



HAL
open science

Variations on the Mumford-Shah functional for interface detection in degraded images : from proximal algorithms to unrolled architectures

Hoang Trieu Vy Le

► **To cite this version:**

Hoang Trieu Vy Le. Variations on the Mumford-Shah functional for interface detection in degraded images : from proximal algorithms to unrolled architectures. Signal and Image Processing. Ecole normale supérieure de lyon - ENS LYON, 2023. English. NNT : 2023ENSL0126 . tel-04520420

HAL Id: tel-04520420

<https://theses.hal.science/tel-04520420>

Submitted on 25 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THÈSE

en vue de l'obtention du grade de Docteur, délivré par
l'ÉCOLE NORMALE SUPERIEURE DE LYON

École Doctorale N°52
École Doctorale de Physique et Astrophysique de Lyon (PHAST)

Discipline : Physique
Spécialité de doctorat : Traitement du signal et des images

Soutenue publiquement le 13/12/2023, par :

Hoang Trieu Vy LE

Variations sur la fonctionnelle de Mumford-Shah pour la détection d'interfaces dans des images dégradées : des algorithmes proximaux aux architectures déroulées

Devant le jury composé de :

CHAUX, Caroline	Directrice de Recherche CNRS, IPAL Singapour	Rapporteuse
KOWALSKI, Matthieu	Maître de conférences, HDR, Université Paris-Sud	Rapporteur
AUJOL, Jean-François	Professeur, Université de Bordeaux	Examinateur
BRETIN, Elie	Maître de conférence, HDR, INSA de Lyon	Examinateur
CALATRONI, Luca	Chargé de recherche CNRS, Université Côte d'Azur	Examinateur
SEBBAN, Marc	Professeur, Université Jean Monnet	Examinateur
PUSTELNIK, Nelly	Directrice de Recherche CNRS, ENS de Lyon	Directrice de thèse
FOARE, Marion	Enseignante chercheuse, CPE/ENS de Lyon	Examinatrice

Contents

1	Introduction	1
I	Variational approaches for joint image restoration and edge detection	6
2	State-of-the-art on joint image restoration and edge detection using variational methods	7
3	Optimization and proximal algorithms	23
4	Proximal-based strategies for solving Discrete Mumford-Shah with Ambrosio-Tortorelli penalization on edges	40
II	Deep learning methods for joint image restoration and edge detection	58
5	State-of-the-art on deep learning for image restoration or edge detection	59
6	From proximal algorithms to robust proximal unfolded neural networks for image denoising and Plug-and-Play methods	66
7	Proximal neural networks for joint image denoising and edge detection	94
III	Conclusions and perspectives	102
	Appendices	106
A	Auxiliary proofs	107

Introduction

1.1 Motivation

In the era of artificial intelligence, there is a massive amount of imaging data playing a crucial role in our lives. An image is a combination of pixels produced by a sensor which is encountered in photographs, videos, medical images, satellite or physical experiment. In this thesis, we are interested in image edges, a specific image feature. It encodes valuable information which is useful in many applications, for instance, in experimental physics to estimate the contact area's phases helping in the identification of hydrodynamic regimes (as illustrated in Figure 1.1).

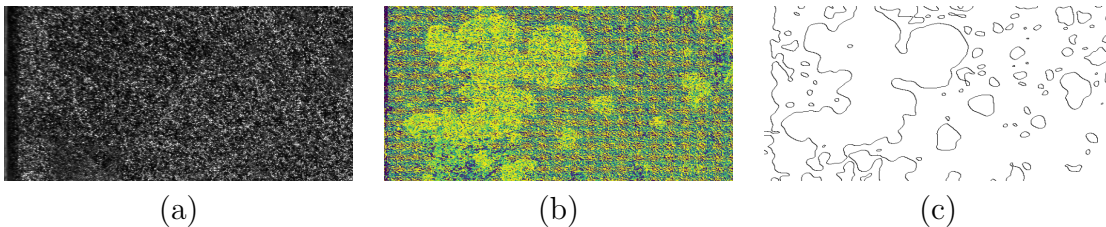


Figure 1.1: (a) Original image capturing joint gas and liquid flow in confined environments (b) Preprocessed image (c) Estimated contact interface/contour.

However, images are often corrupted by noise, motion blur or camera misfocus and valuable image information is lost. The task of **image restoration** is to "remove" noise and "undo" faults while still preserving the important features. Image restoration and edge detection are often treated separately in the image processing literature and the most standard strategies to solve image restoration task is to consider the solution of a convex minimization problem under some practical constraints, also called variational methods. Afterward, to detect image edges, a subsequent step requires a post-processing edge detection to achieve this outcome.

When considering the joint task of image restoration and edge detection, the associated minimization problem turns out to be non-convex and thus much more complex to solve. However, the recent advances in optimization dedicated to non-convex optimization allow us to provide new efficient numerical schemes for these problems. The pioneering works dedicated to the joint task traced back to the 80s

and statistical mechanic literature. Among these works we can refer to [Geman and Geman, 1984, Mumford and Shah, 1989] allowing to estimate a piecewise smooth field and thus to extract contours and a denoised image. Other related work is [Blake and Zisserman, 1987], another non-convex functional for piecewise smooth approximation named weak string model for 1D signal and weak membrane energy for images. Part I of the manuscript will be dedicated to such class of approaches which can be employed in various imaging joint tasks such as restoration and segmentation, e.g [Storath et al., 2014, Storath et al., 2015], which proposed to solve the Blake-Zisserman and Potts models, combining ADMM (the Alternating Direction Method of Multipliers [Beck, 2017]) and dynamic programming. With the same spirit, [Stekalovskiy and Cremers, 2014] proposed to solve the latter problem with primal-dual algorithm. [Foare et al., 2016] proposed a new approach to solve the Mumford-Shah (MS) problem, which is able to specify true 1D contour, set in between pixels ensuring sharp edges and smooth regions.

However, the image restoration problems have been facing an increasing size of data going from the processing of images with 10^3 pixels in the 80s to images with 10^7 pixels twenty years later. All mentioned variational approaches and corresponding minimization strategies struggle with high dimensional data. It is thus necessary to develop a method which is able to process these large images in a reasonable computation time. This gave rise to approaches based on deep learning (DL) techniques that involve: (i) the design of a Deep Neural Network (DNN), (ii) the training of this DNN with a large dataset until the model can make decision or return a desired output from input data. Learning-based methods have offered outstanding results in a wide range of image processing tasks including of course image restoration or edge detection.

For several years, there was a gap between standard image processing and neural network procedures, as the first one was guided by the physics of the data acquisition and prior knowledge about the object while the second one was considered as a very efficient prior-free black-box procedure. However a trade-off between flexibility and performance should be considered. The former can be flexible to deal with different image restoration tasks by well defining prior, provides better control over the underlying physics with fewer parameters, and eliminates the need to create a large database. The latter focuses on the study of a flexible and sophisticated deep architecture model, which requires a huge and accurate labeled database and a heavy training procedure.

For many years, the communities dedicated to Deep Learning (DL) and variational approaches were separate entities. Nowadays, there are many efforts trying to merge these two branches together in order to combine performance and flexibility. In 2020, with the pioneering work by [Gregor and Le Cun, 2010], in the context of sparse coding, the authors proposed a feedforward network built on iterative soft-thresholding algorithm with learnable operators over a fixed number of iterations. In the recent years, NN architectures based on unrolling proximal algorithms have been employed for many image processing tasks and proved their advantages such as light architecture, complexity computation, efficiency and robustness.

As far as we know, no study based on (unfolded) NNs to perform a joint image restoration and edge detection has been conducted yet. The second part of this thesis

will thus be dedicated to this class of approaches.

1.2 Organization of document

This manuscript is composed of three parts.

The first part focuses on variational approaches to perform the joint task of image restoration and edge detection. Chapter 2 is dedicated to a review of the state-of-the-art methods in image restoration and edge detection. Chapter 3 introduces a tour of different minimization algorithms that will be used throughout this manuscript. Chapter 4 presents a first contribution dedicated to the Discrete Mumford-Shah functional with Ambrosio-Tortorelli penalization on edges solved with proximal-based strategies.

The second part focuses on DL-based methods for (joint) image restoration and edge detection. In Chapter 5, we provide a brief tour on deep learning literature with a specific focus on unfolded scheme for image restoration and DL for edge detection. In chapter 6, we introduce the Proximal Neural Networks (PNNs), studying their flexibility, efficiency, and robustness in the context of image denoising and extension to other tasks. In Chapter 7, we explore proximal unfolded networks for the task of joint image restoration and edge detection.

The third part focuses on conclusions and perspectives.

Part1 – Chapter 2: State-of-the-art on joint image restoration and edge detection using variational methods

In this chapter, we give a brief review of the image restoration task including Maximum a Posteriori (MAP) formulation and the resulting variational formulation. More specifically, we focus on how to choose regularization function which addresses the edge detection task and including Mumford-Shah functional.

Part1 – Chapter 3: Optimization and proximal algorithms

This chapter is dedicated to all principal notations and mathematical tools in optimization which will be used throughout this thesis. More specifically, we will introduce proximal operators and associated proximal algorithms dedicated either to convex optimization or bi-convex optimization as encountered in MS functional.

Part1–Chapter 4: Proximal-based strategies for solving Discrete Mumford-Shah with Ambrosio-Tortorelli penalization on edges

This chapter focuses on solving the Discrete Mumford-Shah (DMS) functional with Ambrosio-Tortorelli (AT) penalization on edges. The latter depends on a parameter ε and the theoretical result ensures the Γ -convergence of the Ambrosio-Tortorelli approximation of the MS functional as $\varepsilon \rightarrow 0$. We derive two algorithms: Proximal Alternating Linearized Minimisation (PALM) and Semi-Linearized Proximal Alternating Minimisation (SL-PAM) in this context and associated convergence guarantees. We also propose a strategy for decreasing ε when considering the AT penalization. A

particular attention is paid to the derivation of the involved proximity operators. Numerous experiments are run in order to evaluate the performance of the proposed PALM and SL-PAM for minimizing D-MS with AT penalization. A multiresolution Golden-grid search strategy is proposed to efficiently extract the optimal set of hyperparameters and to provide fair comparisons with state-of-the-art methods for different types of degradation.

This chapter is associated to a work published in [Le et al., 2022a].

Part2 – Chapter5: State-of-the-art on deep learning for image restoration or edge detection

This chapter will be divided into 3 sections aiming to introduce the motivation of DL-based approaches in computer vision. First, we will discuss briefly about DL fundamental tools in image processing. Second, we focus on unfolded schemes for image restoration. Third, we provide a review of DL-based edge detection methods encountered in the literature.

Part2 – Chapter 6: PNN: From proximal algorithms to robust unfolded image denoising networks and Plug-and-Play methods

Before going further into dealing with the joint task, we first explore the unrolled Neural Networks for the single image denoising task. In this context, iterative proximal algorithms are widely used, enabling to handle non-smooth functions and linear operators. Recently, these algorithms have been paired with deep learning strategies, to further improve the estimation quality. The proposed proximal unfolded neural networks (PNNs) are obtained by unrolling a proximal algorithm as for finding a MAP estimate, but over a fixed number of iterations, with learned linear operators and parameters. As PNNs are based on optimization theory, they are very flexible, and can be adapted to any image restoration task, as soon as a proximal algorithm can solve it. They further have much lighter architectures than traditional networks. In this chapter, we introduce a unified framework to build PNNs for Gaussian denoising task based on both the dual-Forward-Backward and the primal-dual Chambolle-Pock algorithms. We further show that accelerated inertial versions of these algorithms enable skip connections in the associated NN layers. We propose different learning strategies for our PNN framework, and investigate their robustness (Lipschitz property) and denoising efficiency. Finally, we assess the robustness of our PNNs when plugged in a forward-backward algorithm for an image deblurring problem and in the context of texture segmentation.

This chapter is associated to the works published in [Le et al., 2022b], [Le et al., 2022c] and [Le et al., 2023].

Part2 – Chapter 7: Unfolded neural networks for joint image denoising and edge detection

This chapter aims to tackle the joint image restoration and edge detection task using unfolded neural networks. We investigate two approaches for addressing this joint task, the first one is built upon the adaptation of unrolling the PALM algorithm

within the framework of the Discrete Mumford-Shah functional, while the second approach involves the combination of the PNNs proposed in Chapter 6 and an additional layer for edge detection.

1.3 Publications

- H.T.V. Le, M. Foare, N. Pustelnik,
Proximal based strategies for solving Discrete Mumford-Shah with Ambrosio-Tortorelli penalization on edges
IEEE Signal Processing Letters, vol. 29, pp. 952-956, 2022. [\[PDF\]](#)[\[Toolbox\]](#)
- H.T.V. Le, N. Pustelnik, M. Foare,
The faster proximal algorithm, the better unfolded deep learning architecture ? The study case of image denoising
30th European Signal Processing Conference (EUSIPCO), Belgrade, Serbia, 2022, pp. 947-951. [\[PDF\]](#)[\[Toolbox\]](#)
- H.T.V. Le, B. Pascal, N. Pustelnik, M. Foare, P. Abry
Fast Unrolled Proximal Algorithms for Piecewise Homogeneous Fractal Image Analysis
26th Colloque Gretsi, Nancy, 2022. [\[PDF\]](#)[\[Toolbox in build currently\]](#)
- H.T.V. Le, A. Repetti, N. Pustelnik
PNN: From proximal algorithms to robust unfolded image denoising networks and Plug-and-Play methods
Submitted in the IEEE Transaction on Image Processing. [\[PDF\]](#)[\[Toolbox in build currently\]](#)

Part I

Variational approaches for joint image restoration and edge detection

State-of-the-art on joint image restoration and edge detection using variational methods

Summary

2.1	Image restoration	8
2.1.1	Preliminaries	8
2.1.2	Maximum <i>a posteriori</i> (MAP) Estimation	9
2.1.3	Regularization functions	10
2.2	Edge-preserving regularization functions	13
2.2.1	Edges detection	13
2.2.2	Total variation and its variations	15
2.2.3	Potts model	17
2.2.4	Blake-Zisserman model	17
2.2.5	Continuous Mumford-Shah functional and the Ambrosio-Tortorelli approach	19
2.2.6	Discrete Mumford-Shah functional	20
2.2.7	Relation between Mumford-Shah functional and Blake-Zisserman model	20
2.3	Conclusion	22

The objective of this chapter is to provide an initial overview of inverse problems and edge detection tasks in image processing with a specific focus on variational approaches.

2.1 Image restoration

2.1.1 Preliminaries

Many applications and tasks in signal and image processing can be reformulated as the resolution of an inverse problem. These problems rely on an estimation framework which aims to estimate parameters of interest from data. Mathematically, an original object is represented by a vector $\bar{\mathbf{x}} \in \mathbb{R}^{CN}$, measured through an acquisition process and leading to a degraded version $\mathbf{y} \in \mathbb{R}^{CM}$ (C denotes the number of image channels and N is the size of image). The degradation model can be formulated by the following expression:

$$\mathbf{y} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{n} \quad (2.1)$$

where $\mathbf{A} \in \mathbb{R}^{CM \times CN}$ is a linear degradation which can model, for example, a blur, and $\mathbf{n} \in \mathbb{R}^{CM}$ denotes a noise produced during the acquisition. For example, in Figure 2.1, the objective is to find an estimate $\hat{\mathbf{x}} \in \mathbb{R}^{CN}$ of the original image $\bar{\mathbf{x}}$ from the observation \mathbf{y} and the knowledge of the blurring kernel \mathbf{a} . In this case, the linear degradation can be modeled by the convolution of \mathbf{a} and \mathbf{x} (i.e $\mathbf{a} * \mathbf{x}$).

The standard approach to find $\hat{\mathbf{x}}$ is to minimize a criterion which is a sum of two functions:

$$\hat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmin}} f(\mathbf{x}) + g(\mathbf{x}) \quad (2.2)$$

where $f : \mathbb{R}^{CN} \rightarrow (-\infty, +\infty]$ is the data fidelity term which encompasses the information related to the degradation model, and $g : \mathbb{R}^{CN} \rightarrow (-\infty, +\infty]$ is the regularization or penalization term which allows us to impose on the estimate some properties or known *a priori* information (e.g. sparsity of the solution, smoothness).

In some cases, the matrix \mathbf{A} is unknown and the problem becomes a **blind deconvolution** task which forces us to reconstruct $\hat{\mathbf{x}}$ while estimating also \mathbf{A} . In this thesis, we focus on the case where \mathbf{A} is known.

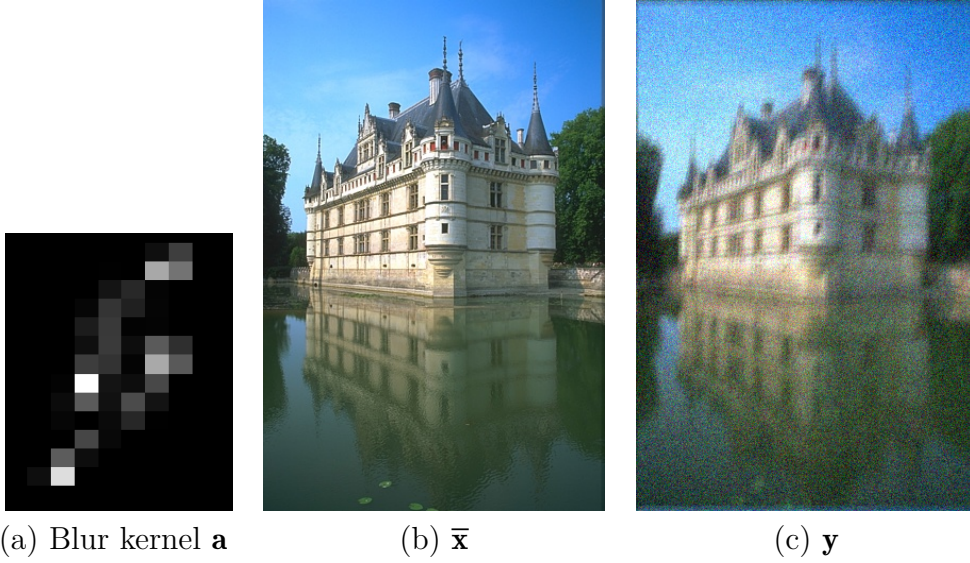


Figure 2.1: Example of an image "102061" from the BSDS500 validation dataset [Arbelaez et al., 2011] degraded by the blur kernel \mathbf{a} and an additive white gaussian noise $\mathbf{n} \sim \mathcal{N}(0, \delta^2 \mathbf{I}_{CM})$ ($\delta = 0.03$).

2.1.2 Maximum *a posteriori* (MAP) Estimation

The **Bayesian** approach provides building blocks to justify a variational approach, especially helping in the understanding of the construction of a data fidelity function adapted to a given problem.

First, let set up general setting for a statistical inference problem where we want to estimate an unknown quantity from observed data. A **Bayesian** approach considers \mathbf{x} and \mathbf{y} as realizations of random vectors \mathbf{X} and \mathbf{Y} . The objective is to find the estimate $\hat{\mathbf{x}}$ which maximizes the posterior probability distribution $\mu_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$. Thanks to Bayes theorem, $\mu_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}$ can be expressed in terms of the likelihood function $\mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}$, the prior distribution $\mu_{\mathbf{X}}$ and the marginal distribution $\mu_{\mathbf{Y}}$, here assumed to be nonzero,

$$\mu_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}) = \frac{\mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y})\mu_{\mathbf{X}}(\mathbf{x})}{\mu_{\mathbf{Y}}(\mathbf{y})}. \quad (2.3)$$

The Maximum A Posteriori (MAP) Estimation of the random vector \mathbf{X} given an observed data $\mathbf{Y} = \mathbf{y}$ is the realization \mathbf{x} that maximizes $\mu_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x})$ which can be reformulated as

$$\hat{\mathbf{x}}_{\text{MAP}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmax}} \mu_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x}). \quad (2.4)$$

We notice that $\mu_{\mathbf{Y}}(\mathbf{y})$ does not depend on \mathbf{x} . Using Eq. (2.3), problem (2.4) can be rewritten as

$$\hat{\mathbf{x}}_{\text{MAP}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmax}} \mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y})\mu_{\mathbf{X}}(\mathbf{x}), \quad (2.5)$$

or equivalently

$$\widehat{\mathbf{x}}_{\text{MAP}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmin}} -\ln \mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}) - \ln \mu_{\mathbf{X}}(\mathbf{x}). \quad (2.6)$$

Thus we can define the data fidelity function to the observed data \mathbf{y} and regularization function of problem (2.2) in the sense of the MAP estimation as

$$(\forall \mathbf{x} \in \mathbb{R}^{CN}) \begin{cases} f(\mathbf{x}) \propto -\ln \mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}), \\ g(\mathbf{x}) \propto -\ln \mu_{\mathbf{X}}(\mathbf{x}). \end{cases} \quad (2.7)$$

Particular case of Additive White Gaussian Noise (AWGN) – We assume that \mathbf{n} is a realization of an independent, identically distributed and drawn from a zero-mean normal distribution with variance δ^2 , that is :

$$\mathbf{n} \sim \mathcal{N}(0_{CM}, \delta^2 \text{Id}_{CM}) \quad (2.8)$$

Using the degradation model (2.1) the likelihood is:

$$\mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}) = \prod_{i=1}^{CM} \frac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\frac{(\mathbf{y}_i - (\mathbf{A}\mathbf{x})_i)^2}{2\delta^2}\right). \quad (2.9)$$

which is equivalent to

$$\mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}) = \frac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\frac{\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2}{2\delta^2}\right) \quad (2.10)$$

In this case, from (2.7) the data fidelity function fits the negative log-likelihood and can be chosen as:

$$f(\mathbf{x}) \propto -\ln \mu_{\mathbf{Y}|\mathbf{X}=\mathbf{x}}(\mathbf{y}) \propto \frac{1}{2\delta^2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2. \quad (2.11)$$

In the remainder of this manuscript, for the reader's convenience, because the data fidelity term depends on the operator \mathbf{A} and observation \mathbf{y} , the function $f(\mathbf{x})$ can be expressed as $\Psi(\mathbf{A}\mathbf{x}; \mathbf{y})$ where $\Psi : \mathbb{R}^{CM} \rightarrow (-\infty, +\infty]$.

2.1.3 Regularization functions

The regularization function plays a very important role in inverse problem which allows us to constrain some desired properties of the solution. In general, the function g can be expressed as $g(\mathbf{x}) = \lambda\Phi(\mathbf{D}\mathbf{x})$, where $\Phi : \mathbb{R}^{CN} \rightarrow (0, +\infty]$, $\mathbf{D} \in \mathbb{R}^{CN \times CN}$ denotes a linear transform, $\lambda > 0$ is a regularization parameter that plays an important role in the trade-off between the data fidelity term and the regularization term. The function Φ and the operator \mathbf{D} are chosen according to the type of images to recover. For instance, we are going to focus on regularization that favors image smoothness or sparsity.

- **Choice of \mathbf{D}** – The operator \mathbf{D} aims to highlight some properties of the images. A vector $\mathbf{x} \in \mathbb{R}^{CN}$ is said to be sparse if most of its elements are set to zero. In signal processing, some signals are sparse intrinsically but most of them are sparse after a linear transform such as wavelets or finite differences operator

as illustrated in Figure 2.2. More generally, \mathbf{D} can be chosen as a sparsifying operator (e.g., wavelet transform, frames or dictionary [Mallat, 1997, Jacques et al., 2011, Pustelnik et al., 2016a].)

Example 1. We consider a grayscale image $\mathbf{x} \in \mathbb{R}^N$ ($N = N_h \times N_v$) displayed in Fig.2.2. Let $h = [1 \ -1]$ and $v = h^T$ be horizontal and vertical difference filters corresponding to the finite differences operators D_h and D_v . In this case, we say that the object $\mathbf{D}\mathbf{x}$ is considered to be sparse difference operator where $\mathbf{D} = [D_h^T \ D_v^T]^T \in \mathbb{R}^{2N \times N}$ and $(D_h\mathbf{x})_{ij}$ and $(D_v\mathbf{x})_{ij}$ are defined as:

$$(D_h\mathbf{x})_{ij} = \begin{cases} \mathbf{x}_{i,j+1} - \mathbf{x}_{i,j} & \text{if } j < N_v, \\ 0 & \text{otherwise,} \end{cases}$$

$$(D_v\mathbf{x})_{ij} = \begin{cases} \mathbf{x}_{i+1,j} - \mathbf{x}_{i,j} & \text{if } i < N_h, \\ 0 & \text{otherwise.} \end{cases}$$

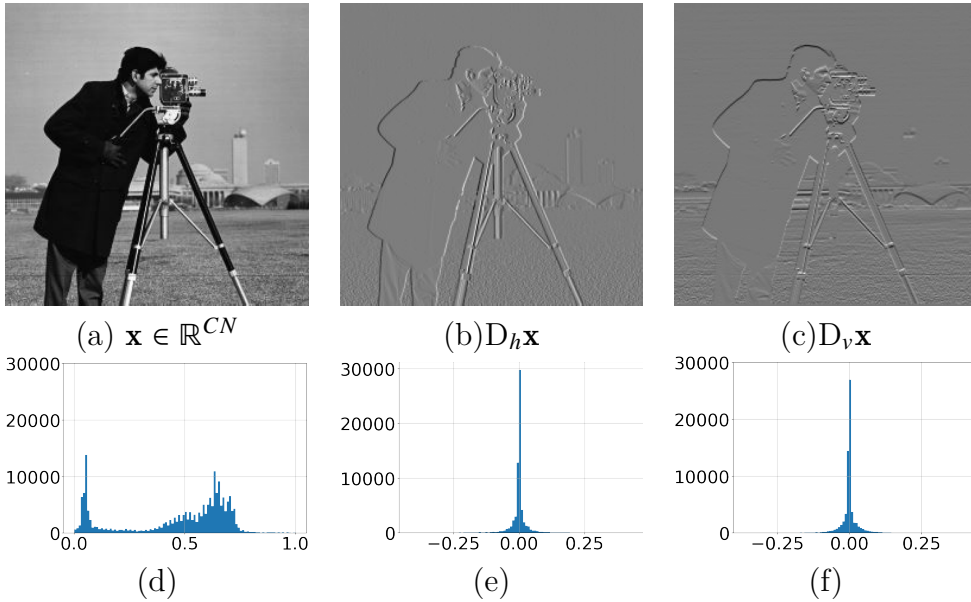


Figure 2.2: (a) \mathbf{x} , (b) $D_h\mathbf{x}$, (c) $D_v\mathbf{x}$, (d),(e) and (f) are respectively histograms of (a), (b) and (c).

- **Choice of Φ** – The pioneering choice is $\|\cdot\|_2^2$ adapted to smooth penalization. However when focusing on sparsity, other choices of Φ should be considered starting with the function that best promotes the sparsity which is the "pseudo-norm" ℓ_0 which counts the number of nonzero coefficients of \mathbf{u} :

$$(\forall \mathbf{u} = (\mathbf{u}_i)_{1 \leq i \leq N}) \quad \Phi(\mathbf{u}) = \|\mathbf{u}\|_0 = \sum_{i=1}^N \phi(\mathbf{u}_i), \quad (2.12)$$

where $\phi : \mathbb{R} \rightarrow \{0, 1\}$ is defined by

$$\phi(\mathbf{u}_i) = \begin{cases} 0 & \text{if } \mathbf{u}_i = 0, \\ 1 & \text{otherwise.} \end{cases} \quad (2.13)$$

In practical applications, employing this non-convex function for designing an optimization algorithm can be challenging. Alternatively, a reasonably effective approach to promote sparsity involves utilizing the ℓ_1 norm:

$$(\forall \mathbf{u} = (\mathbf{u}_i)_{1 \leq i \leq N}) \quad \Phi(\mathbf{u}) = \|\mathbf{u}\|_1 = \sum_{i=1}^N |\mathbf{u}_i|. \quad (2.14)$$

The ℓ_1 -norm addresses the issue of non-convexity, although this function remains non-differentiable. In various scenarios, the Huber function presents a viable convex and smooth alternative, characterized as :

$$(\forall \mathbf{u} = (\mathbf{u}_i)_{1 \leq i \leq N}) \quad \Phi(\mathbf{u}) = \sum_{i=1}^N \phi_\zeta(\mathbf{u}_i) \quad (2.15)$$

where

$$\phi_\zeta(\mathbf{u}_i) = \begin{cases} \frac{\mathbf{u}_i^2}{2\zeta} & \text{if } |\mathbf{u}_i| \leq \zeta \\ |\mathbf{u}_i| - \frac{\zeta}{2} & \text{otherwise.} \end{cases} \quad (2.16)$$

Or we can use the BerHu function which is defined as

$$\phi_\zeta(\mathbf{u}_i) = \begin{cases} \frac{\mathbf{u}_i^2 + \delta^2}{2\zeta}, & \text{if } |\mathbf{u}_i| > \zeta \\ |\mathbf{u}_i|, & \text{if } |\mathbf{u}_i| \leq \zeta. \end{cases} \quad (2.17)$$

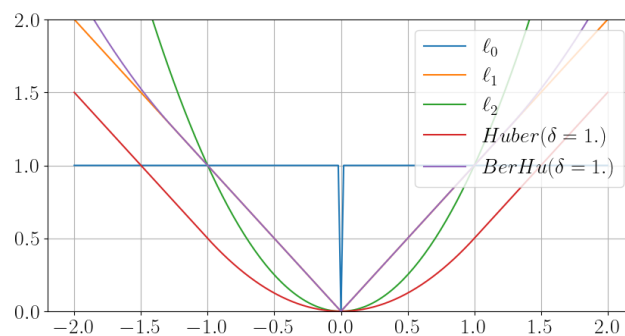


Figure 2.3: Comparison between the ℓ_0 -pseudo-norm, ℓ_1 -norm, ℓ_2^2 -norm and Huber function with $\delta = 1$, BerHu function with $\delta = 1$

2.1.3.1 Tikhonov regularization

One of the most popular and simple regularization methods was proposed by [Tikhonov, 1963], which is defined as:

$$g(\mathbf{x}) = \lambda \|\mathbf{D}\mathbf{x}\|_2^2, \quad (2.18)$$

where \mathbf{D} is a linear operator. This operator can be chosen as an identity matrix $\mathbf{D} = \mathbf{I}$ allowing to impose strong convexity. To impose the smoothness of \mathbf{x} and decrease the irregularity due to the noise in the data, we can choose \mathbf{D} as a high-pass operator e.g. a difference operator $\mathbf{D} = [\mathbf{D}_h^\top \ \mathbf{D}_v^\top]^\top$.

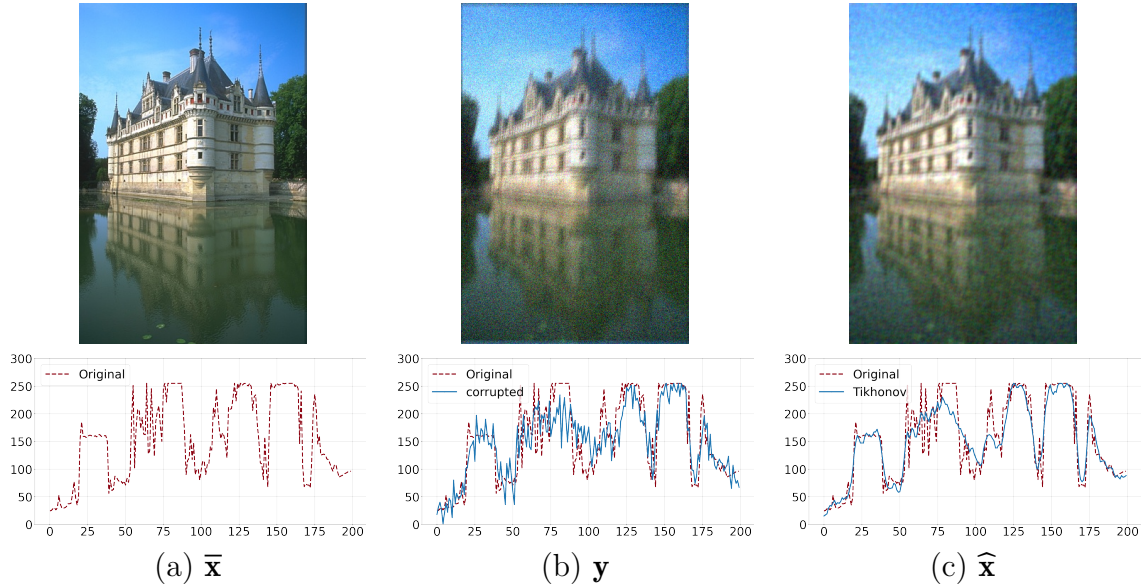


Figure 2.4: Image restoration using Tikhonov-regularization by minimizing (2.2). First row: (a) Original image (b) Degraded image (c) Restored. Second row: (a), (b) (c) display a row of the image to better capture the effect of the restoration including smoothing effect with Tikhonov regularization.

In Figure 2.4, we observe that Tikhonov regularization smooths out the image but does not preserve the discontinuities/edges. In the following, we will review some regularization functions that promote the edge-preserving property.

2.2 Edge-preserving regularization functions

2.2.1 Edges detection

Definition of edges – Edges capture important intensity changes between pixels and thus represent the boundaries between two different regions within an image. When defining edges we need to specify its location and its value.

First, we focus on its location. Since an image can be considered as a discrete surface as illustrated in Figure 2.5-(b) composed of vertices (black dots centered in the middle of pixel), edges (linking two vertices) and faces (colored in gray and blue) [Foare, 2017, Foare et al., 2019], the intensity value \mathbf{x}_i can be placed on the vertices v_i which are seen as point mass on the center of pixels. The contours can either be defined on edges in between two pixels with values living on s_i (Figure 2.5-(c)) or on pixels with values attached to the nodes v_i (Figure 2.5-(d)).

Second, to define edge values s_i , we can use different high pass filters such as finite difference, Sobel [Sobel and Feldman, 1968], Prewitt [Prewitt, 1970]. In this manuscript, we focus on the differences operator $\mathbf{D} = [\mathbf{D}_h^\top \mathbf{D}_v^\top]^\top$ to obtain edges between nodes in the context of discrete calculus (Figure. 2.6).

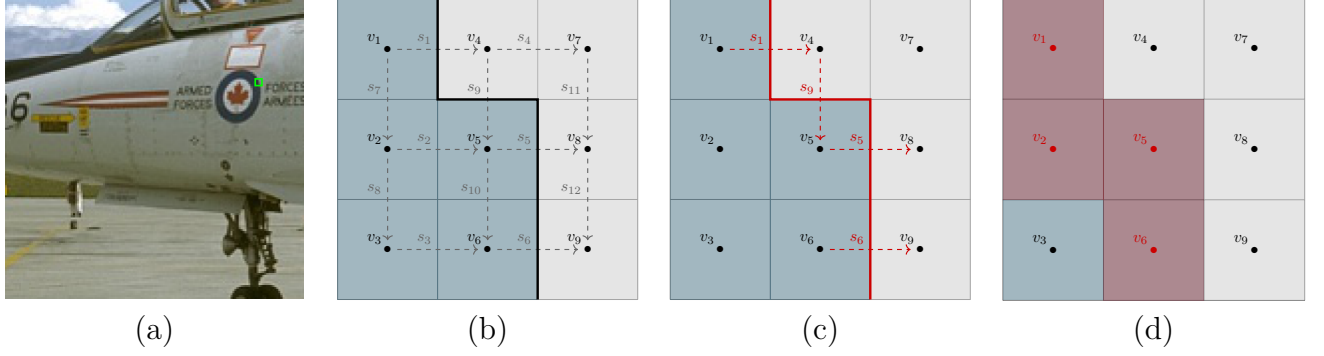


Figure 2.5: Examples of two edges definitions in digital image: (a) Extracted patch (b) Image patch seen as a discrete surface with true contour delineated in black (c) Contours defined in between pixels with values living on edges s_i (d) Contours defined on pixels with values living on vertices v_i

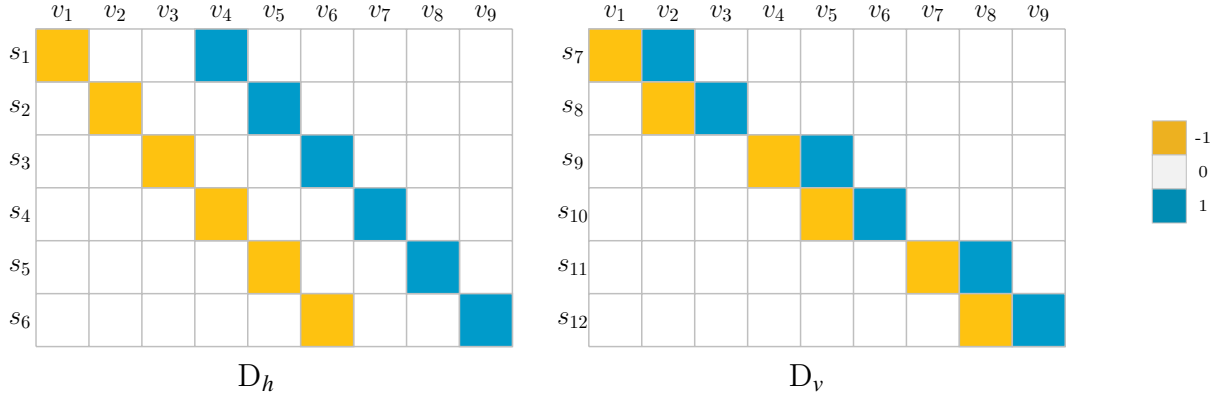


Figure 2.6: Illustration of $\mathbf{D} = [\mathbf{D}_h^\top \mathbf{D}_v^\top]^\top$ with periodic boundary conditions corresponding horizontal and vertical pixelwise discrete gradient operators D_h and D_v .

The gradient magnitude then serves to measure the edge strength. To capture the binary nature of edges a thresholding step is required and is expressed as follows:

$$\hat{\mathbf{e}}_{\bullet,ij} = \begin{cases} 1, & \text{if } |(\mathbf{D} \cdot \hat{\mathbf{x}})_{ij}| > \zeta, \\ 0, & \text{otherwise.} \end{cases} \quad (2.19)$$

where $\mathbf{e}_{\bullet,ij}$ can be $\mathbf{e}_{h,ij}$ horizontal edge between two pixels (i, j) and $(i + 1, j)$ or $\mathbf{e}_{v,ij}$ vertical edge between two pixels (i, j) and $(i, j + 1)$.

Considering (2.19), we can obtain the contours living on edges as illustrated in Figure 2.5-(c-d). And we can convert edges between nodes type to contour living on pixels by defining a single edge map \mathbf{e} as follows:

$$\mathbf{e} = w_h \mathbf{e}_h + w_v \mathbf{e}_v, \quad (2.20)$$

where $w_h = w_v = \frac{1}{2}$. A comparison between a pixel-wise edges (as illustrated in red cells in Figure 2.5-(d)) and edges between nodes (Figure 2.5-(c)) is provided in Figure 2.7.

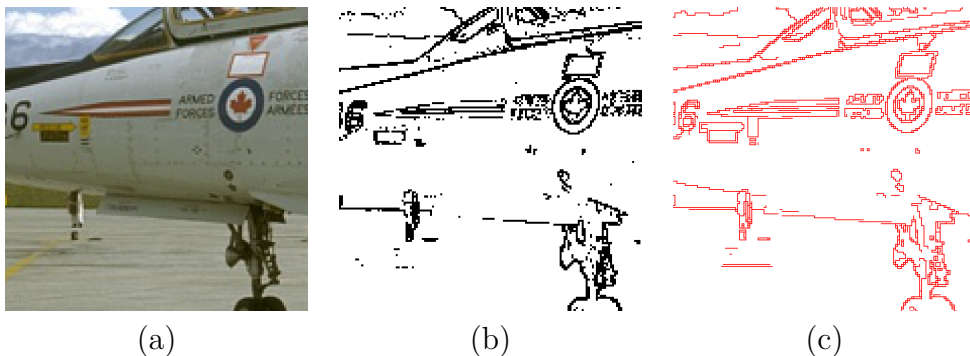


Figure 2.7: An example to compare the difference between pixel-wise edges (b) and edges between nodes (c).

When applying a threshold, the lower the threshold, the more edges will be detected and the result will be increasingly susceptible to noise. In the contrary, a high threshold may miss subtle edges.

However, when dealing with degraded images, these methods will need a pre-processing to reconstruct image beforehand. In this section, we will focus on the variational approach (2.2) where we focus on the choices of regularization function g that favor edge-preserving property.

2.2.2 Total variation and its variations

In [Rudin et al., 1992], a pioneering approach was introduced with the aim of achieving smoothing with edge-preserving for continuous data. This technique later came to be recognized as the ROF model.

Definition 2.2.1. Let $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_C) \in \mathbb{R}^{CN}$ be a tensor modelling a multichannel image of size $(C \times N_h \times N_v)$ and $\mathbf{x}_c \in \mathbb{R}^N$ is an image channel where $1 \leq c \leq C$. The anisotropic total variation of \mathbf{x} is given by

$$\text{TV}_{\text{aniso}}(\mathbf{x}) = \sum_{c=1}^C \sum_{i=1}^{N_h} \sum_{j=1}^{N_v} |(D_h \mathbf{x}_c)_{ij}| + |(D_v \mathbf{x}_c)_{ij}|, \quad (2.21)$$

Definition 2.2.2. Let $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_C) \in \mathbb{R}^{CN}$ be a tensor modelling a multichannel image of size $(C \times N_h \times N_v)$ and $\mathbf{x}_c \in \mathbb{R}^N$ is an image channel where $1 \leq c \leq C$. The

isotropic total variation of \mathbf{x} is given by

$$\text{TV}_{\text{iso}}(\mathbf{x}) = \|\mathbf{D}\mathbf{x}\|_{2,1} = \sum_{c=1}^C \sum_{i=1}^{N_h} \sum_{j=1}^{N_v} \sqrt{(D_h \mathbf{x}_c)_{ij}^2 + (D_v \mathbf{x}_c)_{ij}^2}. \quad (2.22)$$

Finally, we can define the ROF model as follows:

$$\widehat{\mathbf{x}}_{\text{ROF}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmin}} \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \lambda \text{TV}_{\bullet}(\mathbf{x}). \quad (2.23)$$

The objective function (2.23) being convex, but non differentiable, its minimization requires the use of non-smooth convex optimisation tools which will be described in detail in Chapter 3, leading to the implementation of proximal algorithms.

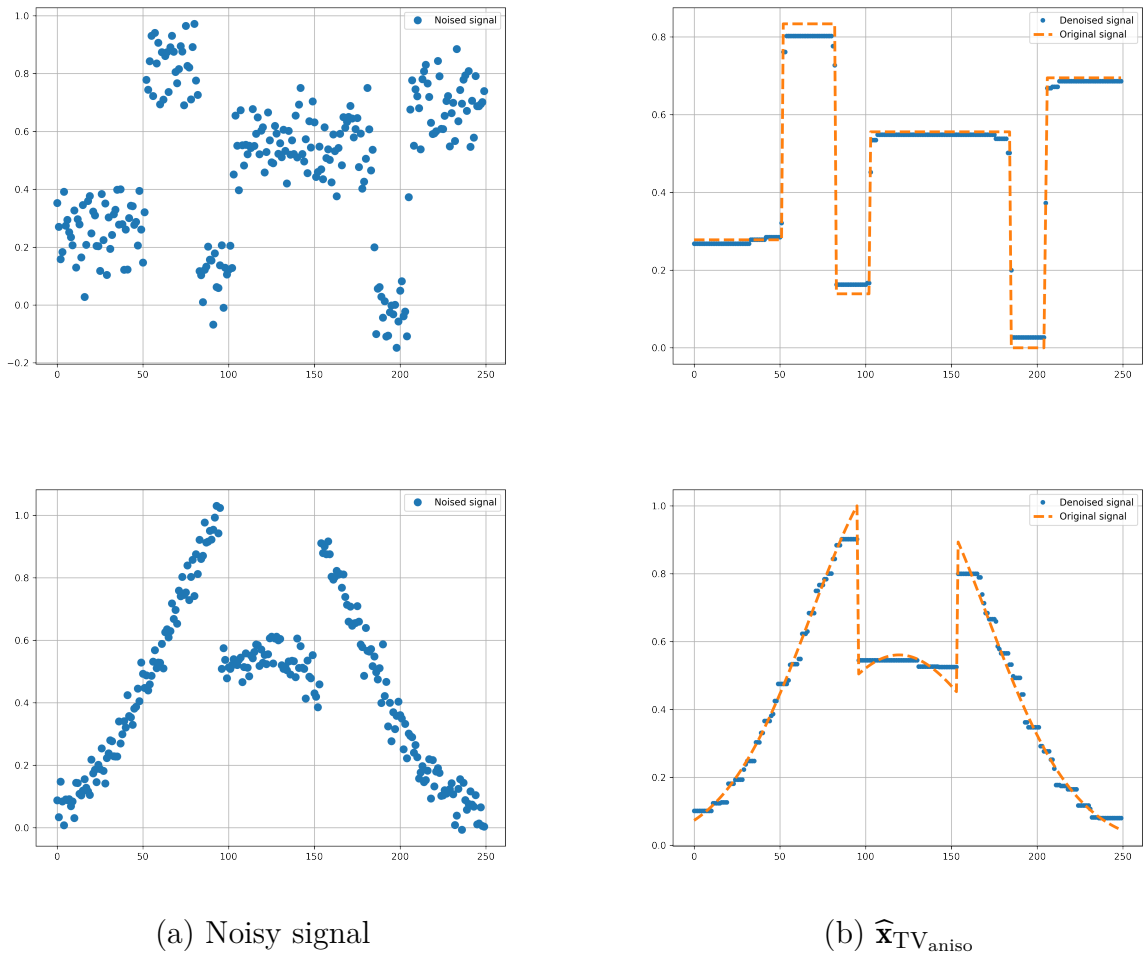


Figure 2.8: Staircasing effects when we use TV_{aniso} for 1D signal.

However, TV regularization has limitations when dealing with textures and structured images that extend beyond the local neighborhood. To overcome this limitation, non-local total variation (NLTV) was proposed by [Gilboa and Osher, 2009]. NLTV takes the concept of TV regularization and goes a step further by considering non-local similarities in image patches. It recognizes that similar patches from different parts

of the image share common characteristics and should influence the regularization process. If in TV, the neighborhood of the location (i, j) is the set of $\{(i-1, j), (i, j-1)\}$ then for NLTV, the neighborhood at this point depends on the similarity degree.

