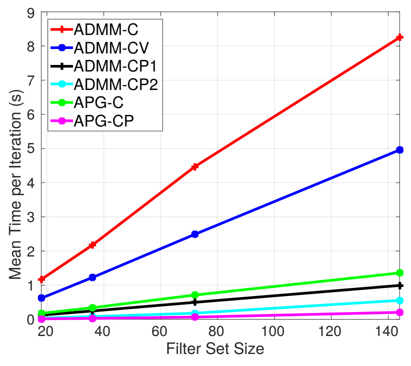


(d) 40 training images

Figure 4.1: Comparison of the functional value decay with respect to execution time on the learning process of 5, 10, 20 and 40 training images.



(a) 64 fixed dictionary filters



(b) 20 fixed training images

Figure 4.2: Average time per iteration of the CDL methods with respect to (a) training set size and (b) filter set size.

As additional information of the Figure 4.2, we present in Table 4.2 the exact values of runtime. In this table, we note that our proposed APG consensus method is approximately 2 to 6 times faster than the ADMM consensus method when comparing either serial or parallel implementations. A constant increase in the computational performance associated to the increase of training set and filter set sizes can also be observed.

		ADMM-C	**ADMM-CV**	**ADMM-CP1**	**ADMM-CP2**	**APG-C**	**APG-CP**
Training	5	1024	557	112	30	262	16
set size	40	7967	4274	855	378	1324	129
Filter	18	1167	618	124	35	175	14
set size	144	8248	4958	992	555	1358	204

Table 4.2: Execution time (seconds) during the learning process of the evaluated CDL methods.

4.3.3 Simulations of the testing stage

As the parallel implementations of the ADMM consensus and APG consensus methods do not affect the accuracy of the estimated dictionaries, only the performance of the dictionaries learned from the serial implementations are compared.

		Mandrill	**Barbara**	**Peppers**	**Boats**	**House**	**Gold hill**	**Monarch**	**Airplane**
PSNR	ADMM-C	20.99	22.94	24.88	23.77	24.42	24.54	24.53	23.82
(dB)	APG-C	21.00	22.97	24.90	23.79	24.42	24.54	24.54	23.84
SSIM	ADMM-C	0.5214	0.6175	0.6654	0.6192	0.6446	0.6266	0.6939	0.7086
	APG-C	0.5210	0.6180	0.6656	0.6199	0.6450	0.6269	0.6958	0.7097
L0 %	ADMM-C	9.12	4.39	1.70	3.67	3.71	1.42	3.76	3.02
	APG-C	8.82	4.26	1.68	3.61	3.85	1.40	3.75	2.99

Table 4.3: Denoising results for a standard set of test images, considering dictionary sets of size $8 \times 8 \times 64$ learned using a set of 40 training images.

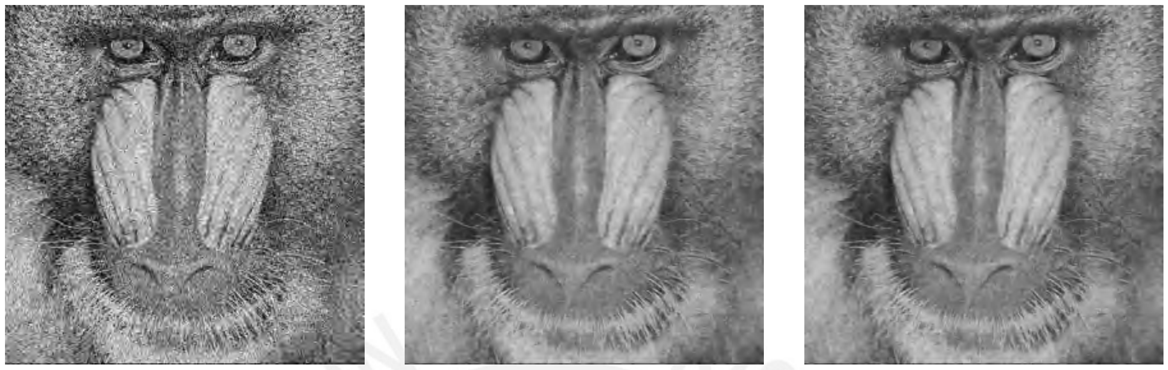
		Mandrill	**Barbara**	**Peppers**	**Boats**	**House**	**Gold hill**	**Monarch**	**Airplane**
PSNR	ADMM-C	30.80	39.64	37.83	43.98	43.10	39.32	41.76	41.70
(dB)	APG-C	30.76	39.93	37.78	44.10	43.90	39.39	41.63	41.75
SSIM	ADMM-C	0.9590	0.9880	0.9503	0.9918	0.9888	0.9780	0.9930	0.9908
	APG-C	0.9610	0.9880	0.9521	0.9920	0.9888	0.9787	0.9930	0.9907
L0 %	ADMM-C	94.45	95.15	76.29	90.36	98.81	91.43	88.02	89.46
	APG-C	92.55	94.71	75.57	89.88	91.20	93.52	88.35	89.75

Table 4.4: Inpainting results for a standard set of test images, considering dictionary sets of size $8 \times 8 \times 64$ learned using a set of 40 training images.

