Deep learning for inverse problems and application to omnidirectional imaging

Rita Fermanian

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Signal, Image, Vision*

Par
Rita FERMANIAN

Deep learning for inverse problems and application to omnidirectional imaging

Thèse présentée et soutenue à Rennes, le 6 Decembre 2023
Unité de recherche : INRIA Rennes - Bretagne Atlantique

Rapporteurs avant soutenance :

Andrés Almansa — Directeur de Recherche, CNRS-Université Paris Cité
Said Moussaoui — Professeur, École centrale de Nantes

Composition du Jury :

Président :		
Examinateurs :	Andrés Almansa	Directeur de Recherche, CNRS - Université Paris Cité
	Charles Kervrann	Directeur de Recherche, INRIA Rennes - Bretagne Atlantique
	Charles Yaacoub	Professeur, Université catholique de Lille
	Nicolas Papadakis	Directeur de Recherche, CNRS - Institut de Mathématiques de Bordeaux
	Said Moussaoui	Professeur, École centrale de Nantes
Dir. de thèse :	Christine Guillemot	Directrice de Recherche, INRIA Rennes - Bretagne Atlantique
Enc. de thèse :	Mikael Le Pendu	Chercheur, Interdigital Rennes

Table of Contents

List of Figures

List of Tables

Résumé

Les images omnidirectionnelles, ou images sphériques, sont des données visuelles qui couvrent un champ de vision à 360°, capturant le champ lumineux convergeant vers un seul point depuis toutes les directions. Les images omnidirectionnelles ont fait l'objet d'une attention particulière ces dernières années, notammant dans les domaines de réalité augmentée (RA) et la robotique. Ces caméras souffrent souvent d'une distorsion en raison de leur large champ de vision et de leurs caractéristiques optiques uniques. Par conséquent, la qualité des images omnidirectionnelles capturées peut être degradée, ce qui nécessite le développement de solutions aux problèmes inverses spécifiques pour ces données.

D'une manière générale, les problèmes inverses font référence à la tâche consistant à trouver les entrées qui ont produit un ensemble d'observations. Plus particulièrement dans le domaine de l'imagerie, les limitations des systèmes d'acquisition, de transmission, ou encore de stockage peuvent conduire à des observations dégradées. Parfois, l'acquisition peut être coûteuse ou même nocive (par exemple, exposer le corps humain à des rayonnements dans le cas d'un scanner), il est donc souhaité de mesurer le moins de données possible. Ainsi, les problèmes inverses en imagerie concernent la récupération d'une image à partir des mesures dégradées que nous observons.

Les problèmes inverses rencontrés sont souvent *mal-posés*, ce qui signifie qu'ils n'ont pas de solution unique. Afin de résoudre le défi posé par la nature mal posée, des connaissances a priori supplémentaires sont utilisées pour restreindre l'ensemble des solutions admissibles.

Cette thèse est dédiée à contribuer aux méthodes de régularisation basées sur l'apprentissage profond pour les problèmes inverses en imagerie perspective et omnidirectionnelle.

Motivation

Au cours des dernières décennies, des recherches approfondies ont été consacrées à la résolution de problèmes inverses, aboutissant à diverses approches différentes pour leur résolution. Par exemple, l'approche bayésienne consiste à considérer une perspective probabiliste et à définir l'a priori et le terme d'attache aux données par des distributions

de probabilité. La distribution a posteriori est exprimée à l'aide de la loi de Bayes :

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}, \tag{1}$$

où $p(y|x)$ et $p(x)$ représentent respectivement la *vraisemblance* des observations et la distribution *a priori*.

La distribution a posteriori représente toutes les solutions possibles au problème inverse en question, compte tenu des données mesurées. Elle peut être évaluée à l'aide de différentes estimations, comme le MAP ou le MMSE.

Pour une bonne reconstruction, le choix du prior (la terme de régularisation) joue un rôle important. Bien que les methodes de régularisation créées manuellement aient prouvé leur efficacité, il n'est pas facile de représenter entièrement la nature complexe des images naturelles à l'aide de telles distributions créées manuellement.

L'apprentissage profond a considérablement révolutionné les sciences de l'imagerie, et le domaine des problèmes inverses n'est pas resté insensible à cette influence. Notre attention se porte sur les progrès en matière de régularisation apprise des problèmes inverses, un domaine qui a récemment suscité beaucoup d'intérêt. Certaines de ces approches comprennent :

- **Régularisation en utilisant le prior implicitement capturé par l'architecture du réseau :** Le Deep Image Prior (DIP) [1] est une approche unique, où les auteurs montrent que l'architecture du réseau d'un modèle génératif peut elle-même être utilisée pour régulariser les problèmes inverses. Le réseau générateur n'est pas appris sur un ensemble de données. Il fonctionne en prenant un vecteur d'entrée aléatoire et en ajustant itérativement ses poids initialisés de manière aléatoire pour minimiser l'erreur quadratique moyenne entre sa sortie et l'observation dégradée. Cela signifie que le réseau capture implicitement un prior au cours du processus d'optimisation, régularisant efficacement les tâches de restauration d'image.
- **Régularisation en utilisant la puissance des débruiteurs :** L'idée d'exploiter les débruiteurs pour régulariser différents problèmes inverses a été proposée pour la première fois en 2013 par Venkatakrishnan et al. dans un framework appelé *Plug-and-play*, où l'opérateur proximal d'un régulariseur est remplacé par un débruiteur. La méthode a été initialement proposée avec un débruiteur classique, mais l'approche a rapidement évolué et est devenue un sous-domaine de recherche important. Les approches Plug-and-play avec des débruiteurs appris sont des méthodes

de pointe pour résoudre les problèmes inverses [2], [3]. Une autre catégorie de méthodes exprime le gradient de la fonction de régularisation comme étant le résidu du débruitage de l'image estimée [4]. Les débruiteurs ont également été utilisés pour remplacer la *fonction de score* dans les approches d'échantillonnage de postérieur [5]-[8]. Ainsi, les approches basées sur le débruiteur constituent sans aucun doute une partie importante de la résolution des problèmes inverses d'une manière générique. Un inconvénient commun à ces approches est que la régularisation est implicitement ou explicitement définie par un débruiteur. Cela implique un hyperparamètre supplémentaire à affiner pour chaque application spécifique, alors qu'en théorie, un régulariseur devrait représenter l'a priori indépendamment du problème inverse spécifique en question.

- **Algorithmes de déroulement profond :** Les méthodes de déroulement profond consistent à itérer un algorithme d'optimisation pour un nombre prédéterminé d'itérations en remplaçant la fonction de régularisation par un réseau de neurones. Le réseau est entraîné de bout en bout pour un problème inverse spécifique. Ces méthodes donnent d'excellents résultats. Cependant, leur principal inconvénient est qu'elles ne sont pas génériques.

D'autre part, un nombre limité de travaux ont abordé le thème des problèmes inverses dans les images omnidirectionnelles, y compris la tâche de débruitage. Cela limite notre capacité à étendre facilement les méthodes de pointe à la sphère.

De plus, les images omnidirectionnelles sont souvent projetées sur des représentations planaires bidimensionnelles (par example en utilisant la projection équirectangulaire [9]) afin de profiter des outils et des méthodes développées pour les images en 2D. Cependant, ces projections entraînent d'importantes distorsions et une résolution spatiale non uniforme. Par conséquent, l'utilisation d'algorithmes conçus pour les images 2D sans tenir compte de la géométrie sphérique conduit à des résultats limités. Une autre approche consiste à travailler directement sur la sphère. Cependant, définir des outils de traitement sur la sphère n'est pas une tâche simple.

L'objectif de cette thèse est de contribuer aux méthodes de régularisation basées sur l'apprentissage profond. Nous nous concentrons principalement sur les approches Plug-and-play, motivées par leur caractère générique et leurs performances exceptionnelles. Nous étudions également l'effet de la combinaison de différentes approches de régularisation. En ce qui concerne les images omnidirectionnelles, nous nous concentrons sur la tâche spécifique du débruitage, étant donné que c'est l'élément clé des méthodes Plug-and-play.

Contributions

Dans cette section, nous discutons des principales contributions de cette thèse et répertorions les publications associées.

Régularisation du Deep Image Prior avec un débruiteur profond

Nous proposons une méthode d'optimisation combinant un débruiteur appris avec le modèle génératif non entraîné, le Deep Image Prior (DIP), dans le cadre de la *Alternating Direction Method of Multipliers* (ADMM). Nous comparons également différents régularisateurs de l'optimisation de DIP, pour des problèmes inverses en imagerie. L'objectif est d'étudier l'effet de la combinaison du DIP non entraîné et d'un régulariseur générique appris de manière supervisée à partir d'une grande collection d'images. Le débruiteur est utilisé comme un opérateur proximal dans le cadre de l'ADMM et peut être appris indépendamment du problème inverse considéré. La régularisation proposée basée sur le débruiteur présente des avantages par rapport aux a priori fabriqués à la main et reste générique, car elle peut être appliquée à différents problèmes inverses.

PnP-ReG : Gradient de régularisation appris pour la descente de gradient Plug-and-play

Le cadre Plug-and-play (PnP) permet d'intégrer des priors avancés de débruitage d'images dans des algorithmes d'optimisation afin de résoudre efficacement diverses tâches de restauration d'images. Cependant, l'algorithme PnP original ne s'applique qu'aux algorithmes proximaux. De plus, un hyperparamètre supplémentaire doit être ajusté pour chaque application lorsque la régularisation est définie par un débruiteur. Nous nous appuyons sur l'hypothèse qu'un débruiteur représente l'opérateur proximal d'un régularisateur différentiable, et nous établissons une relation entre le débruiteur et le gradient du régularisateur correspondant. Nous utilisons cette relation pour entraîner un réseau modélisant directement le gradient du régularisateur, tout en apprenant conjointement le débruiteur correspondant. Nous utilisons ce réseau dans un algorithme de PnP avec la descente de gradient et obtenons de meilleurs résultats par rapport à d'autres approches génériques. Nous montrons également que le réseau de régularisation peut être utilisé comme un réseau pré-entraîné pour la descente de gradient déroulée. Enfin, nous montrons que le débruiteur résultant permet une meilleure convergence de l'ADMM Plug-and-play

en pratique.

SphereDRUNet : un débruiteur sphérique pour les images omnidirectionnelles

Nous abordons le problème du débruitage d'images omnidirectionnelles et nous visons à étudier l'avantage de débruiter directement l'image sphérique plutôt que sa projection. Nous introduisons un nouveau réseau appelé SphereDRUNet pour débruiter des images sphériques en utilisant des outils d'apprentissage profond sur un échantillonnage sphérique. Nous montrons que le débruitage direct de la sphère à l'aide de notre réseau donne de meilleures performances par rapport au débruitage des images équirectangulaires avec un modèle appris de manière similaire. Nous comparons également notre SphereDRUNet à un réseau de convolution basé sur des graphes et confirmons que l'approche de convolution que nous utilisons est plus efficace.

Publications

- R. Fermanian, T. Maugey and C. Guillemot. SphereDRUNet : A Spherical Denoiser for Omnidirectional Images. IEEE 22nd International symposium on mixed and augmented reality adjunct (ISMAR-Adjunct), Sydney, Australia, 2023.
- R. Fermanian, M. Le Pendu and C. Guillemot. PnP-ReG : Learned Regularizing Gradient for Plug-and-Play Gradient Descent. SIAM Journal on Imaging Sciences 16.2 (2023) : 585-613.
- R. Fermanian, M. Le Pendu and C. Guillemot. Regularizing the Deep Image Prior with a Learned Denoiser for Linear Inverse Problems. IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 2021.

Contenu

La suite de cette thèse est divisé en 6 chapitres.

Dans le Chapitre 2, nous élaborons le contexte des problèmes inverses pour l'imagerie. Nous commençons par introduire des problèmes inverses et illustrons plusieurs exemples. Nous discutons ensuite de différentes méthodologies de reconstruction et approches de régularisation.

Dans le chapitre 3, nous proposons une méthode combinant un débruiteur appris et le Deep Image Prior dans le cadre de l'ADMM. Nous discutons de différentes approches existantes qui régularisent le DIP et montrons l'avantage de notre proposition de régularisation appris.

Dans le chapitre 4, nous proposons une nouvelle méthode pour entraîner un réseau modélisant le gradient d'un régulariseur, conjointement avec un débruiteur. En partant de l'hypothèse qu'un débruiteur représente l'opérateur proximal d'un régulariseur différentiable (définissant l'image a priori), nous dérivons une fonction de perte qui relie le débruiteur et le gradient du régulariseur correspondant. Nous utilisons ce réseau dans un algorithme de Plug-and-play avec de la descente de gradient et obtenons de meilleures performances par rapport à d'autres méthodes génériques. Nous utilisons ensuite notre réseau comme stratégie de pré-entraînement dans un algorithme de descente de gradient déroulé.

Dans le Chapitre 5, nous présentons les principaux concepts du traitement d'images omnidirectionnelles. Nous discutons des défis liés à l'acquisition, à la représentation et au traitement des images omnidirectionnelles, et présentons enfin la littérature sur les problèmes inverses.

Dans le chapitre 6, nous proposons SphereDRUNET, un nouveau réseau de débruitage d'images omnidirectionnel, en transférant le DRUNet [2] sur la sphère. Nous montrons que le débruitage d'image donne de meilleures performances lorsqu'il est effectué directement sur la sphère plutôt que via une projection.

Le chapitre 7 conclut cette thèse et propose des perspectives d'études ultérieures.

Chapter 1

Introduction

In this chapter, we present the main problem of interest of this thesis: inverse problems in imaging, with particular attention to omnidirectional imaging. We elaborate on the context and discuss the motivation behind our work. Then, we list our contributions and publications, and outline the remaining of the manuscript.

1.1 Context

Omnidirectional images, also known as 360° images or spherical images, are high-resolution visual data that cover a 360° field of view capturing the light field converging to a single point from all directions. From Augmented Reality (AR) to robotics, omnidirectional images have gained a particular attention in the recent years. These cameras often suffer from inherent distortion and increased noise due to their wide field of view and unique optical characteristics. Consequently, the quality of the captured omnidirectional images can be compromised, limiting the accuracy of subsequent computer vision tasks in AR/MR and necessitating the development of solutions for inverse problems in omnidirectional images.

In a general sense, inverse problems refer to the task of finding the input or underlying factors that have produced a particular set of data or observations. More particularly in the area of imaging, limitations in acquisition systems, transmission or even storage can lead to degraded measurements. Sometimes, acquisition can be costly or even harmful (i.e. exposing the human body to radiation in the case of CT scan), so it is desired to measure as few data points as possible. Hence, inverse problems in imaging concern recovering an image from its degraded measurements that we observe.

Unfortunately, solving inverse problems is not a straightforward task. In fact, the presence of noise and the incomplete measurements lead to challenges in the reconstruction. Moreover, these problems are *ill-posed*, meaning that they do not have a unique solution that is consistent with the observation. In order to address the ill-posed nature,

additional prior knowledge (i.e. regularization) is used to restrict the set of admissible solutions.

This thesis is dedicated to contribute to deep learning-based regularization methods for inverse problems in perspective and omnidirectional imagery.

1.2 Motivation

Over the past few decades, extensive research has been dedicated to solve inverse problems, resulting in various different approaches for their resolution. For instance, the Bayesian approach consists in considering a probabilistic perspective and defining the prior and the data term with probability distributions. The posterior distribution is the probability distribution of the unknown parameters x we wish to recover, conditional on the observed measurement y. It is expressed using Bayes' theorem as:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}, \tag{1.1}$$

where $p(y|x)$ and $p(x)$ represent respectively the *likelihood* of the observations and the *prior*. The posterior distribution represents all potential solutions to the inverse problem in hand, given the measurement. It can be evaluated using different point estimates, such as the MAP or the MMSE.

For a successful reconstruction, the choice of the prior plays a crucial role. Early methods used hand-crafted priors incorporating desired properties of the solution such as Total Variation (TV) [10] or wavelet decomposition [11]. Although these priors have demonstrated significant efficiency, it is not trivial to fully represent the complex nature of natural images using hand-crafted priors.

The breakthrough of deep learning has brought a remarkable shift in different areas of imaging sciences, including the solution of inverse problems. Our focus lies in recent advances on deep regularization of inverse problems, an area that has recently gained a lot of attention. Some of these approaches include:

- **Regularizing using the implicit prior captured by the network architecture:** The Deep Image Prior (DIP) [1] is a unique approach, where the authors show that the network architecture of a generative model itself can be used to regularize inverse problems. The generator network is not trained on a dataset. Instead, it operates by taking a random input vector and iteratively adjusting its randomly

initialized weights to minimize the mean square error between its output and the degraded observation. This means that the network inherently captures an implicit prior during the optimization process, effectively regularizing image restoration tasks. In subsequent studies, researchers have aimed to improve the Deep Image Prior's performance in various ways, particularly by regularizing it using hand-crafted regularizers.

- **Regularizing using the power of denoising engines:** The idea of leveraging denoisers to regularize different inverse problems was first proposed in 2013 by Venkatakrishnan et al. in a framework called *Plug-and-play*, where the proximal operator of a regularizer is replaced by a denoiser. The method was initially proposed with a classical denoiser, but the approach quickly evolved and became an attractive sub-field of research. Deep denoiser-based Plug-and-play approaches are state-of-the-art methods for solving inverse problems [2], [3]. Another category of methods expresses the gradient of the regularization function to be the denoising residual of the estimated image [4]. Denoisers have also been used to replace the *score function* in posterior sampling approaches [5]–[8]. Thus, denoiser-based approaches are unarguably an important part of solving inverse problems in a generic way. A drawback shared by these approaches is that the regularization is implicitly or explicitly defined by a denoiser. This involves an additional hyper-parameter to be fine-tuned for each specific application, whereas in theory, a regularizer should represent the image prior independently of the specific inverse problem at hand.
- **Deep unrolling:** It consists in iterating an optimization algorithm for a predetermined number of iterations by replacing the regularization function by a neural network. The network is trained end-to-end for a specific inverse problem. Deep unrolling methods perform impressively well. However, their main drawback is that they are not generic.

On the other hand, it is worth noting that a limited number of works have addressed the topic of inverse problems in omnidirectional images, including the task of denoising. This limits our capability to easily extend state-of-the-art methods to the sphere.

Moreover, omnidirectional images are often mapped to two-dimensional planar representations (e.g. using equirectangular projection [9]) in order to take advantage of computing tools and methods that have been developed for 2D images. However, these mappings cause significant distortions and non-uniform spatial resolution. Hence, using algorithms designed for 2D images without considering the spherical geometry leads to

limited results. Another approach is to work directly on the sphere. However, defining processing tools on the sphere is not a straightforward task, as it comes with the challenges of spherical geometry.

The goal of this thesis is to contribute to deep learning-based regularization methods. We mainly focus on Plug-and-play approaches, motivated by their genericity and outstanding performance. We also study the effect of combining different regularization approaches. For omnidirectional images, we focus on the specific task of denoising, given that it is the key element of Plug-and-play methods.

1.3 Contributions

In this section, we discuss the main contributions of this thesis and list the associated publications.

Regularization of the Deep Image Prior with a deep denoiser

We propose an optimization method coupling a learned denoiser with the untrained generative model deep image prior (DIP) in the framework of the Alternating Direction Method of Multipliers (ADMM). We also compare different regularizers of DIP optimization, for inverse problems in imaging. The goal is to study the effect of combining the untrained DIP and a generic regularizer learned in a supervised manner from a large collection of images. When placed in the ADMM framework, the denoiser is used as a proximal operator and can be learned independently of the considered inverse problem. The proposed denoiser-based regularization compares favourably with handcrafted priors and remains generic, since it can be applied to different inverse problems.

PnP-ReG: learned regularizing gradient for Plug-and-play gradient descent

The Plug-and-play (PnP) framework makes it possible to integrate advanced image denoising priors into optimization algorithms, to efficiently solve a variety of image restoration tasks. However, the original PnP algorithm only applies to proximal algorithms. When working on the previous contribution, we observed that the parameter tuning of the ADMM algorithm is not a straightforward task. Also, an additional hyper-parameter must be tuned per application when the regularization is defined by a denoiser. We rely

on the assumption that a denoiser represents the proximal operator of a differentiable regularizer, and derive a relation between the denoiser and the corresponding regularizer's gradient. We use this relation to train a network directly modeling the gradient of the regularizer, while jointly training the corresponding denoiser. We use this network in a PnP gradient descent algorithm and obtain better results comparing to other generic approaches. We also show that the regularizer can be used as a pre-trained network for unrolled gradient descent. Lastly, we show that the resulting denoiser allows for a better convergence of the Plug-and-play ADMM.

SphereDRUNet: a spherical denoiser for omnidirectional images

We address the problem of omnidirectional image denoising and we aim to study the advantage of denoising the spherical image directly rather than its mapping. We introduce a novel network called SphereDRUNet to denoise spherical images using deep learning tools on a spherical sampling. We show that denoising directly the sphere using our network gives better performance, compared to denoising the projected equirectangular images with a similarly learned model. We also compare our SphereDRUNet to a graph-based convolution network and confirm that the convolution approach that we use is more efficient.

List of publications

- R. Fermanian, T. Maugey and C. Guillemot. SphereDRUNet: A Spherical Denoiser for Omnidirectional Images. IEEE 22nd International symposium on mixed and augmented reality adjunct (ISMAR-Adjunct), Sydney, Australia, 2023.
- R. Fermanian, M. Le Pendu and C. Guillemot. PnP-ReG: Learned Regularizing Gradient for Plug-and-Play Gradient Descent. SIAM Journal on Imaging Sciences 16.2 (2023): 585-613.
- R. Fermanian, M. Le Pendu and C. Guillemot. Regularizing the Deep Image Prior with a Learned Denoiser for Linear Inverse Problems. IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 2021.

1.4 Outline

The remaining of this thesis is divided into 6 chapters.

In Chapter 2, we elaborate on the background of inverse problems for perspective imagery. We start by introducing inverse problems and illustrate several examples. We then discuss different reconstruction methodologies and regularization approaches.

In Chapter 3, we propose a method coupling a learned denoiser with the Deep Image Prior in the ADMM framework. We discuss different existing approaches that regularize the DIP and show the advantage of our proposed deep regularization.

In Chapter 4, we propose a novel method to train a network modeling the gradient of a regularizer, jointly with a deep denoiser. Based on the assumption that a denoiser represents the proximal operator of an underlying differentiable regularizer (defining the image prior), we derive a loss function that links the denoiser and the gradient of the corresponding regularizer. We use this network in a Plug-and-play gradient descent algorithm and obtain better performance comparing to other generic methods. We then use our network as a pre-training strategy in an unrolled gradient descent algorithm.

In Chapter 5, we present the main concepts of omnidirectional image processing. We discuss the acquisition, representation and processing challenges of omnidirectional images, and finally present the literature of inverse problems.

In Chapter 6, we propose SphereDRUNET, a novel omnidirectional image denoising network, by transferring the DRUNet [2] to the sphere. We show that image denoising gives better performance when it is performed directly on the sphere rather than via a mapping, even though it raises many challenges.

Chapter 7 concludes this thesis and proposes perspectives for further studies.

Part I

Inverse problems in perspective imagery

Chapter 2

An overview of inverse problems in perspective imagery

Picture a doctor seeking to understand the inner workings of the human body from a limited set of diagnostic measurements or an astronomer striving to discover the characteristics of celestial objects, relying only on the faint traces of light that reach its telescope: these are scenarios illustrating instances that represent *inverse problems*. In most real-life applications, we do not have direct measurements of the signal of interest, but only a corrupted version of the original signal. Several phenomena may induce these distortions in practice, such as the acquisition system that is used or the communication channel through which the signal is transmitted. In such cases, we aim to reconstruct the original unknown signal from the distorted observations that we have.

Inverse problems have been broadly studied in the last few decades and several different sort of approaches have been proposed to solve them. This chapter aims, in a first place, to introduce inverse problems in signal and image processing, and secondly, to present an overview of the research and the methodologies that have been proposed to solve them in the literature.

2.1 Inverse Problems

2.1.1 Introduction and problem statement

Inverse problems refer to the task of recovering an image $x \in \mathbb{R}^n$ from its degraded measurements $y \in \mathbb{R}^m$ obtained by a certain degradation model:

$$y = \mathcal{H}(Ax), \tag{2.1}$$

where $A \in \mathbb{R}^{m \times n}$ represents a linear degradation operator depending on the inverse problem and $\mathcal{H}$ is a non-linear model representing noise degradation.

Various different types of non-linear degradation may destroy the signal of interest. For example, the signal may suffer from multiplicative noise [12]–[14], such as speckle noise [15], [16]. The latter can be found in a wide range of systems, including synthetic aperture radar (SAR) images, ultrasound imaging, and many more. Moreover, the statistical nature of electromagnetic waves such as x-rays, visible lights and gamma rays causes shot noise. These ray sources emit a number of photons per unit time and have a random fluctuation of photons. The image has thus a spatial and temporal randomness. The resulting noise is signal-dependent and non-additive and it can be modeled by a Poisson distribution [17]–[20]. Furthermore, acquisition devices usually encounter internal fluctuations. This causes an additive noise that can be represented by a Gaussian distribution [21], [22]. Another possibility is the combination of Poisson and Gaussian noise (called Poisson-Gaussian noise) [23]–[26]. Figure 2.1 shows an image degraded with the aforementioned noise types.

In this work, we consider the case where the noise is additive (independent from the signal itself) and has a Gaussian distribution. Thus, the degraded signal can be obtained by:

$$y = Ax + \eta, \tag{2.2}$$

where $\eta \in \mathbb{R}^m$ represents Additive White Gaussian Noise (AWGN). The probability density function (PDF) of a Gaussian noise follows a normal distribution and is given by:

$$\varphi(\eta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\eta - \mu)^2}{2\sigma^2}\right) \tag{2.3}$$

where μ is the mean of the Gaussian distribution and σ is its standard deviation (i.e. σ^2 is its variance). Figure 2.2 illustrates the PDF of a Gaussian noise.