Another class of penalization was proposed by [Bredies et al., 2010] and named the total generalized variation (TGV) regularization by incorporating higher-order gradients into the regularization process. By considering second-order differences in the image, TGV regularization provides enhanced control over the smoothness and regularity of the reconstructed image.

Post-processing for edge detection – From Figure 2.8, we can observe that these methods allow us to keep the abrupt changes or discontinuities set in 1D signal. Another example is shown in Figure 2.9-(c) for image restoration. To extract the edges, we need a post processing step for example a threshold on the gradient map of image (e.g using Equation (2.19)).

2.2.3 Potts model

The variational formulation of the Potts model relies on the penalization

$$g(\mathbf{x}) = \lambda \|\mathbf{D}\mathbf{x}\|_0. \quad (2.24)$$

[Storath et al., 2015] proposed an extended version of (2.24) which adds more direction in the finite differences operator:

$$g(\mathbf{x}) = \lambda \sum_{s=1}^S w_s \|D_{a_s} \mathbf{x}\|_0, \quad (2.25)$$

where w_s are nonnegative weights, D_{a_s} denotes the operator containing finite differences in axial, diagonal and "knight-moves" direction which was first proposed by [Chambolle, 1999]. Vector a_s denotes the displacement vector so that $D_{a_s} \mathbf{x} = \mathbf{x}(\cdot + a_s) - \mathbf{x}$, where a_s belongs to the neighborhood system $\mathcal{N} = \{a_1, \dots, a_S\} = \{(1, 0), (0, 1), (1, 1), (1, -1), (-2, -1), (-2, 1), (2, 1), (2, -1)\}$ ($S = 8$). With this kind of discretisation, the author was able to divide the Potts problem into distinct subproblems, which they tackled by using dynamic programming and the alternating direction method of multipliers (ADMM) as suggested in their works [Storath et al., 2014, Storath et al., 2015].

An illustration of such a penalization choice is provided in Figure 2.9-(e).

2.2.4 Blake-Zisserman model

In [Blake and Zisserman, 1987], the authors introduced a novel approach for the piecewise smooth reconstruction on discrete domain by introducing weak membrane energy for images. They defined the energy function as follows:

$$\hat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmin}} \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \sum_{c=1}^C \sum_{i,j} \min(\alpha, \gamma^2 |(D_h \mathbf{x}_c)_{ij}|^2) + \sum_{c=1}^C \sum_{i,j} \min(\alpha, \gamma^2 |(D_v \mathbf{x}_c)_{ij}|^2). \quad (2.26)$$

In [Hohm et al., 2015], the authors proposed a more general discretization of the Blake-Zisserman model

$$g(\mathbf{x}) = \omega_s \sum_{c=1}^C \sum_{i,j} \sum_{s=1}^S \min\left(\alpha, \gamma^2 |D_{a_s} \mathbf{x}_c|_{ij}^2\right), \quad (2.27)$$

Furthermore, in [Blake and Zisserman, 1987], to circumvent gradient limits effects; it describes the phenomenon that the first order model penalizes large slopes and produces spurious extra segments to account for them. They also proposed extensions to second order smoothness penalties named weak rode and weak plate models. The discrete formulation is given by

$$\mathcal{E}_{\alpha,\gamma} = \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \sum_{c=1}^C \sum_{ij} \min\{\alpha, \gamma^2 \sqrt{(V_{\mathbf{x}_c})_{ij}}\}, \quad (2.28)$$

where $(V_{\mathbf{x}_c})_{ij} = |(D_h^2 \mathbf{x}_c)_{ij}|^2 + |(D_v^2 \mathbf{x}_c)_{ij}|^2 + 2|(D_h D_v \mathbf{x}_c)_{ij}|^2$. We define here D_h^2 and D_v^2 the second order finite differences in horizontal and vertical directions.

Regarding minimization algorithm, as the objective function (2.26) is not convex, Blake and Zisserman introduced a graduated non-convexity strategy (GNC) [Thacker and Cootes, 1996, Blake and Zisserman, 1987]. The central idea is to solve a sequence of minimization problems which are easier to handle than (2.26). Initially, the non-convex energy \mathcal{E} is approximated by a convex function \mathcal{E}^* . Subsequently, a sequence of energy functions $\mathcal{E}^{(p)}$ is generated, where the construction relies on a parameter $0 \leq p \leq 1$, ensuring that $\mathcal{E}^{(1)}$ corresponds to \mathcal{E}^* , $\mathcal{E}^{(0)}$ corresponds to \mathcal{E} , and $\mathcal{E}^{(p)}$ changes continuously from $\mathcal{E}^{(1)}$ to $\mathcal{E}^{(0)}$. The approximating energy is $\mathcal{E}^{(p)} = \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \sum_{c=1}^C \sum_{ij} \Phi_{\alpha,\gamma}^{(p)}(|D_h \mathbf{x}_c|_{ij}) + \Phi_{\alpha,\gamma}^{(p)}(|D_v \mathbf{x}_c|_{ij})$, where $\Phi_{\alpha,\gamma}^{(p)}$ is expressed as

$$\Phi_{\alpha,\gamma}^{(p)}(t) = \begin{cases} \gamma^2 t^2 & \text{if } |t| < \frac{\gamma}{\alpha^2 r} \\ \alpha - \frac{d}{2} (|t| - r)^2 & \text{if } q \leq |t| < r \\ \alpha & \text{if } |t| \geq r \end{cases} \quad (2.29)$$

with

$$r^2 = \alpha \left(\frac{2}{d} + \frac{1}{\gamma^2} \right), \quad d = \frac{1}{4p}.$$

Each individual subproblem now can then be effectively solved using the gradient descent method.

Post-processing for edge detection – In this case, we can easily retrieve the induced edge set by the following thresholding function:

$$\hat{\mathbf{e}}_{\bullet,ij} = \begin{cases} 1, & \text{if } |(\mathbf{D}_{\bullet} \hat{\mathbf{x}})_{ij}| > \frac{\sqrt{\alpha}}{\gamma}, \\ 0, & \text{otherwise,} \end{cases} \quad (2.30)$$

which corresponds to a truncated quadratic thresholding function.

An illustration of using Blake-Zisserman model is provided in Figure 2.9-(d).

2.2.5 Continuous Mumford-Shah functional and the Ambrosio-Tortorelli approach

In [Mumford and Shah, 1989], the authors proposed to solve a free-discontinuity problem which is a problem formulated in term of a pair (\mathbf{x}, Γ) where $\Gamma \subset \Omega$ is a closed set of discontinuities and \mathbf{x} is a function on the domain Ω .

Considering the data image $\mathbf{z} : \Omega \rightarrow \mathbb{R}$, then the continuous Mumford-Shah functional is to solve the following variational problem:

$$\left(\widehat{\mathbf{x}}, \widehat{\Gamma} \right) \in \underset{\substack{\mathbf{x} \in C^2(\Omega) \\ \Gamma \subset \Omega}}{\text{Argmin}} \mathcal{E}_{\lambda_S, \lambda_E}(\mathbf{x}, \Gamma) \quad (2.31)$$

where

$$\mathcal{E}_{\lambda_S, \lambda_E}(\mathbf{x}, \Gamma) = \int_{\Omega \setminus \Gamma} |\mathbf{x}(t) - \mathbf{z}(t)|^2 dt + \lambda_S \int_{\Omega \setminus \Gamma} |\nabla \mathbf{x}(t)|^2 dt + \lambda_E \mathbf{H}^1(\Omega \cap \Gamma) \quad (2.32)$$

and $\lambda_S, \lambda_E > 0$ are hyperparameters which respectively impose the smoothness everywhere except on Γ and penalize the term $\mathbf{H}^1(\Omega \cap \Gamma)$ which is the Hausdorff measure of dimension 1 of Γ .

This is a challenging question to handle the curve Γ and the function \mathbf{x} in the minimization process. Hence, [Ambrosio and Tortorelli, 1990, Ambrosio, 1989] proposed to approximate Γ by a smooth function \mathbf{e} such that $\mathbf{e} \approx 1$ when the contour is detected and $\mathbf{e} \approx 0$ otherwise. The Ambrosio-Tortorelli (AT) functional is then defined by:

$$\begin{aligned} \mathcal{AT}_{\lambda_S, \lambda_E, \varepsilon}(\mathbf{x}, \mathbf{e}) = & \int_{\Omega} |\mathbf{x}(t) - \mathbf{z}(t)|^2 dt + \lambda_S \int_{\Omega} (1 - \mathbf{e}(t))^2 |\nabla \mathbf{x}(t)|^2 dt \\ & + \lambda_E \int_{\Omega} \left(\varepsilon |\nabla \mathbf{e}(t)|^2 + \frac{1}{4\varepsilon} \mathbf{e}(t)^2 \right) dt. \end{aligned} \quad (2.33)$$

The term $\nabla \mathbf{x}$ penalizes the variation on \mathbf{x} when $\mathbf{e}(t)$ equals 0, while we keep the discontinuity at these points and energy takes into account the third term instead of $\nabla \mathbf{x}$ when $\mathbf{e}(t)$ equals 1. One more important parameter ε appears in the above function. Theoretically, this parameter allows the Γ -convergence of function (2.33) to MS functional (2.32) when $\varepsilon \rightarrow 0$ [Ambrosio and Tortorelli, 1990]. In practice, as ε tends to 0, the penalization of $\|\mathbf{e}\|_2^2$ increases and enforces \mathbf{e} to become sparser, and contours become thinner.

Lately, there were many works making efforts to tackle this problem. In [Richardson and Mitter, 1994], the authors proposed an effective algorithm to solve (2.33) based on the gradient descent. In [Ambrosio et al., 2001], the authors addressed the second order model of (2.33). In [Zanetti et al., 2016], the authors also tried to solve this problem with the minimization of the discrete objective (by means of forward and central differences) which is approached with a vector-valued block coordinate descent as the discrete objective is quadratic in each variable when fixing the other. In each iteration, descent directions for \mathbf{x} and \mathbf{e} are found by using iterative preconditioned conjugate gradient (PCG). More recently, in [Foare et al., 2016, Foare, 2017] the

authors propose to formulate AT using the full framework of Discrete Calculus which is able to sharply represent discontinuities. By rewriting the AT functional with these tools, the authors treated the minimization problem by solving a linear system.

The solution of the Mumford-Shah model (2.32) is piecewise smooth. In the case when $\lambda_S \rightarrow +\infty$, the gradient $\nabla \mathbf{x}(t)$ is strongly penalized for all $\mathbf{x}(t) \notin \Gamma$. Subsequently, \mathbf{x} is forced to be piecewise constant, the discontinuity set Γ now becomes the union of segment boundaries of a partition of the domain Ω . In this situation, we call the resulting model the Potts model which is given by

$$\operatorname{argmin}_{\mathbf{x}_p, \mathcal{P}} \sum_{P \in \mathcal{P}} \int |\mathbf{x}_p(t) - \mathbf{z}(t)|^2 dt + \frac{\lambda}{2} \operatorname{length}(\partial P), \quad (2.34)$$

where \mathcal{P} is the partitions of the domain Ω and $\mathbf{x}_p : \Omega \rightarrow \mathbb{R}$ are constant on each segment $P \in \mathcal{P}$. The parameter $\lambda/2$ compensates the double counting when we calculate the sum of all segments boundaries. The discrete formulation was presented in 2.2.3.

An example of using Potts model is shown in Figure 2.9-(e).

2.2.6 Discrete Mumford-Shah functional

In [Foare et al., 2019], the discrete version of the MS (DMS) functional is defined as follows

$$\operatorname{minimize}_{\mathbf{x} \in \mathbb{R}^{CN}, \mathbf{e} \in \mathbb{R}^{N'}} \mathcal{E}_{\text{DMS}}(\mathbf{x}, \mathbf{e}) := \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \lambda_S \|(1 - \mathbf{e}) \odot \mathbf{D}\mathbf{x}\|^2 + g_E(\mathbf{e}), \quad (2.35)$$

where the first term is the data fidelity term, the second term imposes the smoothness everywhere else on edge set \mathbf{e} and $g_E(\mathbf{e})$ is the edge length penalization which has several choices such as ℓ_0 -pseudo norm, ℓ_1 -norm and the quadratic ℓ_1 also known as BerHu function defined in (3.24).

In [Foare et al., 2019], they also proposed two algorithmic strategies:

- the Proximal Alternating Linearized Minimization (PALM),
- Semi-Linearized Proximal Alternating Minimization (SL-PAM) algorithm which relax the bound associated with the step-size parameter in the updating step of \mathbf{e}

with convergence guarantees presented in details in Chapter 3.

An illustration of the estimation obtained with DMS- ℓ_1 is proposed in Figure 2.9-(f).

2.2.7 Relation between Mumford-Shah functional and Blake-Zisserman model

Recently, in [Pustelnik, 2023], the author establishes a link between the three term DMS-like formulation and Blake-Zisserman functional.

The author proposed to focus on :

$$(\widehat{\mathbf{x}}, \widehat{\mathbf{e}}) \in \underset{\mathbf{x} \in \mathbb{R}^{CN}, \mathbf{e} \in \mathbb{R}^{JN}}{\text{Argmin}} \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \frac{\varsigma^2}{2} \sum_l \phi(D_l \mathbf{x})(1 - \mathbf{e}_l)^2 + \alpha \sum_l \varphi(\mathbf{e}_{lj}), \quad (2.36)$$

where $\mathbf{A} \in \mathbb{R}^{CM \times CN}$ and l th-row of \mathbf{D} is denoted D_l , $\varsigma > 0$, $\alpha > 0$. $f(\cdot) : \mathbb{R}^{CN} \rightarrow (-\infty, +\infty]$ is a proper, convex, continuous function ; $\phi : \mathbb{R} \rightarrow (-\infty, +\infty]$, $\varphi : \mathbb{R} \rightarrow (-\infty, +\infty]$ are proper, lower semi continuous such that $\inf \varphi = 0$, $\varphi(0) = 0$ and $\inf \phi = 0$.

When $\varphi = |\cdot|$ and $\phi = |\cdot|^2$ which correspond to DMS- ℓ_1 , the above minimization problem is equivalent to the following dual minimization problem

$$\begin{cases} \widehat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmin}} \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \varsigma^2 \sum_l \tilde{\phi}_{\alpha/\varsigma^2}(\phi(D_l \mathbf{x})), \\ (\forall l) \widehat{\mathbf{e}} = \begin{cases} \text{prox}_{\frac{\alpha}{\varsigma^2 \phi(D_l \widehat{\mathbf{x}})}|\cdot|}(1) & \text{if } \phi(D_l \widehat{\mathbf{x}}) > 0, \\ 0 & \text{otherwise,} \end{cases} \end{cases} \quad (2.37)$$

where

$$(\forall \eta \geq 0) \tilde{\phi}_{\alpha/\varsigma^2}(\eta) = \begin{cases} \frac{\alpha}{\varsigma^2} (2 - \frac{\alpha}{\varsigma^2 \eta}) & \text{if } \eta > \frac{\alpha}{\varsigma^2}, \\ \eta & \text{if } 0 < \eta \leq \frac{\alpha}{\varsigma^2}, \\ 0 & \text{if } \eta = 0. \end{cases} \quad (2.38)$$

More interestingly, the author shows that when $\varphi(\eta) = 1$ if $\eta \neq 0$ and 0 if $\eta = 0$ (ℓ_0 norm), the minimization problem (2.35) can be reformulated as

$$\begin{cases} \widehat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmin}} \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \frac{\varsigma^2}{2} \sum_l \min(\phi(D_l \mathbf{x}), \frac{2\alpha}{\varsigma^2}), \\ (\forall l \in 1, \dots, J) \widehat{\mathbf{e}}_l = \begin{cases} 0 & \text{if } \phi(D_l \widehat{\mathbf{x}}) < \frac{2\alpha}{\varsigma^2}, \\ 1 & \text{if } \phi(D_l \widehat{\mathbf{x}}) > \frac{2\alpha}{\varsigma^2}, \\ [0, 1] & \text{if } \phi(D_l \widehat{\mathbf{x}}) = \frac{2\alpha}{\varsigma^2}. \end{cases} \end{cases} \quad (2.39)$$

when $\phi = |\cdot|^2$, this formulation can refer to the first-order Blake-Zisserman functional (2.26).

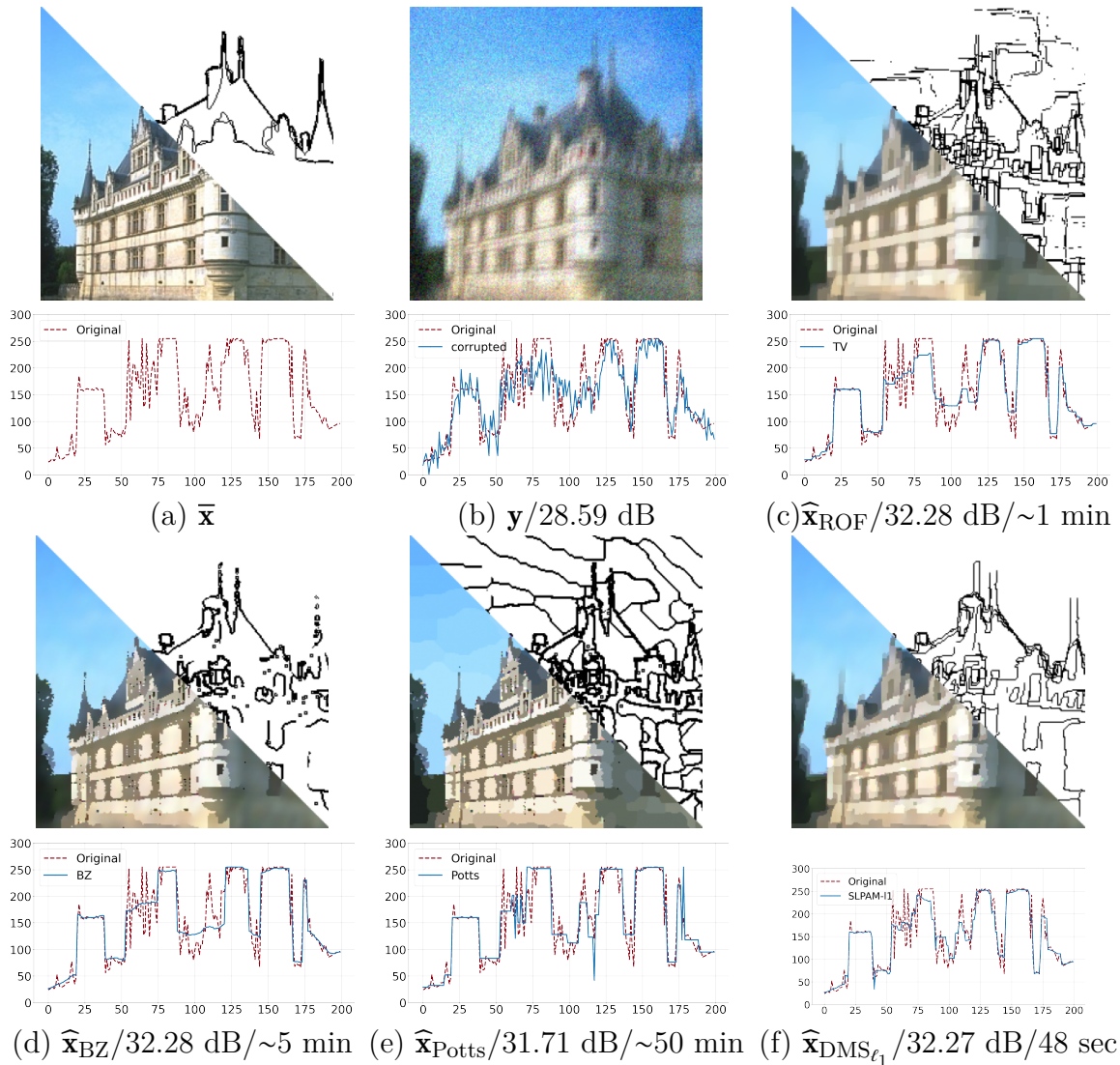


Figure 2.9: Image restoration using different variational methods: (a) Groundtruth \bar{x} (b) degraded image y (c) ROF [Rudin et al., 1992], the induced edges obtained by thresholding the absolute value of gradient map $|\mathbf{D}\mathbf{x}|$ with value 0.5 (d) BZ [Hohm et al., 2015], the induced edges obtained by using Eq.(2.30) (e) Potts [Storath et al., 2015], the induced edge obtained by detecting the jumpset between all segment (f) DMS- ℓ_1 [Foare et al., 2019], the thresholding value for \mathbf{e} is 0.5.

2.3 Conclusion

This chapter summarizes the principal variational approaches for performing joint restoration and contour detection. In Figure 2.9 most of the methods are displayed except AT for which the computation time with existing methods is not competitive. In chapter 4, we will focus on this penalization and we will provide new formulation with associated algorithmic minimization to solve it efficiently.

Chapter 3

Optimization and proximal algorithms

Summary

3.1	Optimization tools	24
3.1.1	Subdifferential of a convex function	24
3.1.2	Proximity operator	25
3.2	Optimization algorithms	28
3.2.1	Preliminary	29
3.2.2	General principal of descent method (Explicit scheme)	30
3.2.3	Implicit schemes	31
3.2.4	Sum of (Non)-differentiable convex functions	32
3.2.5	Bi-convex function	37
3.3	Conclusion	39

3.1 Optimization tools

The objective of this section is to recall some fundamental optimization tools that will be employed throughout this thesis.

3.1.1 Subdifferential of a convex function

In mathematics, concepts such as subgradient, and subdifferential serve as generalizations of the traditional derivative. They come into play when dealing with convex functions which are not necessarily differentiable. These ideas find their significance within convex analysis, a field dedicated to studying convex functions, particularly in their relationship to convex optimization. Such a powerful tools will be used widely in this thesis, we will familiarize with this concept in the following. We will work on \mathcal{H} which represents a real Hilbert space endowed with an inner product $\langle \cdot | \cdot \rangle$ and the associated norm $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x} | \mathbf{x} \rangle}$ (e.g. $\mathcal{H} = \mathbb{R}^N$ denotes real finite N -dimensional Euclidean space).

3.1.1.1 Definition and properties

All the following definitions and properties can be found in [Bauschke and Combettes, 2011].

Definition 3.1.1. *Let \mathcal{H} be a finite Hilbert space. Let $f : \mathcal{H} \rightarrow]-\infty, +\infty]$. The subdifferential of f at point \mathbf{x} denoted $\partial f(\mathbf{x})$ is defined such that*

$$\partial f(\mathbf{x}) = \{\mathbf{w} \in \mathcal{H} \mid (\forall \mathbf{y} \in \text{dom} f) \langle \mathbf{y} - \mathbf{x} | \mathbf{w} \rangle + f(\mathbf{x}) \leq f(\mathbf{y})\}.$$

The importance of the subdifferential is made clear through Fermat's rule, as defined by the following theorem. In the following, we define here $\Gamma_0(\mathcal{H})$ as the class of proper, lower semi-continuous, convex functions. The following definition will give us clue to find a minimizer of a function in $\Gamma_0(\mathcal{H})$.

Theorem 3.1.1. *Let $f \in \Gamma_0(\mathcal{H})$. Then:*

$$\text{Argmin} f = \text{zer } \partial f = \{\mathbf{x} \in \mathcal{H} \mid 0 \in \partial f(\mathbf{x})\}. \quad (3.1)$$

The following lemma will be useful for the next section of the manuscript in the majority of interesting practical cases such as minimization problem of the formulation (2.2).

Lemma 3.1.1. *Let $f \in \Gamma_0(\mathcal{H})$, $g \in \Gamma_0(\mathcal{H})$, $\text{dom} f = \mathcal{H}$ and g is continuous in a point of $\text{dom} f$ then*

$$\partial f + \partial g = \partial(f + g). \quad (3.2)$$

3.1.1.2 Calculus rules of subdifferential

The following is some essential calculus rules of subdifferential when we deal with a sum of multiple functions or the function is incorporated with a linear operator.

Lemma 3.1.2. *Let $f : \mathcal{H} \rightarrow (-\infty, +\infty]$ be a convex and differentiable function at \mathbf{x} then*

$$\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}.$$

Proposition 3.1.1. *Let \mathcal{H} and \mathcal{G} be two real Hilbert spaces.*

- *Let $f : \mathcal{H} \rightarrow]-\infty, +\infty]$, $g : \mathcal{H} \rightarrow]-\infty, +\infty]$ and $\mathbf{D} : \mathcal{H} \rightarrow \mathcal{G}$ be a linear operator. If $\text{dom}g \cap \mathbf{D}(\text{dom}f) \neq \emptyset$, then*

$$(\forall \mathbf{x} \in \mathcal{H}) \partial f(\mathbf{x}) + \mathbf{D}^* \partial g(\mathbf{D}\mathbf{x}) \subset \partial(f + g \circ \mathbf{D})(\mathbf{x}), \quad (3.3)$$

where \mathbf{D}^* is the adjoint of \mathbf{D} such that $\langle \mathbf{y} \mid \mathbf{D}\mathbf{x} \rangle_{\mathcal{G}} = \langle \mathbf{D}^*\mathbf{y} \mid \mathbf{x} \rangle_{\mathcal{H}}$.

- *Let $f \in \Gamma_0(\mathcal{H})$, $g \in \Gamma_0(\mathcal{G})$ and $\mathbf{D} : \mathcal{H} \rightarrow \mathcal{G}$ be a linear operator. If $\text{int}(\text{dom}g) \cap \mathbf{D}(\text{dom}f) \neq \emptyset$, then*

$$\partial f + \mathbf{D}^* \partial g \circ \mathbf{D} = \partial(f + g \circ \mathbf{D}). \quad (3.4)$$

3.1.2 Proximity operator

Jean Jacques Moreau [Moreau, 1962] introduced a valuable extension of the concept of projection operators, applicable to any convex function, resulting in what is now known as the proximity operator. In today's optimization, proximity operators have gained growing significance as essential components within non-smooth optimization throughout the class of proximal algorithms. These algorithms belong to a class of methods used to break down intricate composite convex optimization processes into more manageable steps. In this section, we will recall the definition of this powerful tool and some of its important properties that will be used in this manuscript.

3.1.2.1 Definition

Definition 3.1.2. *Let \mathcal{H} be a real Hilbert space. Let $f \in \Gamma_0(\mathcal{H})$ and $\tau > 0$.*

- *The Moreau envelope of τf at $\mathbf{z} \in \mathcal{H}$ is defined as*

$$M_{\tau f}(\mathbf{z}) = \inf_{\mathbf{x} \in \mathcal{H}} f(\mathbf{x}) + \frac{1}{2\tau} \|\mathbf{x} - \mathbf{z}\|^2. \quad (3.5)$$

- *The proximity operator of τf at $\mathbf{y} \in \mathcal{H}$ is defined as:*

$$\text{prox}_{\tau f}(\mathbf{y}) = \underset{\mathbf{x} \in \mathcal{H}}{\text{argmin}} f(\mathbf{x}) + \frac{1}{2\tau} \|\mathbf{x} - \mathbf{y}\|^2. \quad (3.6)$$

3.1.2.2 Properties

The proximity operator has many remarkable properties that make it a privileged tool when minimizing convex functions.

(i) **Moreau decomposition formula**

Let $f \in \Gamma_0(\mathcal{H})$ and $\tau \in (0, +\infty]$. We have

$$(\forall \mathbf{x} \in \mathcal{H}) \quad \mathbf{x} = \text{prox}_{\tau f^*}(\mathbf{x}) + \tau \text{prox}_{\tau^{-1}f}(\tau^{-1}\mathbf{x}) \quad (3.7)$$

where $f^* : \mathcal{H} \rightarrow (-\infty, +\infty]$ is the conjugate of f such that

$$f^*(\mathbf{u}) = \sup_{\mathbf{x}} (\langle \mathbf{x} | \mathbf{u} \rangle - f(\mathbf{x})). \quad (3.8)$$

(ii) **Additively separable function:**

For every $i \in \{1, \dots, N\}$, let \mathcal{H}_i be a Hilbert space and let $f_i \in \Gamma_0(\mathcal{H}_i)$. If

$$(\forall \mathbf{x} = (\mathbf{x}_i)_{1 \leq i \leq N} \in \mathcal{H} = \mathcal{H}_1 \times \dots \times \mathcal{H}_N) \quad F(\mathbf{x}) = \sum_{i=1}^N f_i(\mathbf{x}_i). \quad (3.9)$$

then $F \in \Gamma_0(\mathcal{H})$ is a separable function.

Furthermore,

$$(\forall \mathbf{x} \in \mathcal{H}) \quad \text{prox}_F(\mathbf{x}) = (\text{prox}_{f_i}(\mathbf{x}_i))_{1 \leq i \leq N} \quad (3.10)$$

Observations:

(i) $M_f(\mathbf{y})$ is real-valued while $\text{prox}_f(\mathbf{y})$ is \mathcal{H} -valued.

(ii) Let \mathcal{H} be a Hilbert space and $f \in \Gamma_0(\mathcal{H})$ then

$$\mathbf{x} = \text{prox}_f(\mathbf{y}) \Leftrightarrow \mathbf{y} - \mathbf{x} \in \partial f(\mathbf{x}). \quad (3.11)$$

(iii) It's important to note that if f is not convex, the proximity operator could exhibit multiple solutions. The convexity of f serves as a sufficient criterion for the existence and uniqueness of the proximity operator but it's not a necessary condition.

(iv) Going forward, we will refer to a "proximable" function if there is a closed form expression of its proximity operator. Many examples are introduced in [the proximity operator repository](#) [Chierchia et al., 2020] and few examples are provided below.

3.1.2.3 Examples

In this manuscript, we will focus on some basic examples in the following.

(i) **Projection operator:**

We recall first the projection operator onto a set $S \subset \mathcal{H}$

Definition 3.1.3. Let S be a subset of \mathcal{H} . The projection operator P_S of $\mathbf{x} \in \mathcal{H}$ is defined as

$$P_S(\mathbf{x}) = \text{prox}_{\iota_S}(\mathbf{x}), \quad (3.12)$$

where ι_S is the indicator function of set S defined as follows

$$\iota_S = \begin{cases} 0, & \text{if } \mathbf{x} \in S, \\ +\infty, & \text{otherwise.} \end{cases} \quad (3.13)$$

(ii) **Power q function [Chaux et al., 2007]:**

$$\left(\mathbf{x} = (\mathbf{x}_i)_{1 \leq i \leq N} \in \mathbb{R}^N \right) \quad \Phi_q = \sum_{i=1}^N |\mathbf{x}_i|^q. \quad (3.14)$$

We observe that Φ_q is additively separable, i.e,

$$\Phi_q = \sum_{i=1}^N \phi_q(\mathbf{x}_i), \quad (3.15)$$

where $\phi_q : x \rightarrow |x|^q$.

- When $q = 1$ we call it *soft-thresholding function*.

Definition 3.1.4. Let $\tau > 0$. We define the soft-thresholding function as follows:

$$(\forall x \in \mathbb{R}) \quad \text{soft}_\tau(x) = \text{sign}(x) \max\{|x| - \tau, 0\}, \quad (3.16)$$

equivalent to

$$(\forall x \in \mathbb{R}) \quad \text{soft}_\tau(x) = \begin{cases} x - \tau & \text{if } x > \tau, \\ 0 & \text{if } x \in [-\tau, \tau], \\ x + \tau & \text{if } x < -\tau. \end{cases} \quad (3.17)$$

- When $q \geq 1$ only some values of q lead to a closed form expression of the proximity operator:

$$(\forall x \in \mathbb{R}) \quad \text{prox}_{\tau|\cdot|^q}(x) = \begin{cases} \text{soft}_\tau(x) & \text{if } q = 1, \\ \frac{x}{1+2\tau} & \text{if } q = 2, \\ \text{sign}(x) \frac{\sqrt{1+12\tau|x|}-1}{6\tau} & \text{if } q = 3, \\ \left(\frac{t+x}{8\tau} \right)^{1/3} - \left(\frac{t-x}{8\tau} \right)^{1/3}, \quad t = \sqrt{x^2 + 1/(27\tau)} & \text{if } q = 4, \end{cases} \quad (3.18)$$

(iii) **Huber loss:**

The Huber function refers to the smooth approximation of the ℓ_1 -norm parametrized by $\delta > 0$, and is defined as

$$\Phi : \mathbb{R}^N \rightarrow \mathbb{R} : (\mathbf{x}_i)_{1 \leq i \leq N} \mapsto \sum_{i=1}^N \phi_\delta(\mathbf{x}_i) \quad (3.19)$$

where

$$\phi_\delta : x \mapsto \begin{cases} |x| - \frac{\delta}{2}, & \text{if } |x| > \delta \\ \frac{|x|^2}{2\delta}, & \text{if } |x| \leq \delta. \end{cases} \quad (3.20)$$

Then,

$$\partial\phi = \nabla\phi : x \mapsto \begin{cases} \text{sign}(x), & \text{if } |x| > \delta \\ \frac{x}{\delta}, & \text{if } |x| \leq \delta. \end{cases} \quad (3.21)$$

The proximity operator of the Huber function is:

$$\text{prox}_{\tau\phi_\delta}(x) \begin{cases} x - \tau\text{sign}(x) & \text{if } |x| \geq \tau + \delta, \\ \frac{\delta x}{\tau + \delta} & \text{otherwise.} \end{cases} \quad (3.22)$$

(iv) **BerHu loss:**

The Berhu function is defined as

$$\Phi : \mathbb{R}^N \rightarrow \mathbb{R} : (\mathbf{x}_i)_{1 \leq i \leq N} \mapsto \sum_{i=1}^N \phi_\delta(\mathbf{x}_i) \quad (3.23)$$

where

$$\phi_\delta : x \mapsto \begin{cases} \frac{x^2 + \delta^2}{2\delta}, & \text{if } |x| > \delta \\ |x|, & \text{if } |x| \leq \delta. \end{cases} \quad (3.24)$$

The proximity operator of the BerHu function is:

$$\text{prox}_{\phi_\delta}(x) \begin{cases} \frac{\delta x}{\delta + 1} & \text{if } |x| > \delta + 1, \\ x - \text{sign}(x) & \text{if } 1 < |x| \leq \delta + 1 \\ 0 & \text{otherwise.} \end{cases} \quad (3.25)$$

3.2 Optimization algorithms

In this section, we will present some optimization methods to approach a solution of the following problem:

$$\min_{\mathbf{x} \in \mathcal{H}} f(\mathbf{x}) \quad (3.26)$$

where $f \in \Gamma_0(\mathcal{H})$.

3.2.1 Preliminary

When considering Fermat rule and that no closed form solution $\widehat{\mathbf{x}}$ is available, we need to design an algorithm to approximate a solution, i.e. building a sequence $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ such that

$$\lim_{k \rightarrow +\infty} \mathbf{x}^{[k]} = \widehat{\mathbf{x}}$$

Definition 3.2.1. Let \mathcal{H} be a Hilbert space. Let $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ be a sequence in \mathcal{H} and $\widehat{\mathbf{x}} \in \mathcal{H}$.

- $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ converges strongly to $\widehat{\mathbf{x}}$ if

$$\lim_{k \rightarrow +\infty} \|\mathbf{x}^{[k]} - \widehat{\mathbf{x}}\| = 0. \quad (3.27)$$

- $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ converges weakly to $\widehat{\mathbf{x}}$ if

$$(\forall \mathbf{y} \in \mathcal{H}) \quad \lim_{k \rightarrow +\infty} \langle \mathbf{y} | \mathbf{x}^{[k]} - \widehat{\mathbf{x}} \rangle = 0. \quad (3.28)$$

- In a finite dimensional Hilbert space, strong and weak convergence are equivalent.

Standard approach is to design an operator $T : \mathcal{H} \rightarrow \mathcal{H}$ that verifies the theorem 3.2.1.

Theorem 3.2.1. [Banach-Picard theorem] Let $L \in [0, 1)$, let $T : \mathcal{H} \rightarrow \mathcal{H}$ be a L -Lipschitz continuous operator, and let $\mathbf{x}^{[0]} \in \mathcal{H}$. We define a sequence $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ as

$$(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = T\mathbf{x}^{[k]}.$$

Then, $\text{Fix}T = \{\widehat{\mathbf{x}}\}$ for some $\widehat{\mathbf{x}} \in \mathcal{H}$ and

$$(\forall k \in \mathbb{N}) \quad \|\mathbf{x}^{[k]} - \widehat{\mathbf{x}}\| \leq L^k \|\mathbf{x}^{[0]} - \widehat{\mathbf{x}}\|.$$

We say $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ converges strongly to $\widehat{\mathbf{x}}$ with linear convergence rate L .

The following is some essential definitions that we need to characterize some important properties of the operator T .

Definition 3.2.2. An operator $T : \mathbb{R}^{\mathcal{H}} \rightarrow \mathbb{R}^{\mathcal{H}}$ is said to be L -Lipschitz (or L -Lipschitz continuous) if for all $(\mathbf{x}, \mathbf{y}) \in (\mathcal{H} \times \mathcal{H})$, we have

$$\|T\mathbf{x} - T\mathbf{y}\| \leq L\|\mathbf{x} - \mathbf{y}\|.$$

If T is 1-Lipschitz continuous then we say that T is non expansive.

Definition 3.2.3. A differentiable function f is said to have an $L_{\nabla f}$ -Lipschitz continuous gradient if

$$(\forall \mathbf{x}, \mathbf{y}) \quad \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L_{\nabla f} \|\mathbf{x} - \mathbf{y}\|.$$

Definition 3.2.4. Let \mathcal{H} be Hilbert space. For every $L_{\nabla f} > 0$, we says that $f : \mathcal{H} \rightarrow \mathbb{R}$ belongs to the class $C_L^{1,1}(\mathcal{H})$ if f satisfies:

- f is Gâteaux differentiable in \mathcal{H} .
- $\nabla f : \mathcal{H} \rightarrow \mathcal{H}$ is $L_{\nabla f}$ -Lipschitz continuous.

Definition 3.2.5. An operator $T : \mathcal{H} \rightarrow \mathcal{H}$ is α -averaged nonexpansive for some $\alpha \in (0, 1]$ if

$$(\forall (\mathbf{x}, \mathbf{y}) \in \mathcal{H} \times \mathcal{H}) \quad \|T\mathbf{x} - T\mathbf{y}\|^2 \leq \|\mathbf{x} - \mathbf{y}\|^2 - \left(\frac{1-\alpha}{\alpha}\right) \|(\text{Id} - T)\mathbf{x} - (\text{Id} - T)\mathbf{y}\|^2.$$

Theorem 3.2.2. Let $\alpha \in (0, 1]$, let $T : \mathcal{H} \rightarrow \mathcal{H}$ be an α -averaged nonexpansive operator such that $\text{Fix}T \neq \emptyset$, and let $\mathbf{x}^{[0]} \in \mathcal{H}$. We define a sequence $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ as

$$(\forall k \in \mathbb{N}) \quad \mathbf{x}^{[k+1]} = T\mathbf{x}^{[k]}.$$

Then $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ converges weakly to a point in $\text{Fix}T$

3.2.2 General principal of descent method (Explicit scheme)

A descent algorithm seeks to generate an iterative sequence $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ starting from an initial point $\mathbf{x}^{[0]} \in \mathcal{H}$ defined as

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + \gamma_k d^{[k]}, \quad (3.29)$$

which verifies

$$(\forall k \in \mathbb{N}) \quad f(\mathbf{x}^{[k+1]}) \leq f(\mathbf{x}^{[k]}), \quad (3.30)$$

where $d^{[k]} \in \mathbb{R}^N$ is the choice of a descent direction defined in Definition 3.2.6 and $\gamma_k > 0$ is the step-size.

Definition 3.2.6. [Descent direction] Let $f \in \Gamma_0(\mathcal{H})$ be continuously differentiable, $\mathbf{d} \in \mathcal{H}$ is a descent direction at point $\mathbf{x} \in \mathcal{H}$ if

$$\langle \nabla f(\mathbf{x}) \mid \mathbf{d} \rangle < 0.$$

Considering $\mathbf{d} = -\nabla f$, the resulting scheme is known as the steepest descent algorithm whose iterations are provided by Algorithm 1.

Algorithm 3.1: Gradient descent algorithm

Initialisation : $\mathbf{x}^{[0]} \in \mathcal{H}$, $\gamma \in (0, \frac{2}{L})$

For $k = 0, 1, \dots$

$$\left[\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \gamma \nabla f(\mathbf{x}^{[k]}) \right].$$

We can denote here operator $T : \mathcal{H} \rightarrow \mathcal{H}$ as follows

$$T\mathbf{x} = (\text{Id} - \gamma \nabla f)(\mathbf{x}),$$

and the sequence $(\mathbf{x}^{[k]})$ now can be simply defined as $\mathbf{x}^{[k+1]} = T\mathbf{x}^{[k]}$.

In order to ensure the convergence, the step-size should be carefully selected.

Proposition 3.2.1. *Let $f \in \Gamma_0(\mathcal{H})$ with $f \in C_{L\nabla f}^{1,1}$. For some $\gamma > 0$, we define $T := \text{Id} - \gamma\nabla f$. For all $\mathbf{x}^{[0]} \in \mathcal{H}$, if $0 < \gamma < \frac{2}{L\nabla f}$ then the sequence $\mathbf{x}^{[k+1]} = T\mathbf{x}^{[k]}$ converges to a minimizer of f .*

When f is no longer differentiable, we can still choose a descent direction at $\mathbf{x}^{[k]}$ by using the subdifferential at this point. The descent algorithm is now called *subgradient descent algorithm* and reads

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \gamma d^{[k]},$$

where $d^{[k]} \in \partial f(\mathbf{x}^{[k]})$ but it requires some conditions over γ to ensure the convergence.

3.2.3 Implicit schemes

Implicit gradient descent refers to a variant of the gradient descent optimization algorithm that incorporates an implicit update rule instead of the explicit update rule. This raises a challenge as it requires solving an implicit equation at each iteration.

We consider $\gamma > 0$ and $\mathbf{x}^{[0]} \in \mathcal{H}$. The implicit gradient descent update at $\mathbf{x}^{[k]}$ is:

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \gamma \partial f(\mathbf{x}^{[k+1]}). \quad (3.31)$$

Considering (3.11) the implicit scheme can be rewritten :

$$\mathbf{x}^{[k+1]} = \text{prox}_{\gamma f}(\mathbf{x}^{[k]}),$$

For such a scheme, we can define the operator

$$T := \text{prox}_{\gamma f}.$$

The advantage of implicit subgradient descent method is the flexibility in the choice of the descent step size but at the price to compute the proximity operator, for which an exhaustive list is provided by [the proximity operator repository \[Chierchia et al., 2020\]](#).

The proximal point algorithm [[Rockafellar, 1976](#)] reads

Algorithm 3.2: Proximal point method

Initialisation : $\mathbf{x}^{[0]} \in \mathcal{H}, \forall k \in \mathbb{N}, \gamma_k \in (0, +\infty)$

For $k = 0, 1, \dots$

$\lfloor \mathbf{x}^{[k+1]} \in \text{prox}_{\gamma_k f}(\mathbf{x}^{[k]}).$

Proposition 3.2.2. *Let $f \in \Gamma_0(\mathcal{H})$. We set $T := \text{prox}_{\gamma f}$ for some $\gamma > 0$ and define sequence $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ as $\mathbf{x}^{[k+1]} = T\mathbf{x}^{[k]}$, then*

- $(\forall \gamma > 0), \text{Fix}T = \text{zer}\partial f$.
- $(\forall \gamma > 0)$ and any $f \in \Gamma_0(\mathcal{H})$, T is firmly nonexpansive. [[Bauschke and Combettes, 2011](#)]
- The sequence $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ converges to a point in $\text{zer}\partial f$.

3.2.4 Sum of (Non)-differentiable convex functions

In the previous section, we introduced some algorithms allowing us to minimize a convex function, differentiable or not. In this section, we focus on the scenario of non-smooth convex optimization problems. A particularly challenging case arises when we encounter a problem involving the sum of two convex functions, where one of the functions is not necessarily smooth, given by

$$\widehat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathcal{H}}{\text{Argmin}} f(\mathbf{x}) + g(\mathbf{x}), \quad (3.32)$$

where $f \in \Gamma_0(\mathcal{H})$ is differentiable with $L_{\nabla f}$ -Lipschitz gradient, $g \in \Gamma_0(\mathcal{H})$.

Some algorithms are capable of minimizing this kind of optimization problem by alternating sub-operation of each function separately, called *splitting algorithms* such as Forward-Backward.

Forward-backward (FB) algorithm – The Forward-Backward algorithm is an algorithm that is used to solve (3.32). The algorithm consists of two alternating steps, one is an explicit gradient descent on f , while the other is a proximal point iteration on g (explicit-implicit algorithm). By starting with an initial point $\mathbf{x}^{[0]} \in \mathcal{H}$ and step-size $\tau > 0$, the algorithm reads:

Algorithm 3.3: Forward-Backward algorithm

Initialisation : $\mathbf{x}^{[0]} \in \mathcal{H}, \forall k \in \mathbb{N}, \tau \in (0, +\infty)$.

For $k = 0, 1, \dots$

$$\left[\mathbf{x}^{[k+1]} = \text{prox}_{\tau g} \left(\mathbf{x}^{[k]} - \tau \nabla f(\mathbf{x}^{[k]}) \right) \right]$$

Theorem 3.2.3. *Let $f + g$ be a sum of two convex, coercive, lower semicontinuous functions that are bounded from below. We suppose that f is differentiable with an $L_{\nabla f}$ -Lipschitz gradient. Let $\tau < \frac{2}{L_{\nabla f}}$, $\mathbf{x}^{[0]} \in \mathcal{H}$ be an initial guess and let $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ be the sequence defined by*

$$\mathbf{x}^{[k+1]} = \text{prox}_{\tau g}(\text{Id} - \tau \nabla f)(\mathbf{x}^{[k]}). \quad (3.33)$$

Then the sequence $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$ converges to a minimizer of $f + g$.