In Table 4.3 and Table 4.4, we report the denoising and inpainting comparisons between both evaluated dictionaries in terms of the PSNR, SSIM and sparsity metrics. It can be observed that the dictionary learned by our proposed method (APG-C) provides matching performance to the base line dictionary

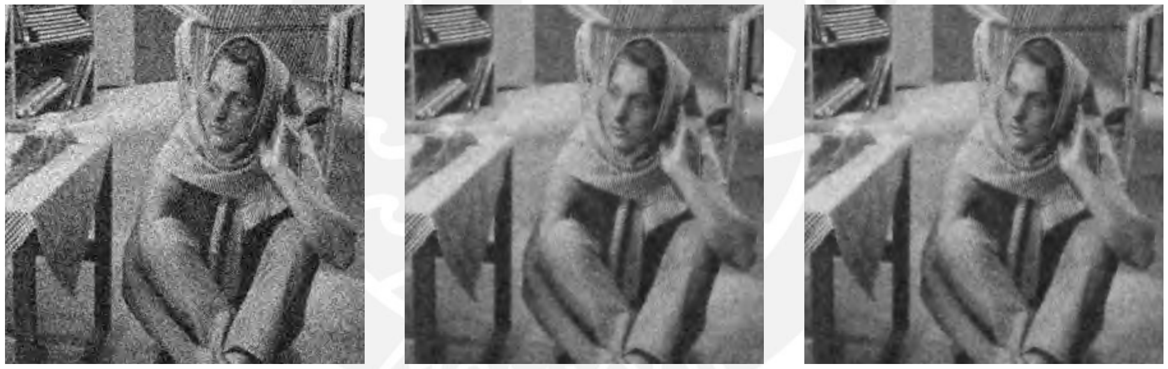
(ADMM-C). The maximum differences in terms of PSNR, SSIM and sparsity between the worst metrics of our method and the best metrics of the other one are only equal to 0.13 dB, 0.0004 and 2.09%.

Visual representations of the denoising process is illustrated in Figure 4.3 and Figure 4.4.



(a) Corrupted image (PSNR = 13.52) (b) ADMM-C (PSNR = 20.99) (c) APG-C (PSNR = 21.00)

Figure 4.3: Denoising of the corrupted Mandrill image using the learned dictionaries by the reported CDL methods.



(a) Corrupted image (PSNR = 13.93) (b) ADMM-C (PSNR = 22.94) (c) APG-C (PSNR = 22.97)

Figure 4.4: Denoising of the corrupted Barbara image using the learned dictionaries by the reported CDL methods.

Chapter 5

Conclusions

We have proposed two computationally efficient algorithms for solving the convolutional dictionary learning problem. Compared to the standard ADMM framework, we have proved that an APG based formulation provides less use of computational resources.

As a first approach, we have presented a novel CDL algorithm that consists of two particular ideas: the DFT domain APG method and the reminisced BGS model. Our APG-based solution used in both CDL updates has shown to outperform the standard ADMM methods by a speedup factor of $1.5 \sim 5.3$. Additionally, the BGS model has enabled to reduce number of estimated coefficient in our sparse coding update, providing a complementary speedup improvement by a factor of 2.

We have also shown that our second approach implemented in parallel, whose main contribution is the APG consensus approach for solving the DL update, is the fastest method including parallel implementations of the ADMM consensus method. The reported improvements in the training time by our both proposed methods do not negatively affect the quality of the resulting dictionaries when they are evaluated in the denoising and inpainting tasks.

Finally, it is worth noting that both APG methods have successfully been published as conference papers at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [13] and at the IEEE International Conference on Image Processing (ICIP) [16]. Additionally, this thesis report has allowed to produce a subsequent method on efficiently learning separable filters, which has also been published as another conference paper at the IEEE International Workshop on Machine Learning for Signal Processing (MLSP) [17].