The rationale behind commonly restricting the noise to Additive White Gaussian Noise (AWGN) is as follows: one might argue that the Poisson distribution is more suited to model imaging noise given that imaging sensors inherently quantify photons. Although this argument is correct, it is worth noting that in situations of high photon counts, the Poisson distribution becomes a Gaussian one [28]. As a result, when dealing with high photon counts, the measurements can be converted into a form akin to Gaussian contamination by using a variance-stabilizing transformation, such as the *Anscombe* transform [29]. Finally, since the formulations become simpler when using the Gaussian distribution,

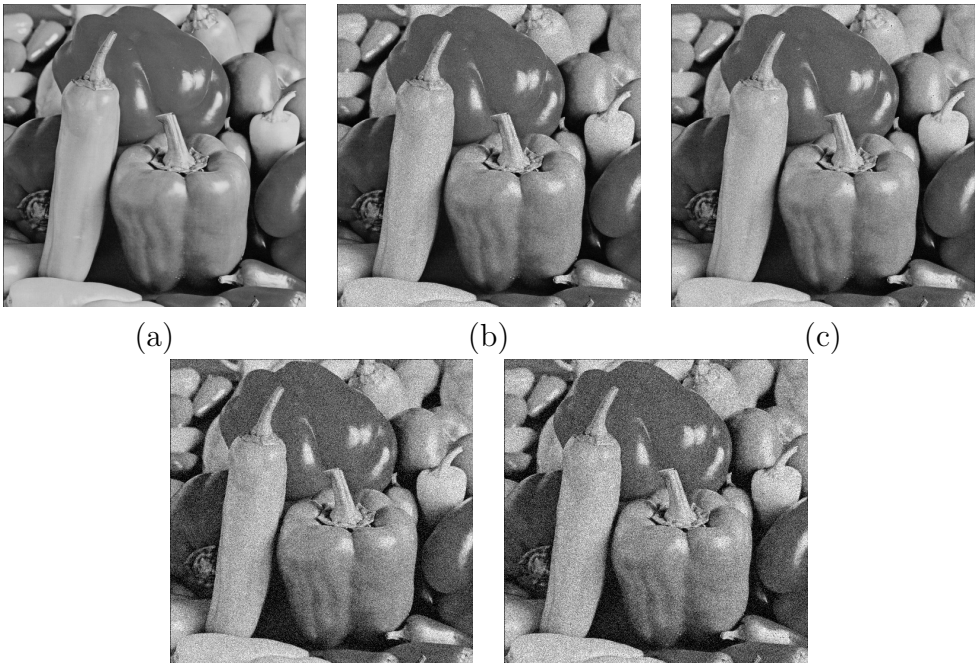


Figure 2.1 – (a) Grayscale version of the pepper image from the Set14 dataset [27] (pixel intensities are ranged from 0 to 255). The original image is degraded with (b) zero mean Speckle noise of standard deviation 0.1, (c) Poisson noise with average intensity of photons equal to 1, (d) additive zero mean Gaussian noise of standard deviation 25 and (e) Poisson-Gaussian noise (poisson noise with average intensity of photons of 1, and additive zero mean Gaussian noise of standard deviation of 25).

researchers have considered the AWGN when designing solutions to inverse problems.

2.1.2 Inverse problems in imaging: examples and definitions

In the image restoration tasks that we consider, we aim to recover the original image x, which is degraded by a certain operation that is represented by the degradation operator A. The latter depends on the inverse problem in hand. In this section, we define some inverse problems in imaging that we will solve in this work, such as denoising, super-resolution, deblurring and pixel-wise inpainting. Solution methodologies will be discussed in the upcoming sections.

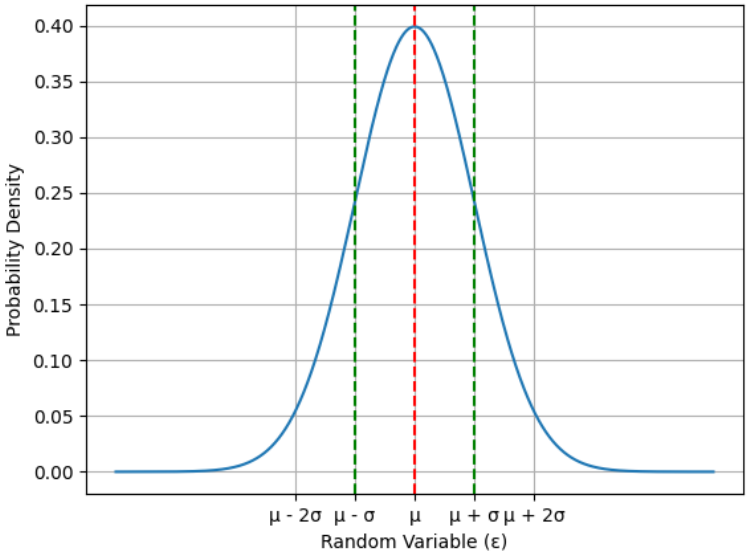


Figure 2.2 – Probability Density Function of Gaussian noise of mean μ and standard deviation σ.

2.1.2.1 Denoising

Image denoising is unarguably the most extensively studied among the inverse problems in imaging. The task of denoising refers to the process of removing unwanted noise from an observed image to enhance its quality and improve visual clarity. The presence of noise in images can be attributed to various factors such as sensor limitations, transmission errors, or environmental conditions during image acquisition. The different types of noise were discussed in the previous section.

In the case where the observed image is degraded with Additive White Gaussian noise of variance σ^2, Eq. 2.2 reduces to $y = x + \eta$, where y is the noisy observation, x is the original image and η is an AWGN. Therefore, the degradation matrix A is simply the identity matrix. Figure 2.3 illustrates the inverse problem of image denoising.

2.1.2.2 Deblurring

Image deblurring refers to the task of recovering a sharp, undistorted version of a blurred image. Blur in images can occur due to various factors, such as camera movement, defocusing, or atmospheric conditions. The objective of image deblurring is to estimate

(a) Original image (b) Noisy image (c) Denoised image

Figure 2.3 – Inverse problem of denoising an image corrupted by a White Gaussian noise of variance $\sigma^2 = 30$.

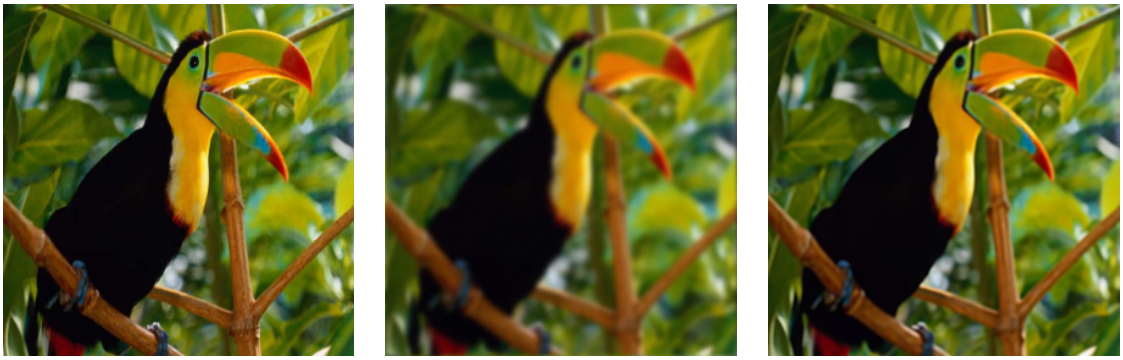
the original, sharp image that might have led to the blurry observation. Figure 2.4 illustrates the inverse problem of deblurring.

The degradation matrix A represents a blurring operator (i.e. convolution), typically followed by the addition of Gaussian noise. The blurring filter used in image deblurring is commonly known as the Point Spread Function (PSF). The PSF characterizes how light from each point in the original scene is spread or blurred across adjacent pixels in the resulting image due to factors like camera shake, defocus, or motion. The PSF is usually represented by a rectangular or a square matrix, with its size increased by the extent of blurring. For instance, in the case of motion blur caused by camera movement, the PSF might be a horizontal line kernel indicating the motion direction. For defocus blur, the PSF may have a circular shape. The PSF of the blurring process must be known or estimated accurately in order to perform image deblurring. Blind deconvolution techniques attempt to estimate the PSF simultaneously with the sharp image, making the reconstruction more complex and challenging.

2.1.2.3 Super-resolution

Super-resolution is the process of enhancing the resolution and quality of a given low-resolution image or video to obtain a higher-resolution version with more details and clarity. The goal is to reconstruct missing high-frequency details, edges, and textures that are lost during the downscaling process of a given image.

Generally, the low-resolution image is obtained after a blurring and a downsampling operation of the high-resolution image. In practice, noise may also be present in the acquired low-resolution. Hence, the degradation operator includes blurring, downsampling,



(a) Original image (b) Blurred image (c) Deblurred image

Figure 2.4 – Inverse problem of deblurring. The blurred image is generated using an isotropic Gaussian kernel followed by adding Gaussian noise of standard deviation $\sigma = \sqrt{2}$.

and possibly noise. Figure 2.5 shows an example of super-resolution.

(a) Original image (b) Low-resolution image (c) Super-resolved image

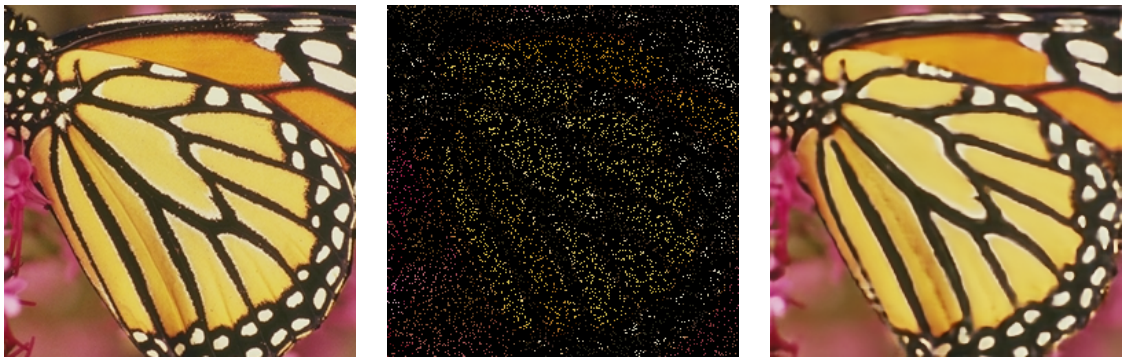
Figure 2.5 – Inverse problem of super-resolution. The low-resolution image is obtained by performing Gaussian blur over the original image and downscaling it with a factor 2.

2.1.2.4 Pixel-wise inpainting

Pixel-wise inpainting, also known as image completion, is a task in image processing that aims to fill in missing pixels in an image. The goal is to generate a complete and visually plausible image from the partially observed or degraded version.

The missing pixels are often represented by masked pixels, where the pixel values are unknown or marked as invalid. The task is to estimate the missing pixel values and complete the image in a way that blends seamlessly with the surrounding content. The

degraded image is typically obtained by multiplying the original image with a binary mask. This binary mask is used to indicate the pixels that need to be inpainted or completed. The mask is a matrix of the same size as the original image, with values of 0 or 1. A value of 0 in the mask denotes a missing or corrupted pixel, while a value of 1 represents a valid pixel that should be kept unchanged. Figure 2.6 shows an example of the inverse problem of completion.



(a) Original image (b) Masked image (c) Inpainted image

Figure 2.6 – Inverse problem of pixel-wise inpainting. The degraded image has a known pixel rate of $p = 10\%$.

2.1.3 An ill-posed problem and need for regularization

The task of solving inverse problems defined by Eq. 2.2 is equivalent to finding an optimal solution $\hat{x}$ that best represents the original signal of interest x using the measurement y. In practice, even in relatively simpler applications, obtaining a restored image of good quality is not as easy as it seems. This is due to the ill-posed nature of inverse problems, that we will elaborate in this section.

A problem is considered to be mathematically well-posed according to Hadamard [30] if the solution $\hat{x}$ satisfies all three conditions below:

- A solution exists
- The solution is unique
- The solution is stable (the solution depends continuously on the measurement)

The first condition requires the system to be consistent. The second condition restricts the solution to be unique. It ensures that there is only one possible solution that can be

determined from the available data. In order to meet this condition, the system should not be under-determined (e.g. the degradation matrix should not have $m < n$). Moreover, the stability of the solution with respect to the input data ensures that minor changes (i.e. noise) in the observed data do not drastically alter the solution. Stability is essential to prevent the problem from becoming ill-posed, where small errors in the data could lead to significant inaccuracies or even to the non-existence of the solution.

Inverse problems are ill-posed because they do not satisfy all of the aforementioned conditions. To address this ill-posed nature, regularization techniques are employed. Regularization involves incorporating additional prior knowledge or constraints into the problem to guide the solution process. By imposing constraints, such as smoothness or sparsity on the unknowns, regularization helps to narrow down the solution space, making it more stable and robust. The topic of regularization will be further detailed in the following sections.

2.2 Reconstruction approaches for solving inverse problems

To this end, we have discussed inverse problems without actually diving into the actual task of their reconstruction. So, how can we solve inverse problems? From classical approaches such as simple interpolation and filtering to deep learning methods, the answer to this question has evolved over time with the advancements in signal and image processing. In this section, we elaborate the most common reconstruction approaches that have been adopted in the literature.

2.2.1 Variational approach for solving inverse problems

A natural and naive approach to solve inverse problems is to apply a function enforcing similarity between the observation and the searched solution (e.g. the least squares method), as follows:

$$\hat{x} = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ \mathcal{L}(Ax, y), \tag{2.4}$$

However, since inverse problems are ill-posed, we use a regularizing function as discussed before. The variational approach for solving inverse problem amounts to solving an opti-

mization problem defined by:

$$\hat{x} \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ \mathcal{L}(Ax, y) + \lambda \phi(x), \tag{2.5}$$

where $\mathcal{L}(Ax, y)$ represents the data-fidelity term, $\phi(x)$ is the regularization function and $\lambda > 0$ balances the trade-off between the two terms. This parameter should be tuned in order to compromise between the two terms.

Note that the most common choice for $\mathcal{L}(Ax, y)$ is the $\mathcal{L}_2$ distance form, thus we generally get:

$$\hat{x} \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - Ax\|_2^2 + \lambda \phi(x). \tag{2.6}$$

The problem in Eq. 2.5 usually does not have a closed-form solution except in some simple cases, for instance when both the data-fidelity term and the regularization are quadratic. Consequently, the variational approach involves solving an optimization problem by using iterative schemes. Section 2.3 provides an overview of some iterative algorithms that are commonly used to solve variational problems.

The choice of the regularization function plays a crucial role in the solution of inverse problems. The prior should balance between incorporating desired properties of the solution and avoiding an excessive computational burden. In the following, we provide a non exhaustive discussion about some of the traditional priors (often referred to as "hand-crafted" priors) that have been designed throughout years.

- **Tikhonov Regularization:** Tikhonov regularization [31] is probably one of the most commonly used classical regularization methods to solve inverse problems. The regularization expression is given as $\phi(x) = \|\Gamma x\|_2^2$, where Γ is the Tikhonov operator promoting some desirable properties. For example, if the signal is believed to be smooth, Γ can assume the role of a high-pass operator like a Laplacian filter (i.e. $\phi(x) = \|Lx\|_2^2$). A particular case of Tikhonov regularization is the energy regularization (L_2 regularization) when $\Gamma = I$. Thus ϕ reduces to $\phi(x) = \|x\|_2^2$ encouraging solutions with smaller norms.
- **Weighted Smoothness:** Imposing a uniform constraint on the smoothness of the solution may result in overly-smoothed regions in the image. To overcome this, a weighted smoothing regularization can be considered, such as in [32], [33]. By assigning varying weights, the regularization process can effectively adapt to the inherent structure of the data: a high weight is assigned to regions where the

assumption of smoothness is believed to be valid, while a lower weight is allocated to textured regions or edges. Forming a diagonal weight matrix W containing the assigned weights in the diagonal, and considering the Laplacian regularization, the regularization function becomes $\phi(x) = \|Lx\|_W^2$.

- **Total-Variation:** Total-Variation [10] is a regularization function quantifying the intensity variation or discontinuity across neighboring pixels, hence promoting gradient sparsity. In the anisotropic form, total variation computes the $\mathcal{L}_1$ norm of the image gradient: $TV_{aniso}(u) = \sum_{i=1}^{n} \ |(D_h u)_i| + |(D_v u)_i|$, while isotropic TV corresponds to the $\mathcal{L}_2$ norm of the gradient vector: $TV_{iso}(u) = \sum_{i=1}^{n} \sqrt{(D_h u)_i^2 + (D_v u)_i^2}$. The variables D_h and D_v denote the first order finite difference discrete operators along the horizontal and vertical axes. Extensions of this line of research can be found in [11], [34].
- **Sparsity methods:** A sparse signal refers to a type of signal in which most of the values are either zero or very close to zero, while only a few values are non-zero and carry a significant information. For example, wavelets provide a compact representation of data by using a minimal number of coefficients, thereby introducing the concept of sparsity, which conveniently links to the regularization $\phi(x) = \|Wx\|_1$ encouraging a reduction in the presence of non-zero Wavelet coefficients [35]. The concept of sparsity gained further interest as subsequent research explored redundant and learned representations. Ideal images are assumed to be linear combinations of atoms from a pre-defined dictionary D, denoted as $x = D\alpha$. Sparsity can hence be enforced through $\phi(x) = \|\alpha\|_0$. Different algorithms have been developed along these lines considering global and patch-based dictionaries [36]–[40].
- **Self-similarity:** The concept of self-similarity in image processing is rooted in the idea that small patches within an image often exhibit strong similarities to other patches in the overall image content. Hence, treating these similar patches collectively is believed to contribute to enhanced restoration. For example, $\phi(x) = \sum_i \sum_{j \in \Lambda(i)} d\{R_i x, R_j x\}$ promotes proximity between $R_i x$ and $R_j x$, where $R_i x$ denotes the patch extracted from the image at location i, $\Lambda(i)$ refers to the set of indices in the neighborhood of i and d denotes a dissimilarity score.
- **Low-rank assumption:** This approach is very similar to self-similarity regularization, but rather than imposing proximity between a set of closely related patches, a low-rank structure is forced on a matrix having these related patches as columns. This infers that these patches are spanned by a few main directions. For instance,

let $Z_{\Lambda(k)}$ be a matrix denoting a group of similar patches extracted from the image, where $\Lambda(.)$ is the collection of related patches and k indicates the specific patch group. Let $\|.\|_*$ denote the nuclear norm, which is used as a measure of the rank of the matrix. The low-rank assumption can be imposed by $\phi(x) = \sum_k \|Z_{\Lambda(k)}\|_*$ [41], [42].

It's important to note that the landscape of regularization techniques is vast and continually evolving. Hand-crafted regularization approaches have historically demonstrated significant efficiency in addressing specific challenges. However, in pursuit of enhanced performance, current research has shifted its focus towards learning prior knowledge. We detail the idea of leveraging deep learning tools to regularize inverse problems in Section 2.4.3.

2.2.2 Learning approach for solving inverse problems: Neural network regression

With the advancement of deep learning and the success of convolutional neural networks in different computer vision tasks, researchers started to develop learning-based solutions for inverse problems. The simplest approach is to train problem-specific neural networks that approximate the mapping between the space of measurements (i.e. degraded images) and the corresponding solution space (i.e. ground-truth images), without having knowledge of the degradation operator A (an agnostic learner). In fact, using a large amount of training data, the network is able to learn the patterns and relationships between the distorted inputs and the corresponding clean outputs. As a result, the network can effectively reconstruct the high-quality outputs from given corrupted inputs. This approach yielded a significant performance improvement in the field of image restoration.

A wide range of network architectures have been designed and trained for different inverse problems, such as DnCNN [43], NLRN [44], FOCNet [45] and DRUNet [2] for image denoising, SRCNN [46], SRGAN [47], ESRGAN [48] and RCAN [49] for Super-Resolution and DCNN [50] for blind deconvolution.

However, the success of these networks highly depends on the complexity of the forward operator A and the available amount of training data. For instance, in the medical imaging field, obtaining large datasets can be particularly challenging due to privacy concerns and ethical considerations. Another drawback of this end-to-end learning procedure is that

the network has to be re-trained again if a single parameter of the degradation process changes. Therefore, these limitations highlight the necessity for more adaptable and generalized methods in handling diverse degradation scenarios.

2.2.3 Bayesian approach for solving inverse problems

A common approach to solve inverse problems consists in considering a probabilistic perspective. Rather than treating x and y as constant values, it is assumed that the noise in the measured data causes some randomness, which further induces uncertainty in the restoration. It can be inferred that x and y are random variables having probability distributions. In this setting, the reconstructed image from the measurements can be calculated through its posterior density $p_{x|y}(x|y)$. For notation simplicity, we will denote $p_u(u)$ as $p(u)$ in the remainder of the manuscript. This approach is the core of this thesis.

Bayes' theorem expresses the posterior density $p(x|y)$ using the likelihood and the prior density:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}, \tag{2.7}$$

where $p(y|x)$ is the probability density of the measurement y given x, representing the likelihood of the observations and $p(x)$ represents the prior distribution. $p(y)$ is the distribution of y and can be considered as a normalization constant for $p(x|y)$, as it is independent of x.

In the assumption of an Additive White Gaussian Noise η with standard deviation σ, the noise is assumed to be independent and identically distributed across pixels, which means that each pixel y_i follows a Gaussian distribution with mean Ax_i and standard deviation σ. Since the noise in each pixel is independent, the joint probability density function of all noisy pixels y given the true image x is the product of the PDFs of each pixel. Therefore, the likelihood can be expressed as:

$$\begin{aligned} p(y|x) &= \prod_{i=0}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - Ax_i)^2}{2\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{\sum_{i=0}^{n}(y_i - Ax_i)^2}{2\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}\|y - Ax\|_2^2}. \end{aligned} \tag{2.8}$$

It therefore follows that $p(y|x) \propto e^{-\frac{1}{2\sigma^2}\|y - Ax\|_2^2}$.

Once the likelihood is derived and the prior is chosen, the posterior density can be evaluated using various point estimates, such as Maximum A Posteriori (MAP) and the Minimum Mean Square Error (MMSE).

The bayesian estimator can be obtained by calculating the minimum expectation of a certain cost function $\mathcal{C}(\hat{x} - x)$, where the latter assesses the accuracy of the estimated $\hat{x}$ in comparison to the unknown signal x with posterior distribution $p(x|y)$:

$$\begin{aligned} \hat{x}_C &\in \underset{\hat{x}}{\operatorname{argmin}}\, \mathbb{E}_{x|y}[\mathcal{C}(\hat{x} - x)], \\ \hat{x}_C &\in \underset{\hat{x}}{\operatorname{argmin}} \int \mathcal{C}(\hat{x} - x) p(x|y)\, dx. \end{aligned} \tag{2.9}$$

Minimum Mean Squared Error estimator (MMSE)

For the cost function $\mathcal{C}(\hat{x} - x) = \|\hat{x} - x\|_2^2$, solving Eq. 2.9 amounts to finding the Minimum Mean Squared Error (MMSE) estimate. Having the following minimization problem:

$$\hat{x}_{MMSE} = \underset{\hat{x}}{\operatorname{argmin}} \int \|\hat{x} - x\|_2^2 \, p(x|y) dx, \tag{2.10}$$

Let us note $\mathcal{Q}(\hat{x}) = \int \|\hat{x} - x\|_2^2 \, p(x|y) dx$. We can thus state that finding the minimizer in Eq. 2.10 amounts to solving $\frac{\partial \mathcal{Q}(\hat{x})}{\partial \hat{x}} = 0$.

$$\begin{aligned} \frac{\partial \mathcal{Q}(\hat{x})}{\partial \hat{x}} &= \int \frac{\partial \|\hat{x} - x\|_2^2}{\partial \hat{x}} p(x|y) dx, \\ &= \int 2(\hat{x} - x) p(x|y) dx, \\ &= 2\hat{x} - 2 \int x p(x|y) dx = 0. \end{aligned} \tag{2.11}$$

Therefore, the MMSE estimator is given by the posterior mean, namely:

$$\hat{x}_{MMSE} = \int x p(x|y) dx. \tag{2.12}$$

While the formulation seems simple and clear, computing the MMSE estimate can be challenging, as it consists in estimating high dimensional integrals. Sampling algorithms are often used in order to perform numerical integration.

Maximum A Posteriori estimator (MAP)

Maximum A Posteriori estimator is derived from choosing the *Hit-or-miss* cost function:

$$\mathcal{C}(\hat{x}-x) = \begin{cases} 0 & \text{if } \|\hat{x}-x\|_2 < \delta \\ 1 & \text{if } \|\hat{x}-x\|_2 \geqslant \delta, \end{cases} \tag{2.13}$$

$\delta > 0$ being a predefined threshold.

In this case, the estimation in Eq. 2.9 reduces to:

$$\hat{x}_{MAP} \in \underset{\hat{x}}{\operatorname{argmin}}\, 1 - \int_{\|\hat{x}-x\|_2<\delta} p(x|y)dx. \tag{2.14}$$

As $\delta \to 0$ we can observe that this minimization is obtained when the posterior distribution is maximized. Therefore, the estimation is computed by maximizing the probability density of the posterior $p(x|y)$:

$$\begin{aligned} &\underset{x}{\operatorname{argmax}}\, p(x|y) \\ &\underset{x}{\operatorname{argmax}}\, \frac{p(y|x)p(x)}{p(y)} \\ &\underset{x}{\operatorname{argmin}} - \log p(y|x) - \log p(x) \end{aligned} \tag{2.15}$$

As established in Eq. 2.8, $p(y|x) \propto e^{-\frac{1}{2\sigma^2}\|y-Ax\|_2^2}$ in the case of AWGN. Eq. 2.15 is therefore equivalent to:

$$\underset{x}{\operatorname{argmin}} - \log(e^{-\frac{1}{2\sigma^2}\|y-Ax\|_2^2}) + \phi(x) \tag{2.16}$$

We finally get the MAP estimation:

$$\hat{x}_{MAP} \in \underset{x}{\operatorname{argmin}}\, \frac{1}{2}\|y-Ax\|_2^2 + \sigma^2\phi(x). \tag{2.17}$$

By combining the likelihood (i.e. data fidelity term) and the prior (i.e. regularization term), the MAP estimate effectively balances the trade-off between fitting the data and conforming to prior knowledge, leading to a reliable restoration despite the degraded observation. One key point is that the term $-\log p(x)$ should assume a closed-form formulation that facilitates feasible numerical optimization. Consequently, most attempts at describing $p(x)$ have opted for the Gibbs distribution representation [51], $p(x) = c.e^{-\phi(x)}$,

thereby redirecting our attention towards the energy function $\phi(x)$ rather than the prior distribution $p(x)$. It's worth noting that within the research community, there's a prevalent simplification where the term 'prior' is sometimes used interchangeably to refer to the regularization function (or even a denoiser which is also related to the prior distribution).

In most cases, Eq. 2.17 does not have a closed-form solution, therefore the estimation is computed using iterative optimization methods, as in the variational case. In Section 2.3, we discuss some optimization methods that can be used to solve common variational problems.