Fast Iterative Shrinkage thresholding Algorithm (FISTA) – To improve the convergence of the algorithm, a popular technique is to use inertial-type methods [Nesterov, 1983, Moudafi and Oliny, 2003, Attouch and Peypouquet, 2016]. Starting with $\mathbf{v}^{[0]} = \mathbf{v}^{[1]} \in \mathcal{H}$ and $\mathbf{x}^{[0]} \in \mathcal{H}$, for $k \geq 1$, the inertial forward-backward method reads:

$$\begin{cases} \mathbf{v}^{[k+1]} = \text{prox}_{\tau_k g}(\mathbf{x}^{[k]} - \tau_k \nabla f(\mathbf{x}^{[k]})), \\ \mathbf{x}^{[k]} = \mathbf{v}^{[k]} + \rho_k(\mathbf{v}^{[k]} - \mathbf{v}^{[k-1]}), \end{cases} \quad (3.34)$$

where $\tau_k > 0$, $\rho_k \geq 0$ is the inertial parameter.

In [Beck and Teboulle, 2009], the authors proposed a fast iterative shrinkage-threshold algorithm method (FISTA) when choosing $\tau_k = \frac{1}{L_{\nabla f}}$ and

$$\rho_k = \frac{t_k - 1}{t_{k+1}}, \quad \text{where } t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2} \text{ and } t_0 = 1.$$

In [Chambolle and Dossal, 2015], the author proposed another way to define the sequence $(\rho_k)_{k \in \mathbb{N}}$ allowing to ensure the convergence of the iterates. Their FISTA version reads:

Algorithm 3.4: FISTA [Chambolle and Dossal, 2015]

Initialisation : $\mathbf{x}^{[0]}, \mathbf{y}^{[0]} \in \mathcal{H}, \forall k \in \mathbb{N}, \tau_k \in (0, +\infty)$ and $a > 2$

For $k = 0, 1, 2 \dots$

$$\left[\begin{array}{l} \mathbf{x}^{[k+1]} = \text{prox}_{\tau_k g} \left(\mathbf{x}^{[k]} - \tau_k \nabla f(\mathbf{e}^{[k]}) \right), \\ \mathbf{e}^{[k+1]} = (1 + \rho_k) \mathbf{x}^{[k+1]} - \rho_k \mathbf{x}^{[k]}, \\ \rho_k = \frac{t_k - 1}{t_{k+1}} \text{ with } t_k = \frac{k+a-1}{a}. \end{array} \right.$$

3.2.4.1 Dealing with constraints

We can also generalize (3.32) as a sum of many functions (non necessary differentiable). For example, the variational minimization problem can take into account constraints. The minimization problem (3.32) now becomes

$$\widehat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathcal{H}}{\text{Argmin}} f(\mathbf{x}) + g(\mathbf{x}) + \iota_S(\mathbf{x}), \quad (3.35)$$

where S is a closed, non-empty, convex set. In the remainder of this manuscript, for the convenience for the reader, the problem (3.35) will be expressed as:

$$\widehat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathcal{H}}{\text{Argmin}} f(\mathbf{x}) + \lambda \Phi(\mathbf{D}\mathbf{x}) + \iota_S(\mathbf{x}) \quad (3.36)$$

or equivalently

$$\widehat{\mathbf{x}} \in \underset{\mathbf{x} \in S}{\text{Argmin}} f(\mathbf{x}) + \lambda \Phi(\mathbf{D}\mathbf{x}). \quad (3.37)$$

3.2.4.2 Chambolle-Pock algorithm

This section is based on a part of work in [Chambolle and Pock, 2011]. The idea of Chambolle-Pock algorithm is to facilitate the computation of $\text{prox}_{\Phi \circ \mathbf{D}}$ which has rarely a closed form expression.

Theorem 3.2.4. *Let $f \in \Gamma_0(\mathcal{H})$, $\Phi \in \Gamma_0(\mathcal{G})$ and $\mathbf{D} : \mathcal{H} \rightarrow \mathcal{G}$ is a linear operator and S a closed, non-empty convex subset of \mathcal{H} . We choose $\tau_k \in]0, +\infty]$, $\mu_k \in]0, +\infty]$ such that $\tau_k \mu_k \|\mathbf{D}\|_{sp}^2 < 1$ ¹ for every $k \in \mathbb{N}$. Let $(\mathbf{x}^{[0]}, \mathbf{y}^{[0]}) \in \mathcal{H} \times \mathcal{G}$ and $\mathbf{v}^{[0]} = \mathbf{x}^{[0]}$. We define the sequences $(\mathbf{x}^{[k]})_{k \in \mathbb{N}}$, $(\mathbf{y}^{[k]})_{k \in \mathbb{N}}$ and $(\mathbf{v}^{[k]})_{k \in \mathbb{N}}$ by*

¹ $\|\mathbf{D}\|_{sp}$ is the spectral norm of \mathbf{D} .

Algorithm 3.5: CP algorithm

Initialisation : $\mathbf{x}^{[0]}, \mathbf{y}^{[0]} \in \mathcal{H}, \forall k \in \mathbb{N}, \tau_k \in (0, +\infty)$

For $k = 0, 1, \dots$

$$\begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\mu(f+\iota_S)}(\mathbf{x}^{[k]} - \mu \mathbf{D}^* \mathbf{v}^{[k]}) \\ \mathbf{y}^{[k]} = 2\mathbf{x}^{[k+1]} - \mathbf{x}^{[k]} \\ \mathbf{v}^{[k+1]} = \text{prox}_{\tau(\lambda\Phi)^*}(\mathbf{v}^{[k]} + \tau \mathbf{D} \mathbf{y}^{[k]}) \end{cases}$$

Then the sequence $(\mathbf{x}^{[k]}, \mathbf{y}^{[k]})_{k \in \mathbb{N}}$ converges to a critical point $(\mathbf{x}^*, \mathbf{y}^*)$ of (3.35).

When f is ξ -strongly convex (the definition is provided in Def. 3.2.8 or Def. 3.2.7), the Strongly Convex CP algorithm reads:

Algorithm 3.6: Strongly convex CP algorithm

Initialisation : $\mathbf{x}^{[0]}, \mathbf{y}^{[0]} \in \mathcal{H}, \forall k \in \mathbb{N}, \tau_k \in (0, +\infty)$

For $k = 0, 1, \dots$

$$\begin{cases} \mathbf{x}^{[k+1]} = \text{prox}_{\mu_k(f+\iota_S)}(\mathbf{x}^{[k]} - \mu_k \mathbf{D}^* \mathbf{v}^{[k]}) \\ \mathbf{u}^{[k+1]} = \text{prox}_{\tau_k(\lambda\Phi)^*}(\mathbf{v}^{[k]} + \tau_k \mathbf{D}((1 + \alpha_k)\mathbf{x}^{[k+1]} - \alpha_k \mathbf{x}^{[k]})) \\ \alpha_k = (1 + 2\mu_k)^{-1/2} \\ \tau_{k+1} = \tau_k \alpha_k^{-1} \\ \mu_{k+1} = \alpha_k \mu_k \end{cases}$$

where for every $k \in \mathbb{N}$, $\mu_0 \tau_0 \|\mathbf{D}\|_{sp}^2 \leq 1$.

Definition 3.2.7. $f : \mathcal{H} \rightarrow (-\infty, +\infty]$ is a strongly convex function with constant $\mu_k > 0$ if

$$(\forall (\mathbf{x}, \mathbf{y}) \in (\text{dom} f)^2), \forall \lambda \in (0, 1), \quad f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) + \frac{\xi f}{2} \|\mathbf{y} - \mathbf{x}\|_2^2.$$

Definition 3.2.8. Let $f : \mathcal{H} \rightarrow (-\infty, +\infty]$. f is a strongly convex function with constant $\xi > 0$ if

$$(\forall (\mathbf{x}, \mathbf{y}) \in (\text{dom} f)^2), \forall \lambda \in (0, 1), \quad f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}) - \frac{\lambda(1-\lambda)\xi}{2} \|\mathbf{x} - \mathbf{y}\|_2^2. \quad (3.38)$$

The following convergence result applies.

Theorem 3.2.5 ([Chambolle and Pock, 2011]). Let $(\mathbf{u}^{[k]}, \mathbf{x}^{[k]})_{k \in \mathbb{N}}$ be sequences generated by Algorithm 6. Assume that $(\tau_k)_{k \in \mathbb{N}}$ and $(\mu_k)_{k \in \mathbb{N}}$ are positive sequences, and that one of the following conditions is satisfied.

1. For every $k \in \mathbb{N}$, $\tau_k \mu_k \|\mathbf{D}\|_{sp}^2 < 1$, and $\alpha_k = 1$.
2. For every $k \in \mathbb{N}$, $\alpha_k = (1 + 2\mu_k)^{-1/2}$, $\mu_{k+1} = \alpha_k \mu_k$, and $\tau_{k+1} = \tau_k \alpha_k^{-1}$ with $\mu_0 \tau_0 \|\mathbf{D}\|_{sp}^2 \leq 1$.

Then we have

$$\widehat{\mathbf{x}} = \lim_{k \rightarrow \infty} \mathbf{x}^{[k]}, \quad (3.39)$$

where $\widehat{\mathbf{x}}$ is defined in (3.35).

3.2.4.3 FB and FISTA in the dual

Function (3.36) can be minimized efficiently using proximal splitting methods [Bauschke and Combettes, 2017a, Combettes and Pesquet, 2011a, Chambolle and Pock, 2016]. The choice of the most appropriate algorithm will depend on the properties of f , Φ , \mathbf{D} and S . For instance, when $\Phi \circ \mathbf{D}$ is proximable, Douglas-Rachford (DR) scheme [Combettes and Pesquet, 2007] can be considered, alternating, at each iteration, between a proximity step on the sum of f and the indicator function, and a proximity step on the penalization function $\Phi \circ \mathbf{D}$. However, when $\Phi \circ \mathbf{D}$ is not proximable nor differentiable (e.g., TV penalization, or when \mathbf{D} is a redundant wavelet transform), more advanced algorithms, relying on duality, must be used.

Such methods usually rely on the Fenchel-Rockafellar duality [Komodakis and Pesquet, 2015, Bauschke and Combettes, 2017a]. On the one hand, some algorithms can evolve fully in the dual space, such as, e.g., ADMM [Gabay and Mercier, 1976, Fortin and Glowinski, 1983, Boyd et al., 2011] or the dual-Forward-Backward (FB) [Combettes et al., 2009, Combettes et al., 2010]). Note however that ADMM requires the inversion of $\mathbf{D}^\top \mathbf{D}$. On the other hand, other algorithms can alternate between the primal and the dual spaces, namely primal-dual algorithms [Chambolle and Pock, 2011, Condat, 2013, Vũ, 2013, Combettes and Pesquet, 2012]).

The following definitions introduce the notion of conjugate of a function, some fundamentals of Fenchel's duality theorem which will help to the understanding of many primal-dual optimization algorithm.

(i) Conjugate and duality:

Definition 3.2.9. Let $f \in \Gamma_0(\mathcal{H})$, $g \in \Gamma_0(\mathcal{G})$ and $\mathbf{D} : \mathcal{H} \rightarrow \mathcal{G}$ is a bounded linear operator where \mathcal{H}, \mathcal{G} are two real Hilbert spaces. The primal problem is a minimization problem defined as

$$\underset{\mathbf{x} \in \mathcal{H}}{\text{minimize}} f(\mathbf{x}) + g(\mathbf{D}\mathbf{x}) \quad (3.40)$$

and the corresponding dual problem is defined as

$$\underset{\mathbf{u} \in \mathcal{G}}{\text{minimize}} f^*(-\mathbf{D}^*\mathbf{u}) + g^*(\mathbf{u}) \quad (3.41)$$

(ii) Duality theorem:

Theorem 3.2.6. Let \mathcal{H} and \mathcal{G} be real Hilbert spaces. Let $f \in \Gamma_0(\mathcal{H})$, $g \in \Gamma_0(\mathcal{G})$, $\mathbf{D} \in \mathcal{B}(\mathcal{H}, \mathcal{G})$. We have

$$\text{zer}(\partial f + \mathbf{D}\partial g\mathbf{D}^\top) \neq \emptyset \Leftrightarrow \text{zer}((-\mathbf{D})\partial f^*(-\mathbf{D}^\top) + \partial g^*) \neq \emptyset. \quad (3.42)$$

(iii) Karush-Kuhn-Tucker condition:

Theorem 3.2.7. Let \mathcal{H} and \mathcal{G} be real Hilbert spaces. Let $f \in \Gamma_0(\mathcal{H})$, $g \in \Gamma_0(\mathcal{G})$, $\mathbf{D} \in \mathcal{B}(\mathcal{H}, \mathcal{G})$.

- If $\exists \hat{\mathbf{x}} \in \mathcal{H}$ s.t $0 \in \partial f(\hat{\mathbf{x}}) + \mathbf{D}^\top \partial g(\mathbf{D}\hat{\mathbf{x}})$, then $\hat{\mathbf{x}}$ is a solution of the primal problem (3.35). Moreover, there exists a solution $\hat{\mathbf{u}}$ of the dual problem s.t $-\mathbf{D}^*\hat{\mathbf{u}} \in \partial f(\hat{\mathbf{x}})$ and $\mathbf{D}\hat{\mathbf{x}} \in \partial g^*(\hat{\mathbf{u}})$.

- If $\exists(\widehat{\mathbf{x}}, \widehat{\mathbf{u}}) \in \mathcal{H} \times \mathcal{G}$ such that $-\mathbf{D}^* \widehat{\mathbf{u}} \in \partial f(\widehat{\mathbf{x}})$ and $\mathbf{D} \widehat{\mathbf{x}} \in \partial g^*(\widehat{\mathbf{u}})$ then $\exists(\widehat{\mathbf{x}}, \widehat{\mathbf{u}})$ is called a Kuhn-Tucker point.

Dual formulation in denoising case – When $\mathbf{A} = \text{Id}$, the degradation model (2.1) boils down to the Gaussian denoising problem

$$\mathbf{z} = \bar{\mathbf{x}} + \mathbf{n}, \quad (3.43)$$

where $\mathbf{n} \in \mathbb{R}^{CN}$ models an additive white Gaussian noise with standard deviation $\delta > 0$.

Thus, the standard method to denoise the image is to minimize the general formulation (3.35) where $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2$ and $g(\mathbf{x}) = \lambda \Phi(\mathbf{D}\mathbf{x})$, the primal formulation of (3.35) reads:

$$\widehat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathbb{R}^N}{\text{Argmin}} \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda \Phi(\mathbf{D}\mathbf{x}) + \iota_S(\mathbf{x}), \quad (3.44)$$

Based on Lemma 2.5 in [Combettes et al., 2010], set $\varphi : \mathbf{x} \mapsto \psi(\mathbf{x}) + \|\mathbf{x} - \mathbf{w}\|_2^2$. Then the conjugate $\varphi^* : \mathbf{y} \mapsto M_{\psi^*}(\mathbf{y} + \mathbf{w}) - \frac{1}{2} \|\mathbf{w}\|_2^2$ where M_{ψ^*} and M_{ψ} are Moreau envelopes (see definition (3.5)) of ψ^* and ψ . The dual formulation of problem (3.35) then reads:

$$\widehat{\mathbf{u}} \in \underset{\mathbf{u} \in \mathbb{R}^M}{\text{Argmin}} \widetilde{\mathcal{G}} := M_{\iota_S^*}(-\mathbf{D}^\top \mathbf{u} + \mathbf{z}) - \frac{1}{2} \|\mathbf{z}\|_2^2 + (\lambda \Phi)^*(\mathbf{u}) \quad (3.45)$$

and $\widehat{\mathbf{x}} = P_S(\mathbf{z} - \mathbf{D}^\top \widehat{\mathbf{u}})$.

Based on (iii) and (iv) in Lemma 2.3 in [Combettes et al., 2010]:

$$\nabla M_{\psi^*} = \text{prox}_{\psi} = \text{Id} - \text{prox}_{\psi^*}. \quad (3.46)$$

Applying the lemma in the case of $\psi = \iota_S$, we can write

$$\nabla_{\mathbf{u}} \left(M_{\iota_S^*}(-\mathbf{D}^\top \mathbf{u} + \mathbf{z}) - \frac{1}{2} \|\mathbf{z}\|_2^2 \right) = P_S(\mathbf{z} - \mathbf{D}^\top \mathbf{u}). \quad (3.47)$$

3.2.4.4 Dual (i)FB

A first strategy to solve (3.44) consists in applying (i)FB to its dual formulation (3.45). The resulting iterations are presented in Algorithm 7.

Algorithm 3.7: Dual Forward-Backward

Initialisation : $\mathbf{u}^{[0]}, \mathbf{v}^{[0]} \in \mathcal{G}, \forall k \in \mathbb{N}, \tau_k \in (0, +\infty), a > 2$

For $k = 0, 1, \dots$

$$\begin{cases} \mathbf{u}^{[k+1]} = \text{prox}_{\tau_k(\lambda \Phi)^*} \left(\mathbf{v}^{[k]} + \tau_k \mathbf{D} P_S(\mathbf{z} - \mathbf{D}^\top \mathbf{v}^{[k]}) \right), \\ \mathbf{v}^{[k+1]} = (1 + \rho_k) \mathbf{u}^{[k+1]} - \rho_k \mathbf{u}^{[k]}, \end{cases}$$

Note that when, for every $k \in \mathbb{N}, \rho_k = 0$, then algorithm 7 reduces to FB.

The following convergence result applies.

Theorem 3.2.8. [Combettes et al., 2010, Chambolle and Dossal, 2015] Let $(\mathbf{u}^{[k]}, \mathbf{v}^{[k]})_{k \in \mathbb{N}}$ be sequences generated by Algorithm 7. Assume that one of the following conditions is satisfied.

1. For every $k \in \mathbb{N}$, $\tau_k \in (0, 2/\|\mathbf{D}\|_{sp}^2)$, and $\rho_k = 0$.
2. For every $k \in \mathbb{N}$, $\tau_k \in (0, 1/\|\mathbf{D}\|_{sp}^2)$, and $\rho_k = \frac{t_k-1}{t_{k+1}}$ with $t_k = \frac{k+a-1}{a}$ and $a > 2$.

Then we have

$$\widehat{\mathbf{x}} = \lim_{k \rightarrow \infty} P_S(\mathbf{z} - \mathbf{D}^\top \mathbf{u}^{[k]}), \quad (3.48)$$

where $\widehat{\mathbf{x}}$ is defined in (3.44).

3.2.5 Bi-convex function

In this section, we consider the following generic minimization problem:

$$\min_{\mathbf{x}, \mathbf{e}} \mathcal{E}(\mathbf{x}, \mathbf{e}) := f(\mathbf{x}) + g_S(\mathbf{x}, \mathbf{e}) + g_E(\mathbf{e}) \quad (3.49)$$

where $f : \mathcal{H} \rightarrow (-\infty, +\infty]$, $g_E : \mathcal{G} \rightarrow (-\infty, +\infty]$ are proper lower semicontinuous, $g_S : \mathcal{H} \times \mathcal{G} \rightarrow (-\infty, +\infty]$ a C^1 function and ∇g_S is Lipschitz continuous on bounded subsets of $\mathcal{H} \times \mathcal{G}$. This function can cover many applications in image processing such as image restoration, sparse approximation of images, compressed sensing and blind decomposition, etc. Such a model can be also adapted in the context of Discrete Mumford-Shah functional that we presented (2.35).

The standard approach to solve (3.49) is a Gauss-Seidel scheme or alternating minimization, presented in Algorithm 8, in which we generate a sequence of $(\mathbf{x}^{[k]}, \mathbf{e}^{[k]})$.

Algorithm 3.8: Gauss-Seidel

$$\begin{array}{l} \text{For } k = 0, 1, \dots \\ \left[\begin{array}{l} \mathbf{x}^{[k+1]} \in \underset{\mathbf{x}}{\text{Argmin}} \mathcal{E}(\mathbf{x}, \mathbf{e}^{[k]}) \\ \mathbf{e}^{[k+1]} \in \underset{\mathbf{e}}{\text{Argmin}} \mathcal{E}(\mathbf{x}^{[k+1]}, \mathbf{e}) \end{array} \right. \end{array}$$

The convergence results for the Gauss-Seidel method is ensured under restricted conditions such as strict convexity w.r.t each variable. In the convex setting, if \mathcal{E} is a continuously differentiable function, and assuming that \mathcal{E} is strictly convex on each variable while the other is fixed then the sequence $\{\mathbf{x}^{[k]}, \mathbf{e}^{[k]}\}_{k \in \mathbb{N}}$ generated by Algorithm 8 minimizes \mathcal{E} .

To relax these assumptions, the above algorithm can be tackled in a different way by adding a quadratic term or can be seen as a proximal regularization of the two block Gauss-Seidel method:

Algorithm 3.9: Proximal Alternating Minimization algorithm

$$\begin{array}{l} \text{For } k = 0, 1, \dots \\ \left[\begin{array}{l} \mathbf{x}^{[k+1]} \in \underset{\mathbf{x}}{\text{Argmin}} \mathcal{E}(\mathbf{x}, \mathbf{e}^{[k]}) + \frac{1}{2\mu_k} \|\mathbf{x} - \mathbf{x}^{[k]}\|^2 \\ \mathbf{e}^{[k+1]} \in \underset{\mathbf{e}}{\text{Argmin}} \mathcal{E}(\mathbf{x}^{[k+1]}, \mathbf{e}) + \frac{1}{2\kappa_k} \|\mathbf{e} - \mathbf{e}^{[k]}\|^2, \end{array} \right. \end{array}$$

where μ_k and κ_k are real positive. This approach was also proposed by [Auslender, 1971]. In the non-convex and nonsmooth setting, one of the first works was established

by [Attouch et al., 2010]. The idea relies on assuming that the objective function \mathcal{E} satisfies the Kurdyka-Łojasiewicz property.

However, each step in this scheme can be reformulated as calculating a proximity operator of the sum of 2 functions which does not always have a closed form or can be easily computed. Thus this give rise to an idea that for each step, we perform one gradient descent step on the smooth function $g_S(\mathbf{x}, \mathbf{e})$ then a proximal step activate on the (non)-smooth function $f(\mathbf{x})$ or $g_E(\mathbf{e})$. This yield to the Proximal Alternating Linearized Minimization (PALM) algorithm described in Algorithm 10.

Algorithm 3.10: Proximal Alternating Linearized Minimization algorithm

$$\begin{array}{l} \text{For } k = 0, 1, \dots \\ \left[\begin{array}{l} \mathbf{x}^{[k+1]} \in \text{prox}_{\mu_k f} \left(\mathbf{x}^{[k]} - \mu_k \nabla_{\mathbf{x}} g_S(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) \right) \\ \mathbf{e}^{[k+1]} \in \text{prox}_{\kappa_k g_E} \left(\mathbf{e}^{[k]} - \kappa_k \nabla_{\mathbf{e}} g_S(\mathbf{x}^{[k+1]}, \mathbf{e}^{[k]}) \right), \end{array} \right. \end{array}$$

Under Proposition 3.2.3, the PALM algorithm described in Algorithm 10 fits the requirements for convergence in [Bolte et al., 2014][Assumptions A-B, Thm. 3.1].

Proposition 3.2.3. *The sequence $(\mathbf{x}^{[k]}, \mathbf{e}^{[k]})_{k \in \mathbb{N}}$ generated by Algorithm 10 converges to a critical point of Problem (2.35) if*

1. *the updating steps of $\mathbf{x}^{[k]}$ and $\mathbf{e}^{[k]}$ have closed form expression;*
2. *the sequence $(\mathbf{x}^{[k]}, \mathbf{e}^{[k]})_{k \in \mathbb{N}}$ generated by Algorithm 10 is bounded;*
3. *f , g_E and $\mathcal{E}(\cdot, \cdot)$ are bounded below;*
4. *\mathcal{E} is a Kurdyka-Łojasiewicz function;*
5. *$\nabla_{\mathbf{x}} g_S$ and $\nabla_{\mathbf{e}} g_E$ are globally Lipschitz continuous with moduli $\pi(\mathbf{e})$ and $\varpi(\mathbf{x})$ respectively, and for all $k \in \mathbb{N}$, $\pi(\mathbf{e}^{[k]})$, $\varpi(\mathbf{x}^{[k]})$ are bounded by positive constants.*

In [Foare et al., 2019], the authors proposed an alternative to PALM to deal with the DMS functional where the update \mathbf{e} relies on the proximity operator of the function $\{g_S(\mathbf{x}, \mathbf{e}) + g_E(\mathbf{e})\}$ whose iterations are provided in Algorithm 11.

Algorithm 3.11: Semi-Linearized Proximal Alternating Minimization Algorithm

$$\begin{array}{l} \text{For } k = 0, 1, \dots \\ \left[\begin{array}{l} \mathbf{x}^{[k+1]} = \text{prox}_{\mu_k f} \left(\mathbf{x}^{[k]} - \mu_k \nabla_{\mathbf{x}} g_S(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) \right) \\ \mathbf{e}^{[k+1]} = \text{prox}_{\kappa_k (g_E(\cdot) + g_S(\mathbf{x}^{[k+1]}, \cdot))} \left(\mathbf{e}^{[k]} \right) \end{array} \right. \end{array}$$

- Assumption 1.**
1. The updating steps of $\mathbf{x}^{[k+1]}$ and $\mathbf{e}^{[k+1]}$ have closed form expression;
 2. \mathcal{E} is a Kurdyka-Łojasiewicz function;
 3. $\nabla_{\mathbf{x}}g_S$ is globally Lipschitz continuous with moduli $\pi(\mathbf{e}^{[k]})$, $k \in \mathbb{N}$ and there exists $\pi^-, \pi^+ > 0$ such that $\pi^- \leq \pi(\mathbf{e}^{[k]}) \leq \pi^+$;
 4. $(\kappa_k)_{k \in \mathbb{N}}$ is a positive sequence such that the stepsize κ_k belong to (κ^-, κ^+) for some $\kappa^- \leq \kappa^+$.

Under Assumption 1, based on the proof in [Bolte et al., 2014], in [Foare et al., 2019] the authors show that the sequence $(\mathbf{x}^{[k]}, \mathbf{e}^{[k]})_{k \in \mathbb{N}}$ generated by Algorithm 11 converges to a critical point $(\mathbf{x}^*, \mathbf{e}^*)$ of \mathcal{E} . This approach allows us to choose a larger κ_k which accelerates the convergence of the algorithm to the critical point $(\widehat{\mathbf{x}}, \widehat{\mathbf{e}})$ of (2.35). The leveraged difficulty is that the computation of the proximity operator of the function $g_S(\mathbf{x}, \mathbf{e}) + g_E(\mathbf{e})$ is not simple in all cases.

3.3 Conclusion

In Chapter 2, we see that standard approaches for solving inverse problems is to frame them as variational problems, seeking a solution that minimizes a sum of data fidelity function and regularization functions. Throughout this chapter, we had observed that the objective functions to be minimized may have diverse mathematical characteristics. Consequently, devising algorithms that capitalize on these distinct properties becomes imperative. In Chapter 3, we thus introduced some algorithms capable of minimizing these different functions.

When applying these algorithms, there are three crucial aspects that demand attention: performance, the convergence and the computational speed for large scale data. The subsequent chapter aims to explore deeper into these aspects, focusing on the case of PALM and SL-PAM algorithms within the framework of the Discrete Mumford-Shah functional.

Proximal-based strategies for solving Discrete Mumford-Shah with Ambrosio-Tortorelli penalization on edges

Summary

4.1	Discrete Ambrosio-Tortorelli approximation	41
4.1.1	Motivation	41
4.1.2	Discrete exterior calculus	42
4.1.3	Design of the linear operator \mathbf{D} and \mathbf{D}_1	43
4.2	Minimization algorithm	45
4.2.1	Proximal Alternating Linearized Minimization	45
4.2.2	Semi-Linearized Proximal Alternating Minimization	48
4.2.3	Choice of ε	49
4.3	Choice of the hyperparameters	52
4.4	Comparisons with state-of-the-art methods	53
4.5	Conclusion	56

This chapter focuses on a special instance of (2.35), when g_E models the AT penalization over \mathbf{e} [Ambrosio and Tortorelli, 1990]. We derive two proximal alternating schemes PALM and SL-PAM in this context, leading to algorithmic schemes with convergence guarantees to a critical point of (2.35). A particular attention is paid to the definition of the involved linear operators and the derivation of the associated proximity operators. Numerous experiments are run in order to evaluate the performance of the proposed PALM and SL-PAM for minimizing D-MS with AT penalization over \mathbf{e} . A multiresolution Golden-grid search strategy is proposed to efficiently extract the optimal set of hyperparameters (λ_S, λ_E) and to provide fair comparisons with state-of-the-art methods for different types of degradation.

This chapter is based on the publication [Le et al., 2022a].

4.1 Discrete Ambrosio-Tortorelli approximation

4.1.1 Motivation

As we thoroughly discussed in Chapter 2, when dealing with the joint image restoration and edge detection task, there are three main variational approaches: (i) convex objective function such as ROF model which requires a post-processing step to retrieve edge map, (ii) non-convex function such as Blake-Zisserman (BZ) functional or (iii) Mumford-Shah (MS) functional allowing to reconstruct degraded image and estimate edges. Even though relying on a single minimization process, the BZ functional is a non-convex two-term minimization problem and the D-MS [Foare et al., 2019] is a bi-convex three-term minimization problem leading to more challenging algorithms.

In Chapter 2, we have become familiar with the original continuous Mumford-Shah functional (2.32) for which we wanted to find a pair (\mathbf{x}, Γ) where $\Gamma \subset \overline{\Omega}$ denotes the discontinuities and \mathbf{x} is a smooth reconstructed field. In the continuous setting, the edge length penalization is defined by the term $\mathcal{H}(\Gamma \cap \Omega)$. Such a concept is difficult to model in the discrete setting. In [Foare et al., 2019], the authors proposed different types of penalizations on edges involving either the ℓ_1 norm or BerHu penalization (cf. Section 2.1.3). As shown in [Pustelnik, 2023], there is a link between these BZ functional and DMS under appropriate choices of penalization on edges. For this reason, in this chapter we only focus on DMS formalism. However, in practice, when dealing with discrete setting, the latter penalization on edges leads to some false edges. For instance, in Figure (4.1)(c) we observe a good denoising performance, but missing edges. One possibility to recover more edges is to choose a smaller value λ_E (Figure (4.1) (d)). However, the resulting edges are not more satisfying, with a lot of outliers detected, leading to poor denoising performance.

A challenging question is then how to design a better regularizer g_E so that the DMS model is able to detect relevant edges (e.g Figure (4.1) (e)). To achieve this goal we will focus on Ambrosio-Tortorelli formalism (AT) [Ambrosio and Tortorelli, 1990] since it has an interesting Γ -convergence properties in this continuous setting. In [Foare et al., 2016], the authors propose to reformulate AT using the framework of Discrete Calculus (DC) expressed as

$$\underset{\mathbf{x} \in \mathbb{R}^{CN}, \mathbf{e} \in \mathbb{R}^{2N}}{\text{minimize}} \quad \mathcal{E}_{\mathcal{AT}}(\mathbf{x}, \mathbf{e}) := \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2 + \lambda_S \|(1 - \mathbf{e}) \odot \mathbf{Dx}\|^2 + \lambda_E \left(\frac{1}{4\varepsilon} \|\mathbf{e}\|_2^2 + \varepsilon \|\mathbf{D}_1 \mathbf{e}\|_2^2 \right), \quad (4.1)$$

where $\varepsilon > 0$ denotes the Γ -convergence parameter, \mathbf{D}_1 is a derivative operator that will be thoroughly discussed in section 4.1.3. Theoretically, large values of ε lead to thick contours but help to detect the set of discontinuities. As ε tends to 0, the penalization of $\|\mathbf{e}\|_2^2$ increases and enforces \mathbf{e} to become sparser, and contours become thinner. However the numerical scheme proposed in [Foare et al., 2016, Foare, 2017] got a huge price in computational time. In this chapter, we focus on minimizing the Discrete Mumford-Shah functional with the approximating AT penalization on edges and the design of algorithmic schemes to minimize such a functional. Inspired from [Foare, 2017] we modify the definition of \mathbf{D} and \mathbf{D}_1 originally proposed to make it numerically efficient and to allow us to obtain results as displayed in Figure 4.1 (e).

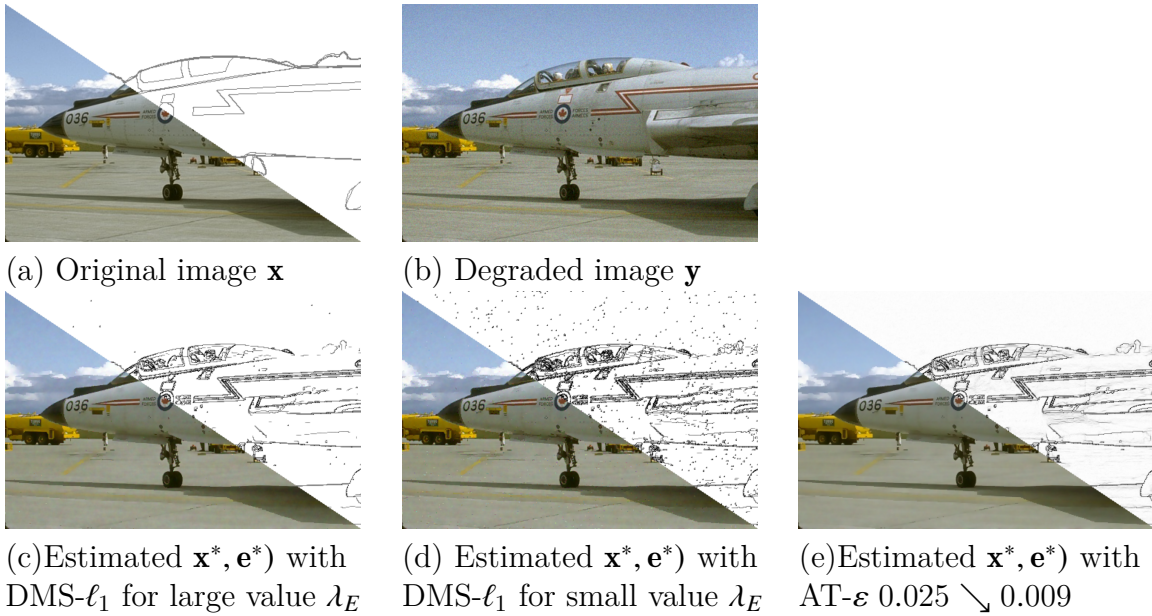


Figure 4.1: (a) Original image (b) Degraded image with $\delta = 0.05$ according to (2.1)-(2.8) with $\mathbf{A} = \text{Id}$ (c) Estimated images and edges obtained with DMS- ℓ_1 (2.35) solved by SL-PAM algorithm for $\lambda_S = 5.7$ and $\lambda_E = 0.006$, (d) Estimated images and edges obtained with DMS- ℓ_1 (2.35) solved by SL-PAM algorithm for $\lambda_S = 5.7$ and $\lambda_E = 0.01$, $\overline{\lambda_E} = 0.01$ (e) AT (4.1) solved by SL-PAM with decreasing at each stage of $\varepsilon = 0.2 \searrow 0.02$ and $(\lambda_S, \lambda_E) = (5.7, 0.008)$.

4.1.2 Discrete exterior calculus

When going from continuous formulation of (2.33) to numerical schemes, we need to reformulate the objective function in a discrete setting. Following [Foare et al., 2016, Foare, 2017] we will focus on a discretization relying on discrete exterior calculus (DEC). We will recall some fundamental definitions and important tools in discrete differential geometry that are essential in this chapter.

Definition 4.1.1. *A discrete surface or 2-dimensional cubical complex is composed of:*

- a set of vertices $V = \{v_i\}$ (0-cells),
- a set of edges $S = \{s_i\}$ such that each edge connect two distinct vertices (1-cells),
- a set of faces $F = \{f_i\}$ such that face f denotes $f = (v_1 \dots v_n)$, where $(v_i v_{i+1}) \in E \forall i = 1, \dots, n$ and $v_{n+1} = v_1$ (2-cells).

In [Foare et al., 2016], the authors proposed two options to discretize the image domain on a 2-dimensional cell complex or discrete surface: (i) we set \mathbf{x} to live on the faces and \mathbf{e} to live on the vertices and edges or (ii) we set \mathbf{x} on the vertices and \mathbf{e} on the edges then the intensity value of \mathbf{x} is the point on the center of pixels.

In this work, we focus on the option where pixel value \mathbf{x} lies on vertices and the edges \mathbf{e} are defined to locate in between them (as illustrated in Figure 2.5-(c)).

Discrete differential form of degree k – In differential geometry, a differential k -forms is a unified approach to define integrands over curves, surfaces or higher-dimensional manifolds. A discrete differential k -form is a quantity that associates a scalar value to a k -cell. In our case when the intensity value of pixels lives on vertices and edges live between pixels, image \mathbf{x} is a 0-form and the edges \mathbf{e} is a 1-form.

Derivative of k -forms – The exterior derivative is an operation on differential forms that, given a k -form ω , produces a $(k+1)$ -form $d\omega$. For instance, the derivative of \mathbf{x} produces a 1-form attached to the edges. This operation extends the differential of a function, and it is summarized in the following theorem.

Theorem 4.1.1. *(Stokes' theorem). The integral of a differential form ω over the boundary $\partial\Omega$ of some orientable manifold Ω is equal to the integral of its exterior derivative $d\omega$ over the whole Ω , i.e.,*

$$\int_{\partial\Omega} \omega = \int_{\Omega} d\omega. \quad (4.2)$$

4.1.3 Design of the linear operator \mathbf{D} and \mathbf{D}_1

In this section, we focus on the design of the discrete operators \mathbf{D} and \mathbf{D}_1 .

Differential operator \mathbf{D} of vertices – Let \mathbf{x} be a 0-form (that is, scalar living on vertices) in \mathbb{R} defined on an oriented grid (e.g. Figure 4.2). Based on the Stoke's theorem, on the oriented edge $s_1 = (v_1 v_4)$, there exists a 1-form $d\mathbf{x}$ that verifies

$$d\mathbf{x}(s_1) = \int_{s_1} d\mathbf{x} = \int_{\partial s_1} \mathbf{x} = \mathbf{x}(v_4) - \mathbf{x}(v_1). \quad (4.3)$$

We define then

$$d : V \rightarrow S \quad (4.4)$$

$$\mathbf{x} \mapsto d\mathbf{x} \quad (4.5)$$

which is equivalent to the finite difference operator and the differential value between two vertices associated with an edge.

Under matrix form, we can rewrite d as:

$$d\mathbf{x} \sim \mathbf{D}\mathbf{x}, \quad (4.6)$$

where \mathbf{D} is the matrix of operator d and \mathbf{x} is the associated vector to the 0-form \mathbf{x} . We can refer to the illustration in the Figure 4.2 where $\mathbf{D} = [\mathbf{D}_h^\top \mathbf{D}_v^\top]^\top$ is a vertex-to-edge oriented incident matrix of the discrete surface where \mathbf{D}_h and \mathbf{D}_v are defined in Figure (4.2)-(b).

Differential operator \mathbf{D}_1 of edges – Let \mathbf{e} be a 1-form in \mathbb{R} defined on a discrete surface. We rewrite the Stoke's theorem on the face $f_1 = (v_1, v_4, v_5, v_2)$ of \mathcal{G} (Figure 4.2), there exists a 2-form noted $d_1\mathbf{e}$ that verifies:

$$d_1\mathbf{e}(f_1) = \int_{f_1} d_1\mathbf{e} = \int_{\partial f_1} \mathbf{e} = -\mathbf{e}(s_1) + \mathbf{e}(s_2) + \mathbf{e}(s_7) - \mathbf{e}(s_9). \quad (4.7)$$

Under matrix form, we can rewrite:

$$d_1\mathbf{e} = \mathbf{D}_1\mathbf{e},$$

where for any faces f surrounding by vectorized edges s , $\mathbf{D}_1 \in \mathbb{R}^{|\mathcal{F}| \times 2N}$ is defined as the edge-to-face oriented incidence matrix. By computing a kind of *curl* on the faces, it acts as a differential operator on edges. We can observe that it results in a combination of the vertical derivatives of horizontal edges. By observation, we can simply rewrite \mathbf{D}_1 as

$$\mathbf{D}_1 = [\mathbf{D}_v \quad -\mathbf{D}_h]. \quad (4.8)$$

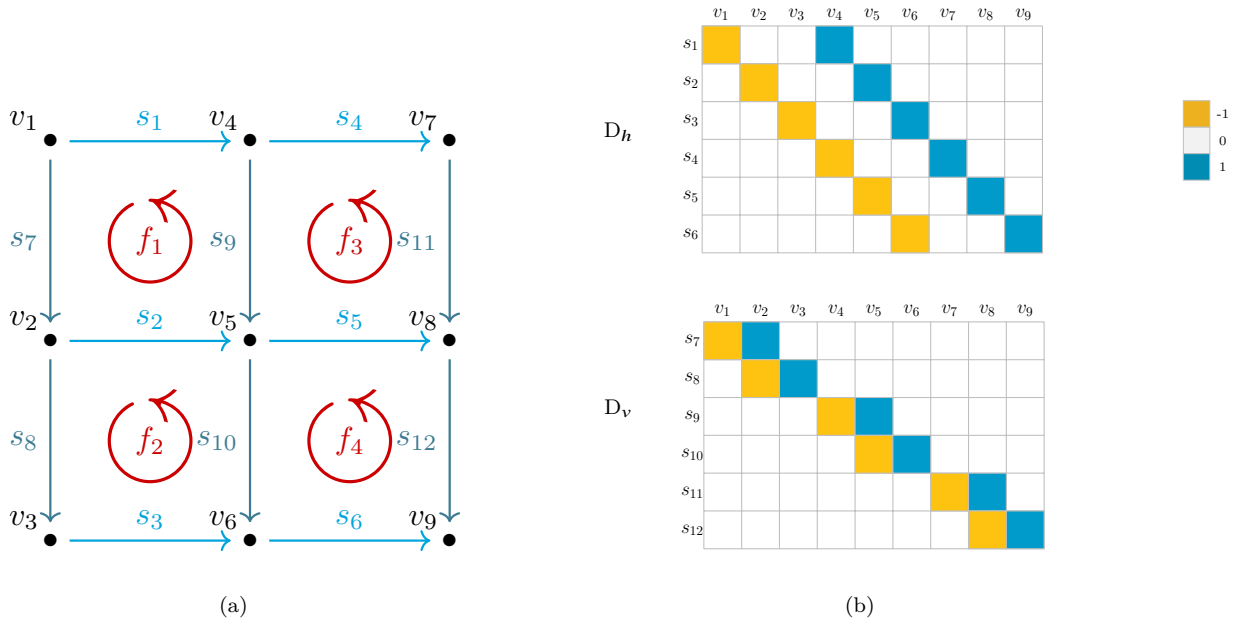


Figure 4.2: Left: example of a discrete image complex for illustrating $\mathbf{D} = [\mathbf{D}_h^\top \mathbf{D}_v^\top]^\top$. Right: corresponding horizontal and vertical pixelwise discrete gradient operators \mathbf{D}_h and \mathbf{D}_v .

4.2 Minimization algorithm

The minimization of (4.1) appears to be a bi-convex minimization problem. In [Foare et al., 2016], the authors propose to alternate the resolution of two linear systems, derived from the optimality conditions. Mimicking the Γ -convergence process, they decrease ε during the optimization process. It allows them to better capture thin structures. This numerical scheme converges to a stationary point, but at the price of a huge computational time.

We propose to derive two proximal alternating schemes (following [Attouch et al., 2010, Bolte et al., 2014, Foare et al., 2019]) relying on proximal steps.

The first scheme, referred as PALM-AT, is presented in Algorithm 4.1, while the second one, called SL-PAM-AT, is presented in Algorithm 4.2. The benefit of considering a full proximal step in the update of $\mathbf{e}^{[k+1]}$ in Algorithm 4.2 is to relax the bound associated with the step-size parameter, computed from the Lipschitz constant of the gradient of the linearized coupling term. The sequence $\{(\mathbf{x}^{[k]}, \mathbf{e}^{[k]})\}_{k \in \mathbb{N}}$ generated in Algorithm 4.1 (resp. Algorithm 4.2) to a critical point of (4.1) following similar arguments than those in [Bolte et al., 2014, Ass. A-B, Theorem 3.1] (resp. [Foare et al., 2019]). However the involved proximal step-sizes require specific attention in our context.

4.2.1 Proximal Alternating Linearized Minimization

Using the generic PALM iterations defined in Algorithm 10, the PALM-AT iterations reads:

Algorithm 4.1: PALM-AT

Input: Degraded image \mathbf{y}

Output: Restored image \mathbf{x}^* , contour \mathbf{e}^*

Set: $g_S = \lambda_S \|(1 - \mathbf{e}) \odot \mathbf{D}\mathbf{x}\|^2$, $g_E = \lambda_E \left(\frac{1}{4\varepsilon} \|\mathbf{e}\|_2^2 + \varepsilon \|\mathbf{D}_1 \mathbf{e}\|_2^2\right)$ $L_{\nabla_{\mathbf{x}} g_S}^{[k]}$ and $L_{\nabla_{\mathbf{e}} g_S}^{[k]}$ the

Lipschitz constants of $\nabla_{\mathbf{x}} g_S(\cdot, \mathbf{e}^{[k]})$ and $\nabla_{\mathbf{e}} g_S(\mathbf{x}^{[k+1]}, \cdot)$

Initialization: $\mathbf{x}^{[0]} \in \mathbb{R}^{CN}$, $\mathbf{e}^{[0]} \in \mathbb{R}^{JN}$

while $\frac{\mathfrak{G}^{[k+1]} - \mathfrak{G}^{[k]}}{\mathfrak{G}^{[k]}} < \zeta$ and $k \in \mathbb{N}$ **do**

Choose $\mu_k < 1/(L_{\nabla_{\mathbf{x}} g_S}^{[k]})$

$\mathbf{x}^{[k+1]} = \text{prox}_{\frac{\mu_k}{2} \|\mathbf{A} \cdot - \mathbf{y}\|_2^2} \left(\mathbf{x}^{[k]} - \mu_k \nabla_{\mathbf{x}} g_S(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) \right)$

Choose $\kappa_k < 1/(L_{\nabla_{\mathbf{e}} g_S}^{[k]})$

$\mathbf{e}^{[k+1]} = \text{prox}_{\kappa_k g_E} \left(\mathbf{e}^{[k]} - \kappa_k \nabla_{\mathbf{e}} g_S(\mathbf{x}^{[k+1]}, \mathbf{e}^{[k]}) \right)$

In Algorithm 4.1, there are two essential steps:

- The computation of $\text{prox}_{\frac{\mu_k}{2}\|\mathbf{A}\cdot-\mathbf{y}\|_2^2}$ which can be efficiently computed when matrix \mathbf{A} is block-circulant with circulant blocks, thus diagonalized by the 2-D DFT.