Recommendations

- Even though our parallel algorithm has shown to be the fastest implementation with respect to the existing parallel ones, its separable structure across training image could not be completely exploited due to the high-level nature of the MATLAB programming framework. In order to achieve the best performance, it is recommended to develop a CUDA C algorithm running on multiple GPUs.
- As in the CDL topic, a good computational performance is highly required, we suggest to develop our methods on more flexible languages such as C, C++, or Fortran.
- Although this thesis work has been focused on CDL algorithms for learning non-separable filters, it would be interesting to explore the benefits of learning separable filters by taking as basis APG algorithms such as APG consensus.

References

[1] Julien Mairal, Francis Bach, and Jean Ponce, "Sparse modeling for image and vision processing," *Foundations and Trends® in Computer Graphics and Vision*, vol. 8, no. 2-3, pp. 85–283, 2014.

[2] Michael Elad and Michal Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[3] Matthew Zeiler, Dilip Krishnan, Graham Taylor, and Rob Fergus, "Deconvolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2528–2535.

[4] Brendt Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 301–315, Jan. 2016.

[5] James Bezdek and Richard Hathaway, "Some notes on alternating optimization," in *AFSS International Conference on Fuzzy Systems*. Springer, 2002, pp. 288–300.

[6] Michal Ŝorel and Filip Ŝroubek, "Fast convolutional sparse coding using matrix inversion lemma," *Digital Signal Processing*, vol. 55, pp. 44 – 51, 2016.

[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[8] Morten Mørup and Mikkel Schmidt, "Transformation invariant sparse coding," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2011, pp. 1–6.

[9] Matthew Zeiler, Graham Taylor, and Rob Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2018–2025.

[10] Cristina Garcia-Cardona and Brendt Wohlberg, "Convolutional dictionary learning," *arXiv preprint arXiv:1709.02893*, 2017.

[11] Gene Golub and Charles Van Loan, *Matrix computations*, vol. 3, JHU Press, 2012.

[12] Neal Parikh and Stephen Boyd, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[13] Gustavo Silva and Paul Rodríguez, "Efficient convolutional dictionary learning using partial update fast iterative shrinkage-thresholding algorithm," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.

[14] Brendt Wohlberg, "Efficient convolutional sparse coding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7173–7177.

[15] Luigi Grippo and Marco Sciandrone, "On the convergence of the block nonlinear gauss–seidel method under convex constraints," *Operations research letters*, vol. 26, no. 3, pp. 127–136, 2000.

[16] Gustavo Silva and Paul Rodríguez, "Efficient algorithm for convolutional dictionary learning via accelerated proximal gradient consensus," in *IEEE International Conference on Image Processing (ICIP)*, 2018.

[17] Gustavo Silva, Jorge Quesada, and Paul Rodríguez, "Efficient separable filter estimation using rank-1 convolutional dictionary learning," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP*, 2018.

[18] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck, *Discrete-time Signal Processing (2Nd Ed.)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[19] Rakesh Chalasani, Jose C Principe, and Naveen Ramakrishnan, "A fast proximal method for convolutional sparse coding," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–5.

[20] Kjersti Engan, Sven Ole Aase, and J Hakon Husoy, "Method of optimal directions for frame design," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1999, vol. 5, pp. 2443–2446.

[21] Cristina Garcia-Cardona and Brendt Wohlberg, "Subproblem coupling in convolutional dictionary learning," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Sept. 2017.

[22] Amos Sironi, Bugra Tekin, Roberto Rigamonti, Vincent Lepetit, and Pascal Fua, "Learning separable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 1, pp. 94–106, Jan 2015.

[23] Gustavo Silva, Jorge Quesada, Paul Rodríguez, and B. Wohlberg, "Fast convolutional sparse coding with separable filters," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 6035–6039.

[24] Mehiddin Al-Baali, "Descent property and global convergence of the fletcher—reeves method with inexact line search," *IMA Journal of Numerical Analysis*, vol. 5, no. 1, pp. 121–124, 1985.

[25] Amir Beck and Marc Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[26] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[27] Syazni Shoid, Mohd Rivaie, Mustafa Mamat, and Zabidin Salleh, "A new conjugate gradient method with exact line search," *Applied Mathematical Sciences*, vol. 9, no. 96, pp. 4799–4812, 2015.

[28] Mark Huiskes, Bart Thomee, and Michael Lew, "New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative," in *Proceedings of the international conference on Multimedia information retrieval*. ACM, 2010, pp. 527–536.

[29] Huiyi Hu, Brendt Wohlberg, and Rick Chartrand, "Task-driven dictionary learning for inpainting," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3543–3547.

[30] Brendt Wohlberg, “Sparse optimization research code (SPORCO),” *Matlab and Python library available from goo.gl/BjVgH5*, 2017.

[31] Gustavo Silva, “APG based code,” *Matlab and Python library available from goo.gl/1rvfDq*, 2018.