2.3 Optimization algorithms

This section aims to provide a brief overview of some of the optimization methods used to solve common variational problems. While numerous categories of methods exist, our focus here is primarily on commonly employed algorithms in the recent literature of inverse problems, notably first-order optimization methods and proximal methods.

2.3.1 Derivative-based optimization algorithms: first-order methods

Let's consider minimizing functions that are smooth and differentiable. In this case, derivative-based algorithms, such as gradient descent, are effective tools for solving such optimization problems.

2.3.1.1 Gradient Descent

Let us consider minimizing the following optimization problem:

$$\hat{x} = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} f(x). \tag{2.18}$$

If f is a convex differentiable function with a gradient that is β-Lipschitz, then we can solve Eq. 2.18 by finding $\hat{x}$ such that:

$$\nabla f(\hat{x}) = 0. \tag{2.19}$$

Thus, we can easily solve 2.18 in the case where ∇f is sufficiently straightforward to

derive a closed-form solution from Eq. 2.19 which is computationally tractable. However, finding such a solution can become challenging when f has a more complex form and we can no longer solve 2.19. In this case, the solution of the minimization can be defined as the limit of a sequence constructed iteratively through some algorithm. One category of methods to find an iterative solution of the minimization problem defined in Eq. 2.18 is the gradient-based algorithms [52].

Gradient methods are based on the principle that for any x, the negative gradient $-\nabla f(x)$ points in the direction of the steepest decrease, making it a favorable choice for finding the minimizer of f. Let γ_k be the step-size of the gradient descent at iteration k and $K \in \mathbb{N}$ be the total number of iterations until convergence. Starting with an initial point $x_0 \in \mathbb{R}^n$, we can derive $\hat{x}$ by applying the Algorithm 1:

Algorithm 1 Gradient Descent

1: Initialize $x_0 \in \mathbb{R}^n$
2: **for** $k \leftarrow 0$ to $K-1$ **do**
3: $\quad x_{k+1} \leftarrow x_k - \gamma_k \nabla f(x_k)$
4: **end for**

2.3.1.2 Projected Gradient Descent

Let us now consider a constrained optimization problem, where the objective function needs to be minimized while satisfying a certain constraint.

$$\hat{x} = \underset{\substack{x \in \mathbb{R}^n \\ s.t.\ g(x)=0}}{\operatorname{argmin}} f(x), \tag{2.20}$$

where f is the original objective function to be minimized and g is a convex function defining the constraint or the set of feasible solutions.

Projected gradient descent consists in projecting the current iterate onto the feasible set at each iteration, to ensure that the constraints are satisfied.

Algorithm 2 Projected Gradient Descent

1: Initialize $x_0 \in \mathbb{R}^n$
2: **for** $k \leftarrow 0$ to $K-1$ **do**
3: $\quad x_{k+\frac{1}{2}} \leftarrow x_k - \gamma_k \nabla f(x_k)$
4: $\quad x_{k+1} \leftarrow \mathrm{Proj}_{g(x)}(x_{k+\frac{1}{2}})$
5: **end for**

Algorithm 2 describes the method of projected gradient descent. The projection operator is defined as

$$\text{Proj}_{g(x)}(x_k) = \underset{\substack{\hat{x}\in\mathbb{R}^n \\ s.t.\ g(\hat{x})=0}}{\text{argmin}} \|\hat{x} - x_k\|_2, \tag{2.21}$$

where the feasible set is defined by the function g. The projection itself is an optimization problem to be solved.

2.3.2 Proximal algorithms

Proximal Operator

Before diving into proximal algorithms, let us first define a proximal operator. Consider $\mathcal{V}$ to be a real Hilbert space and $\Gamma_0(\mathcal{V})$ to represent the set of lower semi-continuous convex functions from $\mathcal{V}$ to $[-\infty, +\infty]$. Let $h \in \Gamma_0(\mathcal{V})$ be a closed proper convex function. Jean Jacques Moreau [53] proposed that for each $x \in \mathcal{V}$, the minimization on the right-hand side of Eq. 2.22 has a unique solution and is denoted by $\text{prox}_{\lambda h}(u)$.

$$\text{prox}_{\lambda h}(u) = \underset{x\in\mathcal{V}}{\text{argmin}}\, h(x) + \frac{1}{2\lambda}\|x - u\|_2^2. \tag{2.22}$$

The term $\text{prox}_{\lambda h}(u)$ is called the proximal operator of the scaled function λh, or the proximal operator of h with parameter λ. Let us note that, in practice, if h is not convex, there may be an abuse of notation, as Eq. 2.22 will have multiple solutions.

If h is differentiable and λ is small, the proximal operator of λh can also be evaluated by doing a gradient step on h:

$$\text{prox}_{\lambda h}(u) \approx u - \lambda \nabla h(u). \tag{2.23}$$

Proximal algorithms are advantageous tools to solve non-smooth minimization problems. The base operation of a proximal algorithm is to compute the proximal operator of a function, which reduces to solve a simple convex optimization sub-problem. The latter often has a closed-form solution or can be easily solved with simple methods. Examples of proximal algorithms are presented next.

2.3.2.1 Proximal Gradient Descent

Let's consider a minimization optimization as follows:

$$\hat{x} = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} f(x) + g(x), \tag{2.24}$$

where f is a smooth and differentiable function whereas g is a non-smooth, possibly non-differentiable function that enforces some constraint.

Proximal gradient descent (also called Forward-Backward Splitting algorithm) consists in computing the gradient of the smooth part of the objective function at the current point x_k (i.e. $\nabla f(x_k)$) and updating the latter, then evaluating the proximal operator at the updated point $x_k - \gamma_k \nabla f(x_k)$. The proximal operator is used to handle the non-smooth regularization term g. These 2 steps are performed repeatedly at each iteration until reaching convergence. Algorithm 3 details the proximal gradient descent algorithm.

Algorithm 3 Proximal Gradient Descent

1: Initialize $x_0 \in \mathbb{R}^n$
2: **for** $k \leftarrow 0$ to $K - 1$ **do**
3: $\quad x_{k+\frac{1}{2}} \leftarrow x_k - \gamma_k \nabla f(x_k)$
4: $\quad x_{k+1} \leftarrow \text{Prox}_{\gamma_k g}(x_{k+\frac{1}{2}})$
5: **end for**

2.3.2.2 Half-Quadratic Splitting

Half-Quadratic Splitting (HQS) [54] is a simple yet effective splitting method for solving optimization problems. The basic idea behind HQS is to split the original optimization problem into two sub-problems, each of which addressing one part of the original problem. The algorithm first reformulates the problem by introducing an auxiliary variable to the Eq. 2.24, as follows:

$$\begin{aligned} &\underset{x,z \in \mathbb{R}^n}{\operatorname{argmin}} \quad f(x) + g(z). \\ &\text{subject to} \quad x = z \end{aligned} \tag{2.25}$$

The HQS iteration consists in alternatively applying the proximal operators of f and g, as follows:

$$x^{k+1} = \underset{x}{\operatorname{argmin}}\, f(x) + \frac{\rho^k}{2}\|x - z^k\|_2^2 = \operatorname{prox}_{\frac{1}{\rho^k} f}(z^k), \tag{2.26}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}}\, g(z) + \frac{\rho^k}{2}\|z - x^{k+1}\|_2^2 = \operatorname{prox}_{\frac{1}{\rho^k} g}(x^{k+1}), \tag{2.27}$$

where ρ^k is a positive penalty parameter for weighting the penalty terms. In order to enforce the constraint $x = z$, it is necessary to progressively increase the penalty parameter ρ^k towards infinity within the HQS iterations.

2.3.2.3 Alternating Direction Method of Multipliers

Given the optimization problem given in Eq. 2.24, the Alternating Direction Method of Multipliers (ADMM) algorithm aims to solve the problem by first decoupling the data term and the prior term by adding an auxiliary variable z just like the HQS, yielding a constrained optimization problem which is equivalent to 2.25.

In order to deal with the constraint $x = z$, the ADMM introduces a Lagrange multiplier: the original problem is reformulated into an augmented Lagrangian form, which includes the Lagrange multiplier (dual variable) associated with the constraints. The augmented Lagrangian of the optimization problem with the equality constraint $x = z$ is given by:

$$\begin{aligned} \mathcal{L}(x, z, l) &= f(x) + g(z) + l^T(x - z) + \frac{\rho}{2}\|x - z\|_2^2, \\ &= f(x) + g(z) + \frac{\rho}{2}\|x - z + \frac{l}{\rho}\|_2^2 - \frac{1}{2\rho}\|l\|_2^2, \end{aligned} \tag{2.28}$$

where l is the dual variable and ρ is a positive penalty parameter.

Each ADMM iteration then consists in an alternate minimization over x and z, as follows:

$$x^{k+1} = \arg\min_x \mathcal{L}(x, z^k, l^k) = \operatorname{prox}_{\frac{1}{\rho^k} f}(z^k - \frac{l^k}{\rho^k}) \tag{2.29}$$

$$z^{k+1} = \arg\min_z \mathcal{L}(x^{k+1}, z, l^k) = \operatorname{prox}_{\frac{1}{\rho^k} g}(x^{k+1} + \frac{l^k}{\rho^k}) \tag{2.30}$$

$$l^{k+1} = l^k + \rho^k(x^{k+1} - z^{k+1}), \tag{2.31}$$

where the dual variable is typically zero-initialized. The penalty parameter ρ^k can be

adjusted during the iterations to balance between convergence speed and accuracy. In Appendix A.1, we will take a closer look at the ADMM algorithm, providing detailed insights and exploring parameter settings in the context of the Plug-and-play framework.

Closed-form solution of the data-fidelity optimization sub-problem

Both ADMM and HQS split the problem into two sub-problems. Hence, they can be used to solve the MAP estimation for linear inverse problems in Eq. 2.17, by taking $f(x) = \frac{1}{2}\|y - Ax\|_2^2$ and $g(x) = \sigma^2\phi(x)$. Therefore, the data-fidelity term f and the prior term g are handled as independent sub-problems that consist in evaluating the corresponding proximal operator $\text{prox}_{\frac{1}{\rho^k}f}$ (Eq. 2.26 and Eq. 2.29) and the regularization term $\text{prox}_{\frac{1}{\rho^k}g}$ (Eq. 2.27 and Eq. 2.30).

In particular, for the data fidelity term, the proximal operator is expressed as:

$$\text{prox}_{\frac{1}{\rho^k}f}(u) = \underset{x}{\text{argmin}}\ \mathcal{Q}(x,u) \tag{2.32}$$

$$\text{with}\ \ \mathcal{Q}(x,u) = \frac{1}{2}\|y - Ax\|_2^2 + \frac{\rho^k}{2}\|x - u\|_2^2. \tag{2.33}$$

One can thus derive a general closed-form solution by finding x that cancels out $\nabla_x \mathcal{Q}(x,u)$. This gives:

$$\text{prox}_{\frac{1}{\rho^k}f}(u) = (A^TA + \rho^k I)^{-1}(A^Ty + \rho^k u). \tag{2.34}$$

In general, calculating the inverse directly might be computationally expensive, especially for large matrices. However, for many specific problems such as super-resolution, deblurring, and denoising, the matrix A can often be represented in a particular form (e.g. diagonal matrix or block-diagonal matrix) which simplifies the computation and make the closed-form solution in Eq. 2.34 feasible.

2.4 Learning to Regularize inverse problems

The choice of the regularization function plays a crucial role in the efficient reconstruction of inverse problems. The advent of deep learning has remarkably transformed various domains, including the field of inverse problems and its regularization. In the last years, researchers have elaborated different ways to use learned priors, be it through generative models, deep denoisers or other approaches. In this section, we elaborate some

of these methodologies.

2.4.1 Regularizing by using the implicit prior captured by a neural network

A unique approach called the Deep Image Prior (DIP) was introduced by Ulyanov et al. [1], where it was shown that the CNN structure itself can inherently regularize image restoration without having to be trained on a specific dataset. DIP represents an untrained generative model, that takes a random input vector (i.e. noise) and adjusts its randomly initialized weights to minimize the mean square error (MSE) between its output and the degraded observation. The regularization typically employed in conventional methods is replaced by the implicit prior captured by the architecture of the generator network. The DIP formulation can be expressed as follows:

$$\Theta^* \in \underset{\Theta}{\operatorname{argmin}} \; E(f_\Theta(z); x_0), \quad s.t.\; x^* = f_{\Theta^*}(z) \tag{2.35}$$

where f_Θ is the generator network with random input z and Θ its parameters to be optimized. x_0 is the degraded observation. The goal is to find the optimal set of parameters Θ that minimize the task-dependent data term $E(f_\Theta(z); x_0)$.

To effectively control the regularization in DIP, early stopping is commonly employed during the optimization process, which prevents the network from overfitting to the observed degradation. In practice, this early stopping is hard to control.

Subsequent works have aimed to enhance the performance of the Deep Image Prior by different methods. We will dedicate our next chapter to discuss these advancements and to study the effect of combining the DIP regularization with a deep regularization.

2.4.2 Regularizing by leveraging the power of denoisers

Image denoising has garnered significant research attention in the literature. Clarivate Web-Of-Science (WoS) identifies nearly 30,000 papers targeting image denoising published in the last 25 years, with an important increase in the number of publications over time. This growth in interest can be attributed not only to the crucial role of image denoising in pre-processing, but also to the recent revelation that denoisers can be exploited for broader applications, such as solving general inverse problems or synthesizing high-quality images. In the following, we discuss the ability to deploy denoising engines for regularizing inverse

problems.

2.4.2.1 Plug-and-play (PnP) framework

The Plug-and-play framework revolutionized traditional inverse problem-solving by seamlessly incorporating the capabilities of advanced denoisers. First, let's recall the equation of the proximal operator of a function h with parameter λ:

$$\text{prox}_{\lambda h}(u) = \underset{x \in \mathbb{R}^n}{\text{argmin}} \frac{1}{2} \|x - u\|_2^2 + \lambda h(x). \tag{2.36}$$

Observing Eq. 2.17, we can state that the proximal operator in Eq. 2.36 can be seen as a particular case of inverse problem where the degradation operator is the identity matrix, and the degradation only consists in the addition of White Gaussian Noise of standard deviation $\sqrt{\lambda}$.

Inspired by this perspective, and motivated by the significant advancements in the field of image denoising, Venkatakrishnan el al. [55] introduced the Plug-and-play (PnP) framework, which stimulated an important sub-field of research in image restoration. The authors suggest to solve different types of inverse problems by replacing the proximal operator of the regularization term by a powerful denoiser in an ADMM framework.

In the context of MAP estimation, the constrained optimization can be formulated as:

$$\begin{aligned} \hat{\boldsymbol{x}} = \underset{\boldsymbol{x},\boldsymbol{z}}{\text{argmin}} \quad & \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sigma^2 \phi(\boldsymbol{z}), \\ \text{subject to} \quad & \boldsymbol{x} = \boldsymbol{z}. \end{aligned} \tag{2.37}$$

Each ADMM iteration then consists in an alternate minimization over $\boldsymbol{x}$ and $\boldsymbol{z}$ with the introduction of an additional variable $\boldsymbol{l}$ called dual variable as well as a penalty parameter ρ, as follows:

$$\boldsymbol{x}^{\boldsymbol{k+1}} = \underset{\boldsymbol{x}}{\text{argmin}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \rho^k \left\| \boldsymbol{x} - \left(\boldsymbol{z}^{\boldsymbol{k}} - \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k} \right) \right\|_2^2, \tag{2.38}$$

$$\boldsymbol{z}^{\boldsymbol{k+1}} = \underset{\boldsymbol{z}}{\text{argmin}} \frac{1}{2} \left\| \boldsymbol{z} - \left(\boldsymbol{x}^{\boldsymbol{k+1}} + \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k} \right) \right\|_2^2 + \frac{\sigma^2}{\rho^k} \phi(\boldsymbol{z}) = \text{prox}_{\frac{\sigma^2}{\rho^k}\phi}(\boldsymbol{x}^{\boldsymbol{k+1}} + \boldsymbol{l}^{\boldsymbol{k}}/\rho), \tag{2.39}$$

$$\boldsymbol{l}^{\boldsymbol{k+1}} = \boldsymbol{l}^{\boldsymbol{k}} + \rho^k (\boldsymbol{x}^{\boldsymbol{k+1}} - \boldsymbol{z}^{\boldsymbol{k+1}}), \tag{2.40}$$

where the dual variable is typically zero-initialized.

The Plug-and-play ADMM thus replaces the z-update in 2.39 by the following denoising step:

$$\boldsymbol{z^{k+1}} = \mathcal{D}_s(\boldsymbol{x^{k+1}} + \boldsymbol{l^k}/\rho), \tag{2.41}$$

where D is a denoiser suitable for Gaussian noise of standard deviation s, with $s = \sqrt{\sigma^2/\rho} = \sigma/\sqrt{\rho}$.

While early works used traditional denoising methods such as BM3D [55], [56] or non-local means [57], recent research extended the PnP framework to powerful deep CNNs such as DnCNN [43] or DRUNet [2]. The latter leads to a State-Of-the-Art image restoration technique. Also, the Plug-and-play framework was applied using different proximal algorithms such as HQS [2], [3] and Proximal Gradient Descent [58].

An advantage of PnP methods is that, unlike agnostic learning approaches where the training takes place for each inverse problem separately, integrating deep denoisers in PnP frameworks decouples the learning phase from the inverse problem in hand, which makes it generic.

However, a limitation of proximal PnP algorithms lies in the fact that the regularizer is only defined implicitly via the denoiser, meaning that the objective function is not clearly defined. This limits the interpretability of the algorithm, since it becomes challenging to keep tractability of the optimization problem. Moreover, for most denoisers, there is no theoretical guarantee that there exists a potential regularization function whose proximal operator can be accurately represented by the denoiser [59]. This induces convergence issues in proximal PnP algorithms.

Although still an open question, designing convergence proofs for the Plug-and-play framework has become a major focus in the recent literature [3], [56], [58], [60]–[64]. The theoretical convergence analysis of these algorithms being outside the scope of this thesis, we refer interested readers to the review papers [65], [66]. In summary, most of the proposed solutions come with certain limitations, such as imposing strong assumptions on the denoiser (often not suitable for deep CNNs) or necessitating performance compromises.

2.4.2.2 Regularization by Denoising (RED)

In order to overcome some limitations arising from the absence of a clear objective function, Romano et al. [4] introduced an alternative approach for denosing-based priors called Regularization by Denoising (RED). Unlike some other methods where the prior

is implicitly defined, RED takes a different approach by explicitly expressing the regularization function ϕ. The latter is defined to be proportional to the inner product between the estimated image and its denoising residual, which can be written as follows:

$$\phi(x) = \frac{1}{2} x^T [x - D(x)]. \tag{2.42}$$

where D is an off-the-shelf denoiser. This explicit formulation results in an image-adaptive Laplacian-based regularization term, effectively promoting smoothness in the restored image. Under the assumption that the denoiser D is locally homogeneous and its Jacobian is symmetric, the gradient of the regularization term defined in Eq. 2.42 is computed as:

$$\nabla \phi(x) = x - D(x). \tag{2.43}$$

Using this formulation, RED proposes to solve Eq. 2.17 by using different optimization algorithms such as gradient descent or ADMM. Unlike Plug-and-play methods, RED is not restricted by the choice of the algorithm, making it adaptable to different optimization strategies. Another key advantage of the RED algorithm is its explicit formulation of the regularization function $\phi(x)$ as given in Eq. 2.42. This explicit expression allows for better understanding of the overall Bayesian objective function.

However, Reehorst and Schniter [59] proved later that the gradient expression proposed in RED is not justified with denoisers that lack Jacobian symmetry, which excludes most practical denoisers such as block-matching and 3D filtering (BM3D) [67], Non-local means (NLM) [68] or state-of-the-art deep neural networks.

Another extension to the RED algorithm proposed by Cohen et al. [69] explores a projected RED via Fixed-Point Projection approach (RED-PRO). Rather than explicitly defining the regularization term, the fixed-point set of a demi-contractive denoiser D $\{x \in \mathbb{R}^n : x = D(x)\}$ is considered as a prior for the image restoration. However, in practice, verifying the assumption that the denoiser is demi-contractive is not straightforward [70].

2.4.2.3 The score function in the context of Plug-and-play MAP estimators

A different approach leading to the same formulation as RED uses Tweedie's identity to match the score function with an MMSE denoiser.

Starting from the MAP estimation for solving an inverse problem, we have:

$$\hat{x}_{MAP} = \underset{x}{\operatorname{argmin}} \frac{1}{2} \|y - Ax\|_2^2 - \sigma^2 \log(p(x)). \tag{2.44}$$

Solving Eq. 2.44 by Steepest Descent (SD) amounts to solving the iterative formula:

$$\hat{x}_{k+1} = \hat{x}_k - \gamma_k [A^T(Ax_k - y) - \sigma^2 \nabla_x \log(p(x))|_{\hat{x}_k}], \tag{2.45}$$

where $\nabla_x \log(p(x))$ is defined as the *score function* in statistical literature. Moreover, a well-known mathematical finding commonly credited to Tweedie [71], [72], Stein [73] or Miyasawa [74] outlines:

$$\nabla_y \log(p_{\sigma_0}(y)) = \frac{D(y, \sigma_0) - y}{\sigma_0^2}, \tag{2.46}$$

where $y = x + v$ is a degraded version of x, corrupted with AWGN $v \sim \mathcal{N}(0, \sigma_0^2 I)$, and $D(y, \sigma_0)$ is the optimal MMSE denoiser for a Gaussian noise of standard deviation σ_0. The proof of this result is relatively straightforward and can be found in [7].

However, the relevant score function is that of the noisy observation density $p_{\sigma_0}(y)$, rather than the one corresponding to the prior density desired in Eq. 2.45. The noisy observation density $p_{\sigma_0}(y)$ is the result of a convolution between the prior density $p(x)$ and the noise distribution: $p_{\sigma_0}(y) = p(x) \otimes \mathcal{N}(0, \sigma_0^2 I)$. It is assumed, that when σ_0 is sufficiently small, $p(x)$ can be approximated by a slightly blurry PDF, and the update in Eq. 2.45 becomes:

$$\hat{x}_{k+1} = \hat{x}_k - \gamma_k [A^T(Ax_k - y) + c(\hat{x}_k - D(\hat{x}_k, \sigma_0)]. \tag{2.47}$$

Here, $c = \frac{\sigma^2}{\sigma_0^2}$, and this is equivalent to the RED algorithm when solved with Steepest Descent.

In practice, we don't know the real prior distribution of natural images, hence we cannot obtain the MMSE denoiser. However, deep denoisers can often approximate the MMSE denoisers very well, given that they are typically learned using an MSE loss on a large set of natural images.

2.4.2.4 Posterior sampling: The score function in the context of MMSE estimators

Different from the aforementioned approaches, we elaborate here on the computation of MMSE estimators. We recall that the MMSE estimator of the posterior mean is given by:

$$\hat{x}_{MMSE} = \int xp(x|y)dx. \tag{2.48}$$

This computation can be challenging, as it consists in estimating high dimensional integrals. Therefore, sampling algorithms such as the Monte Carlo integration method can be used in order to perform numerical integration. Posterior sampling has been considered in [5]–[8].

When solving 2.48, sampling algorithms consist in taking different samples from the posterior distribution. As the number of samples increases, the average of those samples approaches the expected value of the random variable. In the context of integration, this means that as we consider more samples, the average of the function values over those samples converges to the true integral value.

Posterior sampling approaches are largely based on the annealed Langevin dynamics algorithm [6], a Monte Carlo method with the subsequent transition rule:

$$\begin{aligned} x_{k+1} &= x_k + \gamma \nabla \log p(x_k|y) + \sqrt{2\gamma} z_k \\ &= x_k + \gamma \nabla \log p(y|x_k) + \gamma \nabla \log p(x_k) + \sqrt{2\gamma} z_k, \end{aligned} \tag{2.49}$$

where $z_k \sim \mathcal{N}(0, I)$ and $\gamma > 0$ is a step-size. The term $\nabla \log p(y|x_k)$ reflects the data fidelity and $\nabla \log p(x_k)$ is the score function which can be approximated by an MMSE denoiser as discussed in Section 2.4.2.3. Finally, Eq. 2.49 is an iterative algorithm that converges towards a solution that reflects the posterior distribution.

2.4.3 Regularizing through generative models

Another category of methods define the image prior by generative models such as a Generative Adversarial Network (GAN) [75] or a Variational Audoencoder (VAE) [76]. A generative model $G : \mathbb{R}^k \to \mathbb{R}^n$ learns to take a low-dimenstional latent variable $z \in \mathbb{R}^k$ and map it to a high dimensional sample space $x \in \mathbb{R}^n$ such that $x = G(z)$. The idea is to solve the MAP estimator on the latent variable z and restrict the output to be in the

range of the generator:

$$\hat{z} = \underset{z}{\operatorname{argmin}} - \log p(y|G(z)) - \log p(z), \tag{2.50}$$

The reconstructed image is obtained by $\hat{x} = G(\hat{z})$. This approach was first introduced by Bora et al. [77], motivated by the fact that well-trained generative models approximate the space of natural signals or images.

Subsequently, a lot of research has been done in this direction, resulting in using generative models for regularizing inverse problems either implicitly by restricting the solution in the range of a generative model, or explicitly by learning a CNN regularizer using generative models.

Shah et al. [78] later proposed to solve the approach proposed by [77] with a projected gradient descent algorithm and discussed convergence guarantees.

Furthermore, rather than restricting $\hat{x}$ to fall within the range of the generator and minimizing on the latent variable z, Gonzalez et al. [79] optimize the joint posterior $p(x, z|y)$ of the space variable x and the latent variable z, given the observation y. They evaluate the joint $MAP_{x,z}$ estimator (JPMAP) using a VAE prior. The authors also emphasize the importance of training the VAE with a denoising criterion (denoising VAE) proposed by [80], which results in a more robust encoder that can generalize well for inputs that do not lie in the manifold of the generator network. However, a limitation of JPMAP lies in the fact that it is tailored for VAEs with a fixed Gaussian prior distribution over the latent space, which confines its applicability to toy examples.

In a subsequent work [81], Prost et al. generalize the JPMAP framework to regularize inverse problems with Hierarchical Variational Autoencoders (HVAE). In this context, the regularization strength can be controlled using the temperature of the prior in the latent space (the temperature refers to a hyper-parameter that controls the smoothness of the generated samples). This approach also overcomes the limitation of the previous JPMAP [79] to simple VAEs.