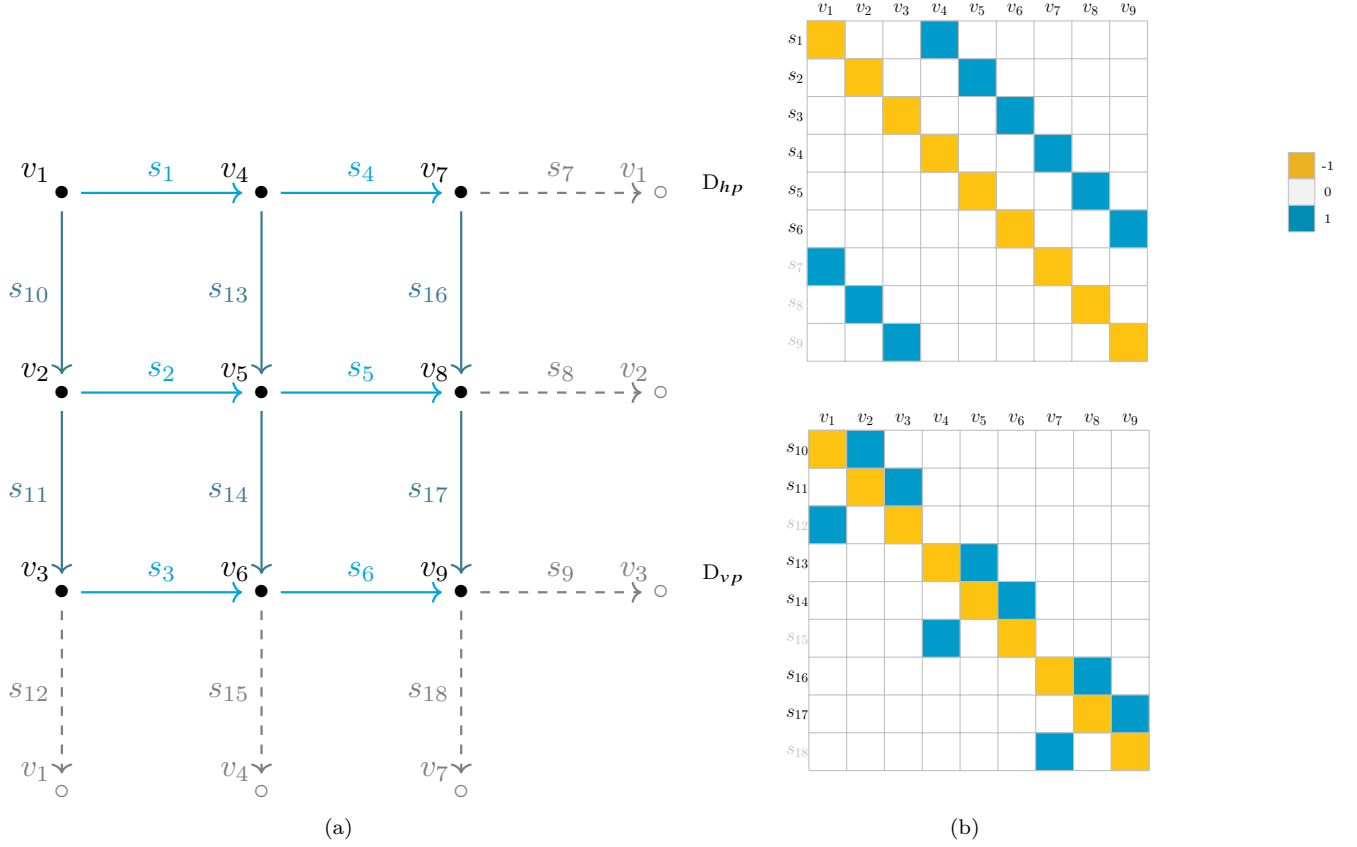


Figure 4.3: Left: example of a discrete image complex for illustrating $\mathbf{D} = [\mathbf{D}_{hp}^\top \mathbf{D}_{vp}^\top]^\top$. Right: corresponding horizontal and vertical pixelwise discrete gradient operators \mathbf{D}_h and \mathbf{D}_v .

- The computation of $\text{prox}_{\kappa_k g_E}$ takes a closed form under Proposition 4.2.1.

Proposition 4.2.1. *The update of the edge variable in Algorithm 4.1 takes a closed form expression that is:*

$$\mathbf{e}^{[k+1]} = \left[2\kappa_k \lambda_E \varepsilon \mathbf{D}_1^* \mathbf{D}_1 + \left(1 + \frac{\kappa_k \lambda_E}{2\varepsilon} \right) \text{Id} \right]^{-1} \mathbf{e}^{[k]}. \quad (4.9)$$

Proof. Using the definition of proximity operator (3.6) for g_E , we obtain:

$$\text{prox}_{\kappa_k g_E}(\mathbf{e}^{[k]}) = \underset{\mathbf{e} \in \mathbb{R}^{2N}}{\text{argmin}} \lambda_E \varepsilon \kappa_k \|\mathbf{D}_1 \mathbf{e}\|_2^2 + \frac{\lambda_E \kappa_k}{4\varepsilon} \|\mathbf{e}\|_2^2 + \frac{1}{2} \|\mathbf{e} - \mathbf{e}^{[k]}\|_2^2. \quad (4.10)$$

Applying the Fermat's rule, we have:

$$\mathbf{e} - \mathbf{e}^{[k]} + 2\kappa_k \lambda_E \varepsilon \mathbf{D}_1^* \mathbf{D}_1 + \frac{2\lambda_E \kappa_k}{4\varepsilon} \mathbf{e} = 0. \quad (4.11)$$

Leading to (4.9),

$$\text{prox}_{\kappa_k \lambda_E}(\mathbf{e}^{[k]}) = \left[2\kappa_k \lambda_E \varepsilon \mathbf{D}_1^* \mathbf{D}_1 + \left(1 + \frac{2\kappa_k \lambda_E}{4\varepsilon} \right) \text{Id} \right]^{-1} \mathbf{e}^{[k]} \quad (4.12)$$

In terms of implementation, the major limitation relies on the inversions involved in Proposition 4.2.1. When \mathbf{D}_1 is defined by $\mathbf{D}_1 = [\mathbf{D}_v \quad -\mathbf{D}_h]$, the update of $\mathbf{e}^{[k+1]}$ can be efficiently obtained by using inversion lemma leading to

$$\mathbf{e}^{[k+1]} = \begin{bmatrix} \mathbf{F} + \mathbf{G}\mathbf{Q}\mathbf{M} & \mathbf{G}\mathbf{Q} \\ \mathbf{Q}\mathbf{M} & \mathbf{Q} \end{bmatrix} \mathbf{e}^{[k]} \quad (4.13)$$

where

$$\left\{ \begin{array}{l} \eta_1 = \left(1 + \frac{\kappa_k \lambda_E}{2\varepsilon} \right) \\ \eta_2 = 2\lambda_E \varepsilon \kappa_k \\ \mathbf{F} = (\eta_2 \mathbf{D}_v^* \mathbf{D}_v + \eta_1 \text{Id})^{-1} \\ \mathbf{G} = \mathbf{F}(\eta_2 \mathbf{D}_v^* \mathbf{D}_h) \\ \mathbf{M} = (\eta_2 \mathbf{D}_h^* \mathbf{D}_v) \mathbf{F} \\ \mathbf{Q} = (\eta_2 \mathbf{D}_h^* \mathbf{D}_h + \eta_1 \text{Id} - (\eta_2 \mathbf{D}_h^* \mathbf{D}_v) \mathbf{F} (\eta_2 \mathbf{D}_v^* \mathbf{D}_h))^{-1} \end{array} \right. ,$$

where inversion is a cheap operation in the Fourier domain requiring to build the discrete gradient operators with boundary effect (as illustrated in Figure 4.3). An auxiliary proof for equation (4.13) is provided in Section A.1.

Convergence of PALM under KL condition – Under Proposition 3.2.3, the PALM algorithm described in Algorithm 4.1 fits the requirements for convergence in [Bolte et al., 2014][Assumptions A-B, Thm. 3.1].

4.2.2 Semi-Linearized Proximal Alternating Minimization

Using the SL-PAM iterations defined in Algo. 11, the SL-PAM algorithm for the DMS with Ambrosio-Tortorelli penalization on edges reads:

Algorithm 4.2: SL-PAM-AT

Input: Degraded image \mathbf{y}

Output: Restored image \mathbf{x}^* , contour \mathbf{e}^*

Set: $g_S = \lambda_S \|(1 - \mathbf{e}) \odot \mathbf{D}\mathbf{x}\|^2$, $g_E = \lambda_E \left(\frac{1}{4\varepsilon} \|\mathbf{e}\|_2^2 + \varepsilon \|\mathbf{D}_1 \mathbf{e}\|_2^2 \right)$ and $L_{\nabla_{\mathbf{x}} g_S}^{[k]}$ is the

Lipschitz constant of $\nabla_{\mathbf{x}} g_S(\cdot, \mathbf{e}^{[k]})$

Initialization: $\mathbf{x}^{[0]} \in \mathbb{R}^{CN}$, $\mathbf{e}^{[0]} \in \mathbb{R}^{JN}$

while $\frac{\mathfrak{g}^{[k+1]} - \mathfrak{g}^{[k]}}{\mathfrak{g}^{[k]}} < \zeta$ and $k \in \mathbb{N}$ **do**

Choose $\mu_k > L_{\nabla_{\mathbf{x}} g_S}^{[k]}$
 $\mathbf{x}^{[k+1]} = \text{prox}_{\frac{\mu_k}{2} \|\mathbf{A}\cdot - \mathbf{y}\|_2^2} \left(\mathbf{x}^{[k]} - \mu_k \nabla_{\mathbf{x}} g_S(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) \right)$
 Choose $\kappa_k > 0$
 $\mathbf{e}^{[k+1]} = \text{prox}_{\kappa_k (g_S(\mathbf{x}^{[k+1]}, \cdot) + g_E)} \left(\mathbf{e}^{[k]} \right)$

In Algorithm 4.2, there are two essential updates:

- The computation of $\text{prox}_{\frac{\mu_k}{2} \|\mathbf{A}\cdot - \mathbf{y}\|_2^2}$ which is similar to Algorithm 4.1.
- The computation of $\text{prox}_{\kappa_k (g_S(\cdot, \mathbf{x}^{[k+1]}) + g_E)} \left(\mathbf{e}^{[k]} \right)$ that will be detailed in Proposition 4.2.2.

Proposition 4.2.2. *The update of the edge variable in Algorithm 4.2 takes a closed form that is:*

$$\mathbf{e}^{[k+1]} = \left[2\kappa_k \lambda_E \varepsilon \mathbf{D}_1^* \mathbf{D}_1 + 2\kappa_k \lambda_S \text{diag} \left((\mathbf{D}\mathbf{x}^{[k+1]})^2 \right) + \dots \right. \\ \left. \left(\frac{\lambda_E \kappa_k}{2\varepsilon} + 1 \right) \text{Id} \right]^{-1} \left(2\kappa_k \lambda_S (\mathbf{D}\mathbf{x}^{[k+1]})^2 + \mathbf{e}^{[k]} \right). \quad (4.14)$$

Proof. Using the definition of proximity of operator (3.6), the update of the edge variable \mathbf{e} in Algorithm 4.2 reads:

$$\text{prox}_{\kappa_k (g_S(\cdot, \mathbf{x}^{[k+1]}) + g_E)} \left(\mathbf{e}^{[k]} \right) = \underset{\mathbf{e} \in \mathbb{R}^{2N}}{\text{argmin}} \lambda_E \varepsilon \kappa_k \|\mathbf{D}_1 \mathbf{e}\|_2^2 + \frac{\lambda_E \kappa_k}{4\varepsilon} \|\mathbf{e}\|_2^2 + \dots \\ \lambda_S \kappa_k \|\mathbf{D}\mathbf{x}^{[k+1]} \odot (1 - \mathbf{e})\|_2^2 + \frac{1}{2} \|\mathbf{e} - \mathbf{e}^{[k]}\|_2^2. \quad (4.15)$$

Applying the Fermat rule, we obtain:

$$2\lambda_E \varepsilon \kappa_k \mathbf{D}_1^* \mathbf{D}_1 \mathbf{e} + \frac{\lambda_E \kappa_k}{2\varepsilon} \mathbf{e} + 2\lambda_S \kappa_k (\mathbf{D}\mathbf{x}^{[k+1]})^2 \odot (\mathbf{e} - 1) + \mathbf{e} - \mathbf{e}^{[k]} = 0. \quad (4.16)$$

Or equivalently

$$2\lambda_E \varepsilon \kappa_k \mathbf{D}_1^* \mathbf{D}_1 \mathbf{e} + \frac{\lambda_E \kappa_k}{2\varepsilon} \mathbf{e} + 2\lambda_S \kappa_k \left((\mathbf{D}\mathbf{x}^{[k+1]})^2 \right) \odot (\mathbf{e} - 1) + \mathbf{e} - \mathbf{e}^{[k]} = 0. \quad (4.17)$$

Leading to

$$\text{prox}_{\kappa_k (g_S(\cdot, \mathbf{x}^{[k+1]}) + g_E)}(\mathbf{e}^{[k]}) = \left[2\kappa_k \lambda_E \varepsilon \mathbf{D}_1^* \mathbf{D}_1 + 2\kappa_k \lambda_S \left((\mathbf{D}\mathbf{x}^{[k+1]})^2 \right) + \dots \right. \\ \left. \left(\frac{\lambda_E \kappa_k}{2\varepsilon} + 1 \right) \text{Id} \right]^{-1} \left(2\kappa_k \lambda_S (\mathbf{D}\mathbf{x}^{[k+1]})^2 + \mathbf{e}^{[k]} \right). \quad (4.18)$$

Contrary to the inversion involved in Algorithm 4.1, the inversion problem (4.18) is more challenging. To accelerate the inversion of $2\lambda_E \varepsilon \kappa_k \mathbf{D}_1^* \mathbf{D}_1 + 2\lambda_S \kappa_k \left[(\mathbf{D}\mathbf{x}^{[k+1]})^2 \right] + \left(\frac{\lambda_E \kappa_k}{2\varepsilon} + 1 \right) \text{Id}$, we can consider it as a sparse matrix and use sparse linear system solver such as GMRES [Baker et al., 2005].

Convergence of PALM under KL condition – Under Assumption 1, based on the proof in [Bolte et al., 2014], SL-PAM-AT also allows us to choose a larger κ_k which accelerates the convergence of the algorithm to the critical point $(\mathbf{x}^*, \mathbf{e}^*)$ of (4.1).

4.2.3 Choice of ε

AT- ε \searrow – The length penalization $g_E(\mathbf{e})$ is controlled by the parameter λ_E and also crucially depends on another parameter ε . It determines the scale at which the penalization operates; when ε has big value, the AT term will have less influence on the length penalization due to $\frac{1}{4\varepsilon} \|\mathbf{e}\|_2^2$ which help us to detect as much as possible the set of potential significant edges (discontinuities). When ε tends to 0, the penalization of $\|\mathbf{e}\|_2^2$ increases and enforces \mathbf{e} to become sparser and contours become thinner.

The idea of the proposed algorithm AT with decreasing ε relies always on solving the (4.1) with PALM or SL-PAM at fixed ε but the initialisation of image reconstructed and edges detected are chosen as the solution of the previous ε as described by Algorithm 4.3. We also observe in Figure 4.4, at each decreasing stage, the energy (4.1) has a jump. This can be explained from the fact that after updating a new smaller ε , the term $\frac{1}{4\varepsilon} \|\mathbf{e}\|_2^2$ will increase suddenly while the others are kept unchanged leading to the jump in the energy.

Algorithm 4.3: AT- ε \searrow

Input: Degraded image \mathbf{y}

Output: Restored image \mathbf{x}^* , contour \mathbf{e}^*

Set: $0 < \tilde{\varepsilon} < \varepsilon_0$ and decreasing factor $\rho > 1$

Initialization: $\mathbf{x}^{[0]} \in \mathbb{R}^{CN}$, $\mathbf{e}^{[0]} \in \mathbb{R}^{JN}$

while $\varepsilon > \tilde{\varepsilon}$ **do**

Solve (4.1) with ε , using Algorithm 4.1 or Algorithm 4.2
 $\varepsilon := \varepsilon / \rho$

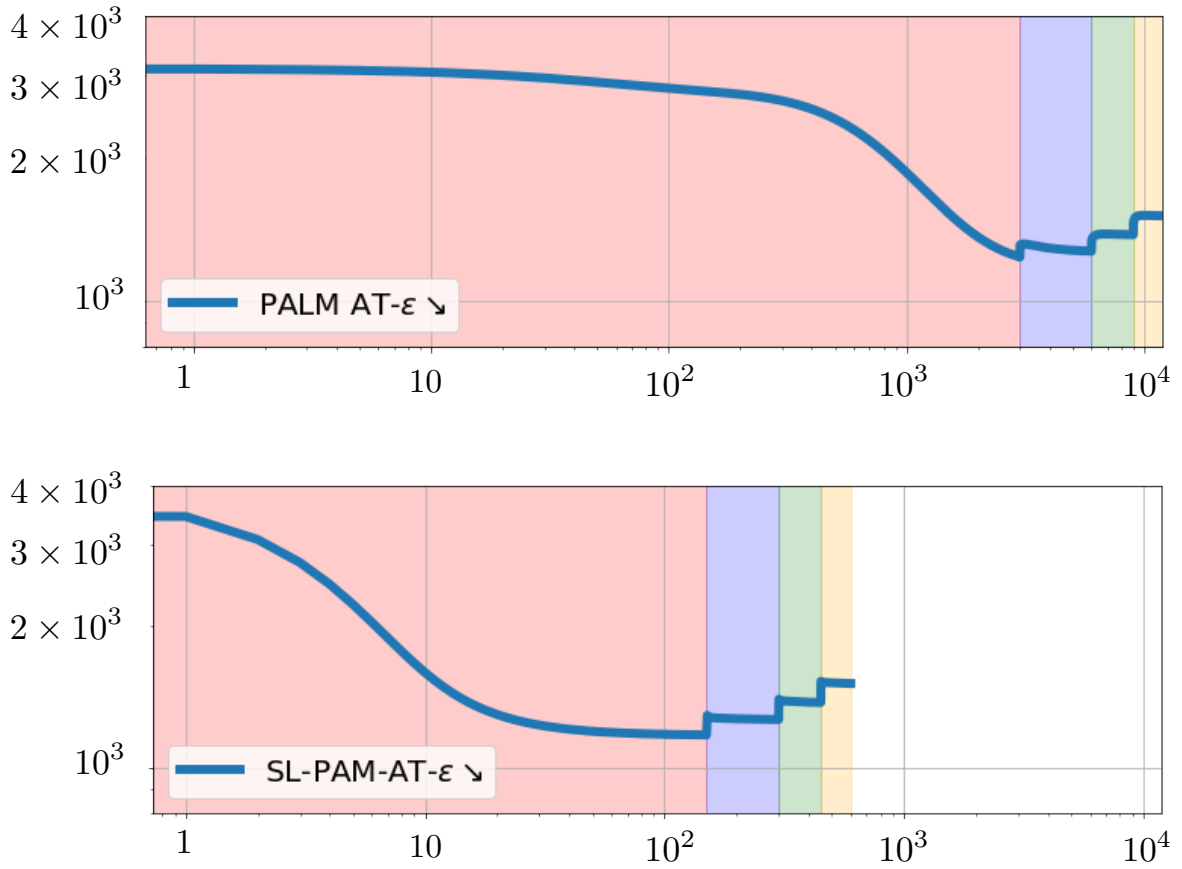


Figure 4.4: AT- ε energy associated to the estimate displayed in Figure 4.1 when $\varepsilon = 0.025 \searrow 0.009$ and $(\lambda_S, \lambda_E) = (5.2, 0.009)$ solved by PALM (top) and SL-PAM (bottom) for each decreasing ε stage with $\varrho = 1.5$.

In Figure 4.5, we display the estimation result at each stage of decreasing ε , we observe that the PSNR score decreases but the edges got clearer. In addition, from Figure 4.5, we can see that as ε decreases, the values of \mathbf{e} converge closer to either 0 or 1. These values correspond to the essential edges and result in a better quality in the sharpness of the edges in some areas with a gradual transition in intensity. At this point, the selection of $\lambda_S, \lambda_E, \varepsilon$ and the decreasing factor ϱ remains a question to enable a better reconstruction and edge detection.

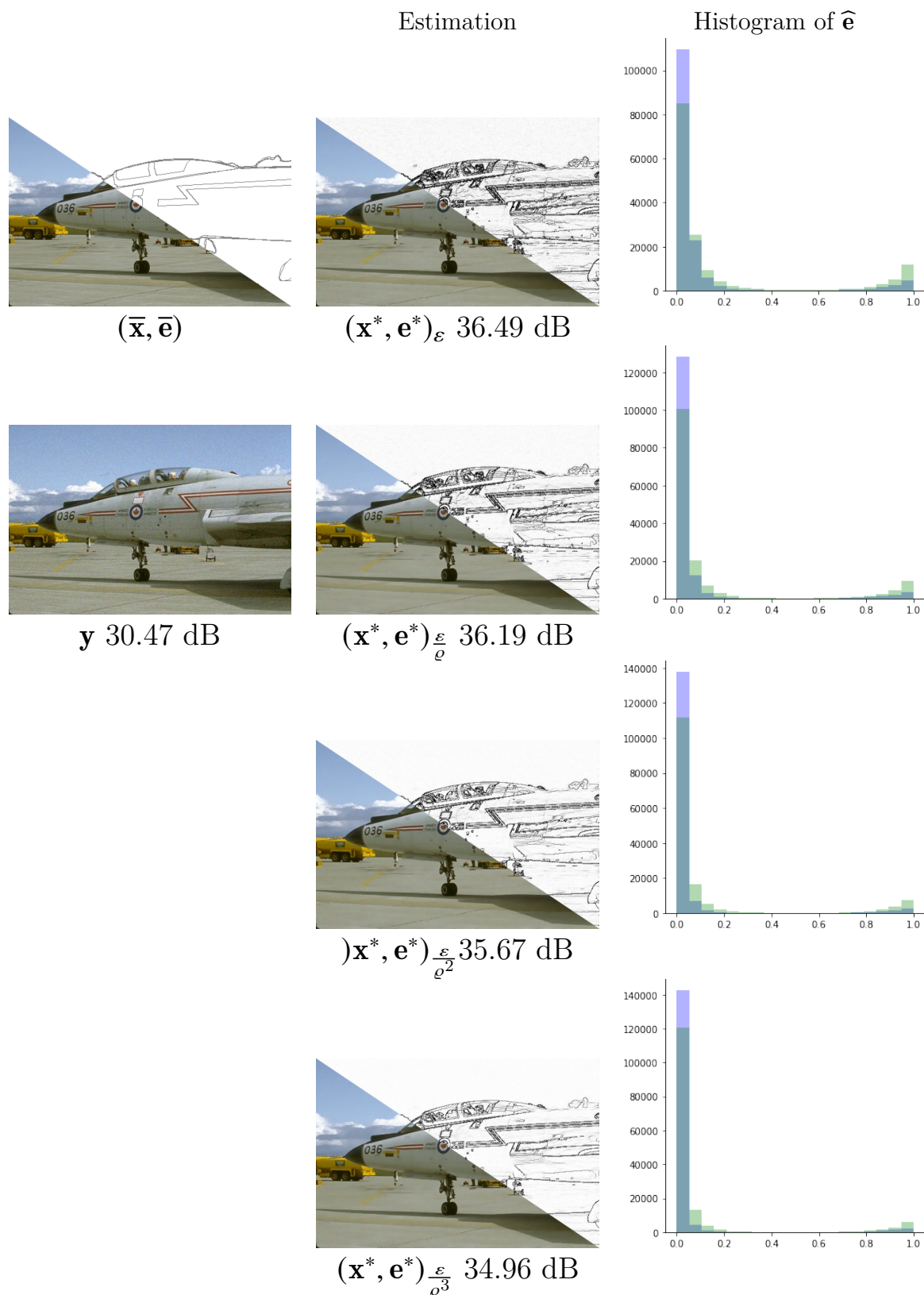


Figure 4.5: 1st column: (Top) original image and groundtruth edge, (Bottom) degraded image with $\delta = 0.03$. 2nd column: Denoised image and edges estimated by SL-PAM-AT- $\varepsilon = 0.2 \searrow 0.02$ with deacresing factor $\varrho = 1.5$ and $(\lambda_S = 5.2, \lambda_E = 0.006)$. 3rd column: Histogram of \mathbf{e}_h (purple) and \mathbf{e}_v (green) or both (gray blue) for each stage.

4.3 Choice of the hyperparameters

In the AT functional (4.1), there are two crucial hyperparameters λ_S and λ_E . The first one controls the smoothness of the image while the second one controls the total edge length or total length of discontinuities. If the λ_S is too big we will over smooth out the image, whereas too small values of λ_S keep some noise in the reconstructed image. Similarly, λ_E controls the penalization on edges where big values of λ_E penalize strongly the edge length and vice versa. The selection of these hyperparameters for DMS or AT functional became crucial for whatever the choice g_E . In the following, we propose a strategy for a better hyperparameter selection procedure, namely, Multiresolution Golden-grid-search.

Algorithm 4.4: Multiresolution Golden-grid search

for $\ell = 0, 1, \dots$ **do**

- 1- Run the Alg.4.1 or Alg.4.2 on a 5×5 -equally spaced grid with $(\lambda_S, \lambda_E) \in [(\lambda_S)_{L,\ell}, (\lambda_S)_{R,\ell}] \times [(\lambda_E)_{L,\ell}, (\lambda_E)_{R,\ell}]$.
 - 2- Identify the pair $((\lambda_S)_\ell^*, (\lambda_E)_\ell^*)$ maximizing the score (e.g. PSNR or Jaccard index).
 - 3- The grid bounds $(\lambda_S)_{L,\ell+1}$, $(\lambda_S)_{R,\ell+1}$, $(\lambda_E)_{L,\ell+1}$, and $(\lambda_E)_{R,\ell+1}$ are updated in order to be centered in $((\lambda_S)_\ell^*, (\lambda_E)_\ell^*)$ and with a twice smaller width.
-

The performance of PALM-AT using a standard grid search strategy and the proposed multiresolution Golden-grid search are provided in Fig. 4.6. We can observe that the proposed hyperparameter selection procedure allows us to reach better scores. We also observe that maximizing the Jaccard index leads to better contour estimation than maximizing PSNR. We then focus on maximizing Jaccard index using multiresolution Golden-grid strategy in order to provide fair comparisons between the different penalizations and algorithmic strategies.

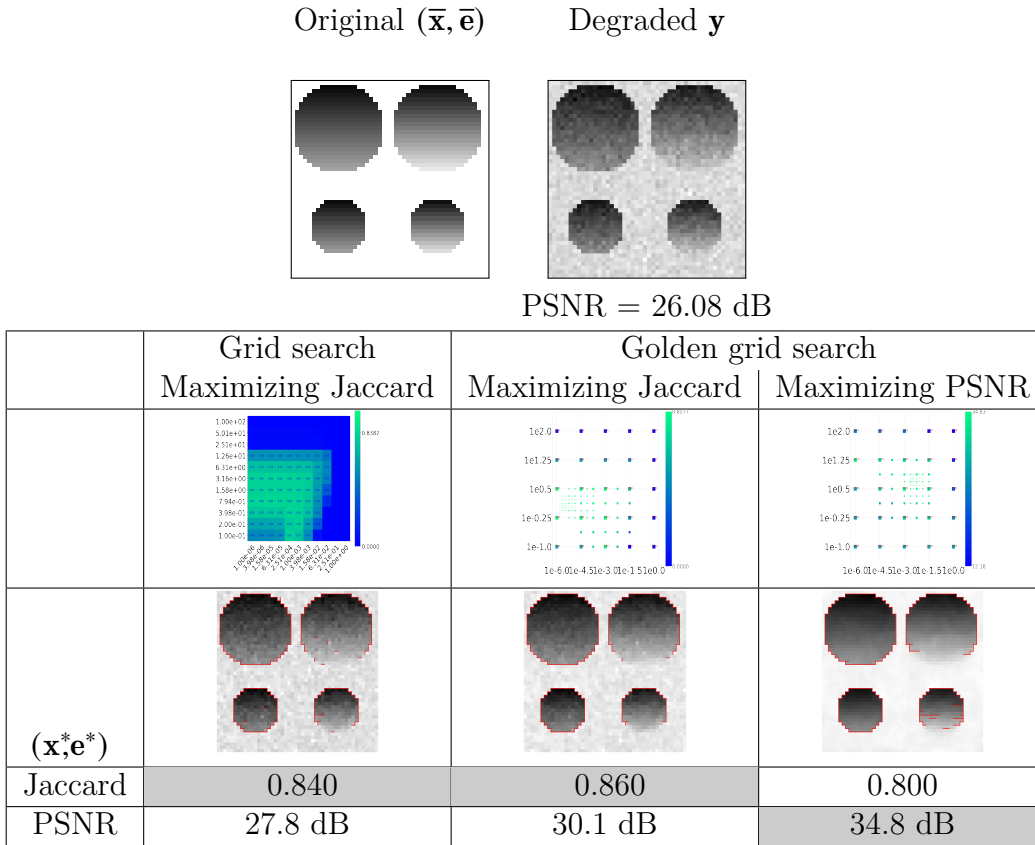


Figure 4.6: Original and degraded data are displayed on the top. Comparison between grid search (1st column) and the proposed Golden-grid search (2nd column) for (λ_S, λ_E) -hyperparameters with PALM-AT- $\varepsilon = 0.2$ when maximizing Jaccard index and when maximizing the PSNR. The grid search (resp. Golden-grid search) map is composed with 121 (resp. 125) values.

4.4 Comparisons with state-of-the-art methods

Considering Algorithm 4.3 with SL-PAM-AT or PALM-AT iterations and multiresolution Golden-Grid search, we are able to perform more systematic experiments.

Simulation settings – Our experiments are first performed on a toy example of size $N = 125 \times 125$ displayed in Fig. 4.6 (top-left), which allows us to have access to the ground truth both in terms of image to restore and contour to extract, and to study carefully the impact of the different algorithmic strategies and penalizations. We consider two degradation models (2.1): (i) a Gaussian noise of variance δ^2 without linear degradation (i.e. $A = \mathbb{I}$) in Fig. 4.7, and (ii) a degradation combining a linear Gaussian blur with standard deviation of 1.1 and Gaussian noise of variance δ^2 , provided in Fig. 4.8, for several realisations of noise.

Second, we evaluate the performance on several images extracted from the BSD database [Meer et al., 1991] degraded with a Gaussian noise and for which contours are provided. As it can be observed on the first column of Figure 4.9, the definition of a contour for complex images is a tedious task and often user-dependent. Among the 5 different contours proposed in the dataset, we considered the one that appears to us

the most realistic. We evaluate the results both in terms of Jaccard index and PSNR, to quantify respectively the contour detection and the image restoration performance.

Algorithm settings – The stopping criterion for all the algorithms is set to $\zeta = 10^{-4}$, $\mu_k = 1/(2.01\lambda_S)$, $\kappa_k = 10^4$ (for SL-PAM) and $\kappa_k = \frac{1.01}{\lambda_S L_{\text{vegS}}^{[k]}}$ (for PALM) where the Lipschitz constant is computed with the power method. ε can be either selected fixed or decreasing $\varepsilon_{\max} \searrow \varepsilon_{\min}$.

Comparison with state-of-the-art methods – In Fig. 4.7, we display the optimal results in terms of Jaccard index using a multiresolution Golden-grid search strategy for PALM and SL-PAM with different choices of g_E . The choice $g_E = \|\cdot\|_1$ refers to the method presented in [Foare et al., 2019] while AT penalization refers to the strategies presented in this chapter.

We observe that the proposed algorithms PALM and SL-PAM with decreasing ε lead to the best performance both in terms of contour detection and restoration. However, SL-PAM-AT is limited by its huge computational cost due to the inversion in (4.14). For this reason, PALM-AT with decreasing ε leads to the best compromise between performance and computational time among the proposed strategies.

	T-ROF [Rudin et al., 1992]	Hohm et al. [Hohm et al., 2015]	PALM- ℓ_1	SL- PAM- ℓ_1	PALM- AT $\varepsilon=0.2$	SL- PAM-AT $\varepsilon=0.2$	PALM- AT $\varepsilon=$ $2 \searrow 0.02$
$(\mathbf{x}^*, \mathbf{e}^*)$							
Jaccard	0.599	0.869	0.860	0.872	0.860	0.860	0.873
PSNR	30.5 dB	30.7 dB	27.5 dB	31.4 dB	30.1 dB	27.5 dB	34.2 dB
CT	~ 10 sec.	~ 10 sec.	~ 10 sec.	~10 sec.	~ 1 min.	~ 1 h.	~ 10 min.

Figure 4.7: Comparison between different schemes using observation \mathbf{y} provided in Fig. 4.6 (top-left). All the results are obtained with a multiresolution golden-grid search strategy (even for T-ROF and Hohm et al. [Hohm et al., 2015]) maximizing the Jaccard index. The second row displays the optimal solution for each methods. The optimum Jaccard index, the associated PSNR and the overall-running time (including the search for the optimal parameters relying on the multiresolution grid-search) are also provided.

Image restoration and edge detection – In the context of image restoration and edge detection, comparisons between T-ROF [Cai and Steidl, 2013], Hohm et al. [Hohm et al., 2015], PALM- ℓ_1 [Foare et al., 2019], and the proposed PALM-AT with decreasing ε are provided in Fig. 4.8 (blur and noisy data for 15 realisations of noise) and Fig. 4.9 (noisy case on more complex data). We focus on PALM strategies to highlight the impact of the penalization, regardless of the algorithm. The performance

of PALM-AT with decreasing ε are systematically better in terms of Jaccard index and most of the time in PSNR.

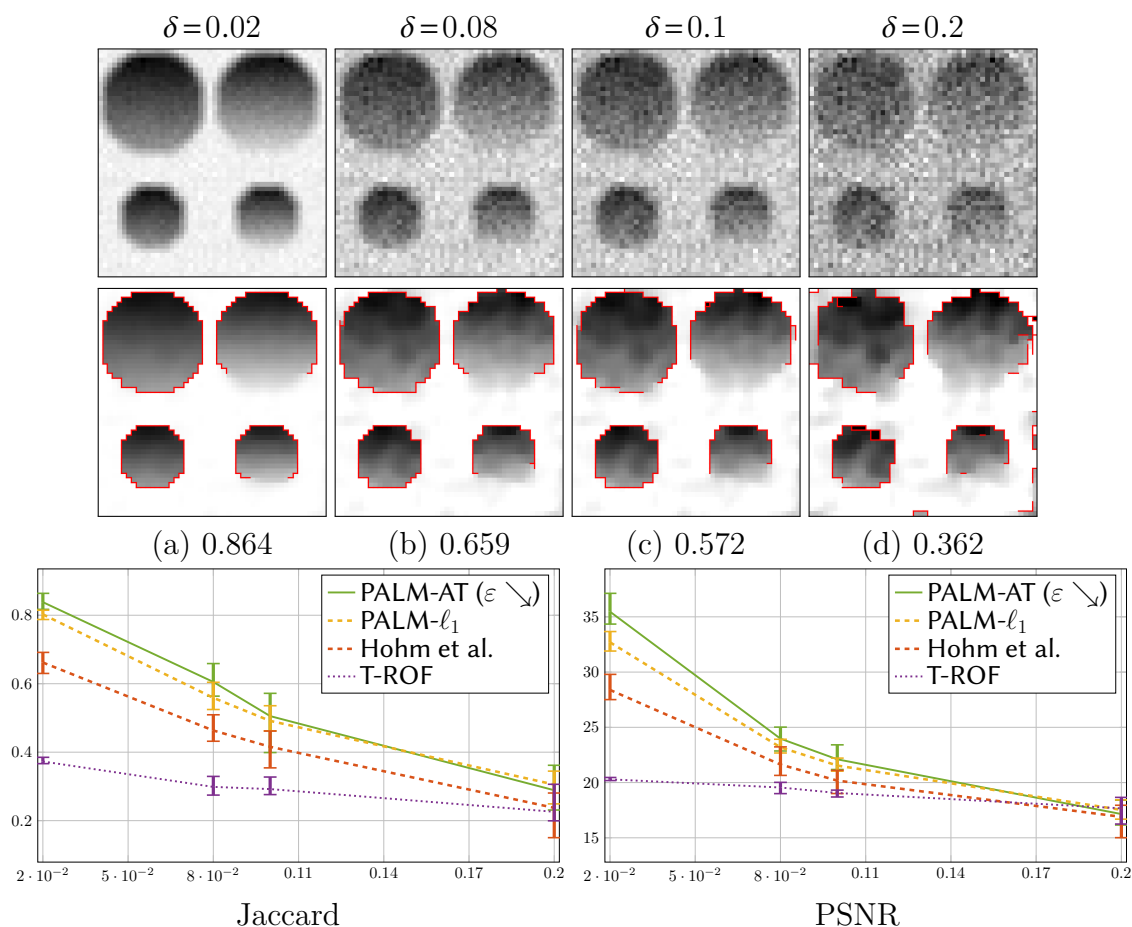


Figure 4.8: (Top) 1st row: observation degraded by both an additive white Gaussian noise and a Gaussian blur. 2nd row: Best estimated image $\hat{\mathbf{x}}$ and contours $\hat{\mathbf{e}}$ (delineated in red) using PALM-AT with $\varepsilon = 2 \searrow 0.02$ maximizing the Jaccard index, provided below. (Bottom) Comparisons between T-ROF, Hohm et al., PALM- ℓ_1 , and PALM-AT with $\varepsilon = 2 \searrow 0.02$ obtained with multiresolution Golden-grid search strategy maximizing the Jaccard index. For each method, we display the mean (min and max) of Jaccard index (middle) and PSNR (right) for 15 realizations of noise.




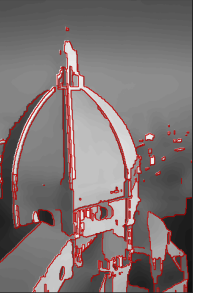

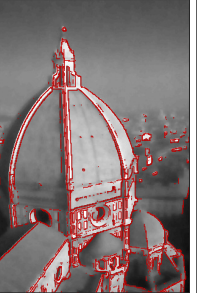

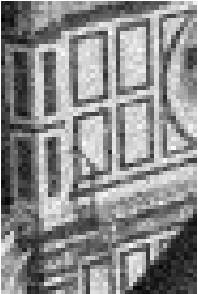
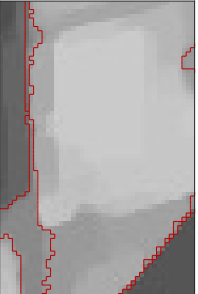


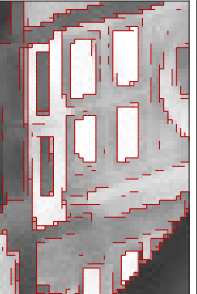

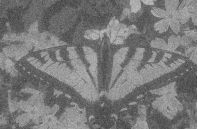
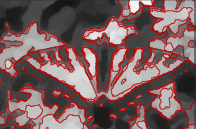

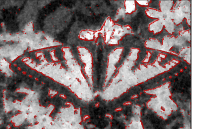
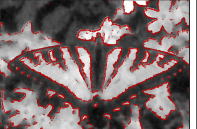


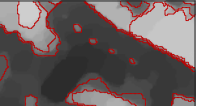

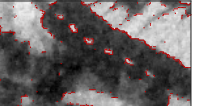
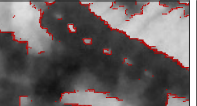


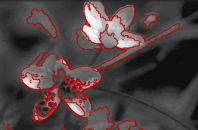
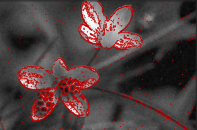
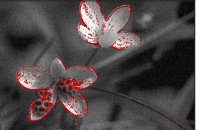
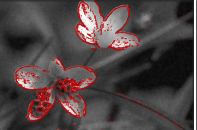


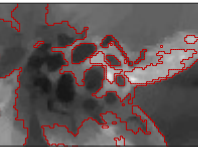
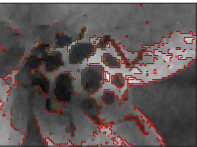
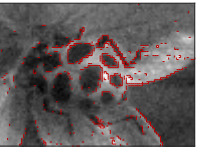
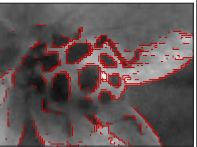
Ground Truth	Data	T-ROF	BZ	DMS- ℓ_1	Proposed
PSNR / Jaccard	26.0 dB / ND	20.5 dB / 0.068	21.9 dB / 0.073	29.7 dB / 0.063	24.0 dB / 0.074
					
					
PSNR / Jaccard	14.0 dB / ND	21.8 dB / 0.104	21.0 dB / 0.101	23.7 dB / 0.094	23.4 dB / 0.121
					
					
PSNR / Jaccard	26.0 dB / ND	34.3 dB / 0.068	33.1 dB / 0.084	30.3 dB / 0.081	34.2 dB / 0.101
					
					

Figure 4.9: Performance obtained with T-ROF, Blake-Zisserman [Blake and Zisserman, 1987], DMS- ℓ_1 [Foare et al., 2019], and the proposed AT- ε \searrow , with associated PSNR/Jaccard.

4.5 Conclusion

In this work, we derive two iterative schemes to solve the AT functional, relying on PALM and SL-PAM. We also introduce a multiresolution Golden-grid search for hyperparameters selection to compare their performances on contour detection and

restoration. From numerical experiments, SL-PAM gives a faster convergence rate than PALM for AT regularization but at the price of larger computational time. Extensive experiments illustrate the benefit of $\text{AT-}\varepsilon = 2 \searrow 0.02$ compared to other state-of-the-art methods such as T-ROF, Blake-Zisserman or DMS- ℓ_1 .

Part II

Deep learning methods for joint image restoration and edge detection

State-of-the-art on deep learning for image restoration or edge detection

Summary

5.1	Preliminaries	60
5.1.1	Objectives	60
5.1.2	Classical Neural Networks	60
5.1.3	Deep Convolutional Neural Networks	61
5.1.4	Learning process	62
5.2	Plug-and-Play and Proximal Neural Networks for image restoration .	63
5.3	Edge detection for degraded image	65

5.1 Preliminaries

In the past twenty years, there has been remarkable progress in both the management of massive data and the computational power available for complex calculations. This has paved the way for innovative approaches known as deep learning, which are increasingly applied to various tasks in the field of image processing including image denoising, image restoration, edge detection. This section is dedicated to introducing essential concepts and tools that will be utilized throughout the remainder of this manuscript.

5.1.1 Objectives

Deep learning methods arise as powerful approach in the era of artificial intelligence to overcome the limitations of variational approaches in term of computation time. Learning-based methods consist of building a prediction function f_{Θ} depending on hyperparameters Θ . For example, in the context of image restoration, the reconstructed image $\hat{\mathbf{x}}$ can be the output of the prediction function $f_{\Theta}(\mathbf{y})$ where \mathbf{y} is the degraded image.

In the context of joint image reconstruction and edges detection, the prediction function acts as $(\hat{\mathbf{x}}, \hat{\mathbf{e}}) = f_{\Theta}(\mathbf{y})$ where $\hat{\mathbf{x}}$ and $\hat{\mathbf{e}}$ are respectively the image reconstructed and edge detected from the degraded image.

We discuss here about supervised learning strategies where the learning parameters Θ are inferred from a training database $(\bar{\mathbf{x}}_s, \mathbf{y}_s)_{s \in \mathbb{I}}$ of size $|\mathbb{I}|$ for image restoration task or in the joint task of image denoising and edge detection where the training set become $(\bar{\mathbf{x}}_s, \mathbf{z}_s, \bar{\mathbf{e}}_s)_{s \in \mathbb{I}}$.

In the following, we start discussing some pioneering structures of f_{Θ} such as classical Neural Networks and Deep Convolutional Networks.

5.1.2 Classical Neural Networks

Neural networks (NNs) form a particular class of prediction functions f_{Θ} composed of several layers or neural nodes. Let f_{Θ}^K be a feedforward NN, with K layers and learnable parameters Θ , f_{Θ}^K can be written as a composition of operators (i.e., layers) $f_{\Theta} = T_{\Theta_K} \circ \dots \circ T_{\Theta_1}$, where, for every $k \in \{1, \dots, K\}$, Θ_k are the learnable parameters of the k -th layer T_{Θ_k} . The k -th layer is defined as

$$T_{\Theta_k} : \mathbf{u}^{[k]} \in \mathbb{R}^{N_k} \mapsto \eta_k(W_k \mathbf{u}^{[k]} + \mathbf{b}_k) \in \mathbb{R}^{N_{k+1}}, \quad (5.1)$$

where η_k is a (non-linear) activation function, W_k is a linear operator, and \mathbf{b}_k is a bias.

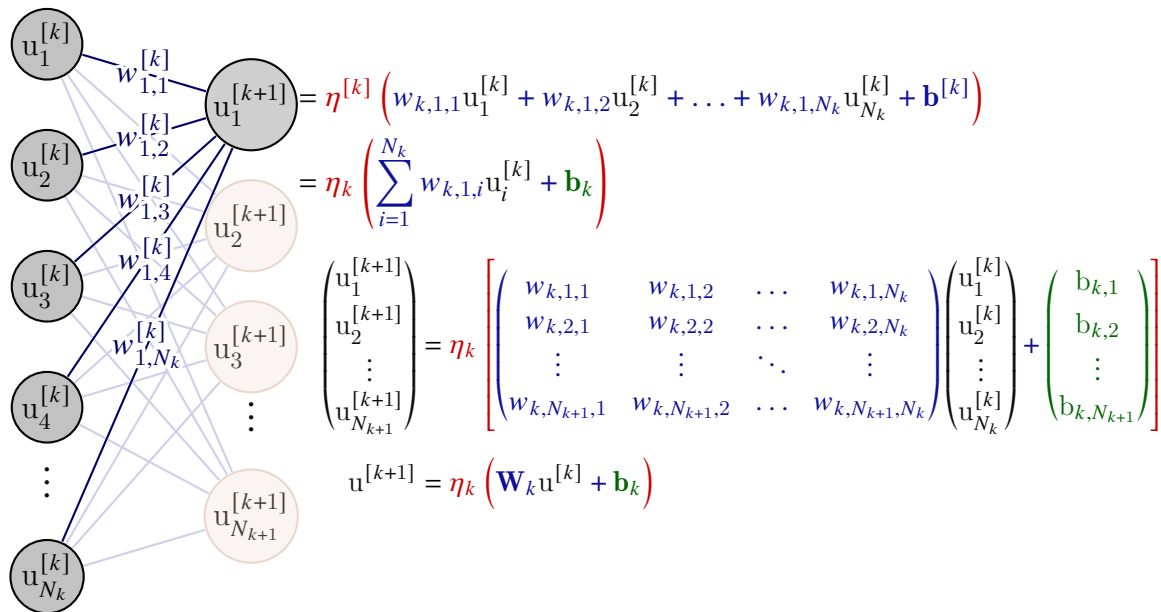


Figure 5.1: The k -th layer of a Feed Forward Neural Network.

Figure 5.1 illustrates the k -th layer of a fully feedforward neural network.

5.1.3 Deep Convolutional Neural Networks

A feedforward neural network (Figure 5.1) is usually a fully connected network, every node in one layer is connected to every nodes in the next layer. It makes them easy to overfit the data. Such a neural network is not flexible and cumbersome when dealing with high dimensional image. This gives rise to Deep convolutional neural networks (DnCNNs) which have proved their success in many image processing problem such as image denoising, edge detection, segmentation, etc. In the following, we introduce step by step how to build a standard DnCNNs.

When dealing with an image processing task, we are already familiarized with different operators (or kernels) such as mean filter in image denoising or Laplacian operator in edge detection. These operators generally rely on a kernel $\mathbf{h} \in \mathbb{R}^{m \times n}$ which is convolved with the image \mathbf{x} to give output \mathbf{u} :

$$\mathbf{u} = \mathbf{h} * \mathbf{x} \Leftrightarrow (\forall i, j) \quad u_{i,j} = \sum_{\tilde{m}=-\lfloor \frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} \sum_{\tilde{n}=-\lfloor \frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \mathbf{h}_{\tilde{m},\tilde{n}} \cdot \mathbf{x}_{i-\tilde{m},j-\tilde{n}} \quad (5.2)$$

Definition 5.1.1. (2D convolutional layer) For $k \in \{1, \dots, K\}$, a convolutional layer with convolutional operator denoted as $(\mathbf{d}_{k,j})_{1 \leq j \leq J}$ can be defined as

$$\mathbf{u}^{[k+1]} = \eta_k \left(\sum_{j=1}^J \mathbf{d}_{k,j} * \mathbf{u}_j^{[k]} + \mathbf{b}_{k,j} \right). \quad (5.3)$$

Definition 5.1.2. (Standard Convolutional Neural Network) A Convolutional Neural Networks refers to several layers of the form (5.3).

In the literature, there is a range of CNNs whose number grows very fast in computer vision. The most famous and standard CNN is DnCNN which proved its efficiency for several tasks like edge detection, image denoising or image classification. In recent years, more and more innovative techniques and design elements such as skip-connections or average pooling have been employed to enable them to extract context-rich features. For example, UNet proposed by [Ronneberger et al., 2015] features a U-shaped architecture with both downsampling and upsampling paths, skip-connections between corresponding layers in these paths allow the network to capture fine details and contextual information. Recently, it was modified to more powerful architecture such as DRUnet [Zhang et al., 2021]. Resnet proposed by [He et al., 2016] introduces the concept of residual connections, which enable the network to learn residual functions, making it easier to train extremely deep networks without vanishing gradient problems.

5.1.4 Learning process

In order to estimate the parameters Θ of the network, the supervised framework relies on a training database $(\bar{\mathbf{x}}_s, \mathbf{z}_s)_{s \in \mathbb{I}}$. The learning relies on a loss function \mathcal{L} which measures the performance of the estimator function $f_{\Theta}(\mathbf{z})$. The goal is to minimize this function with respect to Θ to determine the best estimator of $f_{\hat{\Theta}}$. Formally, we seek to solve the following minimization problem:

$$\hat{\Theta} \in \underset{\Theta}{\text{Argmin}} \frac{1}{|\mathbb{I}|} \sum_{s=0}^{|\mathbb{I}|} \mathcal{L}(\bar{\mathbf{x}}_s, \mathbf{z}_s; \Theta) \quad (5.4)$$

In the literature there are various designs of loss function which is adapted to the structure of the neural networks and the image processing task. After selecting an appropriate loss function, the most simple way to solve the minimization problem (5.4) is to perform a gradient descent algorithm, the update of the parameter Θ at the k -th iteration reads:

$$\Theta^{[k+1]} = \Theta^{[k]} - \gamma \frac{1}{|\mathbb{I}|} \sum_{s=0}^{|\mathbb{I}|} \nabla_{\Theta} \mathcal{L}(\bar{\mathbf{x}}_s, \mathbf{z}_s; \Theta) \quad (5.5)$$

where $\gamma > 0$ is the step-size, $\nabla_{\Theta} \mathcal{L}$ denotes the gradient of the loss function. In practice, the gradient is performed by using the backpropagation procedure, the error or loss obtained from NNs after the forward step is backpropagated through the same layer from the last layer of NNs. However, the computation is more complex and numerically expensive in practice. First, the "size" of the minimization problem depends on the scale of data and the architecture of the designed neural network which is usually very large in order to provide an efficient estimator. To address this problem, there are many minimization algorithms relying on stochastic optimization dedicated to this large scale problem such as Adam [Kingma and Ba, 2014], SGD [Milanfar, 2012] or Adagrad [Duchi et al., 2011], to name the most commonly used optimizers.

5.2 Plug-and-Play and Proximal Neural Networks for image restoration

As we discussed in Chapter 2, from the Bayesian perspective, an estimate solution $\widehat{\mathbf{x}}$ of (2.1) can be obtained by solving a Maximum A Posteriori (MAP) estimation problem. To deal with the inverse problem (2.1), we recall the general minimization problem

$$\widehat{\mathbf{x}}_{\text{MAP}} = \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\operatorname{argmin}} \Psi(\mathbf{A}\mathbf{x}; \mathbf{y}) + \lambda\Phi(\mathbf{D}\mathbf{x}) + \iota_S(\mathbf{x}), \quad (5.6)$$

where $S \subset \mathbb{R}^{CN}$ is a closed, convex, non-empty constraint set, $\gamma > 0$ is a regularization parameter, $\mathbf{D}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{JN}$ is a linear operator mapping an image from \mathbb{R}^{CN} to a feature space \mathbb{R}^{JN} , $\Phi: \mathbb{R}^{JN} \rightarrow (-\infty, +\infty]$ and $\Psi(\cdot, \mathbf{y}): \mathbb{R}^{CM} \rightarrow (-\infty, +\infty]$ denotes a proper, lower-semicontinuous, convex function.

Plug-and-Play – First, standard approach to solve (5.6) is the use of iterative optimization algorithms. For instance, Forward-Backward algorithm (described in Chapter 3) is specified to this context:

$$\begin{aligned} &\text{Let } \mathbf{x}_0 \in \mathbb{R}^{CN} \\ &\text{For } k = 0, 1, \dots \\ &\left[\mathbf{x}^{[k+1]} = \operatorname{prox}_{\gamma(\lambda\Phi(\mathbf{D}\cdot) + \iota_S(\cdot))} \left(\mathbf{x}^{[k]} - \gamma \nabla \Psi(\mathbf{A}\mathbf{x}^{[k]}; \mathbf{y}) \right) \right] \end{aligned} \quad (5.7)$$

During the recent years, the performance of these methods have been pushed to the next level by replacing the proximity operator by a NNs to benefit both respective merits of model-based method and learning-based method. The integration has resulted in the Plug-and-Play (PnP) formalism as described in algorithm (5.8) which was initially introduced in [Venkatakrisnan et al., 2013].

$$\begin{aligned} &\text{Let } \mathbf{x}_0 \in \mathbb{R}^{CN} \\ &\text{For } k = 0, 1, \dots \\ &\left[\mathbf{x}^{[k+1]} = f_{\Theta} \left(\mathbf{x}^{[k]} - \gamma \nabla \Psi(\mathbf{A}\mathbf{x}^{[k]}; \mathbf{y}) \right) \right] \end{aligned} \quad (5.8)$$

While early approaches primarily depend on variational denoising techniques, like BM3D, the subsequent advancements in neural networks for denoising have significantly enhanced the reconstructive potential of PnP algorithms. Lately, there has been a growing focus on investigating the convergence and stability characteristics of PnP algorithms.

Building a denoiser – To understand the above replacement, we first recall the image denoising task which aims to find an estimate of an unknown image $\bar{\mathbf{x}} \in \mathbb{R}^{CN}$, from noisy measurements $\mathbf{z} \in \mathbb{R}^{CN}$. In this part, we focus on the Gaussian denoising problem

$$\mathbf{z} = \bar{\mathbf{x}} + \mathbf{n}, \quad (5.9)$$

where $\mathbf{n} \in \mathbb{R}^{CN}$ models an additive white Gaussian noise with standard deviation $\delta > 0$. Thus, the common method to denoise \mathbf{z} is to rely on a maximum *a posteriori* (MAP)

approach, and to define the estimate $\widehat{\mathbf{x}}_{\text{MAP}} \in \mathbb{R}^{CN}$ as a minimizer of a penalized least-squares objective function. A general formulation of this problem is to find

$$\widehat{\mathbf{x}}_{\text{MAP}} = \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \nu \Phi(\mathbf{D}\mathbf{x}) + \iota_S(\mathbf{x}), \quad (5.10)$$

which can be traced back to the formulation of $\operatorname{prox}_{\nu\lambda\Phi(\mathbf{D}\cdot)+\iota_S}$.

As we discussed in section 5.1.3, beside sophisticated hand-crafted denoisers, there are more and more learning-based denoisers relying on the prediction function f_{Θ}^K where the parameters Θ are learned by minimizing the ℓ^2 empirical loss between noisy and ground-truth images:

$$\mathcal{L}(\bar{\mathbf{x}}_s, \mathbf{z}_s; \Theta) := \frac{1}{2} \|\bar{\mathbf{x}}_s - f_{\Theta}^K(\mathbf{z}_s)\|_2^2, \quad (5.11)$$

In [Zhang et al., 2017], the authors proposed a simple architecture of a Deep CNN by integrating batch normalization into the residual learning framework which outperforms the state-of-the-art traditional denoisers such as BM3D. Due to these remarkable performance a large number of studies were dedicated to build more powerful DNN such as in [Zhang et al., 2017]. However, we still have a little comprehension about insight of these black-box models. Additionally, these DL models are not easy to control and sometime require an updating of the training dataset to adapt the DNN to a different context. Furthermore, many DL architecture nowadays become more and more complex and have a huge number of parameters to learn which require an expensive training phase. Finally, as we know, one of the most important factor impacting the performance of a DNN architecture relies on creating or collecting a training dataset which is not always available or time consuming to have a precise annotation. Motivated by these facts, can we build an architecture which is lighter in term of complexity and number of learnable parameters with has high stability properties? Some answers will be provided in Chapter 6 where we focus on the design of a robust denoiser based on unfolding a proximal algorithms that we will called Proximal unfolded Neural Networks (PNNs).

Unfolded neural networks – During the last decade, the performances of proximal algorithms have been pushed to the next level by mixing them with deep learning approaches [Venkatakrishnan et al., 2013, LeCun et al., 2015, Ongie et al., 2020] leading to unfolded neural networks that consist in unrolling optimisation algorithms over a fixed number of iterations [Adler and Öktem, 2018, Jiu and Pustelnik, 2021].

The most simple example of unfolded neural network relies on forward backward iteration (5.7). It consists of fixing the number of iterations K to a small value (typically ~ 20) then the learned parameters are either the step-sizes and the linear operators (LLO) or the proximal operator via f_{Θ} (LPO-Learned proximity operator). LISTA and DeepPDNet belongs to the class of LLO strategy while the work by [Adler and Öktem, 2018] or IRCNN belongs to the class of LPO.

A huge number of references in image restoration is now related to this field of unfolded neural network see for instance [Adler and Öktem, 2018, Bertocchi et al., 2020, Jiu and Pustelnik, 2021, Repetti et al., 2022, Malézieux et al., 2023, Tan et al., 2023, Nguyen et al., 2023].

In chapter 6, in the context of unfolded LLO scheme, we will study the impact of different algorithmic architectures to solve the same problem. The relying question: does a faster algorithm lead to a better architecture?

5.3 Edge detection for degraded image

Nowadays, benefiting from the power of Convolution Neural Network, many architectures have been developed to tackle edge detection. In the following, we will provide a tour on DL-based edge detectors including 3 principal ingredients:

- **Accurate pixel-level annotation** – All DNN architectures rely on pixel-wise edges (cf Figure 2.7-(c)) as the datasets were built like that. Among the datasets, we can first refer to BSDS500 [Arbelaez et al., 2011] and NYUD [Nathan Silberman and Fergus, 2012]. These datasets seem to be dedicated for other higher-level tasks such as object contour detection and semantic edge detection where the objective is to retrieve the boundary of the object of interest. Many other datasets were designed to better adapt for the comparison such as the multi-cue boundary detection where the annotators label different levels of edges: one for the object boundaries and the other for "lower-level" edges.
- **Loss function** – The most standard choice is the binary cross-entropy defined as:

$$\mathcal{L}(\bar{\mathbf{e}}_s, \mathbf{z}_s; \Theta) := -(f_{\Theta}(\mathbf{z}_s) \log(\bar{\mathbf{e}}_s) + (1 - \bar{\mathbf{e}}_s) \log(1 - f_{\Theta}(\mathbf{z}_s))). \quad (5.12)$$

- **A well-designed model** – Recently, various DNNs architecture were born to push up the cutting edge in this domain. The design of architecture become more complex, for instance DeepEdge [Bertasius et al., 2015] and DeepContour [Shen et al., 2015] are based on image patches and associated features to identify the edges. HED proposed by [Xie and Tu, 2015] relies on a VGGnet which combines edges obtained from different stage into a final fuse edge map. More recent methods such as BDCN calculate the loss between the edge map from different layers and the fuse edge with their associated groundtruth edges at different scales. In [Su et al., 2021], a lightweight model integrates traditional edge detection filter is proposed to enhance the performance. A transformer-based edge detector (EDTER) is proposed by [Pu et al., 2022] which split images into two sequences of 16×16 and 8×8 patches then passing through a transformer encoder for detecting global and local feature to predict object boundaries and local boundaries respectively.

On the one hand, DNN-based architecture for edge detection provide very impressive performance but at the cost of complicated architecture and large dataset. Additionally, the task is always performed in the ideal context of undegraded data.

On the other hand, DMS based procedure relying on standard optimization are adapted to degraded data but at the price of high computational time.

Considering the framework of unfolded neural networks, we will explore several possibilities in the context of edge detection for degraded data in Chapter 7.

From proximal algorithms to robust proximal unfolded neural networks for image denoising and Plug-and-Play methods

Summary

6.1	Denoising (accelerated) proximal schemes	67
6.2	Proposed unfolded denoising NNs	68
6.2.1	Primal-dual building block iteration	68
6.2.2	Arrow-Hurwicz unfolded building block	70
6.2.3	Proposed unfolded strategies	71
6.2.4	Proposed learning strategies	72
6.3	Experiments	73
6.3.1	Denoising training setting	73
6.3.2	Architecture comparison	75
6.3.3	Robustness comparison	76
6.3.4	Denoising performance comparison	77
6.3.5	Denoising performance versus robustness	79
6.4	Image restoration with Plug-and-Play method	81
6.4.1	Restoration problem	81
6.4.2	PnP-FB algorithm	82
6.4.3	Robustness comparison	83
6.4.4	Restoration performance comparison	86
6.5	Application in Piecewise Homogeneous Fractal Image Analysis	89
6.5.1	State-of-the-Art.	89
6.5.2	Local Estimation of Self-Similarity	90
6.5.3	Experiments	91
6.6	Conclusion	93

We investigate the design of an unfolded NNs for denoising purpose. The proposed unfolded NNs build a MAP estimate of a denoising problem, that is equivalent to computing a proximity operator. Precisely, we introduce a generic framework to build such denoisers derived from two proximal algorithms: the dual-FB iterations and the primal-dual Chambolle-Pock (CP) iterations as introduced in Chapter 3. The proposed global architectures also includes skip connections either on the primal or on the dual domains, corresponding to inertial parameters for acceleration purpose. We investigate the robustness and the denoising performances of the proposed NNs, for different training strategies. To evaluate the robustness of our NNs, we evaluate their Lipschitz constants. Further, we inject the resulting denoising unfolded networks in a FB algorithm for solving an image deblurring problem, to obtain a plug-and-play algorithm. We analyse how the performance and stability of the proposed denoising unfolded NNs link with those of the PnP method for solving a restoration problem. Finally, we evaluate the performance of the proposed unfolded schemes for a side image processing task that is texture segmentation, that can be formulated as a denoising problem on the local regularity descriptor. The results presented in this chapter gather the contribution presented in [Le et al., 2022b, Le et al., 2022a, Le et al., 2023].

Outline – The remainder of this chapter is organized as follows. Section 6.1 focuses on MAP denoising estimates, with a recall of the considered iterative schemes. Section 6.2 is dedicated to the design of unfolded NNs relying on algorithmic schemes presented in Section 6.1. In Section 6.3, we first compare the denoising performances of the proposed unfolded NNs with state-of-the-art denoisers. In Section 6.4, we compare the resulting FB-PnP methods for solving a deblurring problem when considering different denoisers. This section also focuses on robustness consideration and its impact on both denoising and restoration. Finally, Section 6.5 is dedicated to the evaluation of the proposed unfolded NNs in the context of texture segmentation.

6.1 Denoising (accelerated) proximal schemes

Let $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2$ and $g(\mathbf{x}) = \nu\Phi(\mathbf{D}\mathbf{x})$ (for the convenience for the reader, we denote ν is the regularization parameter in denoising case ($\mathbf{A} = \text{Id}$) and λ is the regularisation in other cases ($\mathbf{A} \neq \text{Id}$)), we recall that the primal formulation of (3.35) reads:

$$\hat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathbb{R}^{CN}}{\text{Argmin}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 + \nu\Phi(\mathbf{D}\mathbf{x}) + \iota_S(\mathbf{x}), \quad (6.1)$$

We can observe that problem (6.1) does not have a closed form solution in general, iterative methods can be used to approximate it. Multiple proximal algorithms can be used to solve (6.1) as detailed in the Chapter 3. In this section, we describe two schemes enabling minimizing (6.1): the FB algorithm applied to its dual problem, and the primal-dual CP algorithm directly applied to (6.1). In addition, for both schemes we also investigate their accelerated versions, namely DiFB (i.e., Dual inertial Forward Backward, also known as FISTA [Beck and Teboulle, 2009, Chambolle and Dossal, 2015]), and ScCP (i.e., CP for strongly convex functions [Chambolle and Pock, 2011]).

For sake of simplicity, in the remainder of the paper we refer to these schemes as FB, CP, DiFB, and ScCP, respectively.

Dual (i)FB – A first strategy to solve (6.1) is to solve the dual formulation (3.45) consisting in applying (i)FB to the dual formulation of problem (6.1):

$$\begin{aligned} & \text{for } k = 0, 1, \dots \\ & \begin{cases} \mathbf{u}^{[k+1]} = \text{prox}_{\tau_k(\nu\Phi)^*} \left(\mathbf{v}^{[k]} + \tau_k \mathbf{D} \text{P}_S(\mathbf{z} - \mathbf{D}^\top \mathbf{v}^{[k]}) \right), \\ \mathbf{v}^{[k+1]} = (1 + \rho_k) \mathbf{u}^{[k+1]} - \rho_k \mathbf{u}^{[k]}, \end{cases} \end{aligned} \quad (6.2)$$

where $(\mathbf{u}^{[0]}, \mathbf{v}^{[0]}) \in \mathbb{R}^{JN} \times \mathbb{R}^{JN}$ and the step-size parameters, for every $k \in \mathbb{N}$, $\tau_k > 0$ and $\rho_k \geq 0$. Note that when, for every $k \in \mathbb{N}$, $\rho_k = 0$, then Algorithm (6.2) reduces to Dual FB.

(Sc)CP – A second strategy to solve (6.1) is to solve the dual formulation (3.45) consisting in applying the (Sc)CP algorithm to problem (6.1). The data-term being ζ -strongly convex with parameter $\zeta = 1$, the accelerated CP [Chambolle and Pock, 2011], dubbed ScCP, can be employed. This algorithm reads

$$\begin{aligned} & \text{for } k = 0, 1, \dots \\ & \begin{cases} \mathbf{x}^{[k+1]} = \text{P}_S \left(\frac{\mu_k}{1+\mu_k} (\mathbf{z} - \mathbf{D}^\top \mathbf{u}^{[k]}) + \frac{1}{1+\mu_k} \mathbf{x}^{[k]} \right), \\ \mathbf{u}^{[k+1]} = \text{prox}_{\tau_k(\nu\Phi)^*} \left(\mathbf{u}^{[k]} + \tau_k \mathbf{D} \left((1 + \alpha_k) \mathbf{x}^{[k+1]} - \alpha_k \mathbf{x}^{[k]} \right) \right), \end{cases} \end{aligned} \quad (6.3)$$

where $\mathbf{x}^{[0]} \in \mathbb{R}^{CN}$ and $\mathbf{u}^{[0]} \in \mathbb{R}^{JN}$. Note that when, for every $k \in \mathbb{N}$, $\alpha_k = 1$, then Algorithm (6.3) reduces to standard iterations of the primal-dual CP algorithm [Chambolle and Pock, 2011], while when $\alpha_k = 0$, it leads to the classical Arrow-Hurwicz algorithm [Arrow et al., 1958].

6.2 Proposed unfolded denoising NNs

The objective of this section is to design unfolded NNs f_Θ such that

$$\widehat{\mathbf{x}} \approx f_\Theta(\mathbf{z}), \quad (6.4)$$

where $\widehat{\mathbf{x}} \in \mathbb{R}^{CN}$ is an estimate of $\bar{\mathbf{x}}$. As discussed in the introduction, such an estimate can correspond to the penalized least-squares estimate of $\bar{\mathbf{x}}$, defined as in (6.1)-(3.45).

6.2.1 Primal-dual building block iteration

The iterations described previously in (6.2) and (6.3) share a similar framework which yields:

$$\begin{aligned} & \text{for } k = 0, 1, \dots \\ & \begin{cases} \mathbf{u}^{[k+1]} = \text{prox}_{\tau_k(\nu\Phi)^*} \left(\mathbf{u}^{[k]} + \tau_k \mathbf{D} \mathbf{x}^{[k]} \right) \\ \mathbf{x}^{[k+1]} = \text{P}_S \left(\frac{\mu_k}{1+\mu_k} (\mathbf{z} - \mathbf{D}^\top \mathbf{u}^{[k+1]}) + \frac{1}{1+\mu_k} \mathbf{x}^{[k]} \right). \end{cases} \end{aligned} \quad (6.5)$$

On the one hand, this scheme is a reformulation of the Arrow-Hurwicz (AH) iterations, i.e., Algorithm (6.3) with $\alpha_k = 0$. On the other hand, for the limit case when $\mu_k \rightarrow +\infty$, the DFB (6.2) iterations are recovered. Further, the inertia step is activated either on the dual variable for DiFB

$$\mathbf{u}^{[k+1]} \leftarrow (1 + \rho_k)\mathbf{u}^{[k+1]} - \rho_k\mathbf{u}^{[k]} \quad (6.6)$$

or on the primal variable for ScCP

$$\mathbf{x}^{[k+1]} \leftarrow (1 + \alpha_k)\mathbf{x}^{[k+1]} - \alpha_k\mathbf{x}^{[k]}. \quad (6.7)$$

Based on these observations, we propose a strategy to unroll D(i)FB and (Sc)CP algorithms using a primal-dual perspective. This perspective will subsequently be used to build denoising NNs as defined satisfying (6.4).

The result provided below aims to emphasize that each iteration of the primal-building block (6.5) can be viewed as the composition of two layers of feedforward networks, acting either on the image domain (i.e., primal domain \mathbb{R}^{CN}) or in the features domain (i.e., dual domain \mathbb{R}^{JN}). DFB, DiFB, CP, and ScCP, hold the same structure, with extra steps that can be assimilated to skip connections in the specific case of DiFB and ScCP, enabling to keep track of previous layer's outputs (see (6.6) and (6.7)).

Proposition 6.2.1. *Let $\mathbf{z} \in \mathbb{R}^{CN}$, $\mathbf{x} \in \mathbb{R}^{CN}$, $\mathbf{u} \in \mathbb{R}^{JN}$, and $k \in \mathbb{N}$. Let $T_{v, \Theta_{k, \mathcal{D}}, \mathcal{D}}: \mathbb{R}^{CN} \times \mathbb{R}^{JN} \rightarrow \mathbb{R}^{JN}$, defined as*

$$T_{v, \Theta_{k, \mathcal{D}}, \mathcal{D}}(\mathbf{x}, \mathbf{u}) = \eta_{v, k, \mathcal{D}}(W_{k, \mathcal{D}}\mathbf{x} + V_{k, \mathcal{D}}\mathbf{u} + \mathbf{b}_{k, \mathcal{D}}), \quad (6.8)$$

be a sub-layer acting on both the primal variable \mathbf{x} and the dual variable \mathbf{u} and returning a dual (\mathcal{D}) variable. In (6.8) $\eta_{v, k, \mathcal{D}}: \mathbb{R}^{JN} \rightarrow \mathbb{R}^{JN}$ is a fixed activation function with parameter $v > 0$, and $\Theta_{k, \mathcal{D}}$ is a linear parametrization of the learnable parameters including $W_{k, \mathcal{D}}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{JN}$, $V_{k, \mathcal{D}}: \mathbb{R}^{JN} \rightarrow \mathbb{R}^{JN}$ and $\mathbf{b}_{k, \mathcal{D}} \in \mathbb{R}^{JN}$.

Let $T_{z, \Theta_{k, \mathcal{P}}, \mathcal{P}}: \mathbb{R}^{CN} \times \mathbb{R}^{JN} \rightarrow \mathbb{R}^{CN}$, defined as

$$T_{z, \Theta_{k, \mathcal{P}}, \mathcal{P}}(\mathbf{x}, \mathbf{u}) = \eta_{k, \mathcal{P}}(W_{k, \mathcal{P}}\mathbf{x} + V_{k, \mathcal{P}}\mathbf{u} + \mathbf{b}_{k, \mathcal{P}}), \quad (6.9)$$

be a sub-layer acting on both the primal variable \mathbf{x} and the dual variable \mathbf{u} and returning a primal (\mathcal{P}) variable. In (6.9) $\eta_{k, \mathcal{P}}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{CN}$ is a fixed activation functions, and $\Theta_{k, \mathcal{P}}$ is a linear parametrization of the learnable parameters including $W_{k, \mathcal{P}}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{CN}$, $V_{k, \mathcal{P}}: \mathbb{R}^{JN} \rightarrow \mathbb{R}^{CN}$ and $\mathbf{b}_{k, \mathcal{P}} \in \mathbb{R}^{CN}$.

Then, the k -th iteration of the joint formulation (6.5) can be written as a composition of two layers of the form of (5.1):

$$T_{z, \Theta_k}: \mathbb{R}^{CN} \times \mathbb{R}^{JN} \rightarrow \mathbb{R}^{CN} \\ (\mathbf{x}^{[k]}, \mathbf{u}^{[k]}) \mapsto T_{z, \Theta_{k, \mathcal{P}}, \mathcal{P}}(\mathbf{x}, T_{\Theta_{k, \mathcal{D}}, \mathcal{D}}(\mathbf{x}^{[k]}, \mathbf{u}^{[k]})), \quad (6.10)$$

where Θ_k is the combination of $\Theta_{k, \mathcal{P}}$ and $\Theta_{k, \mathcal{D}}$, i.e., the linear parametrization of all learnable parameters for layer k .

Proof. This result is obtained by noticing that, for every $k \in \mathbb{N}$, the k -th iteration (6.5) can be rewritten as (6.10), where the primal (\mathcal{P}) operators are given by $W_{k, \mathcal{P}} = \frac{1}{1+\mu_k}$, $V_{k, \mathcal{P}} = -\frac{\mu_k}{1+\mu_k}\mathbf{D}^\top$, $\mathbf{b}_{k, \mathcal{P}} = \frac{\mu_k}{1+\mu_k}\mathbf{z}$, and $\eta_{k, \mathcal{P}} = P_S$, and the dual (\mathcal{D}) operators are given by $W_{k, \mathcal{D}} = \tau_k\mathbf{D}$, $V_{k, \mathcal{D}} = \text{Id}$, $\mathbf{b}_{k, \mathcal{D}} = 0$, $\eta_{v, k, \mathcal{D}} = \text{prox}_{\tau_k(v\Phi)^*}$.

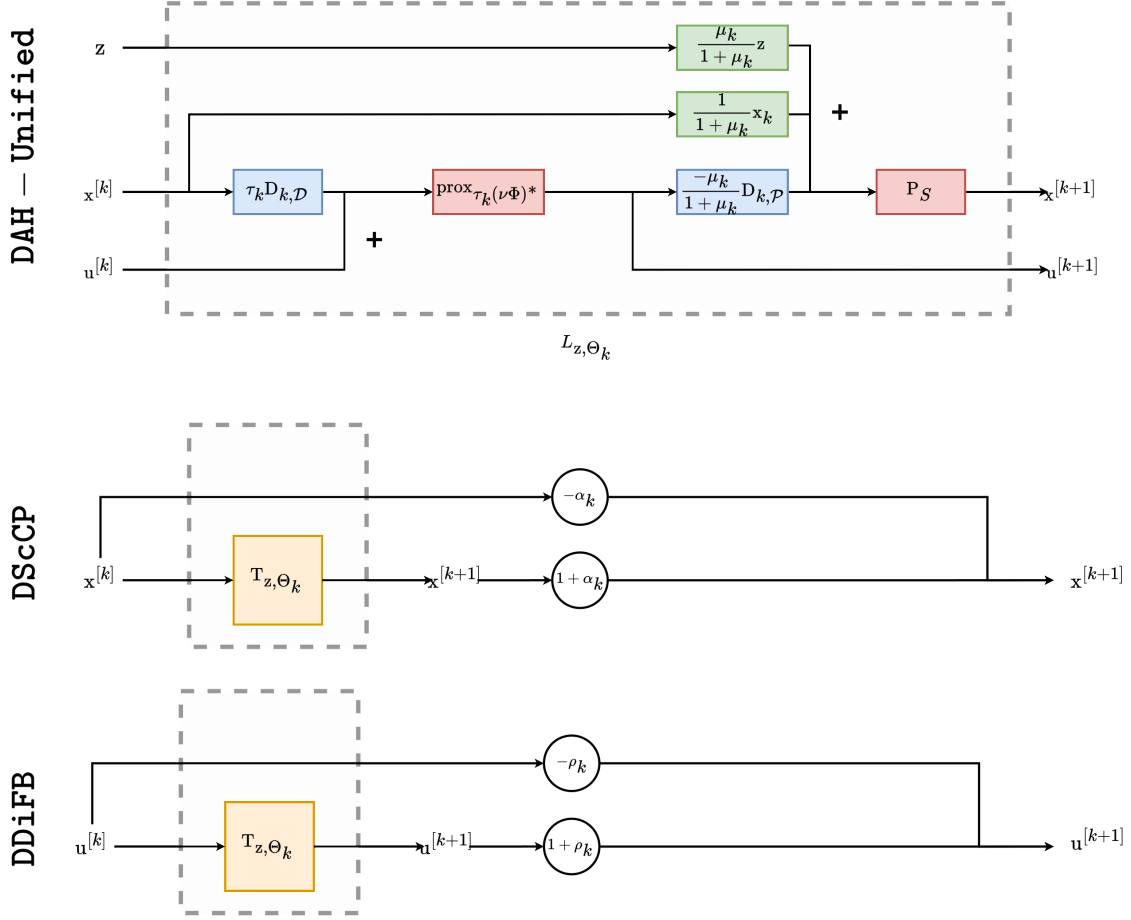


Figure 6.1: Top: Architecture of the proposed DAH-Unified block for the k -th layer. Linearities, biases, and activation functions are shown in blue, green and red, respectively. Bottom: Inertial step for DScCP (top) and DDiFB (bottom), for the k -th layer.

6.2.2 Arrow-Hurwicz unfolded building block

Our unrolled architectures rely on layer structures introduced in Proposition 6.2.1 where we allow the linear operator \mathbf{D} to be different for each layer. For more flexibility, we also introduce, for every $k \in \{1, \dots, K\}$, operators $\mathbf{D}_{k, \mathcal{D}}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{JN}$ and $\mathbf{D}_{k, \mathcal{P}}: \mathbb{R}^{JN} \rightarrow \mathbb{R}^{CN}$, to replace operators \mathbf{D} and \mathbf{D}^\top , respectively, to allow a possible mismatch between operator \mathbf{D} and its adjoint \mathbf{D}^\top . The resulting unfolding Deep Arrow-Hurwicz (DAH) building block is then given below:

$$f_{z, \Theta}^{K, \text{DAH}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}) = T_{z, v, \Theta_K}^{\text{DAH}} \circ \dots \circ T_{z, v, \Theta_1}^{\text{DAH}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}), \quad (6.11)$$

where, for every $k \in \{1, \dots, K\}$,

$$\begin{aligned} \mathbf{u}^{[k]} &= T_{v, \Theta_k, \mathcal{D}, \mathcal{D}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k-1]}) \\ \mathbf{x}^{[k]} &= T_{z, \Theta_k, \mathcal{P}, \mathcal{P}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k]}) \\ T_{z, v, \Theta_k}^{\text{DAH}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k-1]}) &= (\mathbf{x}^{[k]}, \mathbf{u}^{[k]}). \end{aligned}$$

with

$$\begin{cases} W_{k,\mathcal{D}} = \tau_k \mathbf{D}_{k,\mathcal{D}}, \\ V_{k,\mathcal{D}} = \text{Id}, \\ b_{k,\mathcal{D}} = 0, \\ \eta_{\nu,k,\mathcal{D}} = \text{prox}_{\tau_k(\nu\Phi)^*}, \end{cases} \quad \text{and} \quad \begin{cases} W_{k,\mathcal{P}} = \frac{1}{1+\mu_k}, \\ V_{k,\mathcal{P}} = -\frac{\mu_k}{1+\mu_k} \mathbf{D}_{k,\mathcal{P}}, \\ b_{\mathbf{z},k,\mathcal{P}} = \frac{\mu_k}{1+\mu_k} \mathbf{z}, \\ \eta_{k,\mathcal{P}} = P_S. \end{cases}$$

6.2.3 Proposed unfolded strategies

In this section we describe four unfolded strategies for building denoising NNs as defined in (6.4). All strategies rely on Arrow-Hurwicz building block presented in Section 6.2.2.

- **DDFB** stands for Deep Dual Forward-Backward and it fits DAH when $\mu_k \rightarrow +\infty$.

$$f_{\mathbf{z},\nu,\Theta}^{K,\text{DDFB}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}) = T_{\mathbf{z},\nu,\Theta_K}^{\text{DDFB}} \circ \dots \circ T_{\mathbf{z},\nu,\Theta_1}^{\text{DDFB}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}), \quad (6.12)$$

where, for every $k \in \{1, \dots, K\}$,

$$\begin{aligned} \mathbf{u}^{[k]} &= T_{\nu,\Theta_k,\mathcal{D}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k-1]}) \\ \mathbf{x}^{[k]} &= T_{\mathbf{z},\Theta_k,\mathcal{P}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k]}) \\ (\mathbf{x}^{[k]}, \mathbf{u}^{[k]}) &= T_{\mathbf{z},\nu,\Theta_k}^{\text{DDFB}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k-1]}). \end{aligned}$$

with

$$\begin{cases} W_{k,\mathcal{D}} = \tau_k \mathbf{D}_{k,\mathcal{D}}, \\ V_{k,\mathcal{D}} = \text{Id}, \\ b_{k,\mathcal{D}} = 0, \\ \eta_{\nu,k,\mathcal{D}} = \text{prox}_{\tau_k(\nu\Phi)^*}, \end{cases} \quad \text{and} \quad \begin{cases} W_{k,\mathcal{P}} = 0, \\ V_{k,\mathcal{P}} = -\mathbf{D}_{k,\mathcal{P}}, \\ b_{\mathbf{z},k,\mathcal{P}} = \mathbf{z}, \\ \eta_{k,\mathcal{P}} = P_S. \end{cases}$$

- **DDiFB**: stands for Deep Dual inertial Forward-Backward interpreted as a DDFB with skip connections and defined as:

$$f_{\mathbf{z},\nu,\Theta}^{K,\text{DDiFB}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}) = (\mathbf{x}^{[k]}, \mathbf{u}^{[k]}) \quad (6.13)$$

where, for every $k \in \{1, \dots, K\}$,

$$\begin{aligned} (\mathbf{x}^{[k]}, \widetilde{\mathbf{u}}^{[k]}) &= T_{\mathbf{z},\nu,\Theta_k}^{\text{DDFB}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k-1]}) \\ \mathbf{u}^{[k]} &= (1 + \rho_k) \widetilde{\mathbf{u}}^{[k]} - \rho_k \mathbf{u}^{[k-1]}. \end{aligned}$$

- **DCP** stands for Deep Chambolle-Pock relying on DAH with a special update of the primal variable leading to:

$$f_{\mathbf{z},\nu,\Theta}^{K,\text{DCP}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}) = (\mathbf{x}^{[k]}, \mathbf{u}^{[K]}) \quad (6.14)$$

where, for every $k \in \{1, \dots, K\}$,

$$\begin{aligned} (\widetilde{\mathbf{x}}^{[k]}, \mathbf{u}^{[k]}) &= T_{\mathbf{z},\nu,\Theta_k}^{\text{DAH}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k-1]}) \\ \mathbf{x}^{[k]} &= 2\widetilde{\mathbf{x}}^{[k]} - \mathbf{x}^{[k-1]}. \end{aligned}$$

- **DScCP** stands for Deep Strong convexity Chambolle-Pock interpreted as a DAH with skip connections on the primal variable:

$$f_{z,v,\Theta}^{K,\text{DScCP}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}) = (\mathbf{x}^{[K]}, \mathbf{u}^{[K]}) \quad (6.15)$$

where, for every $k \in \{1, \dots, K\}$,

$$\begin{aligned} (\tilde{\mathbf{x}}^{[k]}, \mathbf{u}^{[k]}) &= \mathbb{T}_{z,v,\Theta_k}^{\text{DAH}}(\mathbf{x}^{[k-1]}, \mathbf{u}^{[k-1]}) \\ \mathbf{x}^{[k]} &= (1 + \alpha_k)\tilde{\mathbf{x}}^{[k]} - \alpha_k\mathbf{x}^{[k-1]}. \end{aligned}$$

Illustration of a single layer of the resulting DD(i)FB and D(Sc)CP architectures are provided in Figure 6.1.

Since these architectures are reminiscent of D(i)FB and (Sc)CP, given in Section 6.1, we can deduce limit cases for the proposed unfolded strategies, when $K \rightarrow +\infty$ and linear operators are fixed over the layers.

Corollary 1 (Limit case for deep unfolded NNs). *We consider the NNs DD(i)FB and D(Sc)CP defined in Section 6.2.3. Assume that, for every $k \in \{1, \dots, K\}$, $\mathbf{D}_{k,\mathcal{D}} = \mathbf{D}$ and $\mathbf{D}_{k,\mathcal{P}} = \mathbf{D}^\top$, for $\mathbf{D}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{JN}$. In addition, for each architecture, we further assume that, for every $k \in \{1, \dots, K\}$,*

- DDFB: $\tau_k \in (0, 2/\|\mathbf{D}\|_{sp}^2)$.
- DDiFB: $\tau_k \in (0, 1/\|\mathbf{D}\|_{sp}^2)$ and $\rho_k = \frac{t_k-1}{t_{k+1}}$ with $t_k = \frac{k+a-1}{a}$ and $a > 2$.
- DCP: $(\tau_k, \mu_k) \in (0, +\infty)^2$ such that $\tau_k\mu_k\|\mathbf{D}\|_{sp}^2 < 1$.
- DScCP: $\alpha_k = (1 + 2\mu_k)^{-1/2}$, $\mu_{k+1} = \alpha_k\mu_k$, and $\tau_{k+1} = \tau_k\alpha_k^{-1}$ with $\tau_0\mu_0\|\mathbf{D}\|_{sp}^2 \leq 1$.

Then, we have $\mathbf{x}^{[K]} \rightarrow \hat{\mathbf{x}}$ when $K \rightarrow +\infty$, where $\mathbf{x}^{[k]}$ is the output of either of the unfolded NNs DD(i)FB or D(Sc)CP, and $\hat{\mathbf{x}}$ is a solution to (6.1)

6.2.4 Proposed learning strategies

The proposed DD(i)FB and D(Sc)CP allow for flexibility in the learned parameters. In this work, we propose two learning strategies, either satisfying conditions described in Corollary 1 (LNO), or giving flexibility to the parameters (LFO). These two strategies are described below.

1. **Learned Normalized Operators (LNO):** This strategy aims to use theoretical conditions ensuring convergence of D(i)FB and (Sc)CP, i.e., choosing the stepsizes appearing in deep-D(i)FB and deep-(Sc)CP according to the conditions given in Corollary 1. In this context, for every $k \in \{1, \dots, K\}$, we choose $\mathbf{D}_{k,\mathcal{P}}$ to be equal to the adjoint $\mathbf{D}_{k,\mathcal{D}}^\top$ of $\mathbf{D}_{k,\mathcal{D}}$. However, unlike in Corollary 1, we allow $\mathbf{D}_{k,\mathcal{D}}$ to vary for the different layers $k \in \{1, \dots, K\}$.
2. **Learned Flexible Operator (LFO):** This strategy, introduced in [Le et al., 2022a], consists in learning the stepsizes appearing in deep-D(i)FB and deep-(Sc)CP without constraints, as well as allowing a mismatch in learning the adjoint operator \mathbf{D}^\top of \mathbf{D} , i.e., learning $\mathbf{D}_{k,\mathcal{D}}$ and $\mathbf{D}_{k,\mathcal{P}}$ independently.

The learnable parameters are summarized in Table 6.1.

Table 6.1: Learnable parameters of each unfolded scheme

	Θ_k	Comments
DDFB-LFO	$\mathbf{D}_{k,\mathcal{P}}, \mathbf{D}_{k,\mathcal{D}}$	absorb τ_k in $\mathbf{D}_{k,\mathcal{D}}$
DDiFB-LFO	$\mathbf{D}_{k,\mathcal{P}}, \mathbf{D}_{k,\mathcal{D}}, \alpha_k$	fix α_k , and absorb τ_k in $\mathbf{D}_{k,\mathcal{D}}$
DDFB-LNO	$\mathbf{D}_{k,\mathcal{P}} = \mathbf{D}_{k,\mathcal{D}}^\top$	define $\tau_k = 1.99\ \mathbf{D}_k\ ^{-2}$
DDiFB-LNO	$\mathbf{D}_{k,\mathcal{P}} = \mathbf{D}_{k,\mathcal{D}}^\top$	fix $\alpha_k = \frac{t_k-1}{t_{k+1}}$, $t_{k+1} = \frac{k+a-1}{a}$, $a > 2$, and $\tau_k = 0.99\ \mathbf{D}_k\ ^{-2}$
DCP-LFO	$\mathbf{D}_{k,\mathcal{P}}, \mathbf{D}_{k,\mathcal{D}}, \mu$	learn $\mu = \mu_0 = \dots = \mu_K$, and absorb τ_k in $\mathbf{D}_{k,\mathcal{D}}$
DS _c CP-LFO	$\mathbf{D}_{k,\mathcal{P}}, \mathbf{D}_{k,\mathcal{D}}, \mu_0$	learn μ_0 , absorb τ_k in $\mathbf{D}_{k,\mathcal{D}}$, and fix $\alpha_k = (1 + 2\mu_k)^{-1/2}$, and $\mu_{k+1} = \alpha_k \mu_k$
DCP-LNO	$\mathbf{D}_{k,\mathcal{P}} = \mathbf{D}_{k,\mathcal{D}}^\top, \mu$	learn $\mu = \mu_0 = \dots = \mu_K$, and fix $\tau_k = 0.99\mu^{-1}\ \mathbf{D}_k\ ^{-2}$
DS _c CP-LNO	$\mathbf{D}_{k,\mathcal{P}} = \mathbf{D}_{k,\mathcal{D}}^\top, \mu_k$	learn μ_k , and fix $\alpha_k = (1 + 2\mu_k)^{-1/2}$, and $\tau_k = 0.99\mu_k^{-1}\ \mathbf{D}_k\ ^{-2}$

6.3 Experiments

6.3.1 Denoising training setting

Training dataset – We consider two sets of images: the *training set* $(\bar{\mathbf{x}}_s, \mathbf{z}_s)_{s \in \mathbb{I}}$ of size $|\mathbb{I}|$ and the *test set* $(\bar{\mathbf{x}}_s, \mathbf{z}_s)_{s \in \mathbb{J}}$ of size $|\mathbb{J}|$. For both sets, each couple $(\bar{\mathbf{x}}_s, \mathbf{z}_s)$ consists of a clean multichannel image $\bar{\mathbf{x}}_s$ of size CN_s (where C denotes the number of channels, and N_s the number of pixels in each channel), and a noisy version of this image given by $\mathbf{z}_s = \bar{\mathbf{x}}_s + \mathbf{n}_s$ with $\mathbf{n}_s \sim \mathcal{N}(0, \delta^2 \text{Id})$ for $\delta > 0$.