2.4.4 End-to-end learning of the regularization function via deep unrolling

Introduced by Gregor et al. [82], the idea of deep unrolling quickly grabbed the interest of researchers and made significant progress in subsequent works [83]–[88]. In the context of deep unrolling, often referred to as algorithm unrolling, the regularizing function is

learned end-to-end for a specific inverse problem. The idea is to iterate an optimization algorithm for a pre-determined number of iterations, and to replace the regularization function with a CNN to be learned specifically for a particular problem. The learned network replaces the gradient of the regularizer in the case of a gradient-based algorithm, or its proximal operator if the iterative algorithm is a proximal algorithm. These networks are often referred to as Artifact Removal operators trained to remove artifacts specific to the forward model.

For a more comprehensive understanding of deep unrolling, we illustrate the case of the algorithm of unrolled gradient descent. Solving Eq. 2.17 with the gradient descent algorithm amounts to the iterative process described by:

$$x_{k+1} = x_k - \gamma[A^T(Ax_k - y) + \nabla\phi(x_k)], \tag{2.51}$$

where $\gamma > 0$ denotes the step size of the optimization algorithm. In the context of the unrolled gradient descent, this process is carried out for a specified number of iterations (referred to as blocks), which we will denote by B. The latter remains constant throughout both the training and testing phases.

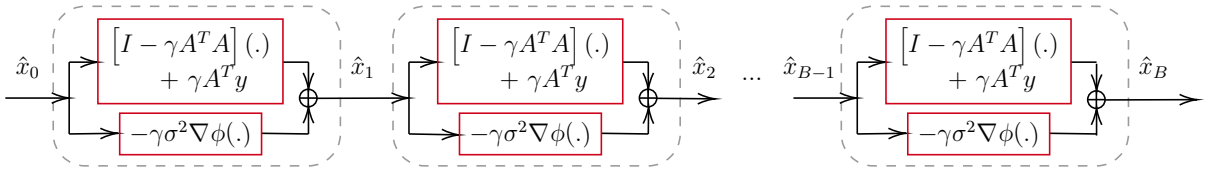


Figure 2.7 – Unrolled gradient descent framework. Here, $\nabla\phi$ is a trained neural network. The resulting output $\hat{x}$ is obtained after B iterations of gradient descent with a fixed step size of γ.

Figure 2.7 provides an illustrative representation. Here, the neural network parameterizes the gradient of the regularizer $\nabla\phi$, which is learned in conjunction with the forward model A and training observations y_i. The learning process involves minimizing a loss function quantifying the discrepancy between the ground truth images x_i and the output of the full network $\hat{x}(y_i)$. Therefore, this end-to-end learning approach tailors the learned component specifically to the underlying forward model.

As a result of the prior being specifically learned for the inverse problem in hand, deep unrolling methods yield significantly improved performance compared to Plug-and-play methods where the prior is generic. However, these end-to-end approaches lose the genericity of Plug-and-play algorithms. Furthermore, it is not clear what interpretation

can be given to the trained network which does not only learn prior knowledge on images, but also task-specific features.

Chapter 3

Regularization of the Deep Image Prior with a deep denoiser

3.1 Introduction

As deep learning evolved in the past decade, the utilization of end-to-end training methods brought about a significant breakthrough in solving inverse problems. Neural networks are typically used to learn to reconstruct degraded images by using a large amount of data, without any knowledge of the degradation operator.

As previously discussed in Section 2.4.1 , an exceptional work by Ulyanov et al. [1] proposed to leverage the power of deep neural networks in a different way. The authors present the Deep Image Prior (DIP), where they show that the architecture of a generator network itself can be used as a regularization for inverse problems. DIP is an un-trained generative model that fits its parameters for the degraded image. Therefore, the prior is essentially hand-crafted, as no component of the model is learned beforehand from data.

Despite its simple formulation, the Deep Image Prior has demonstrated successful performance in solving various problems, including denoising, super-resolution, and inpainting. However, its results still lag behind those achieved by other state-of-the-art methods. This motivates to enhance the DIP by integrating explicit regularization to boost the implicit one.

Interestingly, prior research has explored the regularization of the DIP, yet these studies exclusively employed handcrafted regularization techniques, without exploiting the potential of learned regularizers.

In this chapter, we delve into the regularization of the estimate produced by the untrained generative DIP. We elaborate a method coupling a learned denoiser with the Deep Image Prior in the ADMM framework. The goal is to couple advantages of the DIP only optimized on the input image, with those of a generic prior learned from a large collection of natural images.

3.2 Related work: regularized Deep Image Prior

In this section, we elaborate details about the Deep Image Prior and discuss some regularization methods that have been applied to boost its performance in the literature. The deep generative model $x = f_\Theta(z)$ is learned by mapping a random code vector $z \in \mathrm{R}^N$ to an image x, i.e., by solving

$$\Theta^* \in \underset{\Theta}{\operatorname{argmin}} \ \|Af_\Theta(z) - y\|_2^2 \qquad s.t.\ x^* = f_{\Theta^*}(z) \tag{3.1}$$

where Θ represents the network parameters. The generator is randomly initialized with variables Θ, which are optimized iteratively in a way that the output of the network is close to the target measurements. In most of the applications, a U-Net type architecture with skip-connections is used, having over 2 million parameters (Figure 3.1). Details of the network are adapted to the specific application.

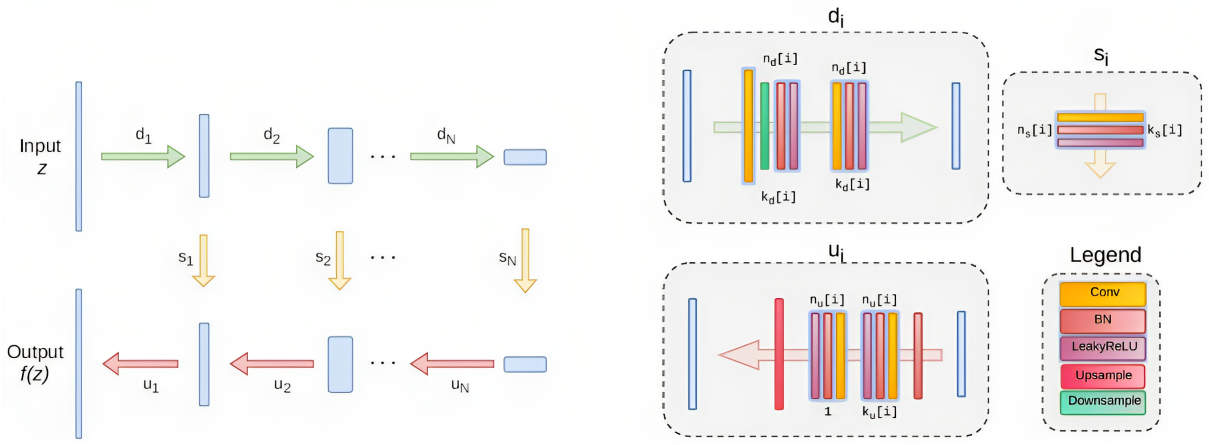


Figure 3.1 – Architecture of the Deep Image Prior. An "hourglass" architecture is used. $n_u[i], n_d[i], n_s[i]$ represent the number of feature maps at the i^{th} layer for the upsampling, downsampling and skip-connections respectively and $k_u[i], k_d[i], k_s[i]$ denote the kernel sizes. Skip connections and parameters are defined depending on the application. The figure is extracted from [1].

TV-regularized Deep Image Prior

Liu et al. [89] propose to use a total-variation regularization to enhance the performance of the Deep Image Prior. They consider the variational problem defined as:

$$\Theta^* \in \underset{\Theta}{\operatorname{argmin}} \ \|Af_\Theta(z) - y\|_2^2 + \lambda TV(f_\Theta(z)) \qquad s.t.\ x^* = f_{\Theta^*}(z), \tag{3.2}$$

and define an l_1-based anisotropic TV given by $TV_{aniso}(u) = \sum_{i=1}^{n} \ |(D_h u)_i| + |(D_v u)_i|$, $D_h u$ and $D_v u$ being the first order finite difference discrete operators respectively along the horizontal and vertical axes. The objective function is minimized using a gradient descent optimization method.

Subsequently, Cascarano et al. [90] propose to solve this objective function with the ADMM. Later, the authors propose to add a weight on the TV regularization in [91]. Another difference between the DIP-TV proposed in [89] and the ADMM-DIP-TV [90], [91] works is that in DIP-TV, anisotropic TV is used, decoupling the contribution of both horizontal and vertical gradient components, whereas ADMM-DIP-TV uses isotropic TV, jointly considering the gradient components.

Denoiser-based DIP regularization

Another approach to boost the performance of the DIP is proposed in [92], bringing-in the concept of Regularization by Denoising (RED) [4], which uses existing denoisers for regularizing inverse problems. By merging the DIP with the RED, the objective function of the resulting DeepRED becomes:

$$\underset{\Theta}{\operatorname{argmin}} \ \frac{1}{2}\|Af_\Theta(z) - y\|_2^2 + \frac{\lambda}{2}[f_\Theta(z)]^T([f_\Theta(z)] - D(f_\Theta(z))), \tag{3.3}$$

where D is a denoiser applied on the output of the generative network, that they replace with the BM3D [67]. Here, the optimization is solved using ADMM.

3.3 Regularizing the Deep Image Prior with a learned denoiser

We propose to regularize the estimate produced by the DIP with a learned denoiser. With this combination, we aim to combine the benefits of a generic prior learned on a

large set of natural images and those of the Deep Image Prior optimized only on the input image, where the network architecture itself plays the role of the regularization.

Let us consider the problem of regularizing the DIP optimization:

$$\begin{aligned} \Theta^* \in \underset{\Theta}{\operatorname{argmin}} \ & \frac{1}{2} \|Af_\Theta(z) - y\|_2^2 + \sigma^2 \phi(x) \\ s.t. \ & x^* = f_{\Theta^*}(z), \end{aligned} \tag{3.4}$$

where we want to replace ϕ with a learned regularizer. For simplicity, we will replace $f_\Theta(z)$ by t wherever convenient. The augmented lagrangian form of the Eq. 3.4 can be written as:

$$L(x,t,l) = \frac{1}{2} \|At - y\|_2^2 + \sigma^2 \ \phi(x) + \frac{\rho}{2} \left\| x - t + \frac{l}{\rho} \right\|_2^2 - \frac{1}{2\rho} \|l\|_2^2, \tag{3.5}$$

with ρ being a positive penalty parameter of the constraint, and l the dual variable. This minimization problem is solved using the iterative ADMM method. By alternatively optimizing x, t and l of the augmented Lagrangian $L(x,t,l)$, the ADMM iterate reads as:

$$t^{k+1} = f_{\Theta^{k+1}}(z) \quad \text{with}$$

$$\Theta^{k+1} = \underset{\Theta}{\operatorname{argmin}} \frac{1}{2} \|Af_\Theta(z) - y\|_2^2 + \rho \left\| x^k - f_\Theta(z) + \frac{l^k}{\rho} \right\|_2^2, \tag{3.6}$$

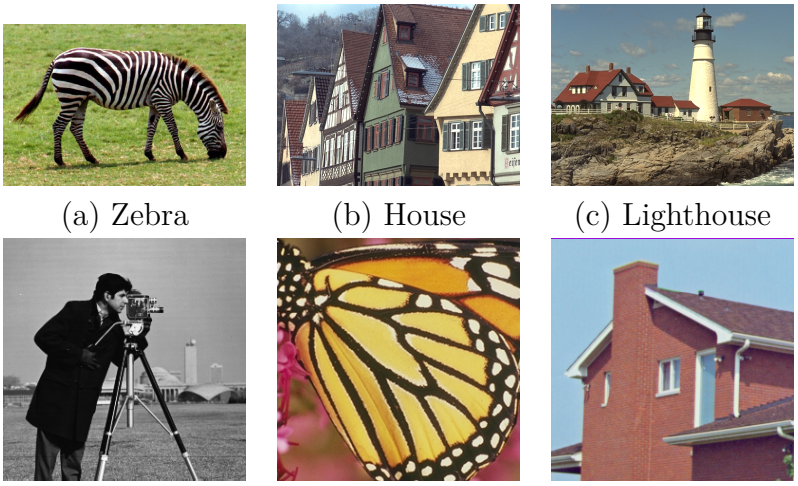
$$x^{k+1} = \underset{x}{\operatorname{argmin}} \frac{1}{2} \left\| x - t^{k+1} + \frac{l^k}{\rho} \right\|_2^2 + \frac{\sigma^2}{\rho} \phi(x), \tag{3.7}$$

$$l^{k+1} = l^k + \rho(x^{k+1} - t^{k+1}). \tag{3.8}$$

As discussed in Section 2.4.2.1, the x-update can be rewritten as $x^{k+1} = \operatorname{prox}_{\frac{\sigma^2}{\rho}\phi}(t^{k+1} - \frac{l^k}{\rho})$ and can be replaced by a denoiser. Hence, inspired by [55], we propose to replace the x-update by a learned denoiser [2]. We thus get:

$$x^{k+1} = D_s(t^{k+1} - \frac{l^k}{\rho}), \tag{3.9}$$

with D_s being a learned denoiser and $s = \frac{\sigma}{\sqrt{\rho}}$. We use the DRUNet [2], a state-of-the-art deep denoiser, which takes as input the noisy image concatenated in the channel dimension with a noise level map. It is worth noting that since ADMM separates the regularization



(a) Zebra (b) House (c) Lighthouse

(d) Cameraman (e) Butterfly (f) Hill

Figure 3.2 – The set of images used for the numerical experiments.

term ϕ from the degradation matrix A, the learned proximal operator can be used with any linear operator.

In practice, Eq. 3.6 can't be solved exactly due to the non-linearity of $f_\Theta(z)$. Furthermore, the DIP should not overfit the exact solution in order to provide an additional regularization effect in complement to the denoiser. Therefore, we solve Eq. 3.6 inexactly by performing a fixed number n_{DIP} of gradient descent iterations within the ADMM iterations. For the initialization, we optimize the DIP using the original method [1] which performs gradient descent to optimize Eq. 3.1. A number n_{init} of iterations is used for this initialization, followed by n_{ADMM} iterations of the ADMM scheme.

We note that this approach can also be seen as a sort of a Plug-and-play ADMM, where the data-term is given by the generative Deep Image Prior.

3.4 Experimental results

We have reproduced the results of DIP [1], DIPTV [89], ADMM-DIPTV [90] and DeepRED [92] for the problems of denoising and super-resolution and we have compared them with our proposed method. In order to evaluate the input of the DIP in the PnP-ADMM algorithm, we have also compared the results with the PnP-ADMM with the denoiser of [2] for super-resolution, and with the denoiser for the inverse problem of denoising.

The test dataset (Figure 3.2) was formed by taking the 5 natural images from the

ADMM-DIPTV paper [90], and the zebra image used in [1], which contains interesting high frequency patterns.

In [90], the authors use isotropic TV and analyze the addition of ADMM by comparing with anisotropic TV in [89]. For the sake of a fair comparison between [89] and [90], we have reproduced the anisotropic version of both of the methods. For DeepRED, we have used the code provided by the authors which applies the BM3D denoiser to regularize the DIP optimization.

For fair comparisons, we have used the same network architecture for the DIP in all the compared methods.

	(i) Denoising		(ii) Super-resolution	
	$\sigma = 20$	$\sigma = 30$	Factor 4	Factor 8
n_{init}	1750	1750	2500	4000
n_{ADMM}	100	100	100	100
n_{DIP}	250	250	200	150
s	20	30	41	45
Step size	0.01	0.01	0.01	0.01

Table 3.1 – Parameters used for the proposed method for (i) Denoising with noise standard deviations of ($\sigma = 20$ and $\sigma = 30$), (ii) Super-Resolution of factor 4 and 8. n_{init}: DIP iterations at initialisation, n_{ADMM}: ADMM iterations, n_{DIP}: number of iterations of Eq. 3.6 in each iteration of ADMM, s: noise level assumed by the learned denoiser in Eq. 3.9. The same step size is used for all gradient descent steps.

Table 3.1 shows the parameters used for the proposed method for each of the cases of denoising and super-resolution. Tables 3.2 and 3.3 show the PSNR [dB] results for each of the compared methods for denoising and super-resolution respectively. The PSNR measures presented in this thesis are computed on the RGB channels in the following way:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right), \tag{3.10}$$

where MAX_I^2 is the maximum possible pixel value in the image (i.e. 1 or 255 depending whether the image is normalized or not), and MSE is the Mean Squared Error calculated as:

$$\text{MSE} = \frac{1}{W.H} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} ((\hat{I}(x,y) - I^{gt}(x,y))^2 \tag{3.11}$$

		DIP	DIPTV	ADMM-DIPTV	denoiser	DIP-denoiser-ADMM
(i) $\sigma = 20$	House	27.93	27.81	27.96	31.55	31.35
	Zebra	29.98	29.63	29.5	31.21	31.1
	Lighthouse	29.03	29.55	29.3	32.18	31.96
	Hill	31.29	32.53	32.07	34.02	33.93
	Butterfly	30.07	30.52	30.62	33.23	32.96
	Cameraman	30.01	31.4	31.04	33.7	33.67
	Average	29.72	30.24	30.08	32.64	32.49
(ii) $\sigma = 30$	House	25.75	26.18	25.72	29.28	29.15
	Zebra	27.34	27.52	27.58	28.93	28.87
	Lighthouse	26.7	27.48	27.2	29.96	29.75
	Hill	29.61	30.81	30.28	32.62	32.51
	Butterfly	28.16	28.48	28.52	30.85	30.83
	Cameraman	27.79	28.88	28.41	31.15	31.14
	Average	27.55	28.23	27.95	30.46	30.37

Table 3.2 – PSNR [dB] of denoised images obtained with TV, DIP[1], DIPTV[89], ADMM-DIPTV[90], the denoiser of [2] and our proposed method, for noise standard deviations of (i) 20 and (ii) 30.

where H and W represent the number of rows and columns in the image respectively and $\hat{I}(x, y)$ and $I^{gt}(x, y)$ represent the pixel value of the reconstructed and original images at position (x, y) respectively.

For denoising (Table 3.2), we can see that, our proposed method significantly improves the result of the DIP. Hence, as expected, learned regularizers outperform handcrafted regularizers when regularizing the DIP optimization. However, the learned denoiser itself performs better alone, and the DIP does not have an added value in this case. This was expected in the case of denoising, since the network is learned end-to-end for the denoising task.

However, for the task of super resolution (Table 3.3), we have a different outcome. In fact, we can see again that regularizing with a learned denoiser over the DIP optimization gives a better performance than using handcrafted priors. But also, as opposed to what we had for the denoising, combining the DIP with the denoiser improves the performance of the learned regularizer alone. This means that combining a prior learned on a large set of natural images with the untrained generative DIP is advantageous in this case.

The degradation filter we have used for the task of super resolution is a Gaussian filter of standard deviation $\sigma_f = 1$ for both x4 and x8 super-resolution factors. Visual comparisons are shown in Figures 3.3 and 3.4 for super-resolution of factor 4. More visual results

		Bicubic	DIP	DIPTV	ADMM-DIPTV	DeepRED	Denoiser-ADMM	DIP-denoiser-ADMM
(i) factor 4	House	19.42	19.72	19.85	19.62	19.88	20.06	20.14
	Zebra	23.00	24.13	24.31	23.84	24.69	24.71	24.9
	Lighthouse	23.09	23.05	23.37	23.17	23.36	23.46	23.45
	Hill	26.37	28.44	27.61	27.01	27.81	28.75	28.97
	Butterfly	20.95	24.23	23.66	22.47	23.79	24.34	24.42
	Cameraman	22.58	23.23	23.27	22.99	23.34	23.34	23.51
	Average	22.58	23.80	23.67	23.18	23.81	24.11	24.23
(ii) factor 8	House	16.25	17.16	17.32	17.11	17.22	16.83	17.21
	Zebra	16.84	18.01	18.15	17.75	18.05	17.37	17.84
	Lighthouse	19.87	20.6	20.65	20.45	20.65	20.47	20.57
	Hill	20.75	22.45	22.1	22.01	22.43	23.46	23.58
	Butterfly	15.38	17.04	16.62	16.63	16.80	17.11	17.4
	Cameraman	18.94	20.2	20.1	19.95	20.15	19.88	20.19
	Average	18.00	19.24	19.15	18.98	19.22	19.18	19.46

Table 3.3 – PSNR [dB] of upsampled images obtained with bicubic interpolation, DIP[1], DIPTV[89], ADMM-DIPTV[90], DeepRED[92], PnP-ADMM with the denoiser of [2] and our proposed method, for (i) SR factor 4 and (ii) SR factor 8.

can be found on the web page: `http://clim.inria.fr/DeepCIM/Regul-DIP/index.html`.

3.5 Conclusion and discussion

In this chapter, we studied different approaches of regularizing the generative Deep Image Prior. We proposed to regularize the DIP with a learned denoiser, in order to combine the benefits of a generic prior learned on a large set of natural images and the regularization provided by the network architecture of the DIP. The proposed denoiser-based regularization compares favourably with handcrafted priors and can be applied to different inverse problems. Moreover, the use of DIP improves the performance of the PnP-ADMM in the case of super-resolution.

However, several aspects were observed while we were working on this contribution. The parameter tuning of the ADMM algorithm is not a straightforward task. Also, employing a denoiser adds the complexity of fine-tuning an extra hyper-parameter for each application, while theoretically, a regularizer should be able to determine the image prior regardless of the inverse problem in hand. Our experimental implementations led us to the reflection that solving the DIP optimization could have been significantly simplified

(a) Ground-truth (b) Bicubic (c) DIP

(d) ADMM-DIPTV (e) denoiser-ADMM (f) DIP-denoiser-ADMM

Figure 3.3 – Visual comparison of super-resolution results on the Butterfly image. Low resolution image is generated with a Gaussian kernel of standard deviation $\sigma_f = 1$ followed by a downsampling by a factor of 4.

(a) Ground-truth (b) Bicubic (c) DIP

(d) ADMM-DIPTV (e) denoiser-ADMM (f) DIP-denoiser-ADMM

Figure 3.4 – Visual comparison of super-resolution results on the Cameraman image. Low resolution image is generated with a Gaussian kernel of standard deviation $\sigma_f = 1$ followed by a downsampling by a factor of 4.

by employing a network that directly models the regularizer within a gradient descent algorithm. In the upcoming chapter, we present a methodology to learn a network modeling the gradient of a regularizer, by using a deep denoiser. Given that a denoiser can be seen as an implicit prior, we aim to learn a network that models this prior. The resulting network enables solving inverse problems using the prior embedded in a denoiser with a simple gradient descent algorithm, avoiding the complexity of the parameter tuning in proximal algorithms.

Chapter 4

PnP-ReG: learned regularizing gradient for Plug-and-play gradient descent

4.1 Introduction

In this chapter, we present a novel approach to regularize inverse problems using deep neural networks. We propose a framework to train a network that models the gradient of a regularizer, which can be used in a Plug-and-play gradient descent algorithm to solve any inverse problem.

The Plug-and-play (PnP) framework, introduced in [55], makes it possible to integrate advanced image denoising priors into optimization algorithms, to efficiently solve a variety of image restoration tasks generally formulated as Maximum A Posteriori (MAP) estimation problems. However, as already discussed in Chapters 2 and 3, PnP methods remain limited to proximal algorithms, which makes the task of parameter tuning quite challenging. This introduces complexity and reduces the algorithm's comprehensibility and ease of interpretation. Subsequent work, such as RED [4] and PnP-SGD [64], used an explicit regularization to circumvent this issue. These methods use the denoising residual as the gradient of the regularization and are not always restricted to proximal algorithms. A recent method called GS-PnP [3] plugs a denoiser trained to perform an exact gradient step on a regularization function represented by a CNN, leading to convergence guarantees. Another line of work considers score matching methods that solve inverse problems in the MMSE framework. However, these methods approximate the true signal density by using a MMSE denoiser's residual which is proportional to the gradient of the log of the noisy signal density.

A common issue shared by all these methods is that the regularization term is (implicitly or explicitly) defined by a denoiser. This involves an additional noise level hyper-

parameter that must be tuned per-application. However, in theory, a regularizer (and thus its gradient) should fully determine the image prior, regardless of the task to be solved.

We propose a novel approach to train a network modeling the gradient of a regularizer. Our method makes use of a second network pre-trained for the denoising task. Our goal is to transfer the prior implicitly defined by a denoiser to a regularizing network. Hence, we aim to use our network in a simple gradient descent algorithm, to avoid the challenging parameter tuning of proximal PnP algorithms. Based on the assumption that the denoiser represents the proximal operator of an underlying differentiable regularizer (defining the image prior), we derive a loss function that links the denoiser and the regularizer's gradient. However, since there is no guarantee that this assumption is mathematically valid for a denoising neural network, we propose an approach where the pre-trained denoiser is modified jointly with the training of our regularizing network. This approach encourages the denoiser to be consistent with the definition of a proximal operator of a differentiable regularizer, and significantly improves our results in comparison to keeping the denoiser fixed.

We use our network to solve different inverse problems such as super-resolution, deblurring and pixel-wise inpainting in a simple gradient-based algorithm, and obtain better results when comparing to other generic methods. We also show that our training method can advantageously serve as a pre-training strategy, later facilitating a per-application tuning of the regularization network in the framework of unrolled gradient descent.

4.2 Notations and problem statement

In this section, we quickly revisit some previously established notations to ensure reader clarity and precision, and we introduce the problem that we treat in this chapter.

In this thesis, we consider linear inverse problems of the form:

$$y = Ax + \eta, \tag{4.1}$$

where $A \in \mathbb{R}^{m \times n}$ represents the degradation operator depending on the inverse problem and $\eta \sim \mathcal{N}(0, \sigma^2 I)$ typically represents Additive White Gaussian Noise (AWGN) of standard deviation σ. The reconstruction can be treated using Bayesian estimation that uses the posterior conditional probability $p(x|y)$. Maximum a posteriori probability (MAP) is

the most popular estimator in this scheme, where the estimation task is modeled as the optimization problem:

$$\widehat{x}_{MAP} = \underset{x}{\operatorname{argmin}} \ \frac{1}{2} \|y - Ax\|_2^2 + \sigma^2 \, \phi(x), \tag{4.2}$$

where $f(x) = \frac{1}{2} \|y - Ax\|_2^2$ is the data fidelity term whereas the $\phi(x)$ is the regularization term. Details of this notation can be found in Section 2.2.3. The problem in Eq. 4.2 does not have a closed-form solution in general. Therefore, it must be solved using different optimization algorithms. The Plug-and-play framework typically considers proximal splitting algorithms which decompose the problem into two sub-problems (one for each term in Eq. 4.2) and solve them alternately. In these algorithms, the regularization sub-problem consists in evaluating the proximal operator of the regularization term defined as:

$$\begin{aligned} \operatorname{prox}_{\sigma^2 \phi}(z) &= \underset{x}{\operatorname{argmin}} \, \mathcal{F}_\phi(x, z, \sigma), \\ \text{with} \quad \mathcal{F}_\phi(x, z, \sigma) &= \frac{1}{2} \|x - z\|_2^2 + \sigma^2 \, \phi(x). \end{aligned} \tag{4.3}$$

This sub-problem can be replaced by a state-of-the-art Gaussian denoiser in a Plug-and-play proximal algorithm, since the proximal operator in Eq. 4.3 can be seen as a particular case of inverse problem where the degradation operator A is the identity matrix, and the degradation only consists in the addition of White Gaussian Noise of standard deviation σ.

However, this approach does not directly generalize to the gradient descent algorithm where the update formula for the minimization in Eq. 4.2 is expressed as:

$$\begin{aligned} \widehat{x}_{k+1} &= \widehat{x}_k - \mu \Big[\nabla f(\widehat{x}_k) + \sigma^2 \nabla \phi(\widehat{x}_k) \Big], \\ &= \widehat{x}_k - \mu \Big[A^T (A\widehat{x}_k - y) + \sigma^2 \nabla \phi(\widehat{x}_k) \Big], \end{aligned} \tag{4.4}$$

μ being the step size. Here, instead of the proximal operator of the regularizer ϕ, we need its gradient $\nabla \phi$, which cannot be replaced by a denoiser. In this chapter, we propose to train a network $\mathcal{G}$ that can serve as the gradient of the regularization term in the Plug-and-play gradient descent (Algorithm 4).

Algorithm 4 Plug-and-play gradient descent

Input: y, σ, μ, N
Output: x

$x_0 \leftarrow y$
for $k \leftarrow 0$ to $N-1$ **do**
    Apply $G_R \leftarrow \mathcal{G}(x_k)$
    $x_{k+1} \leftarrow x_k - \mu[A^T(A.x_k - y) + \sigma^2 G_R]$
end for

4.3 Training of the gradient of a regularizer

4.3.1 Mathematical derivations

We show in the following that it is mathematically possible to train a network that models the gradient of a regularizer by using a deep denoiser. Let us consider a denoiser $\mathcal{D}_\sigma$ defined as the proximal operator in Eq. 4.3:

$$\mathcal{D}_\sigma(z) = \underset{x}{\operatorname{argmin}}\, \mathcal{F}_\phi(x, z, \sigma). \tag{4.5}$$

Hence, for σ and z fixed, the denoised image $x = \mathcal{D}_\sigma(z)$ minimizes $\mathcal{F}_\phi(x, z, \sigma)$. Therefore, we have:

$$\left.\frac{\partial \mathcal{F}_\phi}{\partial x}\right|_{x=\mathcal{D}_\sigma(z)} = 0, \tag{4.6}$$

Furthermore, $\frac{\partial \mathcal{F}_\phi}{\partial x}$ can be computed as:

$$\frac{\partial \mathcal{F}_\phi}{\partial x} = \frac{\partial \left[\frac{1}{2}\left\|x - z\right\|_2^2 + \sigma^2\, \phi(x)\right]}{\partial x}, \tag{4.7}$$

$$= x - z + \sigma^2 . \frac{\partial \phi(x)}{\partial x}. \tag{4.8}$$

Evaluating at the denoised image $x = \mathcal{D}_\sigma(z)$ thus gives:

$$\left.\frac{\partial \mathcal{F}_\phi}{\partial x}\right|_{x=\mathcal{D}_\sigma(z)} = \mathcal{D}_\sigma(z) - z + \sigma^2 . \left.\frac{\partial \phi(x)}{\partial x}\right|_{x=\mathcal{D}_\sigma(z)}. \tag{4.9}$$

Using Eq. 4.6 and Eq. 4.9, we obtain:

$$\mathcal{D}_\sigma(z) - z + \sigma^2 . \frac{\partial \phi(x)}{\partial x}\bigg|_{x=\mathcal{D}_\sigma(z)} = 0, \tag{4.10}$$

$$\sigma^2 . \frac{\partial \phi(x)}{\partial x}\bigg|_{x=\mathcal{D}_\sigma(z)} = z - \mathcal{D}_\sigma(z), \tag{4.11}$$

$$\sigma^2 . \nabla \phi(\mathcal{D}_\sigma(z)) = z - \mathcal{D}_\sigma(z). \tag{4.12}$$

Using Eq. 4.12, we can thus train a network $\mathcal{G}$ to model $\nabla\phi$ (i.e. gradient of the regularizer with respect to its input image) using the loss function:

$$\mathcal{L}_\mathcal{G} = \left\| \sigma^2 \Big[\mathcal{G}(\mathcal{D}_\sigma(z)) \Big] - (z - \mathcal{D}_\sigma(z)) \right\|_2^2 . \tag{4.13}$$

This requires the knowledge of the corresponding denoiser $\mathcal{D}_\sigma$. Note that Eq. 4.12 is valid for any value of σ regardless of the degradation in z. Hence, σ can be seen as a free parameter of our loss $\mathcal{L}_\mathcal{G}$. For small values of σ, the input $\mathcal{D}_\sigma(z)$ of the network $\mathcal{G}$ will be close to the degraded image z. Hence $\mathcal{G}$ will be trained to fit the artifacts in the degraded images (e.g. noise). On the other hand, for high values of σ, the input of $\mathcal{G}$ will be a strongly denoised image, with reduced artifacts but less details. Hence, $\mathcal{G}$ will be trained to recover the missing details. During the training, we vary the value of this parameter so that our network can recover details while also removing artifacts (see details in Section 4.3.2).

Also note that in practice, since ϕ is meant to be used in gradient-based algorithms, we only need the gradient $\nabla\phi$ rather than an explicit definition of ϕ. Hence, we propose in what follows a framework for training $\mathcal{G}$ jointly with the denoiser $\mathcal{D}$, so that $\mathcal{G}$ can then be used in place of $\nabla\phi$ in Plug-and-play gradient descent. In the remainder of the paper, we refer to $\mathcal{G}$ as the regularizing gradient network (ReG).

4.3.2 Training framework for the regularizing gradient network $\mathcal{G}$

The training framework is depicted in Figure 4.1. Let $\nu \to \mathcal{N}(0, \sigma_0)$ be a white Gaussian noise of standard deviation σ_0 that we use to corrupt the ground truth images x_0 of the training dataset to produce degraded images $z = x_0 + \nu$. Let σ be a standard deviation value used as a parameter of our loss $\mathcal{L}_\mathcal{G}$, as defined in Section 4.3.1. In order to

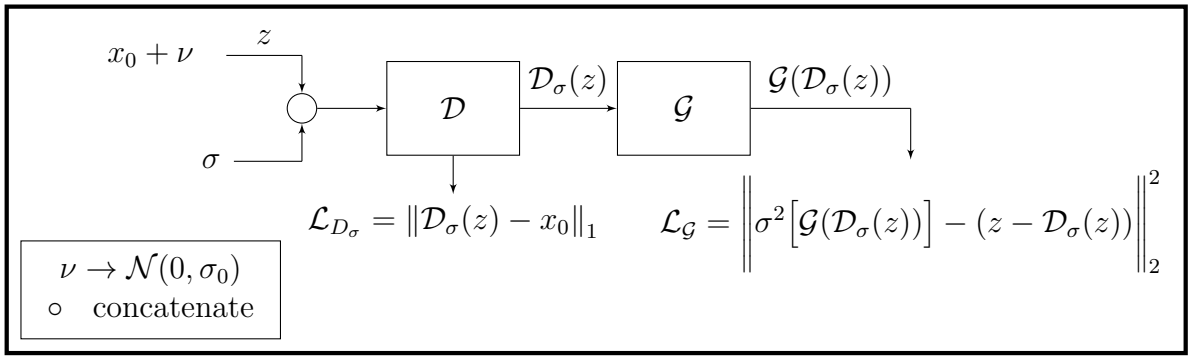


Figure 4.1 – Framework for joint training of $\mathcal{D}$ and $\mathcal{G}$.

handle different values of σ in Eq. 4.13, $\mathcal{D}_\sigma$ is modeled as a non-blind deep denoiser that takes as input a noise level map (i.e. each pixel of the noise level map being equal to σ) concatenated with the noisy image z. This approach has been previously suggested in [2], [93], [94]. Note that if the denoiser D_σ does not satisfy the formal definition of a MAP Gaussian denoiser for some differentiable prior, there may not exist a function $\nabla\phi$ that satisfies Eq. 4.12. We prevent this issue by starting from a pre-trained denoiser which we jointly update along with the training of the ReG network $\mathcal{G}$. In order to preserve the denoising performance of D_σ during the training, we use an additional denoising loss $\mathcal{L}_{D_\sigma}$ defined as:

$$\mathcal{L}_{D_\sigma} = \|\mathcal{D}_\sigma(z) - x_0\|_1 . \tag{4.14}$$

Note that Eq. 4.14 is a suitable loss for the denoiser only when $\sigma = \sigma_0$ since the non-blind denoiser must be parameterized with the true noise level σ_0 of the noisy input. The denoised output $\mathcal{D}_\sigma(z)$ is then inputted to the network modeling the gradient of the regularizer in order to train it using the loss $\mathcal{L}_{\mathcal{G}}$ defined in Eq. 4.13.

Hence, our goal is to minimize the global loss defined as:

$$\mathcal{L} = \delta \, \mathcal{L}_{\mathcal{D}_\sigma} + \lambda \, \mathcal{L}_{\mathcal{G}}, \tag{4.15}$$

where $\lambda > 0$ and $\delta = \begin{cases} 1 & \text{if } \sigma_0 = \sigma \\ 0 & \text{otherwise.} \end{cases}$

For training the deep denoiser network, we should set the noise level σ inputted to the network equal to the actual noise level σ_0 used for generating ν. However, as explained in Section 4.3.1, for the loss $\mathcal{L}_{\mathcal{G}}$, it is preferable to select σ independently of σ_0. Hence, the input $\mathcal{D}_\sigma(z)$ of the ReG network $\mathcal{G}$ can cover a wide range of alterations, including

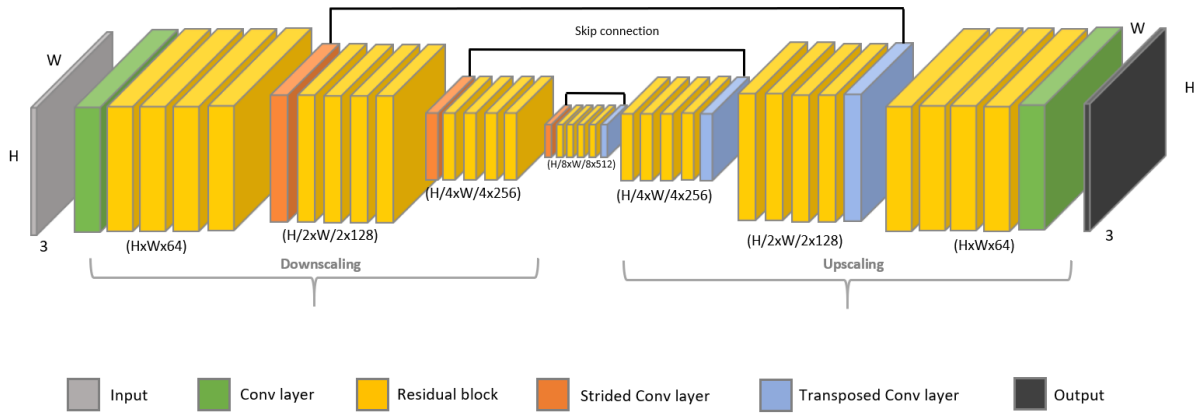


Figure 4.2 – Architecture of $\mathcal{G}$. The architecture is similar to the DRUNet architecture [2], with the input channel set to 3 instead of 4 (the ReG network does not need a noise level map as additional input as it does not depend on noise level).

images with remaining noise (i.e. $\sigma < \sigma_0$) or with too strong denoising, and thus less details (i.e. $\sigma > \sigma_0$). We therefore choose to alternate during the training between either selecting independently σ and σ_0, or setting $\sigma = \sigma_0$ in order to keep $\mathcal{D}$ faithful to the data. Furthermore, since the denoiser loss $\mathcal{L}_{\mathcal{D}_\sigma}$ is only valid when $\sigma = \sigma_0$, we omit this loss when $\sigma \neq \sigma_0$ by setting $\delta = 0$.

Note that we only need to alternate between setting $\sigma = \sigma_0$ and $\sigma \neq \sigma_0$ because we choose to train the denoiser jointly with the ReG network. A simpler training strategy would consist in first training the denoiser separately (only with the loss $\mathcal{L}_{D_\sigma}$ and with $\sigma = \sigma_0$), and then training the regularizer (only with the loss $\mathcal{L}_{\mathcal{G}}$ and with $\sigma \neq \sigma_0$). However, our experimental results in Section 4.4.3 clearly show the advantage of our joint training scheme, which confirms that the separately trained denoiser does not satisfy the assumption of a Gaussian MAP denoiser for a differentiable prior.

4.3.3 Training details

For the training, we have used a state-of-the-art deep denoiser architecture in order to train our ReG network. We have chosen to work with the DRUNet proposed in [2] which is a combination of U-Net [95] and ResNet [96]. DRUNet additionally uses a bias-free network architecture, which has been shown to allow for good generalization of denoisers over various noise levels, even if they were not seen during training [97]. Given that it takes as input the noisy image concatenated in the channel dimension with a noise level map, it can suitably represent the non-blind denoiser $\mathcal{D}_\sigma$.

The architecture of $\mathcal{G}$ is shown in Figure 4.2. It is the same architecture as the bias-free DRUNet denoiser, with the only difference that it does not take a noise level map as additional input.

We have initialized $\mathcal{D}_\sigma$ using the pre-trained DRUNet denoiser (which we have reproduced based on the work in [2]). Then, we have trained $\mathcal{G}$ while jointly updating $\mathcal{D}_\sigma$, following the proposed framework in Section 4.3.2. The weight λ of the loss $\mathcal{L}_\mathcal{G}$ in Eq. 4.15 has been set equal to 0.004. The parameters σ and σ_0 have been selected following the alternating strategy described in Section 4.3.2: for half of the training iterations, we have used $\sigma = \sigma_0$ with a value chosen randomly with uniform distribution in $[0, 50]$; otherwise, σ and σ_0 have been chosen independently with the same uniform distribution.

The remaining training details are similar to the ones presented in [2] for the DRUNet training: the same large dataset of 8694 images composed of images from the Waterloo Exploration Database [98], the Berkeley Segmentation Database [99], the DIV2K dataset [100] and the Flick2K dataset [101] has been used. 16 patches of 128x128 have been randomly sampled from the training dataset for each iteration. We have used the ADAM optimizer [102] to minimize the loss $\mathcal{L}$ defined in Eq. 4.15. The learning rate has been initially set to 1e-4, and decreased by half every 100,000 iterations until reaching 5e-7, where the training stops.

4.4 Experimental results

4.4.1 Plug-and-play gradient descent

In this section, we evaluate the performance of the Plug-and-play gradient descent based on our ReG network. We refer to this approach as Plug-and-play Regularizing Gradient (PnP-ReG). We perform the evaluation on several inverse problems: super-resolution, deblurring and pixel-wise inpainting. Evaluations in this section are performed over the Set5 [103] and the CBSD68 [104] test datasets.

As the main goal of this approach is to solve inverse problems using simple gradient-based algorithms with a generic regularizer, we compare ourselves to algorithms that are designed to solve different inverse problems using a single regularization network in a Plug-and-play framework. Hence, we compare the performance of our PnP-ReG method to the PnP-ADMM with the DRUNet denoiser from [2] (see Appendix A.1 for the classical ADMM formulation and parameterization); the RED method [4] in gradient

descent with the same DRUNet denoiser; GS-PnP [3] where the network architecture used in the denoising function is the DRUNet as well; and Chang's projection operator (One-Net) [105] used in an ADMM framework. We also include a comparison with the Implicit Prior of [7] only for pixel-wise inpainting, since the method is based on the assumption that the kernel matrix is semi-orthogonal, which is not the case for the super-resolution and deblurring applications in our work.

For fair comparisons, we have reproduced all the results under the same conditions, i.e. using the same initialization and the same degradation operator A for each application as described in the following subsection. We have tuned the parameters of each method on the Set5 dataset for each application to obtain the best results, and we have used the same parameters for the CBSD68 dataset.

4.4.1.1 Parameters setting and implementation details

In this section, we give further implementation and parameterization details for our method as well as for the reference methods.

For the experimental results, we have used the ADAM optimizer [102] instead of the simple gradient step (i.e. line 4 in Algorithm 4) to solve Eq. 4.2 in the Plug-and-play framework. Similarly to [2], we have applied a periodical geometric self-ensemble data-augmentation method. This involves one transformation (e.g. flipping, rotation) on the input of the network and the counterpart inverse transformation on the output. Table 4.1 shows the parameters used during testing for PnP-ReG. The number of iterations N is set to 1500. In theory, the parameter σ in Eq. 4.2 should be equal to the true standard deviation σ_n of the Gaussian noise added on the degraded image. However, when $\sigma_n = 0$ (e.g. noiseless super-resolution, pixel-wise inpainting), choosing $\sigma = 0$ would completely remove the regularization term. For these cases, we choose a small non-zero value of σ depending on the application.

To reproduce the results of [105] we have used the model trained by the authors, which takes input images of size 64x64. Hence we have applied the network on quarter-overlapping sample patches in order to enhance the results by avoiding block artifacts. Table A.2 in Appendix A.2 lists the tuned hyper-parameters for the reproduction of [105].

We have implemented RED [4] with gradient descent using the DRUNet denoiser. For fair comparisons with our method, we have used the ADAM optimizer to perform the gradient update step. The number of iterations is set to 300 for the tasks of Super-resolution and Deblurring, and 800 in the case of pixel-wise inpainting. Table A.3 in

Appendix A.2 shows the tuned parameters for the implementation of RED.

For GS-PnP [3], we have used the parameters described in the paper, except for the standard deviation parameter of the denoiser in Super-resolution (noiseless or low-noise cases) and pixel-wise inpainting. In general, for a noise standard deviation σ_n in the degradation, the authors of [3] suggest to parameterize the denoiser with a standard deviation of $2 * \sigma_n$. However, this is not suitable for the noiseless applications where $\sigma_n = 0$ (e.g. noiseless super-resolution, pixel-wise inpainting). Therefore, we have tuned this parameter and have set a denoiser standard deviation of $6/255$ for Super-resolution in our paper. For pixel-wise inpainting, we have set this parameter to $50/255$ for the first 20 iterations and to $20/255$ for subsequent iterations. The remaining parameters were chosen as described in [3].

For the pixel-wise inpainting results of the Implicit Prior in [7], we have used the parameters suggested by the authors in the paper. We have computed the results by averaging 10 samples obtained with their stochastic method, as done in the original paper.

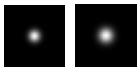
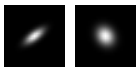
Finally, we have compared with the PnP-ADMM using DRUNet [2]. The PnP-ADMM formulation and its parameterization is elaborated in Appendix A.1. Furthermore, we have applied the periodical geometric self-ensemble data augmentation during the test as done in [2]. Table A.1 in Appendix A.1 shows the parameters that yielded the best results for the PnP-ADMM. We note that in the cases where we have added noise, fixing the standard deviation parameter of the denoiser gave better results than decreasing it throughout the iterations.

4.4.1.2 Super-resolution

First, we test our approach on the task of super-resolution. Low resolution images have been generated by applying a convolution kernel followed by a downsampling by a factor t. We have evaluated our method with bicubic and Gaussian convolutional kernels, with both 2x and 3x downsampling scales and Gaussian noise with 3 different noise levels $\sigma_n = \{0.0, \sqrt{2}, 2.55\}/255$. The Gaussian kernel has a standard deviation of $\sigma_b = 0.5 \cdot t$ (i.e. $\sigma_b = 1$ for x2 and $\sigma_b = 1.5$ for x3). In all the cases, the gradient descent has been initialized with a high resolution image obtained by the bicubic upsampling of the degraded image.

Tables 4.2 and 4.3 give a numerical comparison of our method with the aforementioned generic approaches for super-resolution of factor 2 and 3 respectively, for both bicubic and Gaussian filters. Numerical comparison gives higher values for PnP-ReG compared to the

Table 4.1 – Parameters used for our PnP-ReG method. μ: gradient step size, σ: weight of the regularization, σ_n: standard deviation of the AWGN added on the degraded image.

		$\sigma_n * 255$	μ	$\sigma * 255$
Super-resolution x2	Bicubic	0	0.008	1.2
		$\sqrt{2}$	0.002	$\sqrt{2}$
		2.55	0.002	2.55
	Gaussian	0	0.008	0.8
		$\sqrt{2}$	0.002	$\sqrt{2}$
		2.55	0.002	2.55
Super-resolution x3	Bicubic	0	0.002	0.9
		$\sqrt{2}$	0.002	$\sqrt{2}$
		2.55	0.002	2.55
	Gaussian	0	0.004	0.4
		$\sqrt{2}$	0.002	$\sqrt{2}$
		2.55	0.002	2.55
Deblurring		$\sqrt{2}$	0.004	$\sqrt{2}$
		2.55	0.004	2.55
		7.65	0.004	7.65
		$\sqrt{2}$	0.005	$\sqrt{2}$
		2.55	0.005	2.55
		7.65	0.005	7.65
Pixel-wise inpainting	0.1	0	0.025	3.6/255
	0.2	0	0.01	1/255

existing generic Plug-and-play approaches. Figure 4.3 shows a visual comparison of the results for a noiseless degradation with a bicubic kernel and a downsampling by a factor of 2. We observe sharper images with less aliasing artifacts produced by our approach.

4.4.1.3 Deblurring

We have degraded our images using two 25x25 isotropic Gaussian blur kernels of standard deviations of 1.6 and 2.0, as well as two anisotropic kernels that have been used in [106]. We have considered White Gaussian noise with 3 noise levels $\sigma_n = \{\sqrt{2}, 2.55, 7.65\}/255$. The blurred image is directly used as the initialization of the Plug-and-play gradient descent.

Table 4.4 gives the PSNR results [dB] obtained with our method when deblurring

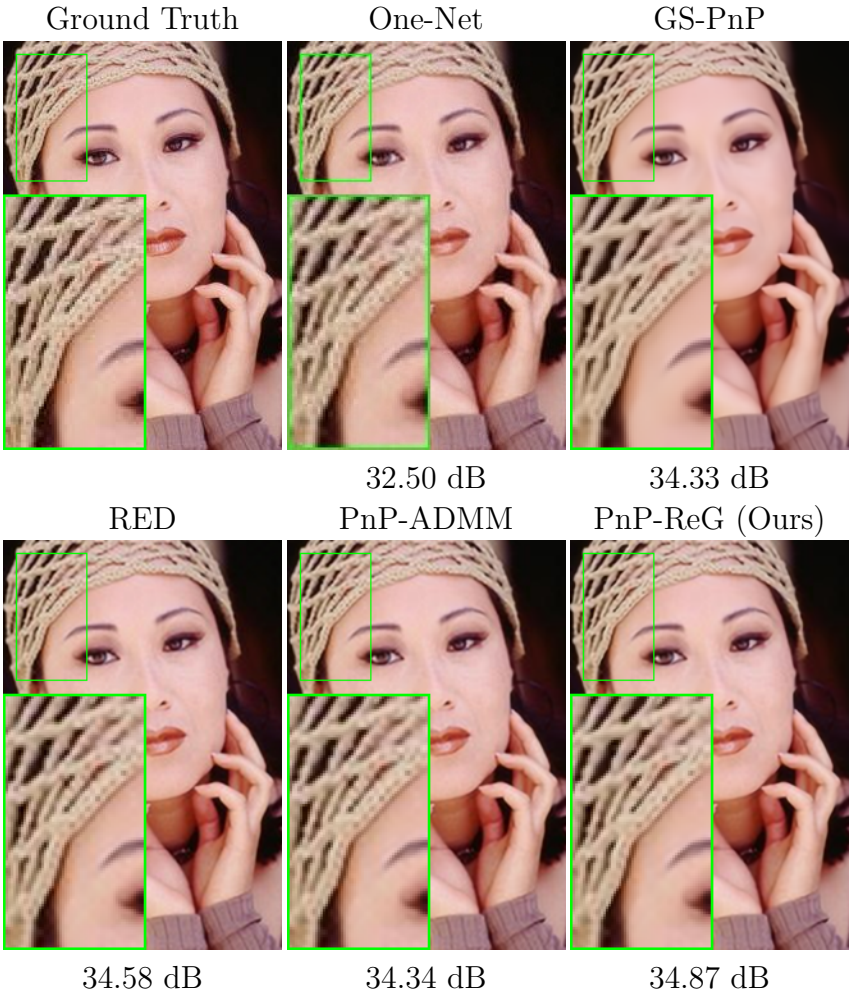


Figure 4.3 – Visual comparison of super-resolution results obtained with the projection operator (One-Net) [105], GS-PnP [3], RED [4], PnP-ADMM with the denoiser of [2] and our PnP-ReG method. Low resolution images generated with a bicubic kernel followed by a downsampling by a factor of 2.

images which have been degraded with isotropic and anisotropic kernels. We observe higher PSNR values with respect to the other generic methods for most cases when the added noise levels are $\sigma_n = \{\sqrt{2}, 2.55\}/255$. However, by increasing the noise level to 7.65/255, the best results have been obtained with GS-PnP. The visual comparison in Figure 4.4 shows that our approach successfully recovers the details without increasing the noise.

Table 4.2 – Super-resolution results (measured in PSNR [dB]) obtained with our PnP-ReG method; the input images have been corrupted using bicubic and Gaussian kernels followed by downsampling by a factor of 2 and adding Gaussian noise with 3 different noise levels $\sigma_n = \{0.0, \sqrt{2}, 2.55\}/255$. Comparison with the projection operator (One-Net) [105], GS-PnP [3], RED [4] and the PnP-ADMM with the denoiser of [2].

		(i) Bicubic			(ii) Gaussian		
	$\sigma_n * 255$	0.0	$\sqrt{2}$	2.55	0.0	$\sqrt{2}$	2.55
Set5	OneNet	33.22	33.70	33.20	32.67	33.08	32.45
	GS-PnP	34.58	34.49	**34.31**	33.98	33.88	**33.59**
	RED	35.05	34.49	33.78	34.99	33.80	32.84
	PnP-ADMM	35.20	34.42	33.80	35.14	33.69	32.74
	PnP-ReG (Ours)	**35.34**	**34.90**	34.29	**35.30**	**34.33**	33.41
CBSD68	OneNet	29.48	29.80	29.56	29.08	29.48	29.06
	GS-PnP	29.95	29.93	29.81	29.57	29.51	29.39
	RED	30.45	30.14	29.73	30.39	29.61	28.99
	PnP-ADMM	30.48	30.17	29.92	30.42	29.78	29.23
	PnP-ReG (Ours)	**30.55**	**30.34**	**30.06**	**30.50**	**30.01**	**29.44**

4.4.1.4 Pixel-wise inpainting

The degradation in the pixel-wise inpainting task consists of multiplying the ground-truth image by a binary mask. We have tested our approach with both 20% and 10% of known pixels rates. For the initialization image $\widehat{x}^0$, we have set the color of the unknown pixels to grey.