Training strategy for unfolded denoising networks – The network parameters are optimized by minimizing the ℓ^2 empirical loss between noisy and ground-truth images:

$$\widehat{\Theta} \in \underset{\Theta}{\text{Argmin}} \frac{1}{|\mathbb{I}|} \sum_{s \in \mathbb{I}} \mathcal{L}(\bar{\mathbf{x}}_s, \mathbf{z}_s; \Theta) \quad (6.16)$$

where

$$\mathcal{L}(\bar{\mathbf{x}}_s, \mathbf{z}_s; \Theta) := \frac{1}{2} \|\bar{\mathbf{x}}_s - f_{z_s, \delta^2, \Theta}^K(\mathbf{z}_s, \mathbf{D}_{K, \mathcal{D}}(\mathbf{z}_s))\|^2,$$

and $f_{z_s, \delta^2, \Theta}^K$ is either of the unfolded networks described in Section 6.2.3. The loss (6.16) will be optimized in Pytorch with Adam algorithm [Kingma and Ba, 2014]. For the sake of simplicity, in the following we drop the indices in the notation of the network $f_{z_s, \delta^2, \Theta}^K$ and use the notation f_Θ .

Architectures – We will compare the different architectures introduced in Section 6.2.3, namely DDFB, DDiFB, DDCP, and DScCP, considering both LNO and LFO learning strategies (see Table 6.1 for details). For each architecture and every layer $k \in \{1, \dots, K\}$, the weight operator $\mathbf{D}_{k, \mathcal{D}}$ consists of J convolution filters (features), mapping an image in \mathbb{R}^{CN_s} to features in \mathbb{R}^{JN} . For LFO strategies, weight operator $\mathbf{D}_{k, \mathcal{F}}$ consists of J convolution filters mapping from \mathbb{R}^{JN_s} to \mathbb{R}^{CN_s} . All convolution filters considered in our work have the same kernel size of 3×3 .

We evaluate the performance of the different proposed models, varying the numbers of layers K and the number of convolution filters J . In our experiments we consider $\Phi = \|\cdot\|_1$ leading to HardTanh activation function and recalled below.

Proposition 6.3.1. *The proximity operator of the conjugate of the ℓ_1 -norm scaled by parameter $\nu > 0$ is equivalent to the HardTanh activation function, i.e., for every $\mathbf{x} = (\mathbf{x}_i)_{1 \leq i \leq N}$:*

$$\begin{aligned} (p_i)_{1 \leq i \leq N} &= \text{prox}_{(\nu \|\cdot\|_1)^*}(\mathbf{x}) = P_{\|\cdot\|_\infty \leq \nu}(\mathbf{x}) \\ &= \text{HardTanh}_\nu(\mathbf{x}) \end{aligned}$$

where

$$p_i = \begin{cases} -\nu & \text{if } \mathbf{x}_i < -\nu, \\ \nu & \text{if } \mathbf{x}_i > \nu, \\ \mathbf{x}_i & \text{otherwise.} \end{cases}$$

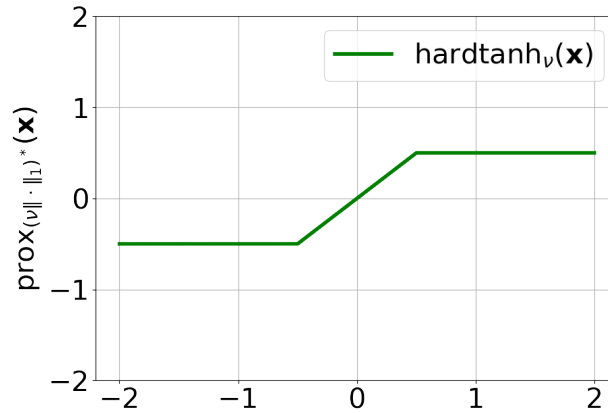


Figure 6.2: Hardtanh function with $\nu = 0.5$

Experimental settings – To evaluate and compare the proposed unfolded architectures, we consider 2 training settings. In both cases, we consider RGB images (i.e., $C = 3$).

- **Training Setting 1 – Fixed noise level:** The NNs are trained with $|\mathbb{I}| = 200$ images extracted from BSDS500 dataset [Arbelaez et al., 2011], with a fixed noise level $\delta = 0.08$. The learning rate for ADAM is set to 8×10^{-5} (all other parameters set as default), we use batches of size of 10 and patches of size 50×50 randomly selected.

We train the proposed unfolded NNs considering multiple sizes of convolution filters $J \in \{8, 16, 32, 64\}$ and for multiple numbers of layers $K \in \{5, 10, 15, 20, 25\}$.

- Training Setting 2 – Variable noise level:** The NNs are trained using the test dataset of ImageNet [Russakovsky et al., 2015] ($|\mathbb{I}| = 5 \times 10^4$), with learning rate for ADAM is set to 1×10^{-3} (all other parameters set as default), a batch size of 200, and patches of size 50×50 randomly selected. Further, we consider a variable noise level, i.e., the images are degraded by a Gaussian noise with standard deviation δ_i , for $i \in \mathbb{I}$, selected randomly with uniform distribution in $[0, 0.1]$.

All unfolded NNs are trained with the same configuration $(J, K) = (64, 20)$, leading to $\sim 34.5 \times 10^3$ parameters for each model.

In our experiments, we aim to compare the proposed unfolded NNs for three metrics: (i) architecture complexity, (ii) robustness, and (iii) denoising performance. For sake of completeness, these metrics will also be provided for a state-of-the-art denoising network, namely DRUnet [Zhang et al., 2021].

For both Settings 1 and 2, our test set \mathbb{J} corresponds to randomly selected subsets of images from the BSDS500 validation set. The size of the test set $|\mathbb{J}|$ will vary depending on the experiments, hence will be specified for each case.

6.3.2 Architecture comparison

We first compare the proposed unfolded NNs (for both LNO and LFO learning strategies) in terms of runtime, FLOPs and number of learnable parameters (i.e., $|\Theta|$). These values, for $(K, J) = (20, 64)$ are summarized in Table 6.2, also including the metrics for DRUnet. The experiments are conducted in PyTorch, using an Nvidia Tesla V100 PCIe 16GB.

From the table, it is obvious that the unfolded NNs have much lighter architectures than DRUnet.

Table 6.2: Architecture comparison. Runtime (in sec.), number of parameters $|\Theta|$ and FLOPs (in G) of the denoisers when used on 100 images of size $3 \times 481 \times 321$. Values for the DUNNs are given for fixed $(K, J) = (20, 64)$.

	average (std) (msec)	$ \Theta $	FLOPs ($\times 10^3$ G)
BM3D	$13 \times 10^3 \pm 317$	–	–
DRUnet	96 ± 21	32,640,960	137.24
LNO	DDFB	3 ± 1.5	34,560
	DDiFB	3 ± 0.5	34,560
	DDCP	6 ± 1	34,561
	DDScCP	7 ± 1	34,580
LFO	DDFB	4 ± 17	69,120
	DDiFB	5 ± 15	69,121
	DDCP	7 ± 14	69,121
	DDScCP	9 ± 15	69,160

6.3.3 Robustness comparison

Multiple works in the literature investigated NN robustness against perturbations [Jakubovitz and Giryes, 2018, Hoffman et al., 2019, Pesquet et al., 2021]. Formally, given an input \mathbf{z} and a perturbation ϵ , the error on the output can be upper bounded via the inequality

$$\|f_{\Theta}(\mathbf{z} + \epsilon) - f_{\Theta}(\mathbf{z})\| \leq \chi \|\epsilon\|. \quad (6.17)$$

The parameter $\chi > 0$ can then be used as a certificate of the robustness for the network. This analysis is important and complementary to quality recovery performance to choose a reliable model. According to [Combettes and Pesquet, 2020], in the context of feedforward NNs as those proposed in this work and described in Section 6.2, χ can be upper bounded by looking at the norms of each linear operator, i.e.,

$$\chi \leq \prod_{k=1}^K \left(\|W_{k,\mathcal{P}}\|_S \times \|W_{k,\mathcal{D}}\|_S \right). \quad (6.18)$$

Unfortunately, as shown in [Le et al., 2022a], this bound can be very pessimistic and not ideal to conclude on the robustness of the network. Instead, a tighter bound can be computed using the definition of Lipschitz continuity. Indeed, by definition the parameter χ in (6.17) is a Lipschitz constant of f_{Θ} . This Lipschitz constant can be computed by noticing that it corresponds to the maximum of $\|J f_{\Theta}(\mathbf{z})\|_{sp}$ over all possible images $\mathbf{z} \in \mathbb{R}^{CN}$, where J denotes the Jacobian operator. Since such a value is impossible to compute for all images of \mathbb{R}^{CN} , we can restrict our study on images similar to those used for training the network, i.e.,

$$\chi \approx \max_{(\mathbf{z}_s)_{s \in \mathbb{I}}} \|J f_{\Theta}(\mathbf{z}_s)\|_{sp}. \quad (6.19)$$

Such an approach has been proposed in [Pesquet et al., 2021] for constraining the value of χ during the training process. In practice, the norm is computed using power iterations coupled with automatic differentiation in Pytorch.

Motivated by these facts, we evaluate the robustness of our models by computing an approximation of χ , as described in (6.19), considering images in the test set \mathbb{J} instead of the training set \mathbb{I} .

Here, \mathbb{J} corresponds to 100 images randomly selected from BSD500 validation set, and for every $s \in \mathbb{J}$, $\mathbf{z}_s = \bar{\mathbf{x}}_s + \mathbf{w}_s$, where $\mathbf{w}_s \sim \mathcal{N}(0, \delta_s \text{Id})$.

Training Setting 1 – Fixed noise level: For this setting we fix $\delta_s \equiv 0.08$. Corresponding values of χ for DD(i)FB and D(Sc)CP, with both LNO and LFO learning strategies, are reported in Figure 6.3. For this setting, we show the evolution of the value of χ for $K \in \{5, 10, 15, 20, 25\}$ and $J \in \{8, 16, 32, 64\}$. We observe that the value of χ for LFO schemes are higher than their LNO counterparts, i.e., LFO schemes are less robust than LNO according to (6.17). In addition, DDFB-LNO and D(Sc)CP-LNO seems to be the most robust schemes. Their χ value decreases slightly when K increases, and increases slightly when J increases.

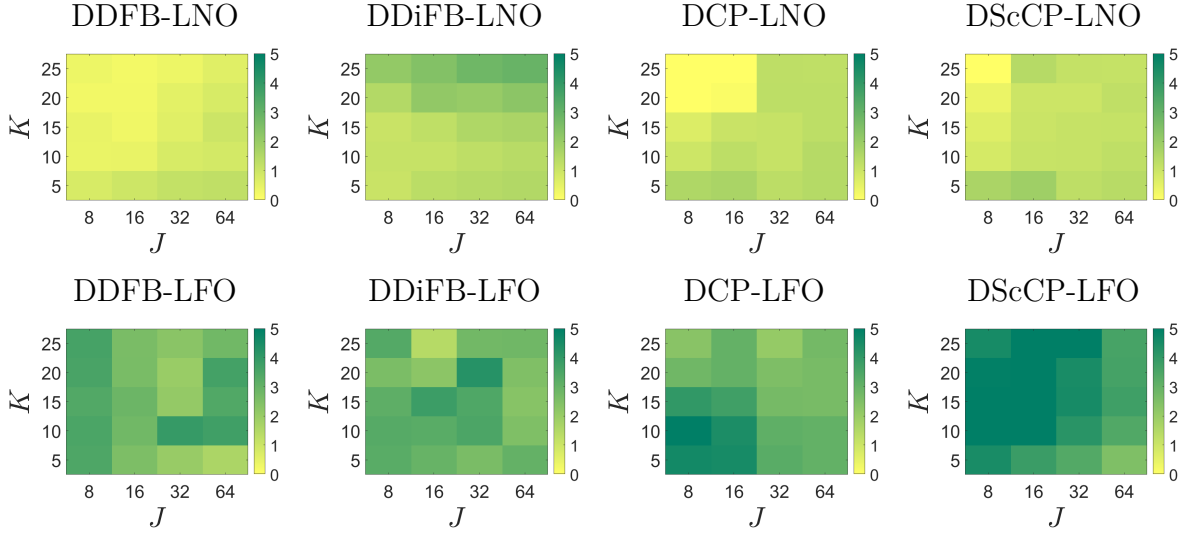


Figure 6.3: Training Setting 1: Robustness. Values $\chi = \max_{s \in \mathbb{J}} \|J f_{\Theta}(\mathbf{z}_s)\|_{sp}$ (\log_2 scale) for the proposed DUNNs, with $J \in \{8, 16, 32, 64\}$ and $K \in \{5, 10, 15, 20, 25\}$. **Top row:** LNO settings. **Bottom row:** LFO settings.

Training Setting 2 – Variable noise level: For this setting, Figure 6.4 gives the box plots showing the distribution of the 100 values of $(\|J f_{\Theta}(\mathbf{z}_s)\|_{sp})_{s \in \mathbb{J}}$, with $\delta_s \sim \mathcal{U}([0, 0.01])$. For the sake of completeness, the norms are also computed for DRUnet[Zhang et al., 2021]. Results are similar to the ones obtained with **Training Setting 1**. Starting from the more robust schemes and moving to the less robust ones, we observe that DDFB-LNO and DScCP-LNO have the smallest values of $(\|J f_{\Theta}(\mathbf{z}_s)\|_{sp})_{s \in \mathbb{J}}$. D(Sc)CP-LFO have similar values to DDFB-LNO and DScCP-LNO, slightly larger and more spread out. Then DCP-LNO and DDiFB-LFO are comparable, with larger values than the previously mentioned schemes, with a few outliers. Finally DDFB-LFO and DRUnet are comparable, with more outliers and higher median, Q1 and Q3 values, followed by DDiFB-LNO that may have very high norm values, depending on the image (although Q3 is smaller than the Q1 value of DRUnet). Note that, overall DRUnet has the worst median, Q1 and Q3 values.

6.3.4 Denoising performance comparison

Training Setting 1 – Fixed noise level: We evaluate the denoising performance of the four proposed architectures considering either the LNO or LFO learning strategy, varying K and J , on $|\mathbb{J}| = 100$ noisy images obtained from the test set, with noise standard deviation $\delta = 0.08$. The average PSNR value for these test noisy images is 21.88 dB.

In Figure 6.5, we show the averaged PSNR values obtained with the proposed unfolded networks. We observe a strong improvement of the denoising performances of all the NNs when increasing the size J of convolution filters, as well as a moderate improvement when increasing the depth K of the NNs. All methods have very similar performances, with the exception of ScCP (both LNO and LFO) which has much higher denoising power.

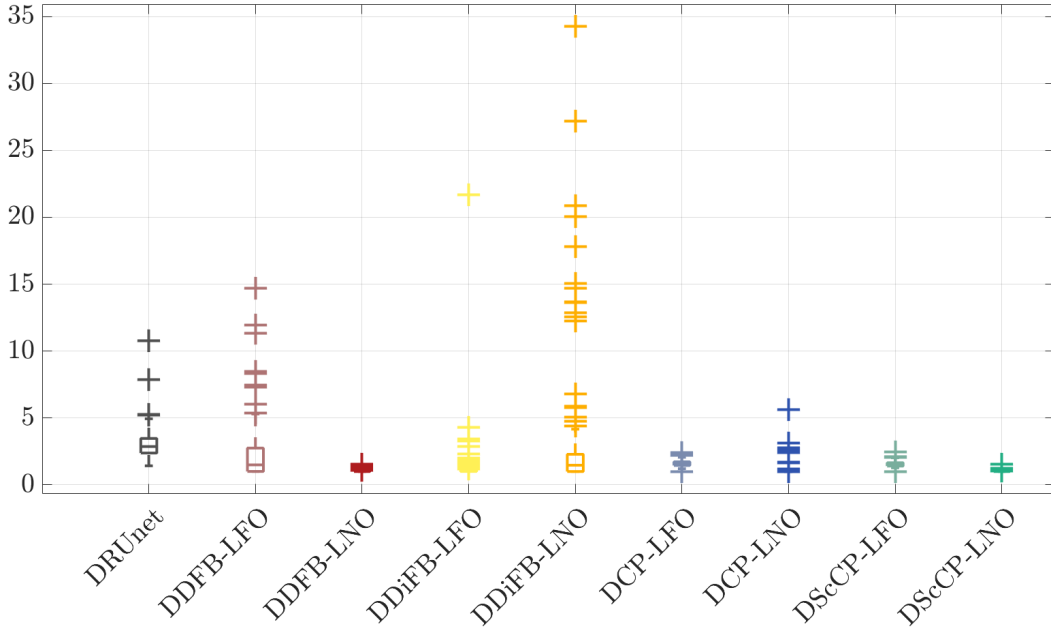


Figure 6.4: Training Setting 2: Robustness. Distribution of $(\|J f_{\theta}(\mathbf{z}_s)\|_S)_{s \in \mathbb{J}}$ for 100 images extracted from BSDS500 validation dataset \mathbb{J} , for the proposed DUNNs and DRUnet.

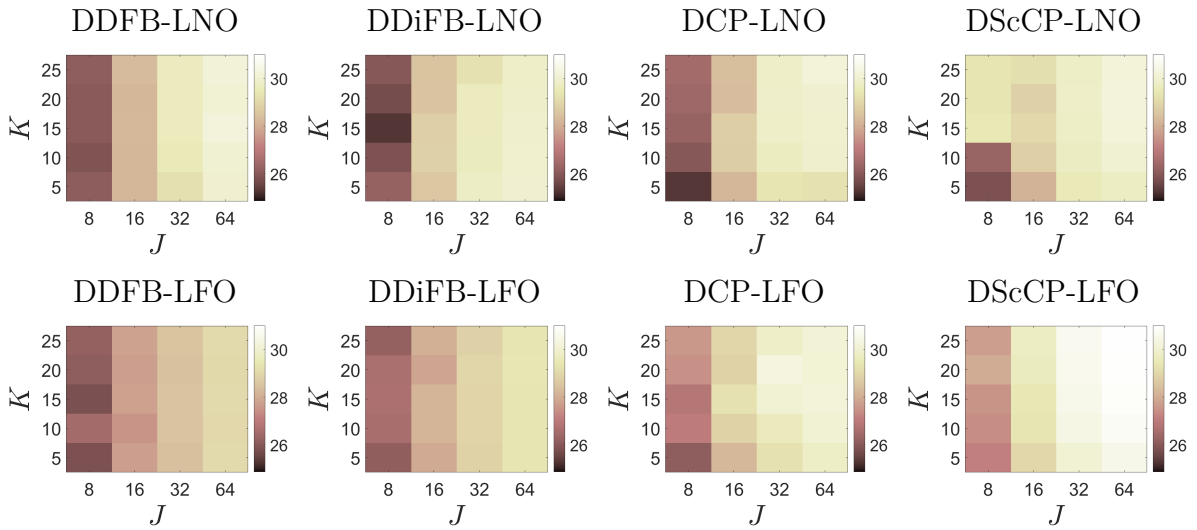


Figure 6.5: Training Setting 1: Denoising performance. Average PSNR obtained with the proposed DUNNs, on 100 images from BSD500 degraded with noise level $\delta = 0.08$. Results are shown for $J \in \{8, 16, 32, 64\}$ and $K \in \{5, 10, 15, 20, 25\}$. **Top row:** LNO settings. **Bottom row:** LFO settings.

Training Setting 2 – Variable noise level: We evaluate the denoising performances of the proposed unfolded NNs on images degraded by a Gaussian noise with standard deviation $\delta = 0.05$. Figure 6.6 gives the PSNR values obtained from the proposed unfolded NNs, when applied to a random subset of $|\mathbb{J}| = 20$ images of BSDS500

validation set. Furthermore, Table 6.3 presents the average PSNR values computed for a separate subset of $|\mathcal{J}| = 100$ images from the BSDS500 validation set. In this table, we further give the average PSNR value for DRUnet. These results show that for all unfolding strategy, the LNO learning strategy improves the denoising performance over LFO. We further observe that D(Sc)CP outperforms DD(i)FB. DRUnet however outperforms the unfolded NNs on this experiments for PSNR values. For completeness, we give examples of a denoised image in Figure 6.7 obtained with DRUnet, DDFB and DScCP. On visual inspection, results from DDFB may appear still slightly noisy compared with DRUnet and DScCP. However results from DRUnet and DScCP are very comparable, and DScCP might reproduce slightly better some textures (e.g., the water around the castle is slightly over-smoothed with DRUnet).

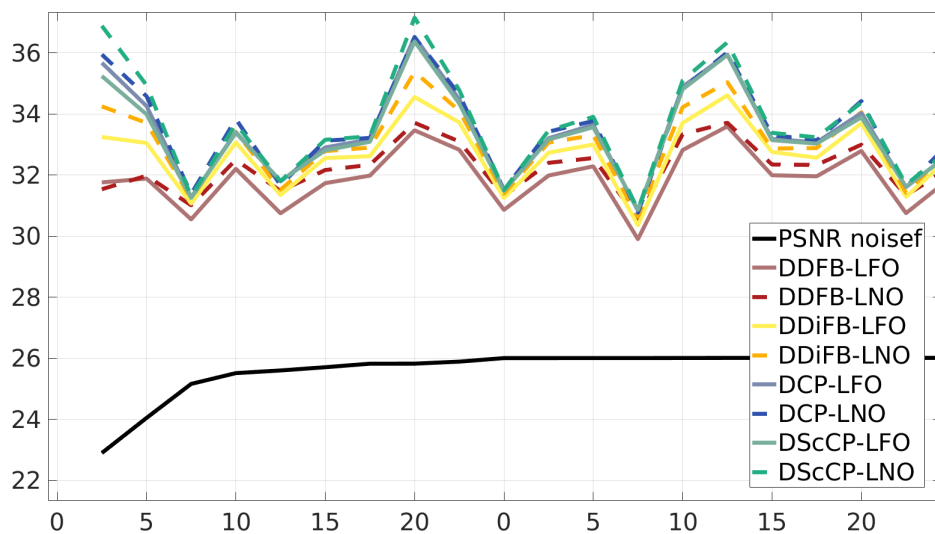


Figure 6.6: Training Setting 2: Denoising performance. PSNR values obtained with the proposed DUNNs (with $(K, J) = (20, 64)$), for 20 images of BSDS500 validation set, degraded with noise level $\delta = 0.05$.

Table 6.3: Training Setting 2: Denoising performance. Average PSNR (and standard deviation) values (in dB) obtained with the proposed PNNs with $(K, J) = (20, 64)$ and with DRUnet, for 100 noisy images of BSDS500 validation set ($\delta = 0.05$, input PSNR= 25.94dB).

DRUnet	DDFB	DDiFB	DCP	DScCP
	LNO			
34.7 ± 1.89	32.42 ± 0.86	33.12 ± 1.27	33.53 ± 1.46	33.57 ± 1.53
	LFO			
	32.09 ± 0.91	32.84 ± 1.12	33.32 ± 1.35	33.27 ± 1.32

6.3.5 Denoising performance versus robustness

In this section, we assess the denoising performances of the proposed DUNNs and DRUnet when evaluated on denoising tasks that are different from those used during

the training process. All networks have been trained as Gaussian denoisers. And we have seen in the previous section that DRUnet, with nearly 100 times more parameters than the proposed DUNNs, demonstrates impressive performance in terms of PSNR for Gaussian denoising. However, we also shown that DRUnet has a higher Lipschitz constant than the proposed DUNNs. Hence we aim to evaluate the performances of the networks when denoising Laplace and Poisson noises (i.e., on different noise settings than the training Gaussian setting).

The evaluation includes both visual and quantitative analysis, presented in Figures 6.8 and 6.9 and Tables 6.4 and 6.5, respectively. Remarkably, the proposed DUNNs lead to higher denoising performances in these scenarios, validating their higher robustness compared to DRUnet.

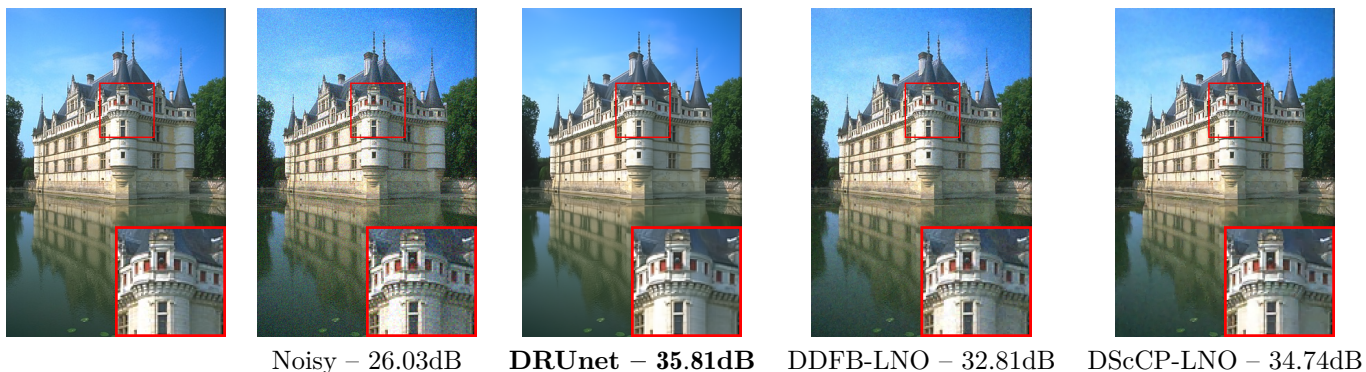


Figure 6.7: Training Setting 2: Denoising performance on Gaussian noise. Example of denoised images (and PSNR values) for Gaussian noise $\delta = 0.05$ obtained with DRUnet and the proposed DDFB-LNO and DScCP-LNO, with $(K, J) = (20, 64)$.

Table 6.4: Training Setting 2: Denoising performance on Poisson noise. Average PSNR (and standard deviation) values (in dB) obtained with the proposed DUNNs with $(K, J) = (20, 64)$ and with DRUnet, for 12 noisy images of BSDS500 validation set degraded with Poisson noise level 50/255.

DRUnet	DDFB	DDiFB	DCP	DScCP
21.59 ± 1.69	LNO			
	23.83 ± 1.80	24.97 ± 1.49	24.57 ± 1.68	25.44 ± 1.25
	LFO			
	23.62 ± 1.70	23.83 ± 1.71	24.74 ± 1.56	24.43 ± 1.63

Table 6.5: Training Setting 2: Denoising performance on Laplace noise. Average PSNR (and standard deviation) values (in dB) obtained with the proposed DUNNs with $(K, J) = (20, 64)$ and with DRUnet, for 12 noisy images of BSDS500 validation set degraded with Laplacian noise level 0.05.

DRUnet	DDFB	DDiFB	DCP	DScCP
19.55 ± 0.83	LNO			
	26.26 ± 0.51	27.73 ± 0.48	27.74 ± 0.47	28.42 ± 1.07
	LFO			
	26.04 ± 0.43	26.43 ± 0.39	27.49 ± 0.54	27.20 ± 0.50

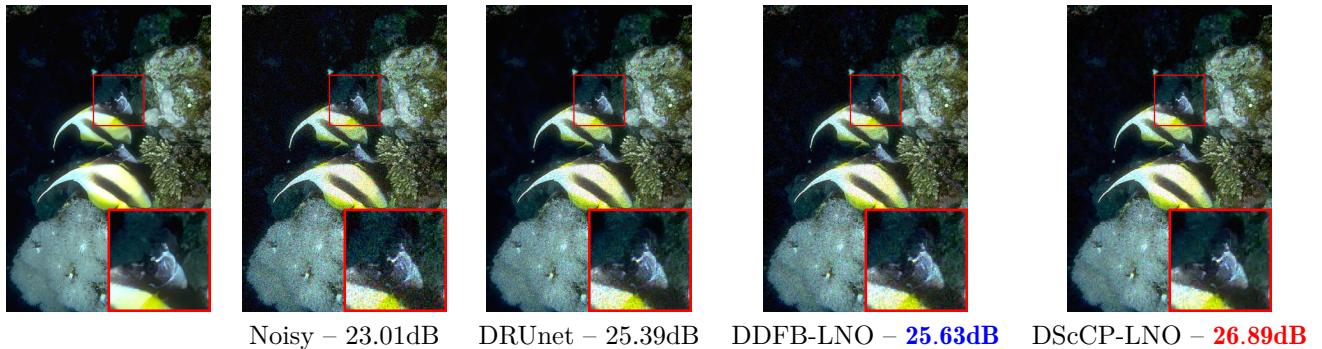


Figure 6.8: Training Setting 2: Denoising performance on Poisson noise. Example of denoised images (and PSNR values) for Poisson noise level 50/255 obtained with DRU-net and the proposed DDFB-LNO and DScCP-LNO, with $(K, J) = (20, 64)$.

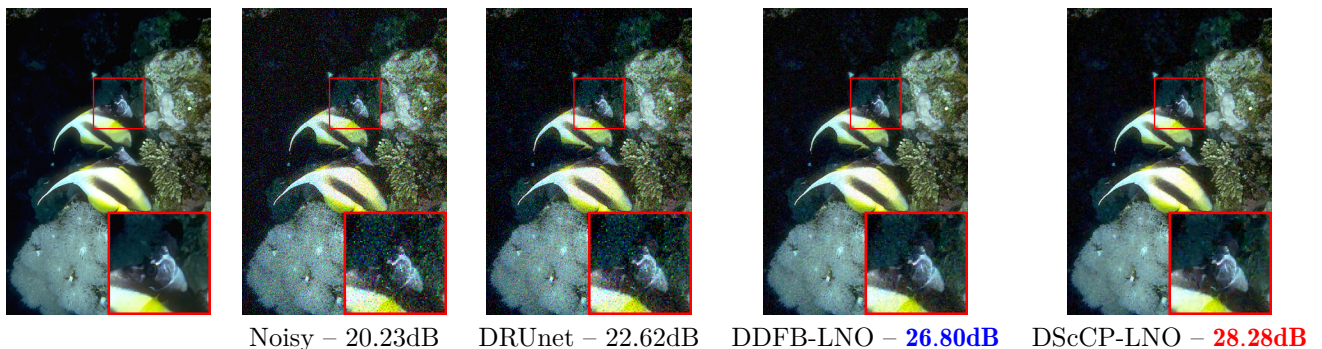


Figure 6.9: Training Setting 2: Denoising performance on Laplace noise. Example of denoised images (and PSNR values) for Laplace noise level 0.05 obtained with DRU-net and the proposed DDFB-LNO and DScCP-LNO, with $(K, J) = (20, 64)$.

6.4 Image restoration with Plug-and-Play method

6.4.1 Restoration problem

Another measure to assess the proposed unfolded NNs is to use them in a plug-and-play framework, for image deblurring. In this context, the objective is to find an estimate $\hat{\mathbf{x}} \in \mathbb{R}^{CN}$ of an original unknown image $\bar{\mathbf{x}} \in \mathbb{R}^{CN}$, from degraded measurements $\mathbf{y} \in \mathbb{R}^{CM}$ obtained through

$$\mathbf{y} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{n}, \quad (6.20)$$

where $\mathbf{A} : \mathbb{R}^{CN} \rightarrow \mathbb{R}^{CM}$ is a linear blurring operator, and $\mathbf{n} \in \mathbb{R}^{CM}$ models an additive white Gaussian noise, with standard deviation $\sigma > 0$. A common method to solve this inverse problem is then to find the MAP estimate $\hat{\mathbf{x}}$ of $\bar{\mathbf{x}}$, defined as a minimizer of a penalized least-squares objective function. A general formulation is given by

$$\text{find } \hat{\mathbf{x}} \in \underset{\mathbf{x} \in \mathcal{S}}{\text{Argmin}} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 + \lambda\Phi(\mathbf{D}\mathbf{x}), \quad (6.21)$$

where $S \subset \mathbb{R}^{CN}$, $\mathbf{D}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{JN}$ and $\Phi: \mathbb{R}^{JN} \rightarrow (-\infty, +\infty]$, and $\lambda \propto \sigma^2$ (i.e., there exists $\beta > 0$ such that $\lambda = (\beta\sigma)^2$) is a regularization parameter.

6.4.2 PnP-FB algorithm

The idea of PnP algorithms is to replace the penalization term (often handled by a proximity operator) by a powerful denoiser. There are multiple choices of denoisers, that can be classified into two main categories: hand-crafted denoisers (e.g. BM3D [Dabov et al., 2007]) and learning-based denoisers (e.g., DnCNN [Zhang et al., 2017] and UNet [Ronneberger et al., 2015]). PnP methods with NNs have recently been extensively studied in the literature, and widely used for image restoration (see, e.g., [Pesquet et al., 2021, Kamilov et al., 2023, Hurault et al., 2021, Repetti et al., 2022]).

In this section, as proposed in [Repetti et al., 2022], we plug the proposed unfolded NNs in a FB algorithm to solve (6.20). The objective is to further assess the robustness of the proposed unfolded strategies. Following the approach proposed in [Repetti et al., 2022], the PnP-FB algorithm is given by

$$\begin{aligned} & \text{Let } \mathbf{x}^{[0]} \in \mathbb{R}^{CN}, \mathbf{u}_0 \in \mathbb{R}^{JN} \\ & \text{For } t = 0, 1, \dots \\ & \left[\begin{array}{l} \mathbf{z}^{[t]} = \mathbf{x}^{[t]} - \gamma \mathbf{A}^\top (\mathbf{A} \mathbf{x}^{[t]} - \mathbf{y}), \\ (\mathbf{x}^{[t+1]}, \mathbf{u}^{[t+1]}) = f_{\mathbf{z}^{[t]}, \lambda \gamma, \Theta}^K(\mathbf{z}^{[t]}, \mathbf{u}^{[t]}), \end{array} \right. \end{aligned} \quad (6.22)$$

where, for every $t \in \mathbb{N}$, $f_{\mathbf{z}^{[t]}, \lambda \gamma, \Theta}^K$ is either DD(i)FB or D(Sc)CP. In Algorithm (6.22), parameters (λ, γ) are given as inputs of $f_{\mathbf{z}^{[t]}, \lambda \gamma, \Theta}^K$. Precisely, the regularization parameter ν for the denoising problem (6.1) is chosen to be the product between the regularization parameter λ for the restoration problem (6.21) and the stepsize of the algorithm γ , i.e., $\nu = \lambda \gamma$.

The following result is a direct consequence of Corollary 1 combined with convergence results of the FB algorithm [Combettes and Wajs, 2005].

Theorem 6.4.1. *Let $(\mathbf{x}^{[t]})_{t \in \mathbb{N}}$ be a sequence generated by (6.22), with $f_{\mathbf{z}^{[t]}, \lambda \gamma, \Theta}^K$ being D(i)FB-LNO or D(Sc)CP-LNO. Assume that $\gamma \in (0, 2/\|\mathbf{A}\|_{sp}^2)$ and that, for every $k \in \{1, \dots, K\}$, $\mathbf{D}_{k, \mathcal{D}} = \mathbf{D}$ and $\mathbf{D}_{k, \mathcal{P}} = \mathbf{D}^\top$ for $\mathbf{D}: \mathbb{R}^{CN} \rightarrow \mathbb{R}^{JN}$. Under the same conditions as Corollary 1, if $K \rightarrow \infty$, then $(\mathbf{x}^{[t]})_{t \in \mathbb{N}}$ converges to a solution \mathbf{x}^* to problem (6.21), and*

$$(\forall t \in \mathbb{N}^*) \quad \|\mathbf{x}^{[t+1]} - \mathbf{x}^{[t]}\| \leq \|\mathbf{x}^{[t]} - \mathbf{x}^{[t-1]}\|. \quad (6.23)$$

A few comments can be made on Theorem 6.4.1. First, [Repetti et al., 2022, Prop. 2] is a particular case of Theorem 6.4.1 for DDFB-LNO. Second, in practice, only a fixed number of layers K are used in the unfolded NNs, although Theorem 6.4.1 holds for $K \rightarrow +\infty$. In [Repetti et al., 2022] the authors studied the behavior of (6.22), using DDFB-LNO. They empirically emphasized that using warm restart for the network (i.e., using both primal and dual outputs from the network of the previous iteration) could add robustness to the PnP-FB algorithm, even when the number of layers K is fixed, due to the monotonic behaviour of the FB algorithm on the dual variable.

In the remainder of the section, we will focus on unfolded NNs trained using **Training setting 2** described in Section 6.3.1 (i.e., with variable noise level) to better fit the noise level of the inverse problem (6.20).

6.4.3 Robustness comparison

In the context of PnP methods, robustness can be measured in terms of convergence of the global algorithm. In this context, it is well known that the PnP-FB algorithm converges if the NN is firmly non-expansive (see, e.g., [Pesquet et al., 2021] for details). According to [Pesquet et al., 2021, Prop. 2.1], an operator f_{Θ} is firmly non-expansive if and only if $h_{\Theta} = 2f_{\Theta} - \text{Id}$ is a 1-Lipschitz operator, i.e., $\chi_h = \max_{\mathbf{z}} \|J h_{\Theta}(\mathbf{z})\|_S < 1$. In the same paper, the authors used this result to develop a training strategy to obtain firmly non-expansive NNs.

Here we propose to use this result as a measure of robustness of the proposed unfolded NNs. Similarly to Section 6.3.3, we approximate χ_h by computing $\chi_h \approx \max_{s \in \mathbb{J}} \|J h_{\Theta}(\mathbf{z}_s)\|_{s,p}$, where \mathbb{J} corresponds to 100 images randomly selected from BSD500 validation set, and \mathbf{z}_s are noisy images with standard deviation uniformly distributed in $[0, 0.01]$. Then, the closer this value is to 1, and the closer the associated NN f_{Θ} is to be firmly non-expansive. Figure 6.10 gives the box plots showing the distribution of $(\|J h_{\Theta}(\mathbf{z}_s)\|_S)_{s \in \mathbb{J}}$. Conclusions on these results are very similar to those in Figure 6.4, in particular that DDFB-LNO and DScCP-LNO have the smallest χ_h values.

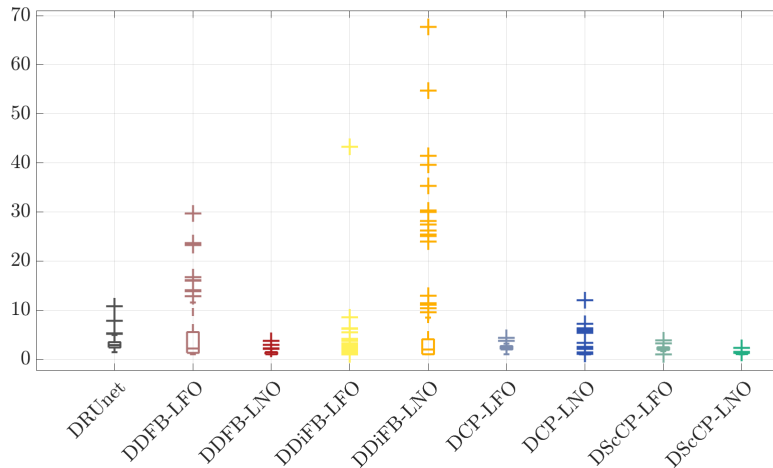


Figure 6.10: Deblurring (Training Setting 2): Robustness. Distribution of $\|J h_{\Theta}(\mathbf{z}_s)\|_S$, where $h_{\Theta} = 2f_{\Theta} - \text{Id}$ for 100 images extracted from BSDS500 validation dataset \mathbb{J} , for the proposed DUNNs and DRUnet.

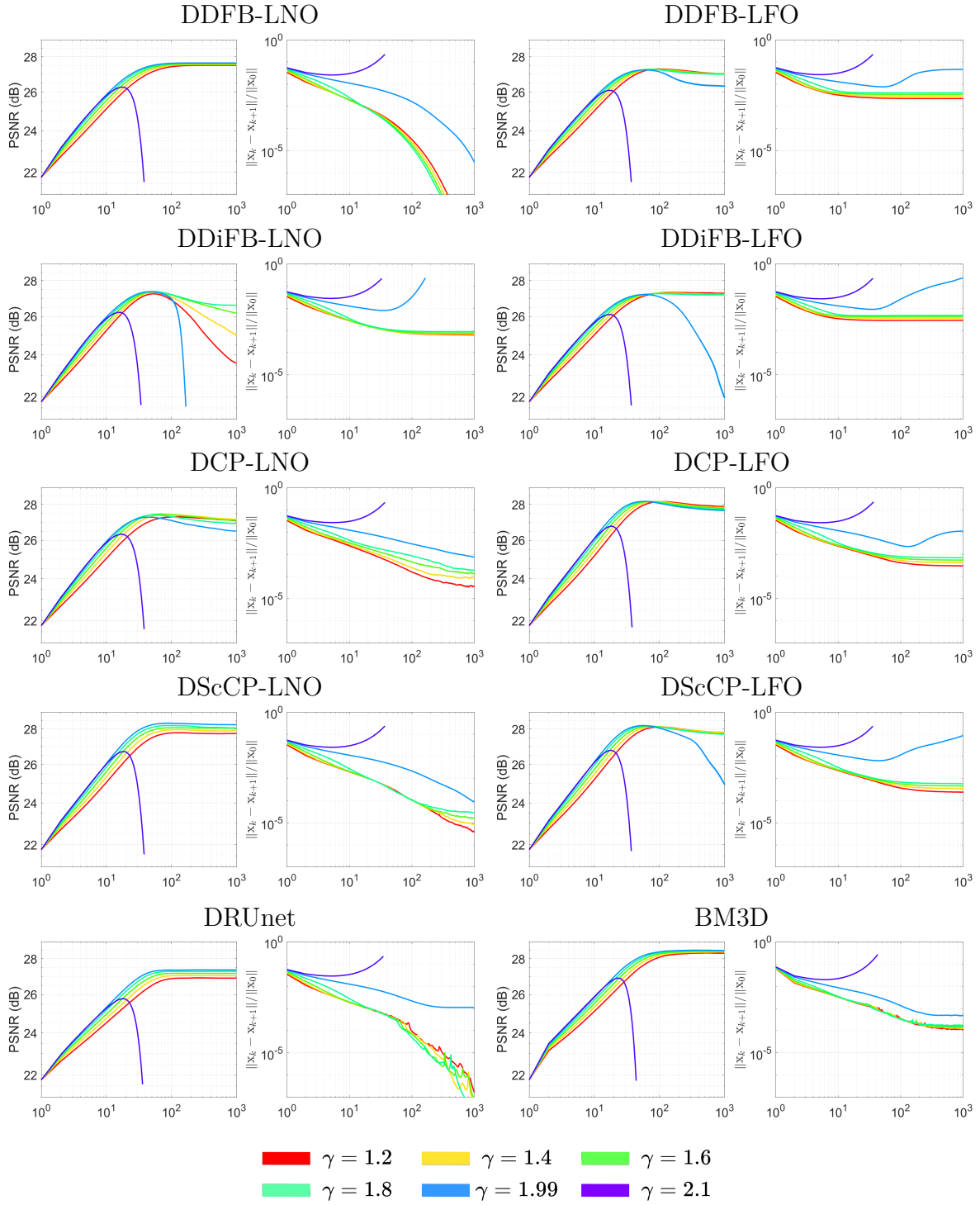


Figure 6.11: Deblurring (Training Setting 2): Parameter choice (γ). Convergence behavior (PSNR and relative error norm) of the PnP-FB algorithms for fixed $\beta = 1$, and varying $\gamma \in \{1.2, 1.4, 1.6, 1.8, 1.99, 2.1\}$.

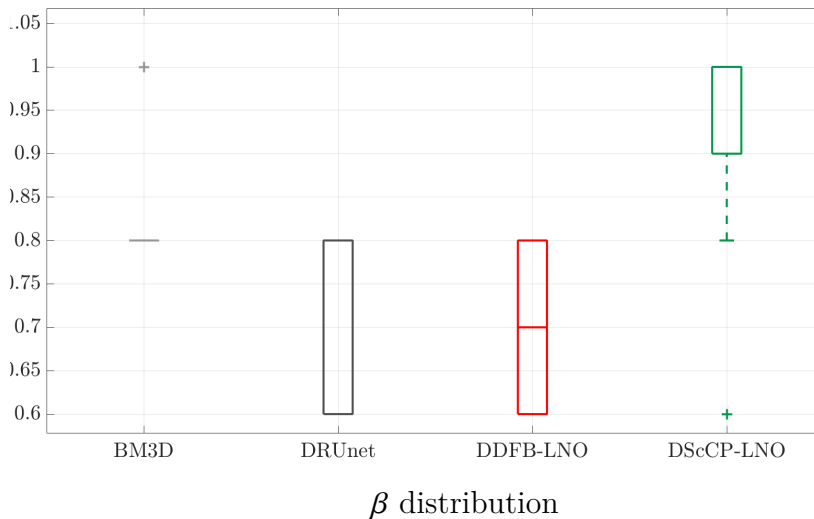
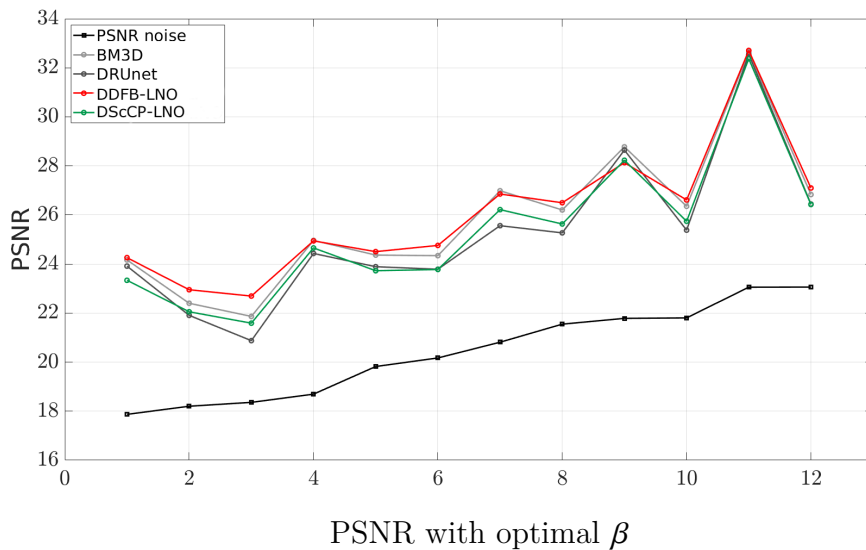


Figure 6.12: Deblurring (Training Setting 2): Parameter choice (β). **Top:** Best PSNR values obtained with DDFB-LNO, DScCP-LNO, DRUnet and BM3D, on 12 images from BSDS500 validation set degraded according to (6.20), with $\sigma = 0.03$. **Bottom:** Box plots for the distribution of β associated with the best PSNR values for DDFB-LNO, DScCP-LNO, DRUnet and BM3D.

Parameter choices

As emphasized previously, the denoising NNs in PnP-FB algorithm (6.22) depend on two parameters: the step-size γ and the regularization parameter $\lambda = \beta^2 \sigma^2$. In this section we investigate the impact of each of these parameters on the results.

Impact of γ . To ensure the convergence of the PnP-FB algorithm, the stepsize must satisfy $\gamma \in]0, 2/\|\mathbf{A}\|_{sp}^2[$. Figure 6.11 aims to evaluate the stability of the proposed unfolded NNs by looking at the convergence of the associated PnP-FB iterations, varying $\gamma \in \{1.2, 1.4, 1.6, 1.8, 1.99, 2.1\}$ (where $\|\mathbf{A}\|_{sp} = 1$). For this experiment, we fix $\lambda = \sigma^2$ (i.e., $\beta = 1$). Further, we compare the PnP iterations with the proposed unfolded NNs to PnP methods with DRUnet and BM3D. The plots show

the convergence profiles for the deblurring of one image in terms of PSNR values and relative error norm of consecutive iterates $\|\mathbf{x}^{[t+1]} - \mathbf{x}^{[t]}\|/\|\mathbf{x}^{[0]}\|$, with respect to the iterations $t \in \{1, \dots, 10^3\}$. In particular, theory ensures that $(\|\mathbf{x}^{[t+1]} - \mathbf{x}^{[t]}\|)$ should decrease monotonically. Interestingly, we can draw similar conclusions from the curves in Figure 6.11 as for Figures 6.10 and 6.4. Looking at the relative error norms, the most robust NNs are DDFB-LNO and DScCP-LNO, both converging monotonically for any choice of $\gamma \leq 1.99$. DCP-LNO seems to have similar convergence profile, with a slower convergence rate. None of the other PnP schemes seems to be stable for $\gamma = 1.99$. For $\gamma \leq 1.8$, DRUnet shows an interesting convergence profile, however the error norm is not decreasing monotonically. The remaining PnP schemes do not seem to converge in iterates, as the error norms seem to reach a plateau. In terms of PSNR values, DScCP-LNO and BM3D seem to have the best performances, followed by DDFB-LNO and DRUnet. For these four schemes, $\gamma = 1.99$ leads to the best PSNR values.

Impact of β . The regularization parameter $\lambda = \beta^2 \sigma^2$ aims to balance the data-fidelity term and the NN denoising power [Repetti et al., 2022]. The proposed unfolded NNs take as an input a parameter $\nu > 0$ that has similar interpretation for the denoising problem (3.45). In the PnP algorithm, we have $\nu = \lambda\gamma$, hence $\delta^2 = \beta^2 \sigma^2 \gamma$, where δ^2 is the training noise level, σ^2 is the noise level of the deblurring problem (6.20). Then β^2 allows for flexibility in the choice of λ , to possibly improve the reconstruction quality.

In Figure 6.12, we provide best PSNR values obtained with DDFB-LNO, DScCP-LNO, DRUnet and BM3D, on 12 images from BSDS500 validation set degraded according to (6.20), with $\sigma = 0.03$. In this figure we also give the distribution of β , associated with the best PSNR values.

6.4.4 Restoration performance comparison

In this section, we perform further comparisons of the different PnP schemes on 12 random images selected from BSDS500 validation set, degraded as per model (6.20). In particular, we will run experiments for three different noise levels $\sigma \in \{0.015, 0.03, 0.05\}$. Since in the previous sections we observed that DDFB-LNO and DScCP-LNO are the more robust unfolded strategies, in this section we only focus on these two schemes, comparing them to BM3D and DRUnet. For each denoiser, we choose $\delta = \lambda\gamma$, with $\gamma = 1.99$ and $\lambda = \beta^2 \sigma^2$, with β chosen according to results displayed in Figure 6.12.

The averaged PSNR values obtained with the four different schemes are given in Table 6.6. It can be observed that, regardless the noise level σ , DDFB-LNO and BM3D always have the highest PSNR values, outperforming DRUnet and DScCP-LNO. However, DDFB-LNO has a much cheaper computation time than BM3D (see Table 6.2 for details). For low noise level $\sigma = 0.015$, DScCP-LNO also outperforms DRUnet. For highest noise levels $\sigma \in \{0.03, 0.05\}$, DScCP-LNO and DRUnet have similar performances.

For visual inspection, we also provide in Figures 6.14, and 6.15 examples of three different images, for noise levels $\sigma = 0.015$, $\sigma = 0.03$ and $\sigma = 0.05$, respectively. We observe that DRUnet and DScCP-LNO tend to better eliminate noise, compared to BM3D and DDFB-LNO. This is consistent with the denoising performance results

observed in Section 6.3.4. However, when integrated in a PnP for the restoration problem, DRUnet seems to smooth out high-frequency details, sometimes resembling an AI generator and generate unrealistic patterns. On the contrary, DDFB-LLNO retains more high-frequency details. DScCP-LNO produces images much smoother than DDFB-LNO, but without unrealistic patterns. Hence, if an image contains a significant amount of high-frequency details compared to piecewise smooth or piecewise constant patterns, unfolded NN denoisers like DScCP-LNO or DDFB-LNO are more suitable choices than DRUnet.

Table 6.6: Deblurring (Training Setting 2): Restoration performance. Average PSNR values over 12 images from BSDS500 validation set, obtained with different PnP-FB schemes. For each NN, β was chosen to obtain the highest PSNR.

σ	Noisy	BM3D	DRUnet	DDFB-LNO	DScCP-LNO
0.015	20.80	28.33	26.47	28.16	27.81
0.03	20.43	25.82	25.14	26.00	25.31
0.05	19.68	24.27	23.98	24.37	23.87

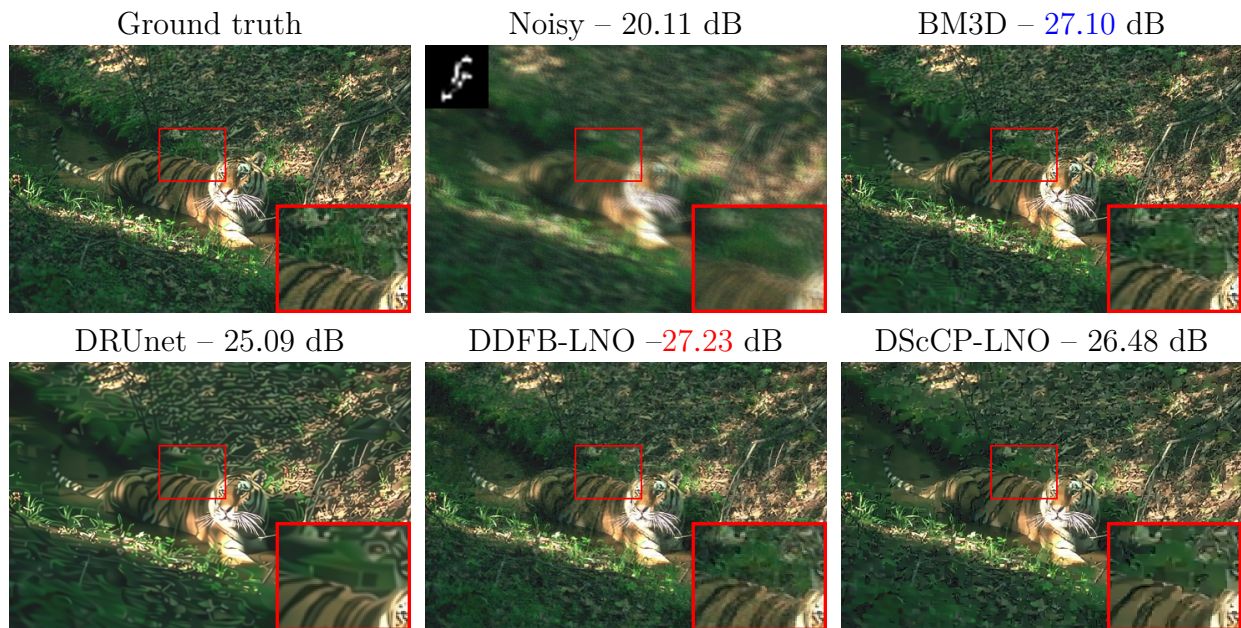


Figure 6.13: Deblurring (Training Setting 2): Restoration performance. Restoration example for $\sigma = 0.015$ with parameters $\gamma = 1.99$ and β chosen optimally for each scheme.

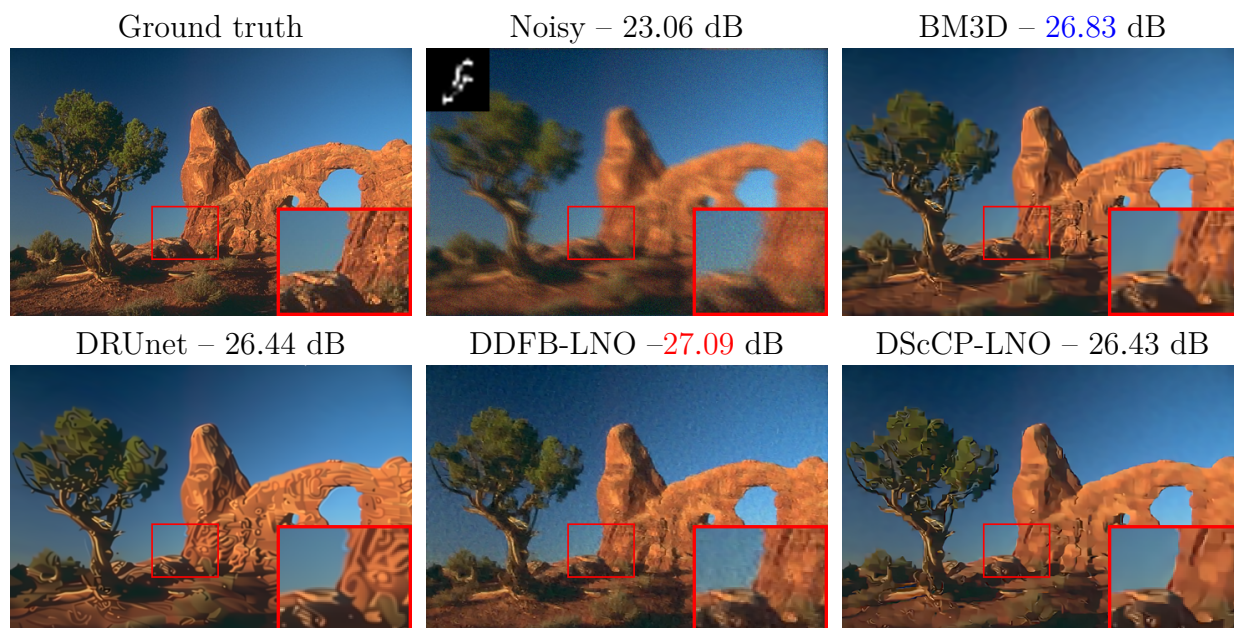


Figure 6.14: Deblurring (Training Setting 2): Restoration performance. Restoration example for $\sigma = 0.03$ with parameters $\gamma = 1.99$ and β chosen optimally for each scheme.

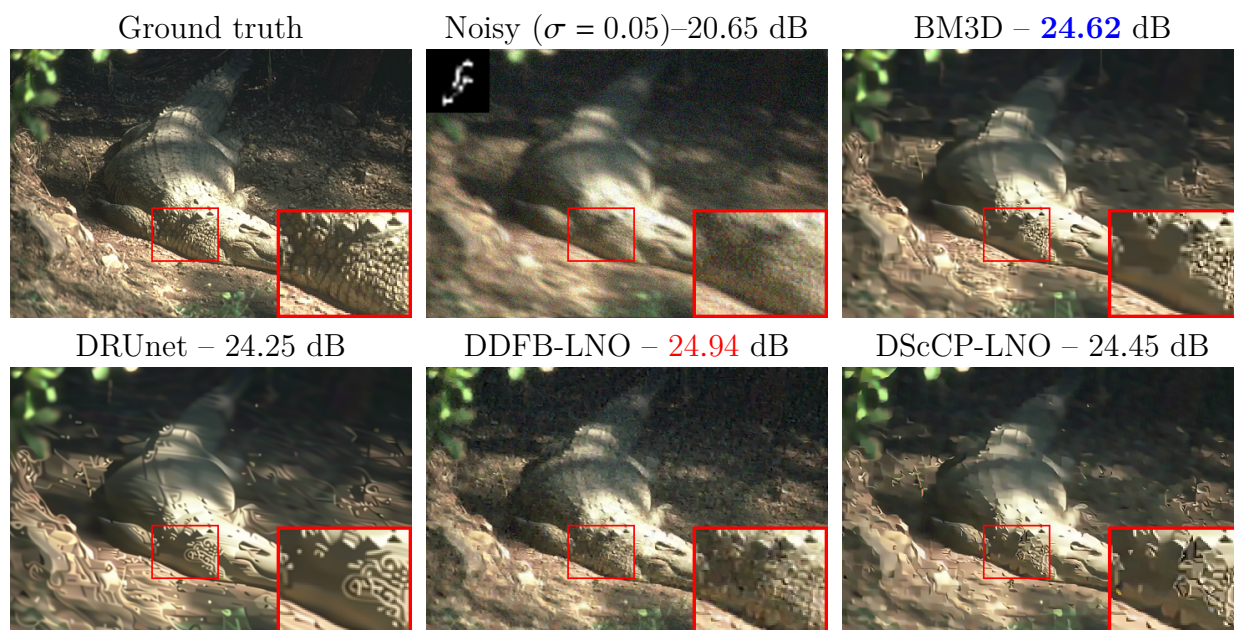


Figure 6.15: Deblurring (Training Setting 2): Restoration performance. Restoration example for $\sigma = 0.05$, with parameters $\gamma = 1.99$ and β chosen optimally for each scheme.

6.5 Application in Piecewise Homogeneous Fractal Image Analysis

In this section, we want to evaluate the performance of the proposed DScCP for the challenging question of estimating local regularity in textured images allowing to identify texture changes in an image.

6.5.1 State-of-the-Art.

Textured images serve as natural representations suitable for a wide range of diverse real-world applications. Frequently, these applications utilize fractal characteristics to effectively describe such textures found in fields such as biological tissues, pathology diagnosis through tomography, art painting analysis, microfluidics, among others. In many of these practical contexts, the task of texture segmentation, which involves dividing images into regions with similar characteristics, remains an ongoing and significant challenge. In the realm of computer vision and scene analysis, there are numerous well-established and efficient methods for image segmentation that primarily rely on the geometric properties of the image. However, when dealing with textured images, segmentation becomes more intricate, as capturing geometric attributes becomes more challenging, and the focus shifts primarily to the statistical properties of the texture features. Typically, traditional approaches to texture segmentation follow a two-step procedure. First, they involve the computation of pre-selected texture features, and second, they group these features into regions with similar statistical characteristics.

To improve this local estimation, one approach involves combining the regression procedure of local multiscale quantities against scales, minimizing least squares, with the introduction of priors on the spatial behavior of these descriptors (*e.g.*, piecewise constancy or linearity), leading to the minimization of a strongly convex non-smooth criterion, effectively solved using proximal algorithms [Nafornita et al., 2014, Pustelnik et al., 2016a]. These can be combined with a SUGAR-type procedure for regularization hyperparameter estimation [Deledalle et al., 2014, Pascal et al., 2021b]. Other approaches rely on Bayesian regularization strategies [Wendt et al., 2018]. It has also been considered to create a learning database close to the a priori model and then learn a deep neural network, referred to as a "black-box," with no guidance from the model's knowledge [Pascal et al., 2021a].

For the segmentation of textured images, assuming that textures are characterized solely by their small-scale statistical properties, a comparison between variational approaches, based on local regularity [Pascal et al., 2020], and deep learning, constructed from the Fully Convolutional Neural Network (FCNN) network, was conducted in [Pascal et al., 2021a]. It highlights that the supervised FCNN network compares favorably to a more classical unsupervised variational approach, but with less robustness and lower-quality interface detection.

In this part, we propose to explore PNN for the task of texture segmentation considering a two-step procedure where fractal descriptor (local regularity) are extracted and then PNN is built to perform denoising/segmentation task.

In this study, we propose to focus on the task of texture analysis through the estimation of local regularity when it is piecewise constant across an image. We propose to adapt two unrolled deep networks that we have recently developed in the context of image denoising [Le et al., 2022c] to the estimation of local regularity. The two proposed networks are DFH (Deep Fista Hloc) and DSH (Deep Strong Convexity Chambolle-Pock Hloc). These networks are based on an unrolled version of the FISTA and Chambolle-Pock algorithms with acceleration strategies which also have similar architecture with DDiFB and DScCP presented in previous section. The performance achieved with these networks will be compared to a competitive black-box deep neural network approach. We also compare the performance with results obtained using a more standard unsupervised variational approach.

6.5.2 Local Estimation of Self-Similarity

Local regularity. *Local regularity* is a mathematical quantity that can be estimated through regression of multiscale quantities (e.g., the logarithm of the absolute values of wavelet coefficients or dominant coefficients [Wendt et al., 2007, Pustelnik et al., 2016a, Pascal et al., 2021b]). Local regularity around position n is measured by the so-called Hölder exponent h_n , such that if h_n is close to 0 (resp. 1), the image is locally very irregular (resp. locally smooth). Formally, if we denote $\bar{\mathbf{x}} = (\bar{x}_n)_{1 \leq n \leq N} \in \mathbb{R}^N$ the image we aim to analyze, and if we denote $\mathcal{L}_{j,n}$ a multiscale quantity associated with the image $\bar{\mathbf{x}}$ defined for scales $j \in \{1, \dots, J\}$ and position n , the local regularity is related to this quantity by the following relation:

$$\mathcal{L}_{j,n} \simeq \eta_n 2^{j h_n}, \quad (6.24)$$

as $2^j \rightarrow 0$, where η_n is proportional to the local variance of $\bar{\mathbf{x}}$ at position n . The local estimation of h_n can be obtained through linear regression across scales:

$$\widehat{h}_n^{(\text{RL})} = \sum_j w_{j,n} \log_2 \mathcal{L}_{j,n} \quad (6.25)$$

where $(w_{j,n})_{j,n}$ model the regression weights at scale j and position n . Unbiased estimation occurs when $\sum_j w_{j,n} \equiv 0$ and $\sum_j j w_{j,n} \equiv 1$ [Wendt et al., 2007].

Estimation with piecewise constant spatial prior In studies dedicated to local regularity estimation, it is customary to consider a homogeneous field. The estimation is then obtained by averaging the values of the estimates $\widehat{h}_n^{(\text{RL})}$ obtained at each position n in the image. To achieve local estimation while reducing estimation variance, [Naornita et al., 2014, Pustelnik et al., 2016b] proposed imposing a piecewise constant spatial prior, leading to the following minimization problem:

$$\widehat{\mathbf{h}}^{(\text{TV})} = \arg \min_{\mathbf{h}} \frac{1}{2} \left\| \mathbf{h} - \widehat{\mathbf{h}}^{(\text{RL})} \right\|_2^2 + \nu \Phi(\mathbf{D}\mathbf{h}). \quad (6.26)$$

where \mathbf{D} models the finite difference operator and Φ models a norm imposing a sparsity prior, usually $\Phi = \|\cdot\|_{1,2}$ for total variation-type penalization. This is a strongly convex non-smooth optimization problem, efficiently solved to estimate $\widehat{\mathbf{h}}^{(\text{TV})}$ using proximal approaches [Combettes and Pesquet, 2011b, Bauschke and Combettes, 2017b].

Objectives – Our study focuses on two fast algorithmic schemes (convergence in $O(1/k^2)$ on the functional) presented in [Pascal et al., 2018] for local regularity analysis: the Fast Iterative Soft Thresholding Algorithm (FISTA) and Chambolle Pock with strong convexity (ScCP). A sequence of K iterations of these algorithmic schemes is respectively denoted as

- $f^{\text{FISTA}}(\mathbf{z}; \mathbf{D}, \mathbf{D}^\top, \nu, \|\cdot\|_\bullet, \tau, \alpha, K)$
- and
- $f^{\text{ScCP}}(\mathbf{z}; \mathbf{D}, \mathbf{D}^\top, \nu, \|\cdot\|_\bullet, \tau, \sigma, \alpha, K)$

highlighting the dependence of iterations on the problem parameters in (6.26) and the descent steps of the algorithmic strategies (τ, α for FISTA and τ, σ, α for ScCP).

6.5.3 Experiments

Learning strategy – Starting from a training set $(\bar{\mathbf{x}}_s, \bar{h}_s)_{s \in \mathbb{I}}$ where $\bar{\mathbf{x}}_s$ represents the image to analyze providing $\widehat{h}_s^{(\text{RL})}$ obtained by linear regression (6.25) from the multiscale transform of the textured image \bar{h}_s , our goal is to learn the parameters Θ of a network f_Θ in order to minimize the empirical error:

$$\text{minimize } \mathcal{L}(\bar{h}_s, \widehat{\mathbf{x}}_s; \Theta) := \frac{1}{\mathbb{I}} \sum_{s=1}^{\mathbb{I}} \|\bar{h}_s - f_\Theta(\widehat{h}_s^{(\text{RL})})\|_2^2. \quad (6.27)$$

Datasets – To evaluate the performance of DFH and DSH, we create a database of textured images $\bar{\mathbf{x}}_s$ generated from a piecewise homogeneous fractal process described in [Pascal et al., 2021b], allowing the assembly of Q fractal textures. Each texture is a stationary Gaussian field whose covariance structure is fully defined by its variance Σ and its fractal parameter H . An illustration of a generated texture $\bar{\mathbf{x}}_s$ and the associated true local regularity values \bar{h}_s are shown in Figure 6.16. Following the procedure described in [Pascal et al., 2021a], three databases are generated: a training dataset consisting of 2000 images with $Q = 2$ regions, $H_1 = 0.5$, $\Sigma_1^2 = 0.6$, $H_2 = 0.8$, $\Sigma_2^2 = 1.1$ (config I), a test dataset consisting of 100 images with the same parameters, and a test dataset of 100 images with $H_1 = 0.5$, $\Sigma_1^2 = 0.33$, $H_2 = 0.65$, $\Sigma_2^2 = 1$ (config II).

Learning parameters – DFH and DSH are trained using PyTorch with the ADAM optimizer [Kingma and Ba, 2014] for 500 iterations, a batch size of 10, and a learning rate of $1e^{-4}$. Two network configurations are considered, varying the number of filters J and the number of layers, allowing the construction of networks with 5×10^3 coefficients (resp. 3×10^4) corresponding to $K = 13$ and $J = 21$ (resp. $K = 45$ and $J = 37$)

Comparisons – We compare the performance of DFH and DSH having the same structure asDDiFB-LFO and DScCP-LFO with a standard unsupervised variational method that minimizes (6.26) when $\Phi = \|\cdot\|_{1,2}$, and λ is automatically chosen using a SUGAR strategy described in [Pascal et al., 2021b]. We also provide a comparison with the reference method DnCNN [Zhang et al., 2017], which proves

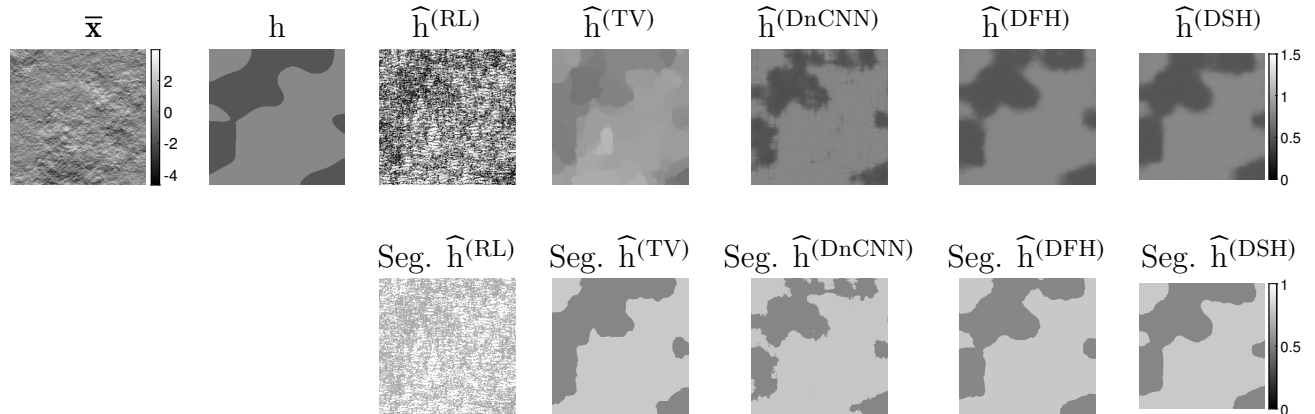


Figure 6.16: Estimation and segmentation results on a texture image from Config I. This configuration involves supervised methods (DnCNN, DFH, and DSH with 3×10^4 weights) trained on Config I and evaluated on Config I. The segmentation is obtained by applying k-means clustering to the estimated values \hat{h} .

	TV-SUGAR	DnCNN 5.10^3	DnCNN 3.10^4	DFH 5.10^3	DFH 3.10^4	DSH 5.10^3	DSH 3.10^4
Estimation error							
Test config I	0.339+/-0.048	0.113+/-0.011	0.104+/-0.010	0.073+/-0.008	0.069+/-0.007	0.072+/-0.007	0.069+/-0.007
Test config II	0.306+/-0.029	0.145+/-0.014	0.130+/-0.012	0.115+/-0.011	0.120+/-0.012	0.116+/-0.012	0.119+/-0.012
Classification score							
Test config I	83.7+/-2.68	81.0+/-1.73	86.2+/-1.29	94.8+/-0.45	95.2+/-0.41	94.8+/-0.42	95.3+/-0.41
Test config II	73.2+/-2.51	68.4+/-1.36	67.9+/-1.45	70.7+/-2.65	70.9+/-2.67	68.3+/-2.74	70.6+/-2.71

Table 6.7: Comparison in terms of normalized error or segmentation accuracy between different methods.

to be competitive with the proposed networks DDiFB-LFO and DScCP-LFO but considered in the context of image denoising [Le et al., 2022c]. The number of layers and parameters in the DnCNN network is chosen to achieve an equivalent total number of parameters as considered with the proposed strategies DFH and DSH (i.e., $K = 9$ or $K = 10$).

Estimation and segmentation performance – Figure 6.17 illustrates the superior performance of our architectures in terms of both training and testing errors compared to the standard DnCNN approach. Moreover, the proposed networks exhibit greater stability (as evidenced by the reduced oscillations in the testing error curve). Table 6.7 (row 1) confirms the improvement of DFH and DSH over an unsupervised TV approach in a scenario where training and testing are performed on a database with the same characteristics (Config I), as well as in comparison to DnCNN. When testing is conducted on a different configuration from the training, the supervised approaches still outperform (see Table 6.7 - row 2). Although the proposed network is designed for estimation tasks, we also evaluate its segmentation performance after applying k-means clustering to the estimates. The results consistently outperform DnCNN, but when the network is evaluated on Config II, the unsupervised procedure yields better results. Illustrations of estimation and segmentation performance are presented in Figures 6.16.

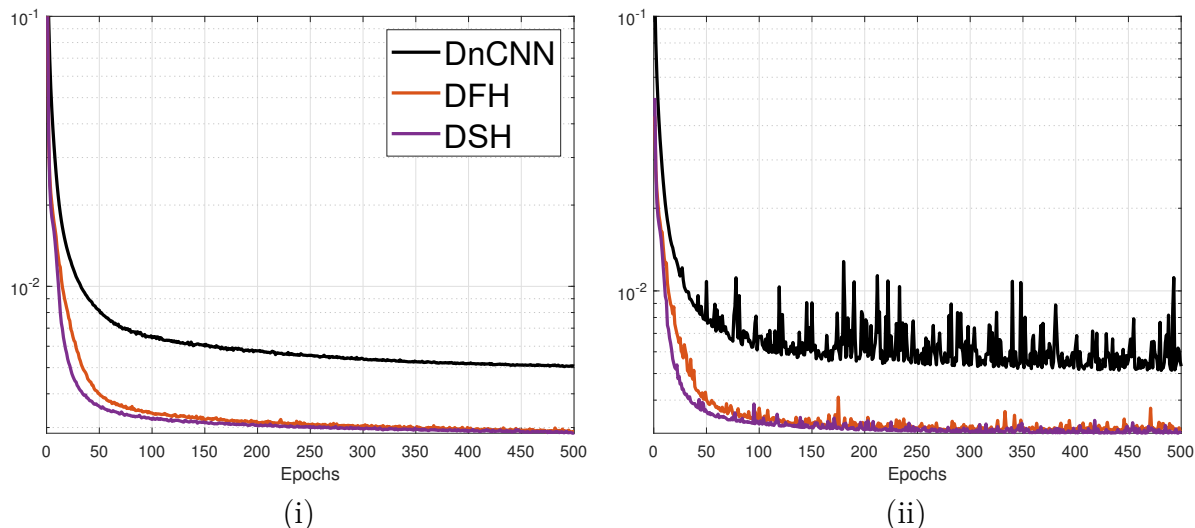


Figure 6.17: (i) Training error as a function of the number of iterations on Config I
(ii) Estimation error as a function of the number of iterations on the test dataset of Config I

6.6 Conclusion

In this chapter we have introduced the concept of proximal unfolded neural networks referring to unfolded schemes where the activation function is a proximity operator. Several PNNs have been proposed with different degrees of flexibility and relying on different algorithmic schemes.

All PNNs provide good numerical performance but some of them appear more stable such as DDFB-LNO and DScCP-LNO. Its good behavior observed for denoising task is also true when plugged into a FB as PnP strategy or for texture segmentation. For all these image processing tasks, PNNs were able to achieve similar or best performances with 100 less parameters.

Proximal neural networks for joint image denoising and edge detection

Summary

7.1	Proposed unrolled DMS with PALM iterations	95
7.2	Edge detector based on DScCP denoiser	97
7.3	Experiments	97
	7.3.1 Training strategy for joint task	97
	7.3.2 Experimental results	98
7.4	Conclusion	101

Traditionally, image restoration and edge detection were treated separately where the former process is needed to enhance image quality and give a better input for the later process which is considered as a post-processing step. As we discussed in Part I, there are many hand-crafted methods that help us to tackle this joint task simultaneously by minimizing an energy function whose estimated solution is the denoised image and estimated edges. In [Blake and Zisserman, 1987], the induced edges can be retrieved easily by using a truncated quadratic function on the gradient of denoised image. In [Le et al., 2022a], both image variable and edges variable are simultaneously estimated, and can both improve the image restoration performance and edge detection as illustrated in chapter 4 with the discrete AT functional. The main drawbacks of the method includes the limitation to handle with large scale data. The method faces an issue of time consumption and necessitates a meticulous tuning of hyperparameters to achieve optimal performance.

In this chapter, we explore unfolded NN strategies in the context of joint restoration and edge detection. We first proposed to unfold the Discrete Mumford-Shah functional where each sub-iteration relies on PALM. Then we tackle this joint task with an approach similar to BZ formalism in which the learning-based model containing two blocks: (i) a PNNs denoiser, (ii) a convolutional layer for edge detection. The efficiency of the proposed unrolled schemes is illustrated on several examples and compared with the state-of-the-art variational methods.

7.1 Proposed unrolled DMS with PALM iterations

Giving the Gaussian degraded model

$$\mathbf{z} = \bar{\mathbf{x}} + \mathbf{n}, \quad (7.1)$$

where $\mathbf{n} \in \mathbb{R}^{CN}$ models an additive white Gaussian noise with standard deviation $\delta > 0$. The objective of this chapter is to design unfolded NNs f_{Θ} such that

$$(\hat{\mathbf{x}}, \hat{\mathbf{e}}) \approx f_{\Theta}(\mathbf{z}), \quad (7.2)$$

where \mathbf{z} is the noisy image, $\hat{\mathbf{x}} \in \mathbb{R}^{CN}$ is an estimate of $\bar{\mathbf{x}}$ and $\hat{\mathbf{e}} \in \mathbb{R}^{JN}$ is an estimate of $\bar{\mathbf{e}}$.

In this section we describe unfolded strategy for building denoising-edge detection NNs relying on PALM iterations presented in Algorithm 10.

Proposition 7.1.1. *Let $\mathbf{z} \in \mathbb{R}^{CN}$, $\mathbf{x} \in \mathbb{R}^{CN}$, $\mathbf{e} \in \mathbb{R}^{JN}$ and $k \in \mathbb{N}$.*

Let $T_{\mathbf{z}, \Theta_{k, \mathcal{E}}, \mathcal{E}} : \mathbb{R}^{CN} \times \mathbb{R}^{JN} \rightarrow \mathbb{R}^{JN}$, defined as

$$T_{\mathbf{z}, \Theta_{k, \mathcal{E}}, \mathcal{E}} = \eta_k(\mathcal{W}_k(\mathbf{x}, \mathbf{e}) + \mathbf{b}_k), \quad (7.3)$$

be a sub-layer acting on both image variable \mathbf{x} and the edge variable \mathbf{e} and returning an edge map. In (7.3) $\eta_k : \mathbb{R}^{JN} \rightarrow \mathbb{R}^{JN}$ is a fixed activation function, $\Theta_{k, \mathcal{E}}$ denotes learnable parameters of the non-linear operator $\mathcal{W}_k : \mathbb{R}^{JN} \times \mathbb{R}^{CN} \rightarrow \mathbb{R}^{JN}$, $\mathbf{b}_{\mathbf{z}, k}$ is the bias.

Let $T_{\mathbf{z}, \Theta_{k, \mathcal{F}}, \mathcal{F}} : \mathbb{R}^{CN} \times \mathbb{R}^{JN} \rightarrow \mathbb{R}^{CN}$, defined as

$$T_{\mathbf{z}, \Theta_{k, \mathcal{F}}, \mathcal{F}} = \tilde{\eta}_k(\tilde{\mathcal{V}}_k(\mathbf{x}, \mathbf{e}) + \tilde{\mathbf{b}}_k), \quad (7.4)$$

be a sub-layer acting on both image variable \mathbf{x} and the edge variable \mathbf{e} and returning image variable. In (7.3) $\tilde{\eta}_k : \mathbb{R}^{CN} \rightarrow \mathbb{R}^{CN}$ is a fixed activation function, $\Theta_{k,\mathcal{F}}$ denotes learnable parameters of the non-linear operator $\mathcal{V}_k : \mathbb{R}^{CN} \times \mathbb{R}^{JN} \rightarrow \mathbb{R}^{CN}$, $\tilde{\mathbf{b}}_k$ is the bias.

Then, the k -th iteration of the joint formulation can be written as a composition of two layers of the form of the feedforward NN:

$$\mathbb{T}_{\mathbf{z},\Theta_k}^{\text{DPMS}} : \mathbb{R}^{CN} \times \mathbb{R}^{JN} \rightarrow \mathbb{R}^{CN} : (\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) \mapsto \mathbb{T}_{\Theta_k,\mathcal{F}} \circ \mathbb{T}_{\Theta_k,\mathcal{E}}(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) \quad (7.5)$$

where Θ_k is the combination of $\Theta_{k,\mathcal{F}}$ and $\Theta_{k,\mathcal{E}}$, i.e., the linear parametrization of all learnable parameters for layer k .

Proof. This result is obtained by noticing that, for every $k \in \mathbb{N}$, the k -th iteration in the PALM algorithm can be rewritten as (7.5), where the image variable \mathcal{F} operators are given by $\mathcal{W}_k = \frac{\mu_k}{\mu_{k+1}} (\text{Id} - 2\mu_k\lambda_S \mathbf{D}^\top ((1 - \mathbf{e})^2 \odot \mathbf{D}\bullet))$, $\mathbf{b}_k = \frac{\mu_k}{\mu_{k+1}} \mathbf{z}$, $\eta_k = \text{Id}$ and the edge \mathcal{E} operators are given by $\mathcal{V}_k = \text{Id} + 2\kappa_k\lambda_S ((\mathbf{D}\mathbf{x})^2 \odot (1 - \bullet))$, $\tilde{\mathbf{b}}_k = 0$ and $\tilde{\eta}_k = \text{prox}_{\kappa_k\lambda_S E}$.

The resulting unfolding Deep PALM on Mumford-Shah functional building block is then given below:

$$f_{\mathbf{z},\Theta}^{K,\text{DPMS}}(\mathbf{x}^{[0]}, \mathbf{e}^{[0]}) = \mathbb{T}_{\mathbf{z},\Theta_K}^{\text{DPMS}} \circ \dots \circ \mathbb{T}_{\mathbf{z},\Theta_1}^{\text{DPMS}}(\mathbf{x}^{[0]}, \mathbf{e}^{[0]}), \quad (7.6)$$

where for every $k \in \{1, \dots, K\}$,

$$\begin{aligned} \mathbf{e}^{[k+1]} &= \mathbb{T}_{\mathbf{z},\Theta_{k,\mathcal{E}}}^{\text{DPMS}}(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}), \\ \mathbf{x}^{[k+1]} &= \mathbb{T}_{\mathbf{z},\Theta_{k,\mathcal{F}}}^{\text{DPMS}}(\mathbf{x}^{[k]}, \mathbf{e}^{[k+1]}), \\ (\mathbf{x}^{[k+1]}, \mathbf{e}^{[k+1]}) &= \mathbb{T}_{\mathbf{z},\Theta_k}^{\text{DPMS}}(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) \end{aligned}$$

From this, we propose two options to unroll PALM iterations:

- **DPMS-LH**: stands for Deep PALM on Mumford-Shah functional with Learned Hyperparameters.

This architecture is nothing else but the PALM iterations for which we fix a number of iterations K and the linear operators \mathbf{D} and \mathbf{D}^\top . The objective is to learn the hyperparameters λ_S and λ_E .

- **DPMS-LNO**: stands for Deep PALM on Mumford-Shah functional with Learned Normalized Operator described as follows:

Our unrolled architecture relies on layer structures introduced in Proposition 7.1.1 where we let the linear operator \mathbf{D} to be different for each layer. In this chapter, we will focus on the Learned Normalized Operator strategies presented in the previous chapter.

Edge pixel approach is the most standard in the DNN literature, we say that \mathbf{e} is the edge map of image \mathbf{x} and we let \mathbf{e} to live on pixels and the edge map \mathbf{e} is in \mathbb{R}^N ($J = 1$). DPMS-LNO architecture relies on:

$$\left\{ \begin{array}{l} \mathcal{V}_{\mathbf{x},k}(\mathbf{x}^{[k]}, \mathbf{e}^{[k]}) = \text{Id} + 2\kappa_k \lambda_S \left(\frac{1}{J} \sum_{j=1}^J (1 - \mathbf{e}^{[k]}) \odot (\mathbf{D}_k \mathbf{x}^{[k]})_j^2 \right) \\ \mathbf{b}_k = 0, \\ \eta_{k,\mathcal{E}} = \text{prox}_{\lambda_{\mathcal{E}} \mathcal{E}}, \\ \mathcal{W}_k(\mathbf{x}^{[k]}, \mathbf{e}^{[k+1]}) = \frac{\mu_k}{\mu_{k+1}} \left(\text{Id} - 2\mu_k \lambda_S \mathbf{D}_k \left(\frac{1}{J} \sum_{j=1}^J (1 - \mathbf{e}^{[k+1]})^2 \odot (\mathbf{D}_k \mathbf{x}^{[k]})_j \right) \right) \\ \tilde{\mathbf{b}}_k = \frac{\mu_k}{\mu_{k+1}} \mathbf{z}, \\ \eta_{k,\mathcal{F}} = \text{Id}. \end{array} \right. \quad (7.7)$$

7.2 Edge detector based on DScCP denoiser

Using variational approaches like ROF or Blake-Zisserman model require to select the edges by thresholding. This strategy is more convenient for learning-based approaches as we can combine these two step as two consecutive black boxes: (i) denoising building block, (ii) edge detector block. To validate this idea, we propose a simple approach where we attach an edge detector block to a stable and powerful PNNs denoiser such as DScCP-LNO proposed in Chapter 6.

- **DScCP-LNO-ED** stands for Deep Strong convexity Chambolle-Pock - with Edge Detection which is defined as follow

$$\left\{ \begin{array}{l} (\mathbf{x}^{[K]}, \mathbf{u}^{[K]}) = f_{\mathbf{z},v,\Theta}^{K,\text{DScCP}}(\mathbf{x}^{[0]}, \mathbf{u}^{[0]}) \\ \mathbf{e}^{[K]} = \eta_{K,\mathcal{E}}(\mathbf{F}_K \mathbf{x}^{[K]}), \end{array} \right. \quad (7.8)$$

where $\mathbf{F}_K \in \mathbb{R}^{JN \times CN}$ and $\eta_{K,\mathcal{E}}(e) = \text{HardTanh}(e^2)$.

Remark 1. *The choice of $\eta_K(e) = \text{HardTanh}(e^2)$ is motivated by the truncated quadratic function that is used in Blake-Zisserman functional.*

Table 7.1: Learnable parameters of each unfolded scheme

	Θ_k	Comments
DPMS-LNO	$\mu_k, \kappa_k, \mathbf{D}_{k,\mathcal{F}} = \mathbf{D}_{k,\mathcal{E}}^T$	
DScCP-LNO-ED	$\mathbf{D}_{k,\mathcal{F}} = \mathbf{D}_{k,\mathcal{D}}^T, \mathbf{F}_K$	absorb τ_k in $\mathbf{D}_{k,\mathcal{D}}$

7.3 Experiments

7.3.1 Training strategy for joint task

Training dataset – We consider two sets of images: the *training set* $(\bar{\mathbf{e}}_s, \bar{\mathbf{x}}_s, \mathbf{z}_s)_{s \in \mathbb{I}}$ of size $|\mathbb{I}|$ and the *test set* $(\bar{\mathbf{e}}_s, \bar{\mathbf{x}}_s, \mathbf{z}_s)_{s \in \mathbb{J}}$ of size $|\mathbb{J}|$. For both sets, $\bar{\mathbf{e}}_s$ are the exact edges and each couple $(\bar{\mathbf{x}}_s, \mathbf{z}_s)$ consists of a clean multichannel image $\bar{\mathbf{x}}_s$ of size CN_s (where

C denotes the number of channels, and N_s the number of pixels in each channel), and a noisy version of this image given by $\mathbf{z}_s = \bar{\mathbf{x}}_s + \mathbf{n}_s$ with $\mathbf{n}_s \sim \mathcal{N}(0, \delta^2 \text{Id})$ for $\delta > 0$.

Choice of loss function – In the context of DL-based approaches, when dealing with the joint task of image denoising and edge detection, before going into designing a model we need to define the loss function. The proposed method is to minimize the following loss:

$$\hat{\Theta} \in \underset{\Theta}{\text{Argmin}} \frac{1}{|\Theta|} \sum_{s \in \Theta} \|\bar{\mathbf{x}}_s - \hat{\mathbf{x}}_s\|_2^2 + \lambda_{CE} \text{BCE}(\bar{\mathbf{e}}_s - \hat{\mathbf{e}}_s) \quad (7.9)$$

where $(\hat{\mathbf{x}}_s, \hat{\mathbf{e}}_s) = f_{\Theta}^K(\mathbf{z}_s)$, BCE is defined in (5.12) and λ_{CE} is the parameter balancing the two losses. The loss (7.9) will be optimized in Pytorch with Adam algorithm [Kingma and Ba, 2014].

This loss has been selected to combine performance in restoration and edge detection.

Architectures – We will compare the two different architectures, namely DPMS-LNO and DScCP-LNO-ED and for every layer $k \in \{1, \dots, K\}$ the weight operator \mathbf{D}_k consists of J convolution filters (features), mapping an image \mathbb{R}^{CN} to features in \mathbb{R}^{JN} .

Experimental settings – To evaluate and compare the proposed unfolded architectures, we consider RGB images (i.e, $C=3$). The unfolded NNs are trained with $|\Theta| = 15$ images extracted from BSDS500 dataset, with a fixed noised level $\delta = 0.05$. The learning rate for ADAM is set to $3e - 2$ (all other parameters set as default), $\lambda_{CE} = 0.005$, we use batches of size of 5 and patches of full size image. Unfolded Denoising-edge detector NNs is trained with $(J, K) = (64, 20)$ and unfolded DPMS NNs is trained with $(J, K) = (24, 10)$.

In our experiments, we aim to compare the proposed unfolded NNs for different metrics: (i) runtime , (ii) denoising and edge detection performance. For sake of completeness, these metrics will also be provided for the variational method proposed in chapter 4, DMS- ℓ_1 with $\lambda_S = 4$ and $\lambda_E = 8 \times 10^{-3}$, AT- ϵ solved with SL-PAM algorithm with ϵ goes from 0.08 to 0.015, $\lambda_S = 4$ and $\lambda_E = 8 \times 10^{-3}$.

7.3.2 Experimental results

Architecture comparison – We first compare the proposed unfolded DPMS and DScCP-LNO-ED in terms of runtime and number of learnable parameters (i.e, $|\Theta|$). The experiments are conducted in PyTorch, using an Nvidia Tesla V100 PCIe 16GB. The results are presented in Table 7.2

Table 7.2: Architecture comparison. Runtime (in sec.), number of parameters $|\Theta|$ of the denoisers when used on 10 images of size $3 \times 481 \times 321$. Values for the NNs are given for fixed $(K, J) = (20, 64)$ in case of DScCP-LNO-ED and $(J, K) = (24, 10)$ in case of DPMS-LNO.

	average (msec)	$ \Theta $
DMS- ℓ_1	300×10^3	
AT- $\varepsilon \searrow$	1102×10^3	
DScCP-LNO-ED	15 ± 3	34,607
DPMS-LNO	11 ± 3	6482

Denoising and edge detection performance – In Figure 7.1, the DScCP-LNO-ED results provide the best PSNR values on most of testing images with $\delta = 0.05$. For visual inspection, we observe that DScCP-LNO-ED tends to estimate closer to the original and preserving more details than DMS- ℓ_1 and AT- $\varepsilon \searrow$, hence, the later methods require to smooth the image to eliminate the noise then some high frequencies pattern is hard to keep when the noise level goes up. The DPMS-LNO also tends to keep image high frequency details but denoising performance is not well achieved.