Table 4.5 shows a numerical evaluation of our method for the application of pixel-wise inpainting in terms of PSNR [dB]. We observe significant performance gains of the PnP-ReG method compared to the other methods, especially in the most challenging case where the known pixel rate is only 10%. Visual comparisons are given in Figure 4.5.

4.4.2 Unrolled gradient descent with $\mathcal{G}$

Aside from the Plug-and-play gradient descent, our approach for training $\mathcal{G}$ can also serve as a pre-training strategy for unrolled gradient descent. In unrolled optimization methods, the regularization network is trained for each inverse problem such that applying a fixed number of iterations of the algorithm (e.g. Eq. 4.4 for gradient descent) best approximates the ground truth image. We describe the unrolled training approach in

Table 4.3 – Super-resolution results (measured in PSNR [dB]) obtained with our PnP-ReG method; the input images have been corrupted using bicubic and Gaussian kernels followed by downsampling by a factor of 3 and adding Gaussian noise with 3 different noise levels $\sigma_n = \{0.0, \sqrt{2}, 2.55\}/255$. Comparison with the projection operator (One-Net) [105], GS-PnP [3], RED [4] and the PnP-ADMM with the denoiser of [2].

		(i) Bicubic			(ii) Gaussian		
	$\sigma_n * 255$	0.0	$\sqrt{2}$	2.55	0.0	$\sqrt{2}$	2.55
Set5	OneNet	29.65	30.18	29.79	29.52	29.71	29.05
	GS-PnP	31.48	31.43	**31.24**	30.91	30.84	**30.59**
	RED	31.47	31.22	30.78	31.44	30.77	30.05
	PnP-ADMM	31.49	31.05	30.39	31.45	30.22	29.17
	PnP-ReG (Ours)	**31.75**	**31.46**	31.13	**31.60**	**31.09**	30.39
CBSD68	OneNet	26.17	26.89	26.67	25.21	26.69	26.33
	GS-PnP	27.11	27.08	26.99	26.80	26.73	26.64
	RED	27.50	27.33	27.11	27.40	27.06	26.24
	PnP-ADMM	27.51	27.33	26.97	**27.47**	26.83	26.20
	PnP-ReG (Ours)	**27.55**	**27.41**	**27.23**	27.46	**27.17**	**26.77**

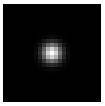
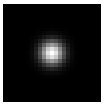
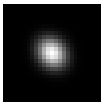
Algorithm 5 for a gradient descent optimization. While this end-to-end training strategy loses the genericity of the Plug-and-play approach, it typically improves the performance.

However, to facilitate the training, it is generally required to initialize the network weights with a generically pre-trained version. When unrolling proximal algorithms such as ADMM, a pre-trained deep denoiser can be used since it can be interpreted as the proximal operator of a generic regularization function. On the other hand, the gradient descent requires instead the gradient of a regularizer. Hence, a denoiser cannot be directly used as a pre-trained network. By transferring the image prior implicitly represented by the denoiser $\mathcal{D}$ to our ReG network $\mathcal{G}$, our method thus provides a satisfying pre-trained network for unrolled gradient descent.

We have tested this approach for noiseless super-resolution of factor 2 and 3, as well as for deblurring using 2 isotropic Gaussian kernels (with standard deviations of 1.6 and 2.0) and additive Gaussian noise of standard deviation $\sqrt{2}/255$. We have compared our results with those obtained when using a network learned end-to-end in an unrolled environment without the pre-training (i.e. random weight initialization).

Both versions (pre-trained and not pre-trained) have been unrolled in the same conditions. Table 4.6 gives the training parameters for each of the different tasks. For all 4

Table 4.4 – Deblurring results (measured in PSNR [dB]) obtained with our PnP-ReG method. The input blurred images have been generated using two isotropic Gaussian kernels of respective standard deviations 1.6 and 2.0, and two anisotropic Gaussian kernels from [2] followed by adding Gaussian noise with 3 different noise levels $\sigma_n = \{\sqrt{2}, 2.55, 7.65\}/255$. We compare with the projection operator (One-Net) [105], GS-PnP [3], RED [4] and the PnP-ADMM with the denoiser of [2].

							
	$\sigma_n * 255$	$\sqrt{2}$	2.55	7.65	$\sqrt{2}$	2.55	7.65
Set5	OneNet	32.18	31.54	27.86	30.41	29.52	27.20
	GS-PnP	32.84	32.31	**30.93**	31.04	29.98	**29.76**
	RED	32.63	31.76	29.83	31.25	30.62	29.21
	PnP-ADMM	32.83	32.06	30.43	31.55	30.88	29.30
	PnP-ReG (Ours)	**33.40**	**32.51**	30.63	**31.96**	**31.19**	29.46
CBSD68	OneNet	28.47	28.27	25.86	26.91	26.97	25.29
	GS-PnP	28.98	**28.64**	**27.64**	27.5	27.33	**26.57**
	RED	29.02	28.16	26.91	27.63	27.24	26.20
	PnP-ADMM	29.21	28.54	27.20	27.91	27.40	26.25
	PnP-ReG (Ours)	**29.38**	28.62	27.07	**27.99**	**27.41**	26.21
							
	$\sigma_n * 255$	$\sqrt{2}$	2.55	7.65	$\sqrt{2}$	2.55	7.65
Set5	OneNet	29.26	28.90	26.71	28.93	28.18	26.76
	GS-PnP	30.18	29.30	**29.13**	29.70	29.07	**28.55**
	RED	30.53	29.98	28.67	29.91	29.53	28.24
	PnP-ADMM	31.21	30.56	28.95	30.33	29.07	28.17
	PnP-ReG (Ours)	**31.26**	**30.57**	28.84	**30.67**	**29.95**	28.31
CBSD68	OneNet	26.52	26.57	25.04	25.88	25.94	24.77
	GS-PnP	27.02	27.04	**26.32**	26.27	26.41	**25.72**
	RED	27.36	26.99	26.00	26.61	26.36	25.53
	PnP-ADMM	**27.83**	**27.32**	26.16	26.9	26.46	25.46
	PnP-ReG (Ours)	27.71	27.16	25.96	**26.93**	**26.47**	25.47

cases, we have trained the networks using the DIV2K dataset [100] of 800 images, over 600 epochs by randomly taking 48x48 patches from the dataset. In addition, we include a comparison with Total Deep Variation (TDV) method [88]. TDV also performs unrolled

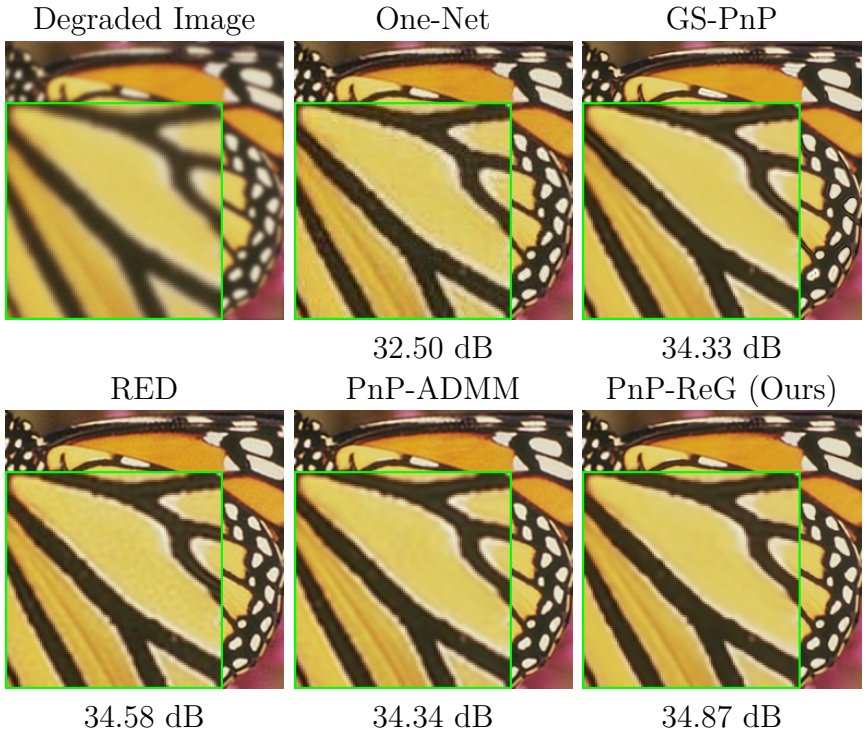


Figure 4.4 – Visual comparison of deblurring results obtained with the projection operator (One-Net) [105], GS-PnP [3], RED [4], PnP-ADMM with the denoiser of [2] and our PnP-ReG method. The blurred images have been generated using an isotropic Gaussian kernel of standard deviation $\sigma_b = 1.6$ followed by adding Gaussian noise of standard deviation $\sigma_n = \sqrt{2}/255$.

optimization, where the network represents the gradient of the regularization function. In [88], the network is not pre-trained. Instead, the training starts with a small number of unrolled iterations $N = 2$, and N is incremented every 700 epochs. Note that the authors originally trained the TDV network for $N = 10$ iterations of an unrolled proximal gradient descent algorithm. However, for fair comparisons with our approach, we re-trained it with $N = 6$ iterations.

Tables 4.7 and 4.8 show the PSNR results [dB] for both networks (pre-trained and not pre-trained) as well as for the TDV, for super-resolution and deblurring respectively, using Set5 [103], Set14 [27] and BSDS100[104]. As expected, using $\mathcal{G}$ for weight initialization improves the results of the unrolled gradient descent for the 4 tested cases (with up to 0.9 dB).

Some visual comparisons of super-resolution results with magnifying factor of 3, and of deblurring images corrupted by a Gaussian kernel of standard deviation 2.0 are respec-

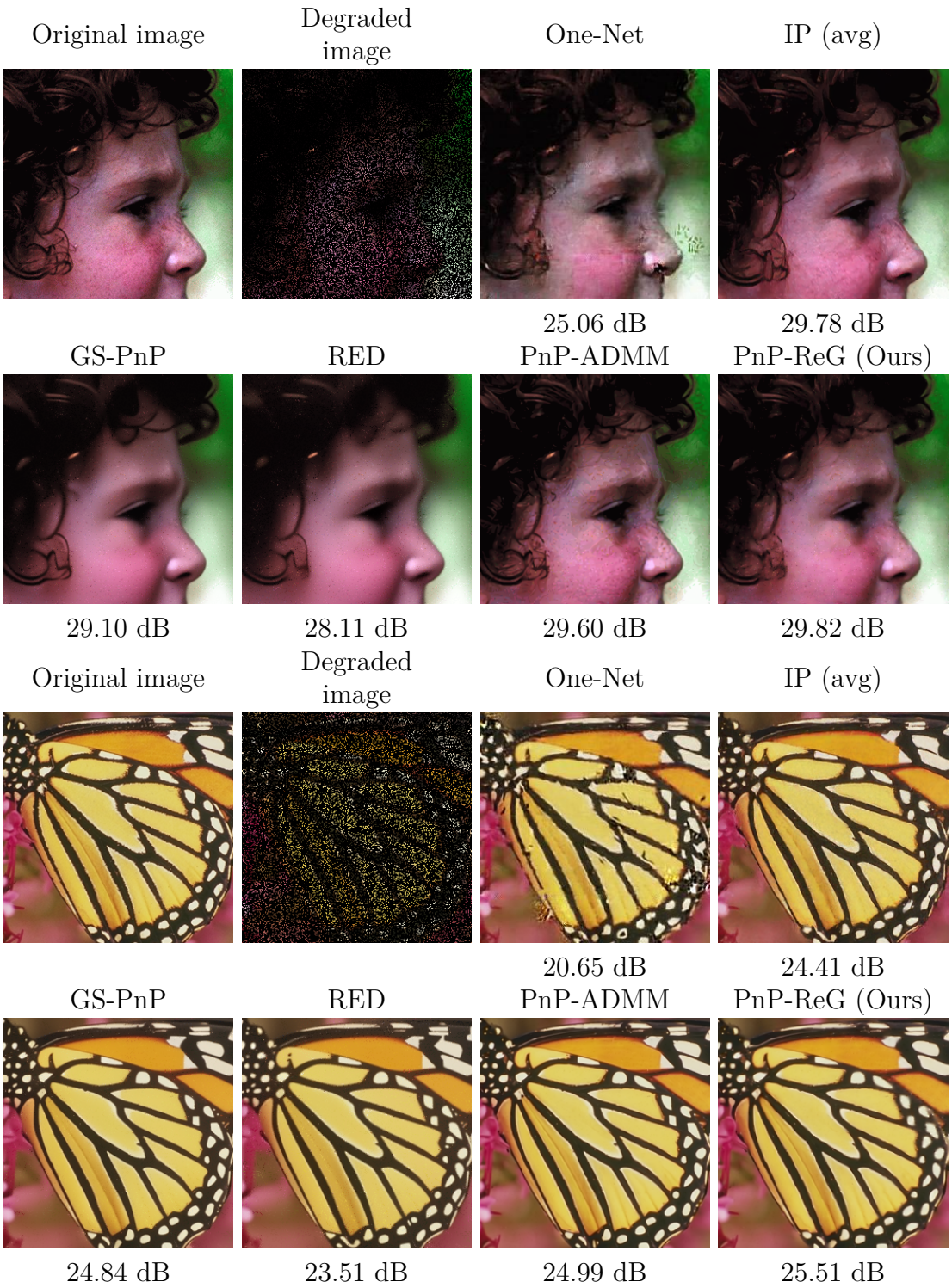


Figure 4.5 – Visual comparison of pixel-wise inpainting results with known pixel rate of $p = 20\%$, obtained with the projection operator (One-Net) [105], the Implicit Prior (IP) of [7], GS-PnP [3], RED [4], PnP-ADMM with the denoiser of [2] and our PnP-ReG method.

Table 4.5 – Pixel-wise inpainting results (measured in PSNR [dB]) obtained with our PnP-ReG method; the corrupted images have been generated by keeping 20% and 10% of the known pixels. Comparison with the projection operator (One-Net) [105], the Implicit Prior (IP) of [7], GS-PnP [3], RED [4] and PnP-ADMM with the denoiser of [2].

	Set5		CBSD68	
	(i) 10%	(ii) 20%	(i) 10%	(ii) 20%
OneNet	17.20	24.06	14.72	20.46
IP (avg)	25.28	28.91	24.12	26.55
GS-PnP	25.08	28.70	23.58	25.91
RED	22.75	27.17	22.24	23.22
PnP-ADMM	26.20	30.20	24.06	26.75
PnP-ReG (Ours)	**26.94**	**30.36**	**24.43**	**27.09**

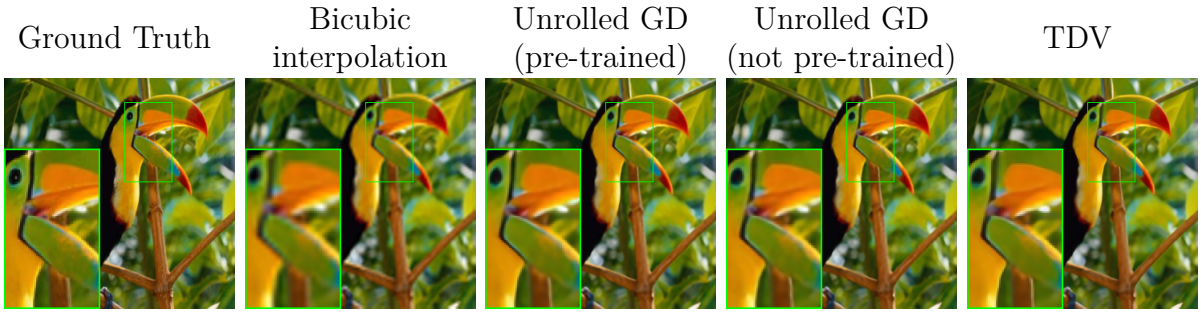


Figure 4.6 – Visual comparison of super-resolution results between unrolled gradient descent with and without pre-training. Low resolution images have been generated with a bicubic kernel followed by a downsampling by a factor of 3.

tively shown in Figures 4.6 and 4.7. The visual results confirm that better reconstruction of the details is obtained when our pre-training is used. While the TDV results display even sharper edges, the PSNR remains lower because of exaggerated sharpness in comparison to the ground truth.

4.4.3 Analysis of the joint training

In this section, we experimentally verify the advantages of updating the denoising network within the training of our ReG network $\mathcal{G}$, compared to letting the denoiser fixed. In the latter case, δ is always set to 0 (i.e. $\mathcal{L} = \lambda \mathcal{L}_{\mathcal{G}}$).

First, we compare the performance of our regularizing gradient network trained in

Algorithm 5 Unrolled gradient descent with $\mathcal{G}$

initialize $\mathcal{G}, \sigma_n, \mu, A, N$
for each epoch **do**
  for each batch **do**
    $\eta \leftarrow \mathcal{N}(0, \sigma_n)$
    $x_{gt} \leftarrow$ ground-truth batch
    $y \leftarrow Ax_{gt} + \eta$
    $x_0 \leftarrow y$
    for $k \leftarrow 0$ to $N - 1$ **do**
      $x_{k+1} \leftarrow x_k - \mu[A^T(Ax_k - y) + \sigma^2\mathcal{G}(x_k)]$
    end for
    loss $\leftarrow \|x_N - x_{gt}\|_2^2$
    Update weights of $\mathcal{G}$ with back-propagation
  end for
end for

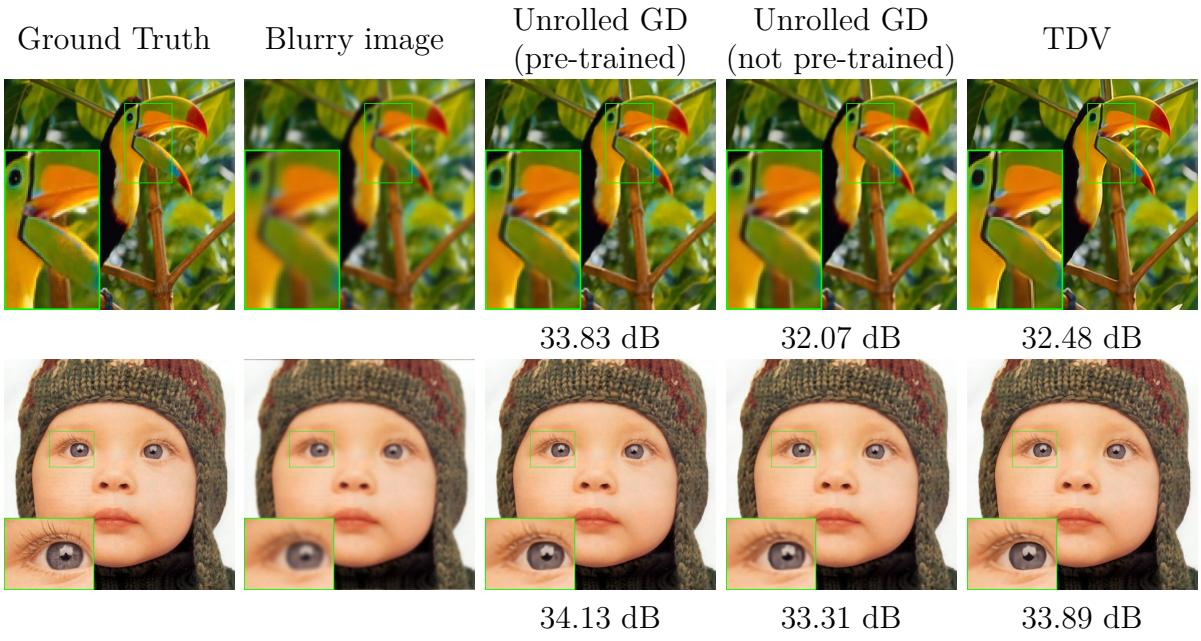


Figure 4.7 – Visual comparison of deblurring results between unrolled gradient descent with and without pre-training. The blurred images have been generated using an isotropic Gaussian kernel of standard deviation 2.0.

Table 4.6 – Parameters used for unrolled gradient descent optimization for both the pre-trained and not pre-trained versions (i) Super-resolution of factor 2 and 3 with input images corrupted using a bicubic filter (ii) Deblurring images corrupted using an isotropic Gaussian kernel of standard deviation σ_b of 1.6 and 2.0. σ_n: Standard deviation of the White Gaussian noise added on the corrupted image, σ: weight of the regularization term, μ: gradient step size , N: number of unrolled iterations. The training is performed over the DIV2K dataset [100].

	(i) Super-resolution		(ii) Deblurring	
	x2	x3	$\sigma_b = 1.6$	$\sigma_b = 2.0$
$\sigma_n * 255$	0	0	$\sqrt{2}$	$\sqrt{2}$
$\sigma * 255$	1.2	1.6	$\sqrt{2}$	$\sqrt{2}$
μ	0.008	0.1	0.004	0.004
N	6	6	6	6

both scenarios. An example of deblurring results with the Plug-and-play gradient descent is shown in Figure 4.8. It is clear that leaving the denoiser fixed to its pre-trained state degrades the performance of $\mathcal{G}$: the reconstructed image in Figure 4.8 (a) remains more blurry than that in Figure 4.8 (b) and it also presents colored fringes artifacts. On the other hand, when the denoiser is updated when training $\mathcal{G}$, the convergence of the Plug-and-play gradient descent is significantly improved as well as the visual result, as shown in Figure 4.8 (b,c).

A possible explanation of the worse results when fixing the denoiser in our training is that the pre-training of $\mathcal{D}_\sigma$ does not guarantee that there exists a differentiable regularizer ϕ for which $\text{prox}_{\sigma^2\phi} = \mathcal{D}_\sigma$ for every value of σ. In other words, the assumption that $\mathcal{D}_\sigma$ is a MAP Gaussian denoiser for a differentiable prior may not be satisfied. However, by jointly updating the denoiser with the ReG network, the modified denoiser better represents such a MAP Gaussian denoiser for the corresponding regularizer.

In a second experiment, we compare the performances of the updated denoiser and the original DRUNet when used in the Plug-and-play ADMM algorithm. Figure 4.9 shows for each ADMM iteration the PSNR of the reconstructed images and the MSE of the difference between two consecutive iterations for both versions of the denoiser. We can observe that although the original DRUNet can obtain better PSNR performances when stopping the ADMM after a few iterations (see subfigure (a)), the algorithm does not converge, and may even strongly diverge after a sufficiently large number of iterations.

Table 4.7 – Super-resolution results (measured in PSNR [dB]) obtained with unrolled gradient descent (the input images have been corrupted using a bicubic kernel and a downsampling factor of 2 and 3). The evaluation has been performed using the Set5, Set14 and BSDS100 datasets. The restoration has been performed using unrolled gradient descent initialized with our pre-trained network $\mathcal{G}$ and without weight initialization, as well as TDV[88].

		pre-trained (ours)	not pre-trained	TDV
(i) x2	Set5	35.61	35.42	34.57
	Set14	31.18	30.93	30.31
	BSDS100	30.77	30.68	30.23
(i) x3	Set5	32.10	31.88	31.47
	Set14	28.00	27.79	27.40
	BSDS100	27.68	27.63	27.34

On the other hand our modified denoiser allows for a better convergence of the Plug-and-play ADMM.

4.5 Discussion

In this section, we discuss the advantages and the limitations of our PnP-ReG method. First, we point out that the proposed approach only requires the tuning of the gradient step μ when the noise level is known, while the reference PnP methods heavily rely on the tuning of several hyper-parameters, sometimes even including the number of iterations. This makes our method easier to use since parameter tuning can take a lot of effort.

Furthermore, the comparison of the convergence of the different PnP algorithms in Figures 4.10 and 4.11 show that although our method requires more iterations than PnP-ADMM and GS-PnP to reach its highest PSNR result, it converges to a fixed point that provides the best quality. On the other hand, PnP-ADMM reaches its highest PSNR after a few iterations, but then diverges, which requires a careful tuning of the number of iterations for each application in practice. The convergence of our PnP-ReG method is more comparable to the RED-GD method (although significant PSNR gains are obtained with our method). This may be explained by the fact that both RED-GD and PnP-ReG are based on gradient descent while PnP-ADMM and GS-PnP use proximal algorithms.