On the contrary, the edge map quality of the state-of-the art methods tend to be more clear. In Figure 4.5, we can observe that the histogram of \mathbf{e} obtained by AT- $\varepsilon \searrow$ lies between interval $[0, 1]$, the methods also give some low level edges and less outlier points compared to DMS- ℓ_1 . The preliminary results on edge detection of unfolded NNs is not bad but still have some limitations. DPMS-LNO tends to give a clearer edge map compared to DScCP-LNO-ED but the contrast is lower. The edge width of both methods is also thicker than the state of the art methods.

From Table 7.2, we can see that DScCP-LNO-ED offers a good compromise between runtime, and the denoising problem. We also emphasize that the training phase for unfolded NNs still was performed on a small dataset.

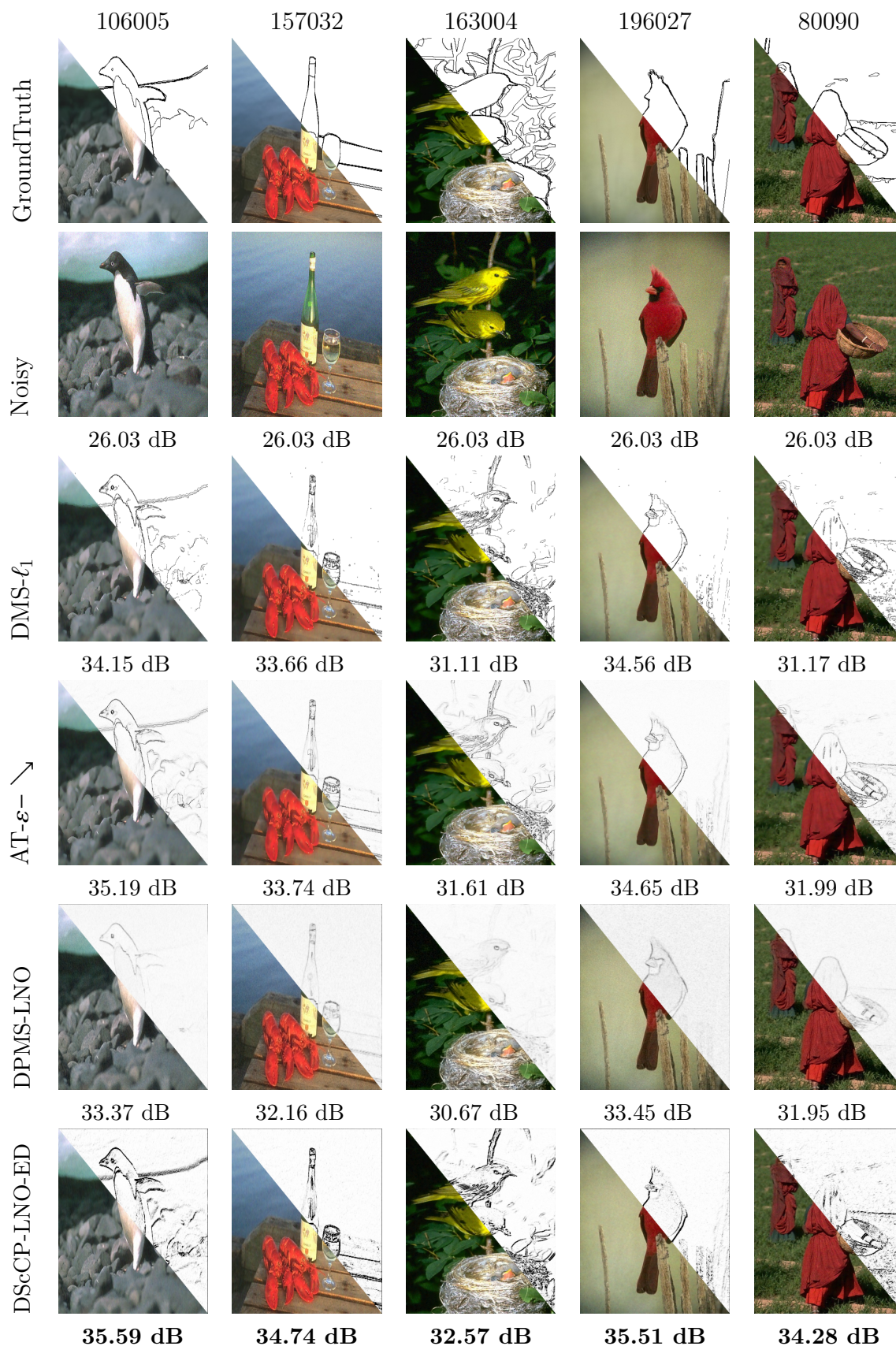


Figure 7.1: Comparison on BSDS500 testing dataset, involving white Gaussian noise with standard deviation $\delta = 0.05$, from top to bottom: Ground truth, noisy image, DMS- ℓ_1 [Foare et al., 2019], AT- ε ↘ [Le et al., 2022a], DPMS-LNO, DScCP-LNO-ED.

7.4 Conclusion

In this chapter, we have presented two frameworks for building a denoising NNs with the capability of edge detection whose architecture are based on PALM applied on DMS functional or DScCP-LNO. We show through numerical experiments that DPMS-LNO does not work as expected and this is left for a deeper study in the future. Beside, the proposed DScCP-LNO-ED is an effective denoiser and also an edge detector which ensures more about its robustness for many tasks by injecting the network in PnP algorithms for deblurring and edge detection task.

Part III

Conclusions and perspectives

This thesis focused mainly on two problems. In the first part, we investigated the problem of the Discrete Mumford-Shah function with the approximating Ambrosio-Tortorelli penalization on edges. In the second part, we proposed several strategies based on the Proximal unfolded Neural Networks deployed for several image processing tasks including its application to edges detection for degraded images. Both approaches can be seen as facets of joint image restoration and edges detection in the future.

Conclusions

In Part I – Chapter 4, we studied the Ambrosio-Tortorelli penalization on edges with the proposed Gold-Grid Search in order to prove the effectiveness of such a choice of g_E for the DMS function (2.35). We described in details how to construct the approximating AT regularizer under the framework of Discrete Exterior Calculus. We also proposed a strategy to favor the Γ -convergence when solving the DMS-AT for many stages for which we restart the initialization and gradually decrease the value of ε . Numerically, we show that this method provides better estimates than the ones obtained with a ℓ_1 norm penalization on edges when including suitable hyperparameters choice. The proposed approach obtains stable results with different noise levels, solves the problem of presenting outlier dots when $g_E = \ell_1$ and high noise level and still preserves important features in the recovered image. Regarding the minimization algorithms to solve Discrete AT, we provide both PALM and SL-PAM strategies with convergence guarantees to a critical point. SL-PAM converges faster but relies on a tedious proximal step that requires a careful processing to ensure that the algorithm stays efficient in running time.

In Part II, we investigate the deep unfolding NNs strategy to design a joint denoising and edge detection architecture. For the first step toward the objective, we placed ourself in the simpler context of denoising. The proposed PNNs are based on the standard convex optimization problem (6.1). Minimization algorithms (FB, CP and inertial versions) were considered through an unified Proximal Neural Network architecture. Various unrolling techniques were proposed lying in the design of the linear operator \mathbf{D} (LNO and LFO). These architectures lead to many variants of denoising PNNs. These architectures appear to be 100 times lighter than the state-of-the-art DNNs (such as DRUnet) encountered in the literature. To train our architecture we also consider a smaller dataset (200 images) compare with ~ 4000 images used in DRUnet but our models were capable to produce competitive results. To guarantee the stability of these models, we also provided a deeper study on their robustness by calculating a tight bound of the norm of Lipschitz constant. Furthermore, along with these advantages, we also investigate their performance in the context of deblurring images and texture segmentation considering a fractal homogeneous data analysis. Both cases prove that proposed unfolded architectures keep being consistent and have effective results compared to state-of-the-art methods.

In Part II – Chapter 7, we proposed two approaches to address the edges detection task for degraded images. One relies on the PALM iterations with the ℓ_1 norm penalization on edges, the second one is based on the denoising PNNs designed in Chapter 6. For both schemes, we proposed to tackle the problem with the loss

function combining the MSE and BCE on a very small dataset. We observe that both architectures work correctly, however these preliminary results have to be deeper studied.

Perspectives

Future studies, following on the findings presented in this thesis, may be considered.

Convergence of PALM and SL-PAM in solving discrete AT functional –

In this manuscript, we did not provide the study on the convergence rate of PALM and SL-PAM solving the problem (4.1). However, based on the assumptions of strong convexity of data term and edge length penalization term, the convergence rates could be established in order to highlight theoretically the better performance of SL-PAM.

Choice of \mathbf{D} – The operator \mathbf{D} plays a crucial role to highlight the properties of the images, both in the variational image processing and DL studies. In the context of this thesis, we did not make any further study on other suitable choice of \mathbf{D} for DMS-AT functional. Some interesting perspectives are :

- to combine hand-designed linear operator with learned-linear operator in unfolded NNs.
- to investigate the relation between constraints on the linear operator and robustness of unfolded NNs.

Unfolding SL-PAM and choice of activator $\text{prox}_{\kappa_K, \lambda \Phi}$ – In Chapter 6, we highlight that a faster minimization algorithm somehow conducted to a better unfolded architecture. Hence, to improve the DPMS-LNO, we could consider

- to unfold the DMS based on SL-PAM iterations.
- to choose the penalization on edges Φ to be ℓ_1 norm or Ambrosio-Tortorelli functional. For instance, we could learn Ambrosio-Tortorelli approximation: through the operator \mathbf{D}_1 in AT regularization. This operator can be generalized and learned in the NNs architecture.

From pixel edge to edge between nodes for unfolded NNs – In the comparison (Figure 7.1), we observe that the Discrete Mumford Shah functional with Ambrosio-Tortorelli penalization on edges tends to better identify the thinner edges of images compared to DL-based method. This can be explained from the fact that edge map and modern edge detector are based on pixel edge rather the edges between pixels. There are many studies focusing on finding a crisp boundary but it remains a challenge in the DL community. In this works, we proposed two frameworks based on unfolded denoising NNs with injected edge detecting sublayer (DPMS-SF and DScCP-SF) but we still use the pixel edge map in the training dataset, it allows us to initially detect

the edges in degraded images but the width of edges are still thick and some places are not well detected. Our objective is to bring these two branches closer in the context of edge detection in degraded images. This could be a path worth looking forward to in the future.

From edges to closed contour – In experimental physics, closed contour refers to a continuous path that forms a closed shape with no breaks. This definition plays an important role in some applications, e.g. to estimate the contact area's phase, aiding in the identification of hydrodynamic regimes or in magnetic resonance imaging measurement of perimeter of the interface is a very useful information. As we observed, functional like MS is not suitable for detecting closed contours but there exist other relaxed variants of MS such as Chan-Vese model that could be considered to design unrolling architecture. Another challenge could be realized on the hydrodynamic regimes dataset in order to develop an architecture that can precisely detect a closed contour and measure interface perimeter.

Light weights but smart unfolded NNs – Intuitively, when combining on one side includes the standard variational approach (where we well understood the context and there exists many efficient well-developed methods), and on the other side the deep learning based approaches (where we less understood the insight of the model), we have shown that such an architecture is robust in simple task of denoising image and even more in edge detection for noisy image. But there are still many questions left behind:

- the study of the theoretical guarantees of PNNs,
- a deeper analysis of improvements of PNNs when using accelerated optimization strategies such as inertia, preconditioning, etc.

Appendices

Auxiliary proofs

A.1 Auxillary proof for Proposition (4.2.1)

Recall: Let A, B, C and D be matrix of arbitrary size.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -A^{-1}B \\ \text{Id} \end{bmatrix} (D - CA^{-1}B)^{-1} [-CA^{-1} \quad \text{Id}] \quad (\text{A.1})$$

where A and $D - CA^{-1}B$ are supposed to be invertible.

We first rewrite the updating on $\mathbf{e}^{[k]}$ in Algorithm 4.1 where \mathbf{D}_1 is defined by (4.8).

$$\begin{aligned} \mathbf{e}^{[k+1]} &= \left[2\kappa_k \lambda_E \varepsilon \mathbf{D}_1^* \mathbf{D}_1 + \left(1 + \frac{2\kappa_k \lambda_E}{4\varepsilon} \right) \text{Id} \right]^{-1} \mathbf{e}^{[k]} \\ &= (\eta_1 \text{Id} + \eta_2 \mathbf{D}_1^* \mathbf{D}_1)^{-1} \mathbf{e}^{[k]} \\ &= \left(\eta_1 \text{Id} + \eta_2 \begin{bmatrix} \mathbf{D}_v^* \mathbf{D}_v & -\mathbf{D}_v^* \mathbf{D}_h \\ -\mathbf{D}_h^* \mathbf{D}_v & \mathbf{D}_h^* \mathbf{D}_h \end{bmatrix} \right) \mathbf{e}^{[k]} \\ &= \begin{bmatrix} \underbrace{\eta_1 \text{Id} + \eta_2 \mathbf{D}_v^* \mathbf{D}_v}_A & \underbrace{-\eta_2 \mathbf{D}_v^* \mathbf{D}_h}_B \\ \underbrace{-\eta_2 \mathbf{D}_h^* \mathbf{D}_v}_C & \underbrace{\eta_1 \text{Id} + \eta_2 \mathbf{D}_h^* \mathbf{D}_h}_D \end{bmatrix}^{-1} \mathbf{e}^{[k]} \end{aligned}$$

where $\eta_1 = 1 + \frac{2\kappa_k \lambda_E}{4\varepsilon}$ and $\eta_2 = 2\kappa_k \lambda_E \varepsilon$.

In (A.1), we can rewrite the above formulation as:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} \overbrace{A^{-1}}^F + \overbrace{(-A^{-1}B)}^G \overbrace{(D - CA^{-1}B)^{-1}}^Q \overbrace{(-CA^{-1})}^M & \overbrace{(-A^{-1}B)}^G \overbrace{(D - CA^{-1}B)^{-1}}^Q \\ \overbrace{(D - CA^{-1}B)^{-1}}^M \overbrace{(-CA^{-1})}^M & \overbrace{(D - CA^{-1}B)^{-1}}^Q \end{bmatrix} \quad (\text{A.2})$$

$$= \begin{bmatrix} F + GQM & GQ \\ QM & Q \end{bmatrix} \quad (\text{A.3})$$

where

$$\begin{cases} \eta_1 = (1 + \frac{\kappa_k \lambda_E}{2\varepsilon}) \\ \eta_2 = 2\lambda_E \varepsilon \kappa_k \\ F = (\eta_2 D_v^* D_v + \eta_1 \text{Id})^{-1} \\ G = F(\eta_2 D_v^* D_h) \\ M = (\eta_2 D_h^* D_v) F \\ Q = (\eta_2 D_h^* D_h + \eta_1 \text{Id} - (\eta_2 D_h^* D_v) F (\eta_2 D_v^* D_h))^{-1} \end{cases} \quad (\text{A.4})$$

List of Figures

1.1	(a) Original image capturing joint gas and liquid flow in confined environments (b) Preprocessed image (c) Estimated contact interface/contour.	1
2.1	Example of an image "102061" from the BSDS500 validation dataset [Arbelaez et al., 2011] degraded by the blur kernel \mathbf{a} and an additive white gaussian noise $\mathbf{n} \sim \mathcal{N}(0, \delta^2 \mathbf{I}_{CM})$ ($\delta = 0.03$).	9
2.2	(a) \mathbf{x} , (b) $D_h \mathbf{x}$, (c) $D_v \mathbf{x}$, (d),(e) and (f) are respectively histograms of (a), (b) and (c).	11
2.3	Comparison between the ℓ_0 -pseudo-norm, ℓ_1 -norm, ℓ_2^2 -norm and Huber function with $\delta = 1$, BerHu function with $\delta = 1$	12
2.4	Image restoration using Tikhonov-regularization by minimizing (2.2). First row: (a) Original image (b) Degraded image (c) Restored. Second row: (a), (b) (c) display a row of the image to better capture the effect of the restoration including smoothing effect with Tikhonov regularization.	13
2.5	Examples of two edges definitions in digital image: (a) Extracted patch (b)Image patch seen as a discrete surface with true contour delineated in black (c) Contours defined in between pixels with values living on edges s_i (d) Contours defined on pixels with values living on vertices v_i	14
2.6	Illustration of $\mathbf{D} = [D_h^\top D_v^\top]^\top$ with periodic boundary conditions corresponding horizontal and vertical pixelwise discrete gradient operators D_h and D_v	14
2.7	An example to compare the difference between pixel-wise edges (b) and edges between nodes (c).	15
2.8	Staircasing effects when we use TV_{aniso} for 1D signal.	16
2.9	Image restoration using different variational methods: (a) Groundtruth $\bar{\mathbf{x}}$ (b) degraded image \mathbf{y} (c) ROF [Rudin et al., 1992], the induced edges obtained by thresholding the absolute value of gradient map $ \mathbf{D}\mathbf{x} $ with value 0.5 (d) BZ [Hohm et al., 2015], the induced egdes obtained by using Eq.(2.30) (e) Potts [Storath et al., 2015], the induced edge obtained by detecting the jumpset between all segment(f) DMS- ℓ_1 [Foare et al., 2019], the thresholding value for \mathbf{e} is 0.5.	22

4.1	(a) Original image (b) Degraded image with $\delta = 0.05$ according to (2.1)-(2.8) with $\mathbf{A} = \text{Id}$ (c) Estimated images and edges obtained with DMS- ℓ_1 (2.35) solved by SL-PAM algorithm for $\lambda_S = 5.7$ and $\lambda_E = 0.006$, (d) Estimated images and edges obtained with DMS- ℓ_1 (2.35) solved by SL-PAM algorithm for $\lambda_S = 5.7$ and $\lambda_E = 0.01$, $\overline{\lambda}_E = 0.01$ (e) AT (4.1) solved by SL-PAM with decreasing at each stage of $\varepsilon = 0.2 \searrow 0.02$ and $(\lambda_S, \lambda_E) = (5.7, 0.008)$	42
4.2	Left: example of a discrete image complex for illustrating $\mathbf{D} = [\mathbf{D}_h^\top \ \mathbf{D}_v^\top]^\top$. Right: corresponding horizontal and vertical pixelwise discrete gradient operators \mathbf{D}_h and \mathbf{D}_v	44
4.3	Left: example of a discrete image complex for illustrating $\mathbf{D} = [\mathbf{D}_{hp}^\top \ \mathbf{D}_{vp}^\top]^\top$. Right: corresponding horizontal and vertical pixelwise discrete gradient operators \mathbf{D}_h and \mathbf{D}_v	46
4.4	AT- $\varepsilon \searrow$ energy associated to the estimate displayed in Figure 4.1 when $\varepsilon = 0.025 \searrow 0.009$ and $(\lambda_S, \lambda_E) = (5.2, 0.009)$ solved by PALM (top) and SL-PAM (bottom) for each decreasing ε stage with $\varrho = 1.5$	50
4.5	1st column: (Top) original image and groundtruth edge, (Bottom) degraded image with $\delta = 0.03$. 2nd column: Denoised image and edges estimated by SL-PAM-AT- $\varepsilon = 0.2 \searrow 0.02$ with decreasing factor $\varrho = 1.5$ and $(\lambda_S = 5.2, \lambda_E = 0.006)$. 3rd column: Histogram of \mathbf{e}_h (purple) and \mathbf{e}_v (green) or both (gray blue) for each stage.	51
4.6	Original and degraded data are displayed on the top. Comparison between grid search (1st column) and the proposed Golden-grid search (2nd column) for (λ_S, λ_E) -hyperparameters with PALM-AT- $\varepsilon = 0.2$ when maximizing Jaccard index and when maximizing the PSNR. The grid search (resp. Golden-grid search) map is composed with 121 (resp.125) values.	53
4.7	Comparison between different schemes using observation \mathbf{y} provided in Fig. 4.6 (top-left). All the results are obtained with a multiresolution golden-grid search strategy (even for T-ROF and Hohm et al. [Hohm et al., 2015]) maximizing the Jaccard index. The second row displays the optimal solution for each methods. The optimum Jaccard index, the associated PSNR and the overall-running time (including the search for the optimal parameters relying on the multiresolution grid-search) are also provided.	54
4.8	(Top) 1st row: observation degraded by both an additive white Gaussian noise and a Gaussian blur. 2nd row: Best estimated image $\hat{\mathbf{x}}$ and contours $\hat{\mathbf{e}}$ (delineated in red) using PALM-AT with $\varepsilon = 2 \searrow 0.02$ maximizing the Jaccard index, provided below. (Bottom) Comparisons between T-ROF, Hohm et al., PALM- ℓ_1 , and PALM-AT with $\varepsilon = 2 \searrow 0.02$ obtained with multiresolution Golden-grid search strategy maximizing the Jaccard index. For each method, we display the mean (min and max) of Jaccard index (middle) and PSNR (right) for 15 realizations of noise.	55

4.9	Performance obtained with T-ROF, Blake-Zisserman [Blake and Zisserman, 1987], DMS- ℓ_1 [Foare et al., 2019], and the proposed AT- ε with associated PSNR/Jaccard.	56
5.1	The k -th layer of a Feed Forward Neural Network.	61
6.1	Top: Architecture of the proposed DAH-Unified block for the k -th layer. Linearities, biases, and activation functions are shown in blue, green and red, respectively. Bottom: Inertial step for DScCP(top) and DDiFB (bottom), for the k -th layer.	70
6.2	Hardtanh function with $\nu = 0.5$	74
6.3	Training Setting 1: Robustness. Values $\chi = \max_{s \in \mathbb{J}} \ J f_{\Theta}(\mathbf{z}_s)\ _{sp}$ (\log_2 scale) for the proposed DUNNs, with $J \in \{8, 16, 32, 64\}$ and $K \in \{5, 10, 15, 20, 25\}$. Top row: LNO settings. Bottom row: LFO settings.	77
6.4	Training Setting 2: Robustness. Distribution of $(\ J f_{\Theta}(\mathbf{z}_s)\ _S)_{s \in \mathbb{J}}$ for 100 images extracted from BSDS500 validation dataset \mathbb{J} , for the proposed DUNNs and DRUnet.	78
6.5	Training Setting 1: Denoising performance. Average PSNR obtained with the proposed DUNNs, on 100 images from BSD500 degraded with noise level $\delta = 0.08$. Results are shown for $J \in \{8, 16, 32, 64\}$ and $K \in \{5, 10, 15, 20, 25\}$. Top row: LNO settings. Bottom row: LFO settings.	78
6.6	Training Setting 2: Denoising performance. PSNR values obtained with the proposed DUNNs (with $(K, J) = (20, 64)$), for 20 images of BSDS500 validation set, degraded with noise level $\delta = 0.05$	79
6.7	Training Setting 2: Denoising performance on Gaussian noise. Example of denoised images (and PSNR values) for Gaussian noise $\delta = 0.05$ obtained with DRUnet and the proposed DDFB-LNO and DScCP-LNO, with $(K, J) = (20, 64)$	80
6.8	Training Setting 2: Denoising performance on Poisson noise. Example of denoised images (and PSNR values) for Poisson noise level 50/255 obtained with DRUnet and the proposed DDFB-LNO and DScCP-LNO, with $(K, J) = (20, 64)$	81
6.9	Training Setting 2: Denoising performance on Laplace noise. Example of denoised images (and PSNR values) for Laplace noise level 0.05 obtained with DRUnet and the proposed DDFB-LNO and DScCP-LNO, with $(K, J) = (20, 64)$	81
6.10	Deblurring (Training Setting 2): Robustness. Distribution of $\ J h_{\Theta}(\mathbf{z}_s)\ _S$, where $h_{\Theta} = 2f_{\Theta} - \text{Id}$ for 100 images extracted from BSDS500 validation dataset \mathbb{J} , for the proposed DUNNs and DRUnet.	83
6.11	Deblurring (Training Setting 2): Parameter choice (γ). Convergence behavior (PSNR and relative error norm) of the PnP-FB algorithms for fixed $\beta = 1$, and varying $\gamma \in \{1.2, 1.4, 1.6, 1.8, 1.99, 2.1\}$	84

6.12	Deblurring (Training Setting 2): Parameter choice (β). Top: Best PSNR values obtained with DDFB-LNO, DScCP-LNO, DRUnet and BM3D, on 12 images from BSDS500 validation set degraded according to (6.20), with $\sigma = 0.03$. Bottom: Box plots for the distribution of β associated with the best PSNR values for DDFB-LNO, DScCP-LNO, DRUnet and BM3D.	85
6.13	Deblurring (Training Setting 2): Restoration performance. Restoration example for $\sigma = 0.015$ with parameters $\gamma = 1.99$ and β chosen optimally for each scheme.	87
6.14	Deblurring (Training Setting 2): Restoration performance. Restoration example for $\sigma = 0.03$ with parameters $\gamma = 1.99$ and β chosen optimally for each scheme.	88
6.15	Deblurring (Training Setting 2): Restoration performance. Restoration example for $\sigma = 0.05$, with parameters $\gamma = 1.99$ and β chosen optimally for each scheme.	88
6.16	Estimation and segmentation results on a texture image from Config I. This configuration involves supervised methods (DnCNN, DFH, and DSH with 3×10^4 weights) trained on Config I and evaluated on Config I. The segmentation is obtained by applying k-means clustering to the estimated values \hat{h}	92
6.17	(i) Training error as a function of the number of iterations on Config I (ii) Estimation error as a function of the number of iterations on the test dataset of Config I	93
7.1	Comparison on BSDS500 testing dataset, involving white Gaussian noise with standard deviation $\delta = 0.05$, from top to bottom: Ground truth, noisy image, DMS- ℓ_1 [Foare et al., 2019], AT- ε \searrow [Le et al., 2022a], DPMS-LNO, DScCP-LNO-ED.	100

List of Tables

6.1	Learnable parameters of each unfolded scheme	73
6.2	Architecture comparison. Runtime (in sec.), number of parameters $ \Theta $ and FLOPs (in G) of the denoisers when used on 100 images of size $3 \times 481 \times 321$. Values for the DUNNs are given for fixed $(K, J) = (20, 64)$	75
6.3	Training Setting 2: Denoising performance. Average PSNR (and standard deviation) values (in dB) obtained with the proposed PNNs with $(K, J) = (20, 64)$ and with DRUnet, for 100 noisy images of BSDS500 validation set ($\delta = 0.05$, input PSNR= 25.94dB).	79

6.4	Training Setting 2: Denoising performance on Poisson noise. Average PSNR (and standard deviation) values (in dB) obtained with the proposed DUNNs with $(K, J) = (20, 64)$ and with DRUnet, for 12 noisy images of BSDS500 validation set degraded with Poisson noise level 50/255.	80
6.5	Training Setting 2: Denoising performance on Laplace noise. Average PSNR (and standard deviation) values (in dB) obtained with the proposed DUNNs with $(K, J) = (20, 64)$ and with DRUnet, for 12 noisy images of BSDS500 validation set degraded with Laplacian noise level 0.05.	80
6.6	Deblurring (Training Setting 2): Restoration performance. Average PSNR values over 12 images from BSDS500 validation set, obtained with different PnP-FB schemes. For each NN, β was chosen to obtain the highest PSNR.	87
6.7	Comparison in terms of normalized error or segmentation accuracy between different methods.	92
7.1	Learnable parameters of each unfolded scheme	97
7.2	Architecture comparison. Runtime (in sec.), number of parameters $ \Theta $ of the denoisers when used on 10 images of size $3 \times 481 \times 321$. Values for the NNs are given for fixed $(K, J) = (20, 64)$ in case of DScCP-LNO-ED and $(J, K) = (24, 10)$ in case of DPMS-LNO.	99

Bibliography

- [Adler and Öktem, 2018] Adler, J. and Öktem, O. (2018). Learned primal-dual reconstruction. *IEEE Trans. Med. Imag.*, 37(6):1322–1332.
- [Ambrosio, 1989] Ambrosio, L. (1989). A compactness theorem for a new class of functions of bounded variation. *Bollettino Dell’Unione Matematica Italiana, B*, 3:857–881.
- [Ambrosio et al., 2001] Ambrosio, L., Faina, L., and March, R. (2001). Variational approximation of a second order free discontinuity problem in computer vision. *SIAM J. Math. Anal.*, 32(6):1171–1197.
- [Ambrosio and Tortorelli, 1990] Ambrosio, L. and Tortorelli, V. M. (1990). Approximation of functional depending on jumps by elliptic functional via t -convergence. *Comm. Pure Applied Math.*, 43(8):999–1036.
- [Arbelaez et al., 2011] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916.
- [Arrow et al., 1958] Arrow, K. J., Hurwicz, L., and Uzawa, H. (1958). *Studies in linear and non-linear programming*. Stanford University Press, Stanford.
- [Attouch et al., 2010] Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. (2010). Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457.
- [Attouch and Peypouquet, 2016] Attouch, H. and Peypouquet, J. (2016). The rate of convergence of Nesterov’s accelerated forward-backward method is actually faster than $1/k^2$. *SIAM J. Opt.*, 26(3):1824–1834.
- [Auslender, 1971] Auslender, A. (1971). Méthodes numériques pour la décomposition et la minimisation de fonctions non différentiables. *Numerische Mathematik*, 18:213–223.
- [Baker et al., 2005] Baker, A. H., Jessup, E. R., and Manteuffel, T. (2005). A technique for accelerating the convergence of restarted GMRES. *SIAM J. Matrix Anal. Appl.*, 26(4):962–984.

- [Bauschke and Combettes, 2011] Bauschke, H. H. and Combettes, P. L. (2011). Convex analysis and monotone operator theory in hilbert spaces. *CMS books in mathematics*. DOI, 10:978–1.
- [Bauschke and Combettes, 2017a] Bauschke, H. H. and Combettes, P. L. (2017a). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, New York.
- [Bauschke and Combettes, 2017b] Bauschke, H. H. and Combettes, P. L. (2017b). *Correction to: Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer.
- [Beck, 2017] Beck, A. (2017). *First-order methods in optimization*. SIAM.
- [Beck and Teboulle, 2009] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202.
- [Bertasius et al., 2015] Bertasius, G., Shi, J., and Torresani, L. (2015). Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4380–4389.
- [Bertocchi et al., 2020] Bertocchi, C., Chouzenoux, E., Corbineau, M.-C., Pesquet, J.-C., and Prato, M. (2020). Deep unfolding of a proximal interior point method for image restoration. *Inverse Problems*, 36(3):034005.
- [Blake and Zisserman, 1987] Blake, A. and Zisserman, A. (1987). *Visual reconstruction*. MIT press.
- [Bolte et al., 2014] Bolte, J., Sabach, S., and Teboulle, M. (2014). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Math. Program.*, 146(1-2):459–494.
- [Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Machine Learn.*, 1:1–122.
- [Bredies et al., 2010] Bredies, K., Kunisch, K., and Pock, T. (2010). Total generalized variation. *SIAM J. Imaging Sci.*, 3(3):492–526.
- [Cai and Steidl, 2013] Cai, X. and Steidl, G. (2013). Multiclass segmentation by iterated ROF thresholding. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 237–250. Springer.
- [Chambolle, 1999] Chambolle, A. (1999). Finite-differences discretizations of the Mumford-Shah functional. *ESAIM: Mathematical Modelling and Numerical Analysis*, 33(2):261–288.

- [Chambolle and Dossal, 2015] Chambolle, A. and Dossal, C. (2015). On the Convergence of the Iterates of the Fast Iterative Shrinkage/Thresholding Algorithm. *J. Optim. Theory Appl.*, 166(3):968–982.
- [Chambolle and Pock, 2011] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imag. Vis.*, 40(1):120–145.
- [Chambolle and Pock, 2016] Chambolle, A. and Pock, T. (2016). An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319.
- [Chaux et al., 2007] Chaux, C., Combettes, P. L., Pesquet, J.-C., and Wajs, V. R. (2007). A variational formulation for frame-based inverse problems. *Inverse Problems*, 23(4):1495–1518.
- [Chierchia et al., 2020] Chierchia, G., Chouzenoux, E., Combettes, P. L., and Pesquet, J.-C. (2020). The proximity operator repository. *User’s guide* <http://proximity-operator.net/download/guide.pdf>. Accessed, 6.
- [Combettes et al., 2009] Combettes, P. L., Dũng, D., and Công Vũ, B. (2009). Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, 18:373–404.
- [Combettes et al., 2010] Combettes, P. L., Dũng, D., and Vũ, B. C. (2010). Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, 18:373–404.
- [Combettes and Pesquet, 2007] Combettes, P. L. and Pesquet, J.-C. (2007). A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery. *IEEE J. Selected Topics Signal Process.*, 1(4):564–574.
- [Combettes and Pesquet, 2011a] Combettes, P. L. and Pesquet, J.-C. (2011a). Proximal splitting methods in signal processing. In Bauschke, H., Burachik, R., Combettes, P., Elser, V., Luke, D., and Wolkowicz, H., editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer-Verlag, New York.
- [Combettes and Pesquet, 2011b] Combettes, P. L. and Pesquet, J.-C. (2011b). Proximal splitting methods in signal processing. *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212.
- [Combettes and Pesquet, 2012] Combettes, P. L. and Pesquet, J.-C. (2012). Primal-dual splitting algorithm for solving inclusions with mixtures of composite, Lipschitzian, and parallel-sum type monotone operators. *Set-Valued and variational analysis*, 20(2):307–330.
- [Combettes and Pesquet, 2020] Combettes, P. L. and Pesquet, J.-C. (2020). Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM J. Math. Data Sci.*, 2(2):529–557.

- [Combettes and Wajs, 2005] Combettes, P. L. and Wajs, V. R. (2005). Signal recovery by proximal forward-backward splitting. *Multiscale model. simul.*, 4(4):1168–1200.
- [Condat, 2013] Condat, L. (2013). A primal-dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms. *Journal of Opt. Theory and Applications*, 158(2):460–479.
- [Dabov et al., 2007] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Trans. Image Process.*, 16(8):2080 – 2095.
- [Deledalle et al., 2014] Deledalle, C., Vaiter, S., Fadili, J., and Peyré, G. (2014). Stein unbiased gradient estimator of the risk (sugar) for multiple parameter selection. *SIAM J. Imaging Sci.*, 7(4):2448–2487.
- [Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(7).
- [Foare, 2017] Foare, M. (2017). *Analyse d’images par des méthodes variationnelles et géométriques*. PhD thesis, Université Grenoble Alpes.
- [Foare et al., 2016] Foare, M., Lachaud, J.-O., and Talbot, H. (2016). Image restoration and segmentation using the Ambrosio-Tortorelli functional and discrete calculus. In *Proc. Int. Conf. Patt. Rec.*, pages 1418–1423. IEEE.
- [Foare et al., 2019] Foare, M., Pustelnik, N., and Condat, L. (2019). Semi-linearized proximal alternating minimization for a discrete Mumford–Shah model. *IEEE Trans. Image Process.*, 29:2176–2189.
- [Fortin and Glowinski, 1983] Fortin, M. and Glowinski, R. (1983). *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*. Elsevier Science Ltd, Amsterdam : North-Holland.
- [Gabay and Mercier, 1976] Gabay, D. and Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite elements approximations. *Comput. Math. Appl.*, 2:17–40.
- [Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Match. Int.*, pages 721–741.
- [Gilboa and Osher, 2009] Gilboa, G. and Osher, S. (2009). Nonlocal operators with applications to image processing. *Multiscale Model. and Simul.*, 7(3):1005–1028.
- [Gregor and Le Cun, 2010] Gregor, K. and Le Cun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the International Conference on Machine Learning*, pages 399–406.

- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [Hoffman et al., 2019] Hoffman, J., Roberts, D. A., and Yaida, S. (2019). Robust learning with Jacobian regularization. *arXiv preprint arXiv:1908.02729*.
- [Hohm et al., 2015] Hohm, K., Storath, M., and Weinmann, A. (2015). An algorithmic framework for Mumford–Shah regularization of inverse problems in imaging. *Inverse Problems*, 31(11):115011.
- [Hurault et al., 2021] Hurault, S., Leclaire, A., and Papadakis, N. (2021). Gradient step denoiser for convergent plug-and-play. *arXiv preprint arXiv:2110.03220*.
- [Jacques et al., 2011] Jacques, L., Duval, L., Chaux, C., and Peyré, G. (2011). A panorama on multiscale geometric representations, intertwining spatial, directional and frequency selectivity. *Signal Processing*, 91(12):2699–2730.
- [Jakubovitz and Giryes, 2018] Jakubovitz, D. and Giryes, R. (2018). Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529.
- [Jiu and Pustelnik, 2021] Jiu, M. and Pustelnik, N. (2021). A deep primal-dual proximal network for image restoration. *IEEE JSTSP Special Issue on Deep Learning for Image/Video Restoration and Compression*, 15(2):190–203.
- [Kamilov et al., 2023] Kamilov, U. S., Bouman, C. A., Buzzard, G. T., and Wohlberg, B. (2023). Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications. *IEEE Signal Process. Mag.*, 40(1):85–97.
- [Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [Komodakis and Pesquet, 2015] Komodakis, N. and Pesquet, J.-C. (2015). Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Process. Mag.*, 32(6):31–54.
- [Le et al., 2022a] Le, H. T. V., Foare, M., and Pustelnik, N. (2022a). A proximal based strategy for solving Discrete Mumford-Shah and Ambrosio-Tortorelli models. *IEEE Signal Process. Lett.*, 29:952–956.
- [Le et al., 2022b] Le, H. T. V., Pascal, B., Pustelnik, N., Foare, M., and Abry, P. (2022b). Algorithmes proximaux rapides déroulés pour l’analyse d’images fractales homogènes par morceaux. *Proc. GRETSI*.
- [Le et al., 2022c] Le, H. T. V., Pustelnik, N., and Foare, M. (2022c). The faster proximal algorithm, the better unfolded deep learning architecture ? the study case of image denoising. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 947–951.

- [Le et al., 2023] Le, H. T. V., Repetti, A., and Pustelnik, N. (2023). PNN: From proximal algorithms to robust unfolded image denoising networks and Plug-and-Play methods. *arXiv preprint arXiv:2308.03139*.
- [LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- [Malézieux et al., 2023] Malézieux, B., Moreau, T., and Kowalski, M. (2023). Dictionary and prior learning with unrolled algorithms for unsupervised inverse problems. *arXiv:2106.06338*.
- [Mallat, 1997] Mallat, S. (1997). *A wavelet tour of signal processing*. Academic Press, San Diego, USA.
- [Meer et al., 1991] Meer, P., Mintz, D., Rosenfeld, A., and Kim, D. Y. (1991). Robust regression methods for computer vision: A review. *Int. J. Comp. Vis.*, 6:59–70.
- [Milanfar, 2012] Milanfar, P. (2012). A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Process. Mag.*, 30(1):106–128.
- [Moreau, 1962] Moreau, J.-J. (1962). Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes rendus hebdomadaires des séances de l’Académie des sciences*, 255:2897–2899.
- [Moudafi and Oliny, 2003] Moudafi, A. and Oliny, M. (2003). Convergence of a splitting inertial proximal method for monotone operators. *J. Comput. Appl. Math.*, 155(2):447–454.
- [Mumford and Shah, 1989] Mumford, D. B. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*.
- [Naornita et al., 2014] Naornita, C., Isar, A., and Nelson, J. D. B. (2014). Regularised, semi-local hurst estimation via generalised lasso and dual-tree complex wavelets. In *2014 IEEE International Conference on Image Processing*, pages 2689–2693. IEEE.
- [Nathan Silberman and Fergus, 2012] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.
- [Nesterov, 1983] Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $\mathcal{O}\left(\frac{1}{k^2}\right)$. In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences.
- [Nguyen et al., 2023] Nguyen, P., Soubies, E., and Chaux, C. (2023). Map-informed unrolled algorithms for hyper-parameter estimation. In *2014 IEEE International Conference on Image Processing*. IEEE.

- [Ongie et al., 2020] Ongie, G., Jalal, A., Metzler, C. A., Baraniuk, R. G., Dimakis, A. G., and Willett, R. (2020). Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56.
- [Pascal et al., 2021a] Pascal, B., Mauduit, V., Pustelnik, N., and Abry, P. (2021a). Scale-free texture segmentation: Expert feature-based versus deep learning strategies. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1367–1371. IEEE.
- [Pascal et al., 2020] Pascal, B., Pustelnik, N., Abry, P., Géminard, J.-C., and Vidal, V. (2020). Parameter-free and fast nonlinear piecewise filtering: application to experimental physics. *Annals of Telecommunications*, 75:655–671.
- [Pascal et al., 2018] Pascal, B., Pustelnik, N., Abry, P., Serres, M., and Vidal, V. (2018). Joint estimation of local variance and local regularity for texture segmentation. application to multiphase flow characterization. In *2018 25th IEEE International Conference on Image Processing*, pages 2092–2096. IEEE.
- [Pascal et al., 2021b] Pascal, B., Vaiteer, S., Pustelnik, N., and Abry, P. (2021b). Automated data-driven selection of the hyperparameters for total-variation-based texture segmentation. *J. Math. Imag. Vis.*, 63(7):923–952.
- [Pesquet et al., 2021] Pesquet, J.-C., Repetti, A., Terris, M., and Wiaux, Y. (2021). Learning maximally monotone operators for image recovery. *IAM J. Imaging Sci.*, 14(3):1206–1237.
- [Prewitt, 1970] Prewitt, J. (1970). Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19.
- [Pu et al., 2022] Pu, M., Huang, Y., Liu, Y., Guan, Q., and Ling, H. (2022). Edter: Edge detection with transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1402–1412.
- [Pustelnik, 2023] Pustelnik, N. (2023). On the primal and dual formulations of the discrete Mumford-Shah functional. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, pages 1–5.
- [Pustelnik et al., 2016a] Pustelnik, N., Benazza-Benhayia, A., Zheng, Y., and Pesquet, J.-C. (2016a). Wavelet-based image deconvolution and reconstruction. *Wiley Encyclopedia of EEE*.
- [Pustelnik et al., 2016b] Pustelnik, N., Wendt, H., Abry, P., and Dobigeon, N. (2016b). Combining local regularity estimation and total variation optimization for scale-free texture segmentation. *IEEE Trans. Comput. Imaging*, 2(4):468–479.
- [Repetti et al., 2022] Repetti, A., Terris, M., Wiaux, Y., and Pesquet, J. (2022). Dual Forward-Backward unfolded network for flexible Plug-and-Play. In *Proc. Eur. Sig. and Image Proc. Conference*, pages 957–961.

- [Richardson and Mitter, 1994] Richardson, T. and Mitter, S. (1994). Approximation, computation, and distortion in the variational formulation. In *Geometry-Driven Diffusion in Computer Vision*, pages 169–190. Springer.
- [Rockafellar, 1976] Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5):877–898.
- [Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.
- [Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear phenomena*, 60(1-4):259–268.
- [Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.
- [Shen et al., 2015] Shen, W., Wang, X., Wang, Y., Bai, X., and Zhang, Z. (2015). Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3982–3991.
- [Sobel and Feldman, 1968] Sobel, I. and Feldman, G. (1968). A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272.
- [Storath et al., 2014] Storath, M., Weinmann, A., and Demaret, L. (2014). Jump-sparse and sparse recovery using Potts functionals. *IEEE Trans. Signal Process.*, 62(14):3654–3666.
- [Storath et al., 2015] Storath, M., Weinmann, A., Friel, J., and Unser, M. (2015). Joint image reconstruction and segmentation using the Potts model. *Inverse Problems*, 31(2):025003.
- [Stekalovskiy and Cremers, 2014] Stekalovskiy, E. and Cremers, D. (2014). Real-time minimization of the piecewise smooth Mumford-Shah functional. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*, pages 127–141. Springer.
- [Su et al., 2021] Su, Z., Liu, W., Yu, Z., Hu, D., Liao, Q., Tian, Q., Pietikäinen, M., and Liu, L. (2021). Pixel difference networks for efficient edge detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5117–5127.

- [Tan et al., 2023] Tan, E. Z. C., Chaux, C., Soubies, E., and Tan, V. Y. F. (2023). Deep unrolling for nonconvex robust principal component analysis. *arXiv:2307.05893*.
- [Thacker and Cootes, 1996] Thacker, N. A. and Cootes, T. F. (1996). Vision through optimization. *BMVC Tutorial Notes*.
- [Tikhonov, 1963] Tikhonov, A. N. (1963). On the solution of ill-posed problems and the method of regularization. In *Doklady akademii nauk*, volume 151, pages 501–504. Russian Academy of Sciences.
- [Vũ, 2013] Vũ, B. C. (2013). A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Adv. Comput. Math.*, 38(3):667–681.
- [Venkatakrisnan et al., 2013] Venkatakrisnan, S. V., Bouman, C. A., and Wohlberg, B. (2013). Plug-and-Play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948. IEEE.
- [Wendt et al., 2007] Wendt, H., Abry, P., and Jaffard, S. (2007). Bootstrap for empirical multifractal analysis. *IEEE Signal Process. Mag.*, 24(4):38–48.
- [Wendt et al., 2018] Wendt, H., Combrexelle, S., Altmann, Y., Tourneret, J.-Y., McLaughlin, S., and Abry, P. (2018). Multifractal analysis of multivariate images using gamma markov random field priors. *SIAM J. Imaging Sci.*, 11(2):1294–1316.
- [Xie and Tu, 2015] Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403.
- [Zanetti et al., 2016] Zanetti, M., Ruggiero, V., and Miranda Jr, M. (2016). Numerical minimization of a second-order functional for image segmentation. *Communications in Nonlinear Science and Numerical Simulation*, 36:528–548.
- [Zhang et al., 2021] Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., and Timofte, R. (2021). Plug-and-play image restoration with deep denoiser prior. *IEEE Trans. Pattern Anal. Match. Int.*
- [Zhang et al., 2017] Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017). Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.*, 26(7):3142–3155.