Aside from the convergence analysis, one may notice that, even though we have de-

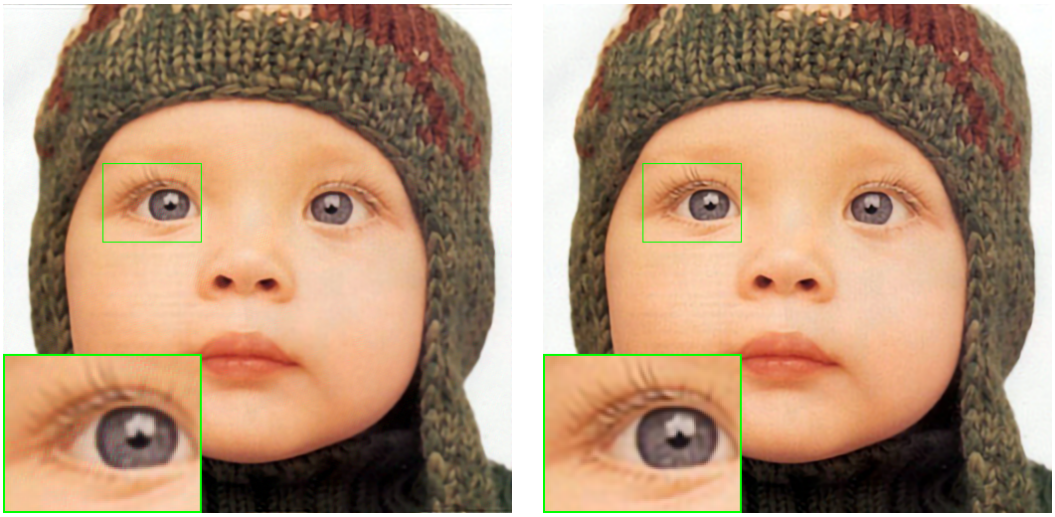
Table 4.8 – Deblurring results (measured in PSNR [dB]) obtained with unrolled gradient descent (the input images have been corrupted using isotropic Gaussian kernels with standard deviation 1.6 and 2.0 and Gaussian noise of standard deviation $\sqrt{2}/255$). The evaluation has been performed using the Set5, Set14 and BSDS100 datasets. The restoration has been performed using unrolled gradient descent initialized with our pre-trained network $\mathcal{G}$ and without weight initialization, as well as [88].

		pre-trained (ours)	not pre-trained	TDV
(i) $\sigma_0 = 1.6$	Set5	33.51	32.57	32.32
	Set14	30.14	29.33	29.17
	BSDS100	29.80	29.12	29.01
(ii) $\sigma_0 = 2$	Set5	31.63	30.94	30.52
	Set14	28.29	27.70	27.46
	BSDS100	28.08	27.52	27.51

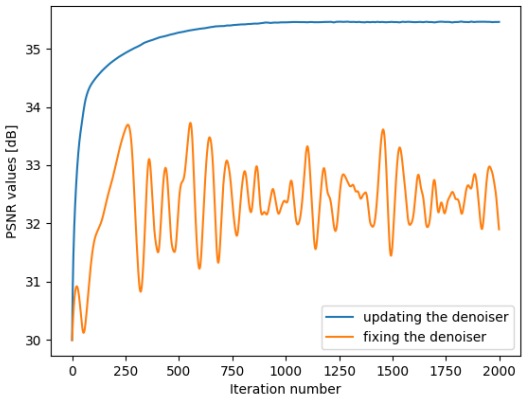
signed our loss from Eq. 4.12 so that $\mathcal{G}$ models $\nabla\phi$, we did not enforce the network to have a symmetric Jacobian matrix. As a result, our training framework does not guarantee that $\mathcal{G}$ can be interpreted mathematically as a conservative vector field, and thus, as the gradient of a scalar potential. However, using a non-conservative vector field for the gradient update step may not necessarily degrade the gradient descent performance. For example, advanced gradient-based algorithms typically alter the gradient (e.g. with momentum), in a way that loses the conservativeness property of the original gradient, while improving the algorithm's performances (more robust to local minima, faster convergence...). Note also that it would be possible to enforce the Jacobian symmetry property in our method by using a network directly modeling ϕ and by explicitly computing its gradient, as done in [3], [107]. However, the explicit gradient computation has the same complexity as the network ϕ, hence doubling the computation cost. A possible direction for future work would be to study whether such a constraint could improve the Plug-and-play gradient descent without sacrificing the computational complexity.

4.6 Conclusion

In this chapter, we have proposed a novel framework for regularizing linear inverse problems. Our approach makes it possible to solve Plug-and-play algorithms using gradient descent, where the gradient of the regularizer is required rather than its proximal



(a) 33.72 dB (b) 35.46 dB

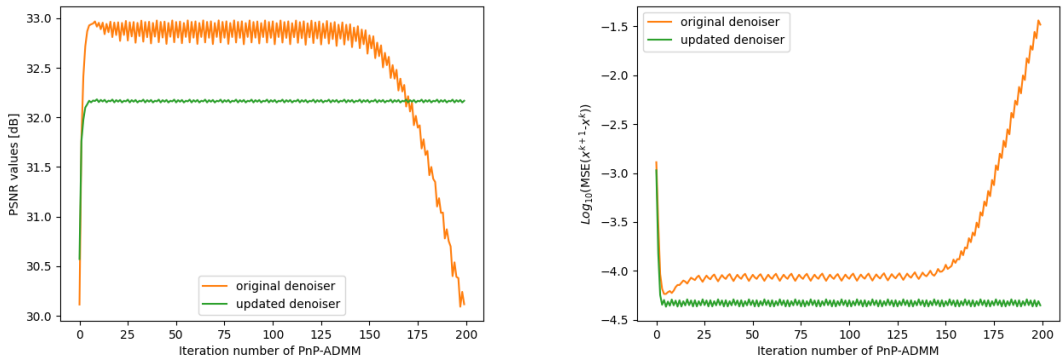


(c)

Figure 4.8 – Comparison of the performances of PnP-ReG, when the ReG network is trained (a) with a fixed denoiser and (b) while jointly updating the denoiser. (c) PSNR [dB] values over the gradient descent iterations. The results are shown for the deblurring problem (the blurred images have been generated using isotropic Gaussian kernels of standard deviation 1.6 followed by adding Gaussian noise of standard deviation $\sqrt{2}/255$).

operator. We have proved that it is mathematically possible to train a network that models the gradient of a regularizer, jointly with a denoising neural network.

The results have demonstrated that the joint training of our network with a DRUNet denoiser has several advantages. First, our network can be used to regularize the gradients in a Plug-and-play gradient descent algorithm and can outperform other generic approaches in different inverse problems such as super-resolution, deblurring and pixel-



(a) (b)

Figure 4.9 – Comparison of the performances of the original and the updated denoisers in PnP-ADMM for deblurring, using the Set5 dataset (the blurred images have been generated using an isotropic Gaussian kernel of standard deviation 1.6 followed by adding Gaussian noise of standard deviation $\sqrt{2}/255$): (a) Average PSNR [dB] over the ADMM iterations (b) MSE of the difference between two consecutive iterations (in log scale).

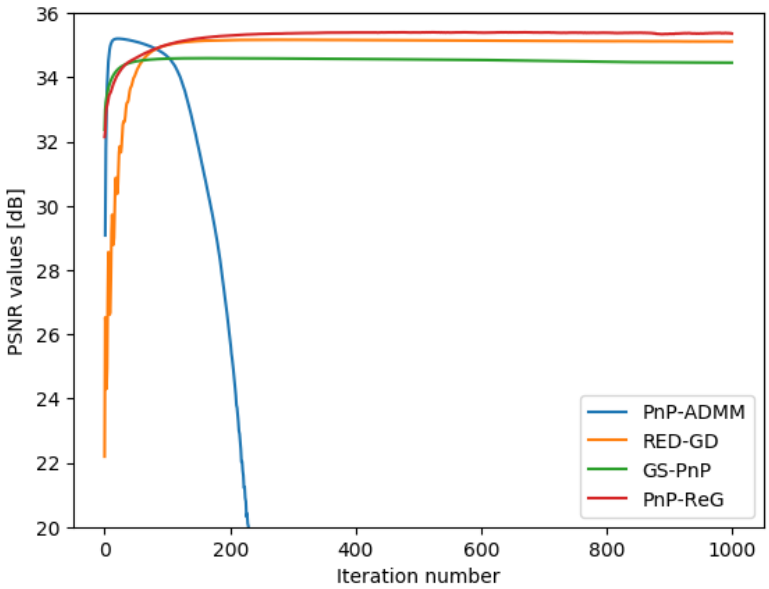


Figure 4.10 – Comparison of the convergence of PnP-ADMM with the denoiser of [2], RED [4], GS-PnP [3] and PnP-ReG for Super-resolution. The input low resolution images have been generated using bicubic downsampling with a factor 2. The results are averaged over the images of the Set5 dataset. For PnP-ADMM, we have used the setting with variable parameter s^k until the 25^{th} iteration for which the best results are obtained (see Table A.1), and let s^k fixed afterwards.

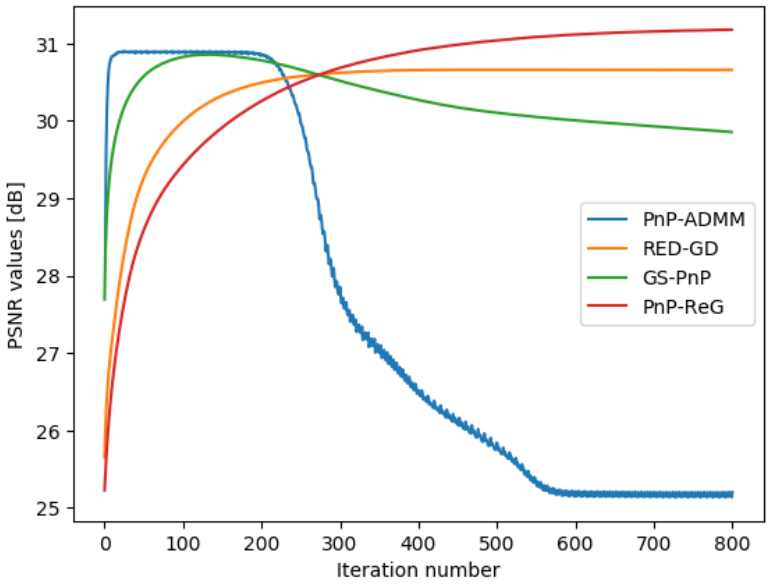


Figure 4.11 – Comparison of the convergence of PnP-ADMM with the denoiser of [2], RED [4], GS-PnP [3] and PnP-ReG for Deblurring. The degraded images have been generated using an isotropic Gaussian kernel of standard deviation 2.0 followed by adding Gaussian noise of standard deviation $\sigma_n = 0.01$. The results are averaged over the images of the Set5 dataset. For PnP-ADMM, we have used the setting with fixed parameter $s^k = 30/255$ for which the best results are obtained (see Table A.1).

wise inpainting. Second, our network can also serve as a pre-training strategy for unrolled gradient descent and yield a significant improvement. Lastly, the joint training of the denoiser with the regularizing gradient network makes the former match better the definition of a proximal operator compared to the original pre-trained DRUNet.

Part II

Inverse problems in omnidirectional imagery

Chapter 5

A Comprehensive Review of omnidirectional images: processing methodologies and inverse problems

---

The second part of this thesis is dedicated to advancing solutions for inverse problems in omnidirectional images. In this chapter, we provide a comprehensive review on omnidirectional images, exploring their acquisition, representation, challenges and the existing literature on inverse problems.

5.1 Introduction to omnidirectional imaging

Omnidirectional images, also known as spherical images or 360° images, are high-resolution visual contents that cover a 360° field of view capturing the light field converging to a single point. Thanks to their human-friendly nature, they have gained a particular attention in the domains of Virtual Reality (VR), Augmented Reality (AR) and robotics.

Nevertheless, the acquisition and the display of omnidirectional images present complex challenges. Also, traditional 2D processing tools can not be directly applied to spherical data. In fact, when dealing with omnidirectional images, the geometry is fundamentally different. A sphere has a curved surface, and points on its surface are not represented by simple x and y coordinates as in a 2D plane. Instead, spherical coordinates (i.e. polar angle $\Theta \in [0, \pi]$ and azimuth angle $\phi \in [-\pi, \pi]$) are used to indicate locations on the surface of the sphere. Hence, processing omnidirectional images is not a straightforward task.

Processing 2D images is relatively straightforward: we visualize and process the captured image after applying minor pre-processing steps (Figure 5.1). However, this sim-

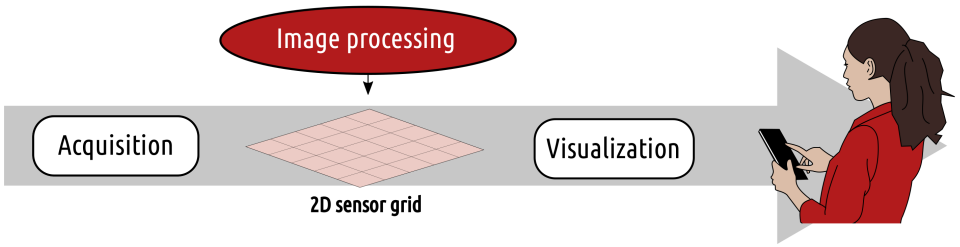


Figure 5.1 – Image processing pipeline for 2D images. Figure extracted from [108].

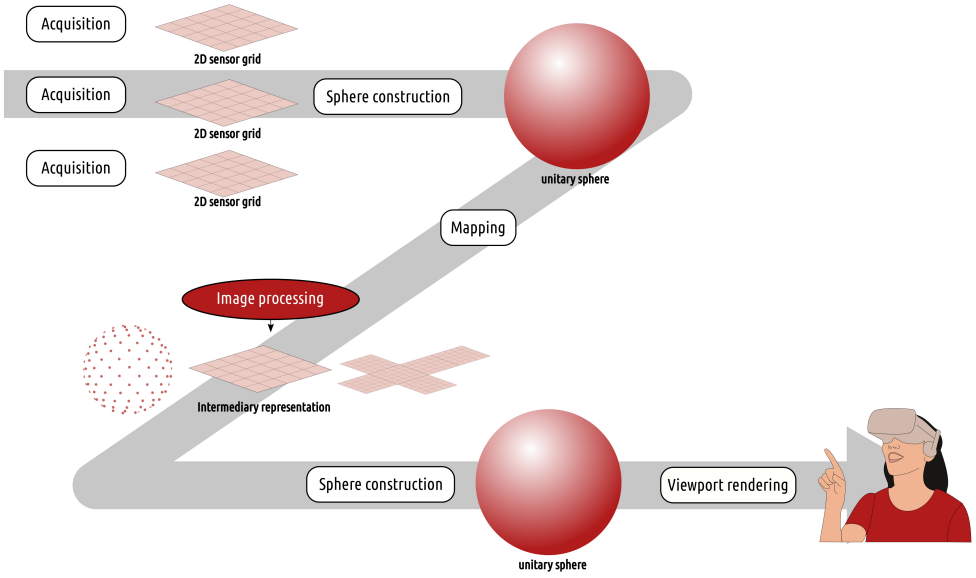


Figure 5.2 – Image processing pipeline for omnidirectional imagery. Figure extracted from [108].

plicity doesn't extend to omnidirectional imaging, where we don't directly process and visualize the captured data. Figure 5.2 represents a typical processing pipeline for omnidirectional imagery. First, it is not feasible, using a single camera, to capture the light ray coming from all directions effectively covering the entire sphere. Multiple captures should be merged in order to build an omnidirectional image that covers the entire sphere surface. Then, a mapping is typically used to process the spherical image. Finally, given the finite field of view in human vision, only a portion of the spherical image becomes visible. This necessitates projecting the relevant section onto a *viewport*. In other words, omnidirectional image processing is not a straightforward task and demands specialized processing tools and techniques. In the next section, we briefly discuss about the acquisition process of 360° images, then elaborate different representation models in Section 5.3.

5.2 Acquisition of omnidirectional images

Capturing spherical data can be done in different ways. It is important to note, that achieving a full 360° coverage using a single camera is not possible. Instead, several captures are typically combined using stitching methods, in order to build one spherical image. One approach involves the utilization of multiple perspective cameras, each covering a small angle of view. This approach is expensive, since it requires a lot of processing and a significant amount of perspective cameras to obtain one spherical image.

Moreover, some cameras can cover a wider angle of view, hence a fewer number of cameras are needed to cover the whole sphere. Two popular such devices are catadoptric and fish-eye based cameras. The former employs a curved mirror in its optical setup, which reflects incoming light rays from the scene onto the camera sensor. However, due to the substantial space required for a second symmetric mirror, it is constrained to capturing only one hemisphere. On the other hand, fish-eye cameras have gained significant popularity in recent years. These cameras use a special fish-eye lens that distorts the incoming light to cover a wide field of view. Fish-eye cameras equip nowadays most budget-friendly cameras available on the public market.

For more details on omnidirectional image acquisition, we refer interested readers to the Chapter 4 of Tzafestas' book [109], as well as to [108], [110].

5.3 Projecting spherical data to the Euclidean space

Omnidirectional images rely on a spherical geometry, which is not an Euclidean space where 2D processing tools are well defined. In order to take advantage of computing tools and methods that have been developed for 2D images, spherical images are often mapped to two-dimensional planar representations such as equirectangular projection (ERP) [9], cubemap projection (CMP) [111] or rhombic dodecahedron projection [112]. While a vast number of such mappings have been designed [1], in this section, we discuss two popular mapping methods that have been commonly adopted in the literature.

5.3.1 Equirectangular projection (ERP)

Equirectangular projection, also known as panorama projection, is the most popular mapping used to represent spherical data. It projects the omnidirectional image to

1. https://map-projections.net

the Euclidean space by mapping each point on the sphere's surface to a corresponding point on the grid, using a linear relationship between longitude (azimuthal angle) and $x-$coordinate, and a linear relationship between latitude (polar angle) and $y-$coordinate. However, this sampling induces significant distortions as we move towards the poles. It can be seen in Figure 5.3, the red and blue portions are mapped to two equal regions on the projection, whereas in reality, their sizes on the sphere are very different. Moreover, this mapping causes strong deformation of objects (Figure 5.5(b)).

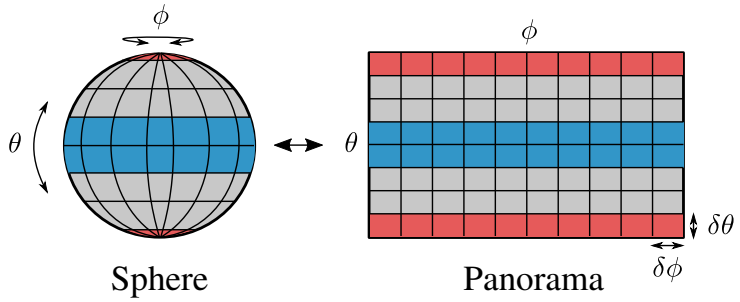


Figure 5.3 – Equirectangular projection. Figure extracted from [108].

5.3.2 Cubemap projection (CMP)

Cubemap projection is a mapping technique that projects the spherical image onto the six faces of a cube (Figure 5.4). The projection on each of the faces imitates the capture of a perspective camera with a $\frac{\pi}{2}$x$\frac{\pi}{2}$ field of view. An illustration of cubemap projection can be seen in Figure 5.5(c). The cube faces have natural image statistics, however, several discontinuities arise when using such mappings, which generates misleading artifacts (i.e. fake corners, cut objects...).

Projecting the spherical image into a 2D plane induces significant distortions, that can lead to misrepresentations and inaccuracies when processing the data. These distortions arise due to the inherent differences in the geometry between the spherical surface and the flat plane. As a consequence, researchers have proposed direct processing of the sphere itself (Figure 5.5 (d)), bypassing the need for intermediary mapping techniques. Nevertheless, as already mentioned, processing on the sphere is not a straightforward task and requires a deep understanding of the underlying non-Euclidean geometry.

Moreover, even elementary processing operations, such as translation or convolution, are not easily defined. In the following section, we briefly review different approaches that have been used to perform convolution on omnidirectional images.

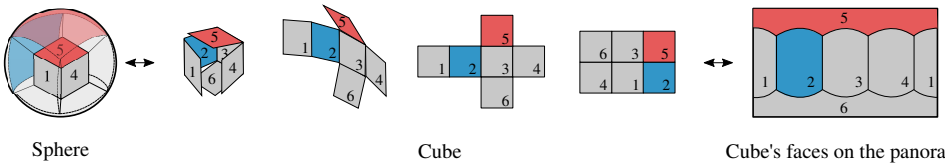


Figure 5.4 – Cubemap projection. Figure extracted from [108].

5.4 Convolution on omnidirectional images

The implementation of spherical convolutions comes with many challenges. First, considering registered images, a convolution is desired to be rotation equivariant for any rotation around the polar axis, meaning that the convolution operation should remain consistent under rotations of the input signal around the north pole. Consequently, the convolution filters of the CNN can be applied consistently all over the sphere. Second, the filters should be able to increase their convolutional range and have anisotropic responses to ensure expressiveness. Finally, computational efficiency in spherical convolutions is very important for real-time processing and memory limitation.

A popular approach to process the spherical image is to map it on a 2D planar grid and apply traditional 2D CNN tools. Primitive existing strategies consider a CNN trained on perspective images and either (i) apply it directly to an equirectangular projection [113], [114] or (ii) apply it to the tangent planes that they sample all over the sphere [115]–[118]. While the former is inaccurate because of the distortions induced by the perspective projection, the latter solution is exact but computationally inefficient. Another solution proposed by Su et al. [119] is based on knowledge distillation where the network takes an equirectangular image as input and for every image position, learns to mimic the output result of applying the corresponding tangent plane to a CNN learned on perspective images. However, they learn kernels of different sizes for every row of the equirectangular projection to ensure that the kernels cover the same surface as the spherical image. This comes with a significant model size and memory footprint, which limits not only the training, but also the usage of the learned model on devices with constrained computational resources. The authors later propose the KTN [120], [121] which solves the problem of memory bloat while achieving high accuracy in object detection, yet still suffers from a significant computational cost because the kernels are generated at run time.

A second category of methods involves directly implementing convolution on the sphere. One approach withing this category involves using the spectral domain [122]–[125]. Spherical harmonics form a complete orthonormal basis on the surface of the

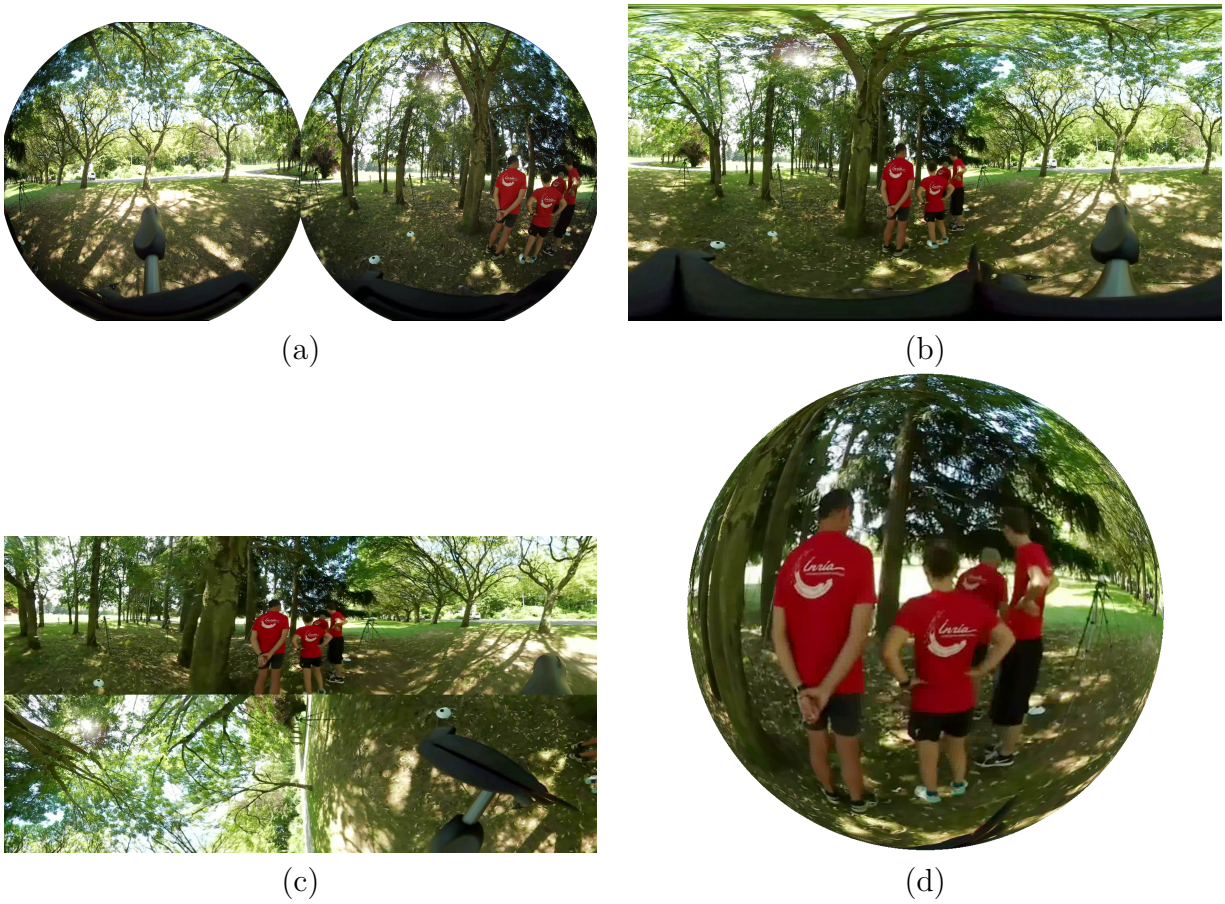


Figure 5.5 – (a) Image acquired with double fish-eye camera and its corresponding (b) equirectangular projection, (c) cubemap projection and (d) spherical representation. Figure extracted from [108].

sphere. Hence, in the spectral domain, spherical convolutions can be performed by the simple product of the spherical harmonics coefficients of the spherical data and the kernel function. While the convolution is consistent and expressive, these methods are very complex and have significant memory overhead, thus are not suitable for real-world high-resolution omnidirectional images.

Another approach is to model the sphere as a graph and to approximate the convolution using a Chebyshev polynomial representation [126], [127]. Although the convolution is rotation equivariant and consistent over the sphere, the learned filters are not expressive because a single coefficient by neighborhood can be learned, therefore they can only be isotropic due to the missing directionality in a graph. To address this drawback of graph-based methods, multi-graph learning was proposed by Khasanova et al. [128] where

multiple directed graphs are used to represent different directions in order to maintain the orientation data and ensure filter expressiveness. Nevertheless, a large storage is required to construct all the graphs.

Finally, to overcome the limitations of existing methods, Bidgoli et al.[129] have proposed a novel framework for spherical processing (OSLO) where they reach the same filter expressiveness, consistency and complexity of 2D CNNs. For this purpose, they sample the sphere using HEALPix [130] and define the convolution in the pixel domain. We elaborate OSLO and HEALPix in the upcoming chapter, as they play a pivotal role in shaping our contribution.

5.5 Inverse problems in omnidirectional images

In contrast to the extensive exploration of inverse problems in 2D imaging, the field of inverse problems in omnidirectional images has received comparatively less attention. Most of the existing work has focused on super-resolution, since omnidirectional images must require very high resolution in order to ensure an acceptable quality of user experience. In this section, we summarize the different approaches that have been proposed to solve omnidirectional inverse problems. In particular, we focus on super-resolution and denoising.

5.5.1 Super-resolution

Early methods for super-resolving omnidirectional images used a sequence of low-resolution images and assembled them to get high resolution outputs [131]–[133]. With the rise of deep learning, Ozcinar et al. [134] designed a generative adversarial network to super-resolve equirectangular projections. Along with the adversarial loss, they use a latitude related loss function to deal with the distortion caused by the projection. Recently, a significant number of end-to-end networks have been introduced to tackle omnidirectional image super-resolution. Deng et al. [135] proposed a latitude adaptive super-resolution network named LAU-Net+ for ERP images, where different upscaling factors can be used for different latitude bands. Furthermore, among the emerging methods, we note TCCL-Net [136], OSRT [137] and OPDN [138]. Finally, it is worth highlighting that prevailing methodologies in super-resolution mainly center around the equirectangular projection.

5.5.2 Image denoising

Although omnidirectional images have gained a lot of attention in the past years, only a few works have been dedicated to denoising them. Bigot et al. [139] adapt the Wiener filter and Tikhonov regularization to spherical images for the denoising application. Demonceaux et al. [140] denoise catadioptric inputs corrupted by White Gaussian noise by proposing a new neighborhood for Markov random fields (MRF) adapted for omnidirectional images. Furthermore, Alibouch et al. [141] adapt the Stein block thresholding method to omnidirectional images to remove additive white Gaussian noise. Finally, Phan et al. [142] solve poisson denoising by using a space-variant total-variation regularization on catadioptric images. They adapt the fidelity and the regularization terms with weighted functions based on the mean curvature of an image surface.

Chapter 6

SphereDRUNet: A spherical denoiser for omnidirectional images

6.1 Introduction

From wavelet thresholding to advanced deep-learning architectures, image denoising is a classical yet still one of the most widely explored topics in image processing. Despite the massive work that has been done to denoise traditional 2D images, very little effort has been dedicated to omnidirectional image denoising.

Most of the existing methods use conventional techniques and are primarily tailored to address the challenges presented by catadioptric inputs, thus they depend on the nature of the acquisition.

We dedicate this chapter to omnidirectional image denoising, as denoisers play an important role in today's era of regularizing inverse problems. Furthermore, a commonly adopted approach for processing spherical images involves applying two-dimensional processing techniques to their corresponding mappings, typically using the equirectangular projection. We propose an approach that is not sensitive to sampling distortion, by working directly on the sphere, and we aim to show that image denoising gives better performance when it is performed directly on the sphere rather than via a mapping, even though it raises many challenges.

Moreover, we introduce a novel spherical CNN denoiser (SphereDRUNet) by transforming the architecture of the DRUNet [2] using the elementary convolutional modules proposed by OSLO [129]. Our network takes directly spherical inputs rather than equirectangular projections. We evaluate the performance of SphereDRUNet in comparison with the initial DRUNet retrained and applied on equirectangular data. Since AWGN is a noise per pixel degradation, one can think that denoising the ERP could yield satisfac-

tory results. However, our results show that working on the sphere leads to a significant performance gain for the task of image denoising. This also confirms that the OSLO solution can be successfully extended to inverse problems such as denoising. We further compare our SphereDRUNet to a graph-based convolution network and confirm that the convolution approach that we use is more efficient.

6.2 Related work

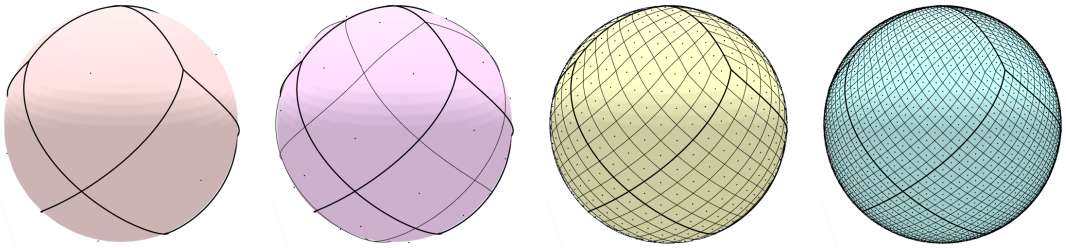
As previously covered in Chapter 5, traditional CNN operations such as convolution and translation can not be straightforwardly extended to the spherical domain. Certain requirements should be respected while constructing specific spherical CNN architectures: the convolution filters should be rotation equivariant, highly expressive and computationally efficient. Although different approaches for spherical convolutions have been proposed such as spectral learning, 2D processing and graph-based representations, it hasn't been possible to assure these 3 key properties simultaneously to guarantee a proper spherical convolution.

To overcome the drawbacks of prevailing methods, Bidgoli et al. [129] have proposed a novel framework for spherical processing (OSLO) where all aforementioned properties are successfully satisfied. For this purpose, they sample the sphere using HEALPix [130] and define the convolution in the pixel domain. We choose to build our work using OSLO because of its efficient ability to conduct spherical convolution. Next, we elaborate on HEALPix and OSLO.

6.2.1 HEALPix sampling

Hierarchical Equal Area isoLatitude Pixelation (HEALPix) [130] is a sampling method for spherical data that produces a configuration where the pixels are arranged in a diamond-shaped pattern. HEALPix starts by forming the base resolution and divides the surface of the sphere into 12 equal-area regions, each representing a specific area on the sphere. Then, each of these regions is iteratively partitioned into 2x2 equal-area sub-pixels, until reaching the target resolution. The HEALPix k^{th} resolution is defined by a parameter N_{side}=2^k and has a total of N_{pix}=$12N_{side}^2$ = 12 x 2^k x 2^k pixels (Figure 6.1). Each pixel in the final arrangement has eight adjacent neighbors (only 24 pixels have seven neighbors, which is far from having a significant effect taking into account the resolution

for an omnidirectional image). This ensures the regularity of the neighborhood, which is a fundamental condition for an expressive and effective convolution. Furthermore, the relative distance and orientation between neighboring pixels remains consistent throughout the entire sphere. Thus, HEALPix offers a rigid structure that was further proved in [129]. Both the regularity and the rigidity of the sampling method are essential properties to ensure an expressive and effective convolution.



(a) N_{side}=1 (b) N_{side}=2 (c) N_{side}=8 (d) N_{side}=16

Figure 6.1 – Visualization of HEALPix sampling over the Sphere. (a) Base resolution. (b) First resolution. (c) Third resolution. (d) Fourth resolution

6.2.2 On-the-Sphere Learning for Omnidirectional Images (OSLO)

On-the-sphere Learning for Omnidirectional Images (OSLO) proposes a new framework that defines all necessary operators to build a CNN on the sphere. The method samples the sphere using HEALPix and performs convolutions in the pixel domain. Pooling, stride, skip-connections and patching are defined as well. The advantage of OSLO is that the convolutions are rotation equivariant, expressive and efficient at the same time.

In fact, let $\mathcal{N}_i(k)$ be the index assigned to the k^{th} neighbor for node i ($k = 1, ..., 8$ for a total of 8 adjacent neighbors of a vertex). Let L_{in} and L_{out} denote respectively the number of input and output features in the convolution. The convolution operation for an output feature l ($1 \leq l \leq L_{out}$) is defined as:

$$x_i^l = \langle \theta_0, x_i \rangle + \sum_{k=1}^{8} \langle \theta_k, x_{\mathcal{N}_i(k)} \rangle \cdot w_{\mathcal{N}_i(k),i} \tag{6.1}$$

where θ_k represents the learnable filter weight at k and x_i is the input data/feature at point i. $w_{\mathcal{N}_i(k),i}$ is set to 0 when the neighbor $\mathcal{N}_i(k)$ is missing and to 1 otherwise, in order to deal with the 24 exceptions of the pixels that have a missing adjacent neighbor.

The described convolution is anisotropic and consistent all over the sphere. The same

weights are applied to compute the convolution output regardless of the position of the kernel on the sphere (Figure 6.2). Yet, it only supports 1-hop neighborhood. In order to increase the potential size of kernels and extend the local support to n-hop neighborhood, the authors propose an iterative computation of the 1-hop convolution with a proper aggregation method such as concatenation, addition or max aggregation. This makes the convolution highly expressive. Furthermore, the convolution being in the pixel domain, it simply consists in translating the filter over the sphere, hence the complexity linearly increases with the number of pixels.

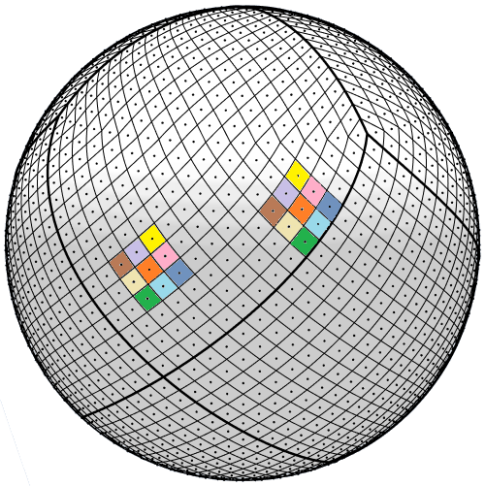


Figure 6.2 – Spherical convolution in the pixel domain proposed by [129]. Each color represents a weight corresponding to a certain orientation. Filter weights remain the same, regardless of the position of the kernel on the sphere.

Moreover, the authors use this framework for the task of compression and confirm that this on-the-sphere solution outperforms similarly learned models applied to equirectangular images.

For these reasons, we choose OSLO to work on the sphere for omnidirectional image denosing. Further information about the other operations can be found in [129].

6.3 Spherical image denoising

Our main objective is to explore whether image denoising is more efficient when performed directly on the sphere, compared to using a projection. We choose the DRUNet[2] CNN denoiser which is a state-of-the-art Gaussian denoiser for perspective images, and use OSLO to transfer its architecture to the sphere. Hence, a novel spherical denoiser

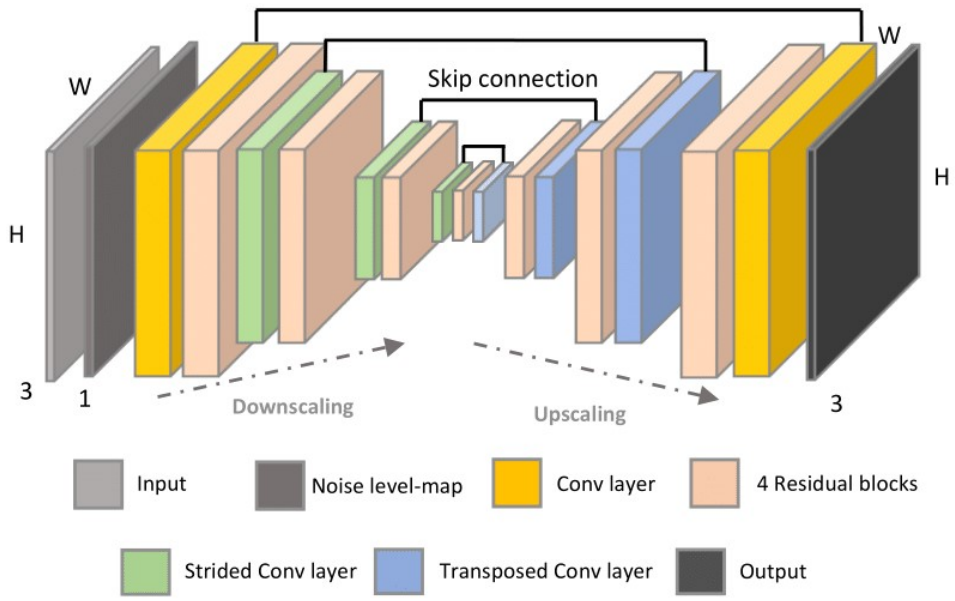


Figure 6.3 – Architecture of the DRUNet network [2]. The network takes as input the noisy image concatenated with a noise level map.

SphereDRUNet is obtained that can be used to denoise spherical images corrupted with any noise level.

The network architecture of DRUNet is shown in Figure 6.3. DRUNet is a bias-free model, which has been observed to be effective for generalization of denoisers over unseen noise levels. The model takes as input a noise level map and integrates Residual blocks (ResNet [96]) into U-Net [95]. It consists of four scales having 64, 128, 256 and 512 number of channels in each of the layers. Each scale in the downscaling consists of 4 residual blocks followed by a 2x2 strided convolution (SConv) whereas a 2x2 transposed convolution (TConv) is followed by 4 residual blocks during the upscaling. Four skip connections are set between strided convolution and transposed convolution blocks at each of the four scales.

We transfer DRUNet to the sphere using the OSLO framework. The architecture of the resulting SphereDRUNet is shown in Figure 6.4. We use 1-hop neighborhood convolution for all the layers because all convolutions in DRUNet are of size 3x3. 2D stride operations of size 2x2 amount to stride 4 on the sphere. For implementation purposes, we replace the 2x2 transpose convolution by the sub-pixel convolution of size 4 (pixel-shuffle) since both unpooling methods have equivalent performance.

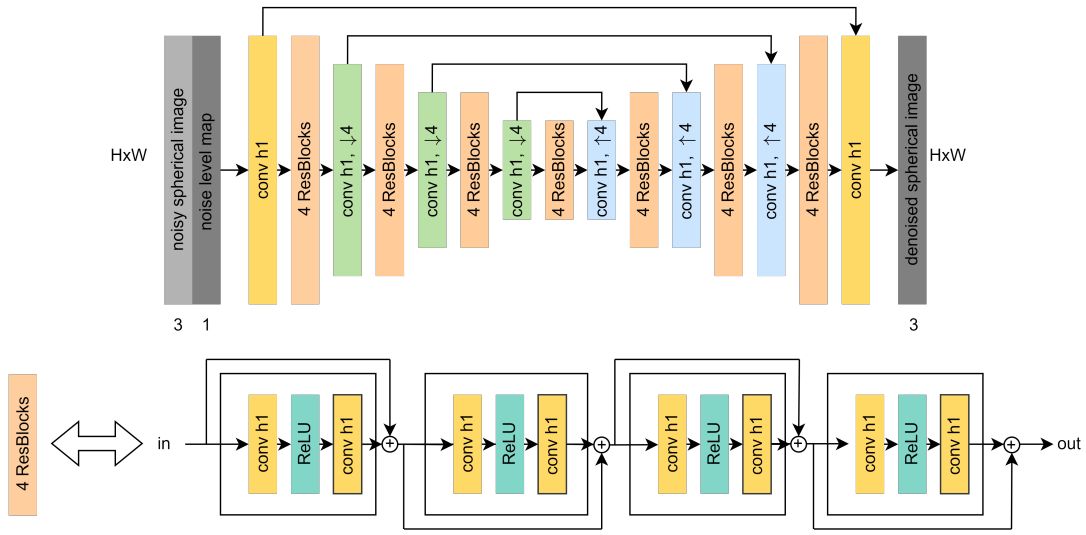


Figure 6.4 – Architecture of the SphereDRUNet network with OSLO-based convolution.

6.3.1 Training details

For the purpose of a fair comparison between denoising a spherical image and its projection, we adapt our training framework to that of the DRUNet. We use the SUN360 equirectangular image dataset [143] and consider 2105 images for training, 22 images for validation and 43 images for testing. All images in the initial dataset have a spatial size of 9104x4552. For each epoch of the training, we degrade the training ERP images by AWGN with a random noise level σ chosen from [0,50], and concatenate a uniform map filled with σ having the same size as the ERP image. We perform HEALPix sampling of resolution 10 (i.e. 12,582,912 pixels) of the original image, the noisy image and the concatenation. SphereDRUNet takes as input the concatenated data in the spherical form, and learns to denoise the image by minimizing the $\mathcal{L}_1$ loss between the estimated clean image and the ground-truth image using the ADAM optimization. The learning rate is set to 5e-5 and decreased by half every 50,000 iterations. The gradient update is performed once every 8 patches, patches being of resolution 8 (2^8 x 2^8 pixels). Finally, we train our network for a total of 25 epochs. Note that most of the modifications to the training framework compared to that of the DRUNet are essentially driven by the higher resolution of spherical images in contrast to perspective images.

Furthermore, since our goal is to study denoising directly on the sphere compared to denoising the equirectangular image, applying the DRUNet trained on perspective images to ERP projections might not seem fair. Therefore, we re-train the DRUNet on ERP

images in the same way as we trained the SphereDRUNet. In order to reduce the unfair bias caused by the presence of ground-truth images in ERP format, we resize the ERP images to 5056x2528 (12,781,568 pixels) so that both ERP and HEALPix have almost the same number of pixels.

6.3.2 Comparison framework

As discussed earlier, we proposed SphereDRUNet, a novel spherical denoising CNN and we re-trained the DRUNet with ERP images in order to apply it on spherical projections. We want to compare the performance of these two approaches in order to study whether spherical denoising is more efficient on the sphere or on a corresponding projection. We also include a comparison with a cubemap projection: we map the equirectangular image to the six faces of a cube and each face is denoised with the DRUNet. We then reconstruct the ERP image by combining the six planes.

To avoid a biased comparison, for testing both spherical and mapping-based approaches, we first corrupt the high-resolution equirectangular input with a Gaussian noise. Then, we downsample the corrupted image to get a HEALPix sampling of resolution 10 (12,582,912 pixels with $N_{side} = 2^{10}$) and resize the ERP image to 5056x2528 (12,781,568 pixels) in order to match the resolution of the HEALPix sampling. Thus, both test images are generated from the same corrupted input.

For quantitative evaluation, we calculate the Spherical PSNR (S-PSNR) [dB] [144] and the Weighted to Spherically uniform PSNR (WS-PSNR) [dB] [145]. The former maps the output and the ground-truth images (ERP or sphere) to a sphere by uniformly sampling 655,362 points, and computes the mean error between them to simulate the PSNR on the sphere. The sampling process for the S-PSNR calculation is different than the HEALPix sampling, hence the metric is not biased. S-PSNR is seen to be an estimation of the overall quality experienced by viewers across all potential views. On the other hand, in the calculation of the WS-PSNR, the mean squared error is weighted by the size of each pixel, as follows:

$$\text{WS-PSNR} = 10 \cdot \log_{10}(\frac{\text{MAX}_I^2}{\text{W-MSE}}), \tag{6.2}$$

where MAX_I is the maximum pixel intensity in the image, and W-MSE is given as:

$$\text{W-MSE} = \frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}((\hat{I}(x,y) - I^{gt}(x,y))^2.w(y)}{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}w(y)}. \tag{6.3}$$

W and H denote the width and height of the image, $\hat{I}$ and I^{gt} represent respectively the reconstructed and the ground-truth images and $w(y)$ represents the weight of the corresponding pixel in the weight matrix. We also show mollweide projections of the denoised images for some visual comparison.

6.4 Experimental results

We perform spherical denoising on images corrupted with an Additive White Gaussian noise (AWGN) of noise level $\sigma * 255 \in [10, 20, 30, 40, 50]$. Tables 6.1 and 6.2 show respectively the WS-PSNR [dB] and the S-PSNR [dB] of denoised equirectangular, cubemap and spherical images of the testing dataset. We can see that the on-the-sphere solution with our SphereDRUNet significantly outperforms denoising the equirectangular and the cubemap projections. This gain is even more important when the noise level is high. Figures 6.5, 6.6 and 6.7 show visual comparisons of the denoised ERP and spherical images for degradations of noise level $\sigma = 30/255$ and $\sigma = 50/255$. We observe that the DRUNet smoothes the equirectangular images and can not recover the high frequency details while the images denoised using SphereDRUNet on the sphere have less distortion and do not lose information (i.e. recover high frequency details such as texture). For instance, in Figure 6.5, we observe that the denoised equirectangular image has lost high frequency elements such as the details on the leaves or the stains on the train, whereas denoising the sphere successfully removes noise while also recovering details. This is probably because in order to recover an equirectangular image with spatial distortions, a network needs a larger amount of weights to reconstruct high frequency components compared to recovering the same high frequency element on the sphere. For instance, a line on the sphere remains a line regardless of its position, whereas it can take different forms on the ERP depending on where it lies, so a network needs more weights to recover the same high frequency components on the ERP compared to the sphere.

While the pixel distribution in the CMP is more uniform than that in the ERP, the mapping between the equirectangular projection and the cubemap projection induces a distortion and a significant shift in the pixels, especially because of the border discontinuities. We show this in Figure 6.8 where we map an ERP image into a CMP image (since the SUN360 dataset is in ERP format) and simply convert it back to ERP. A degraded image with an important shift is observed, with a WS-PSNR of 24.95 dB between the original ERP image and the re-converted projection. This further underlines the advantage

of processing on the sphere rather than via a mapping into 2D ERP or CMP images.

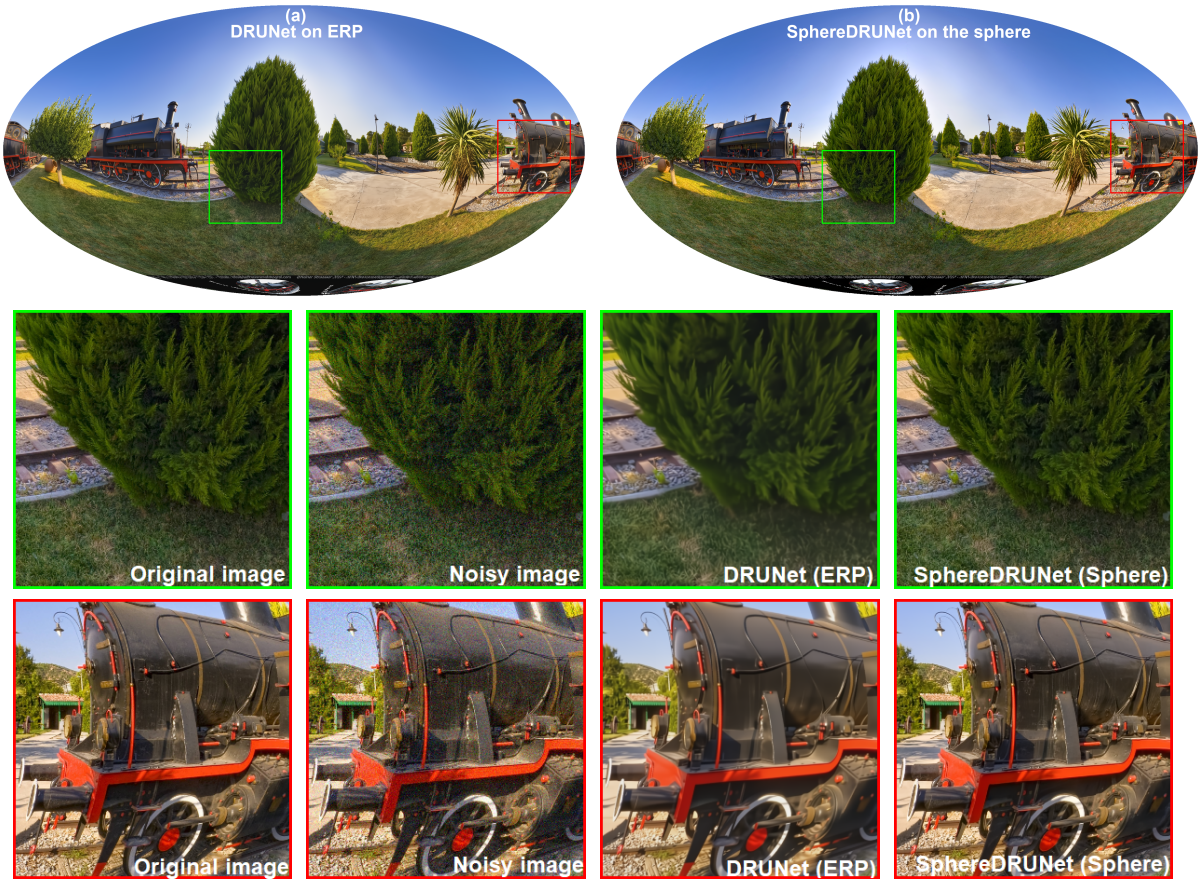
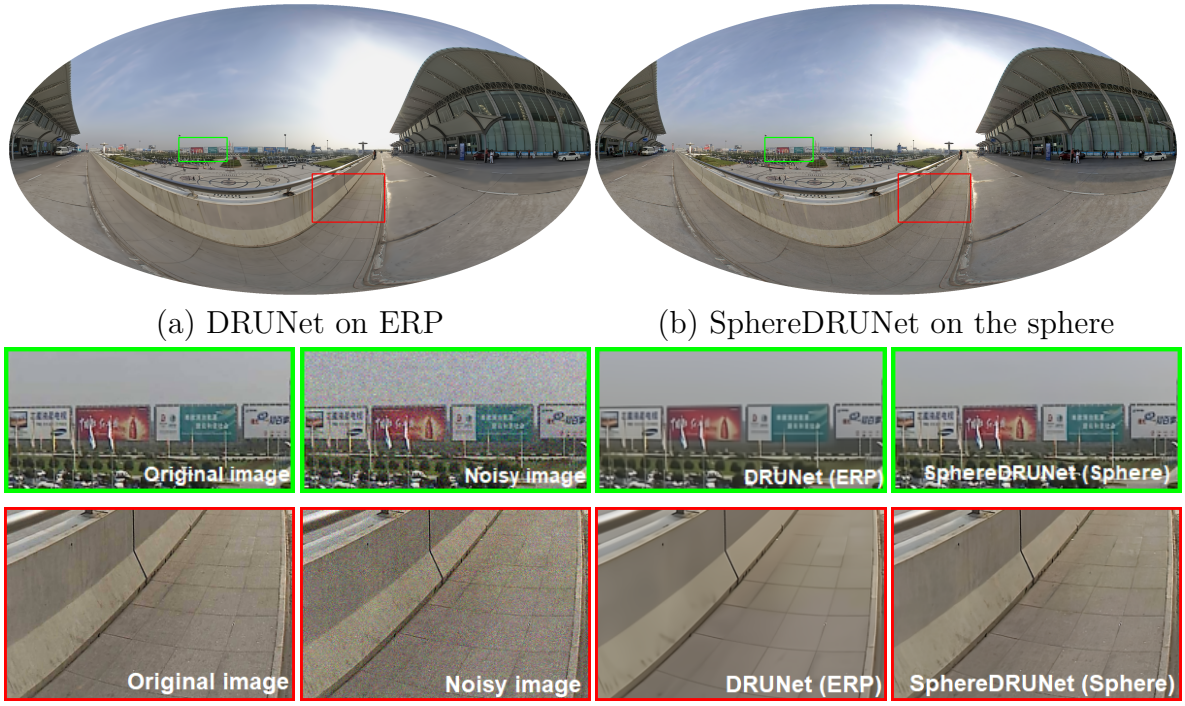


Figure 6.5 – Mollweide projection of the (a) equirectangular image denoised by the DRUNet that was re-trained on ERP images and the (b) spherical image denoised by our SphereDRUNet. AWGN of $\sigma = 30/255$ was added on the original image. The zoomed versions of the red and green rectangular regions are shown in the bottom rows.

Spherical convolution: OSLO-based vs. graph-based (DeepSphere)

Our hypothesis that omnidirectional image denoising is more efficient when performed on the sphere is verified. One popular approach to operate directly on irregular topology relies on graph signal processing. For the sake of completeness, we also want to prove that the OSLO-based spherical convolution that we have chosen is the best approach. For this purpose, we consider a graph-based baseline: the DeepSphere architecture [126]. DeepSphere is also defined on the HEALPix sampling, but uses max-pooling and graph-based convolutions approximated by Chebyshev polynomial formulation. We re-train the same

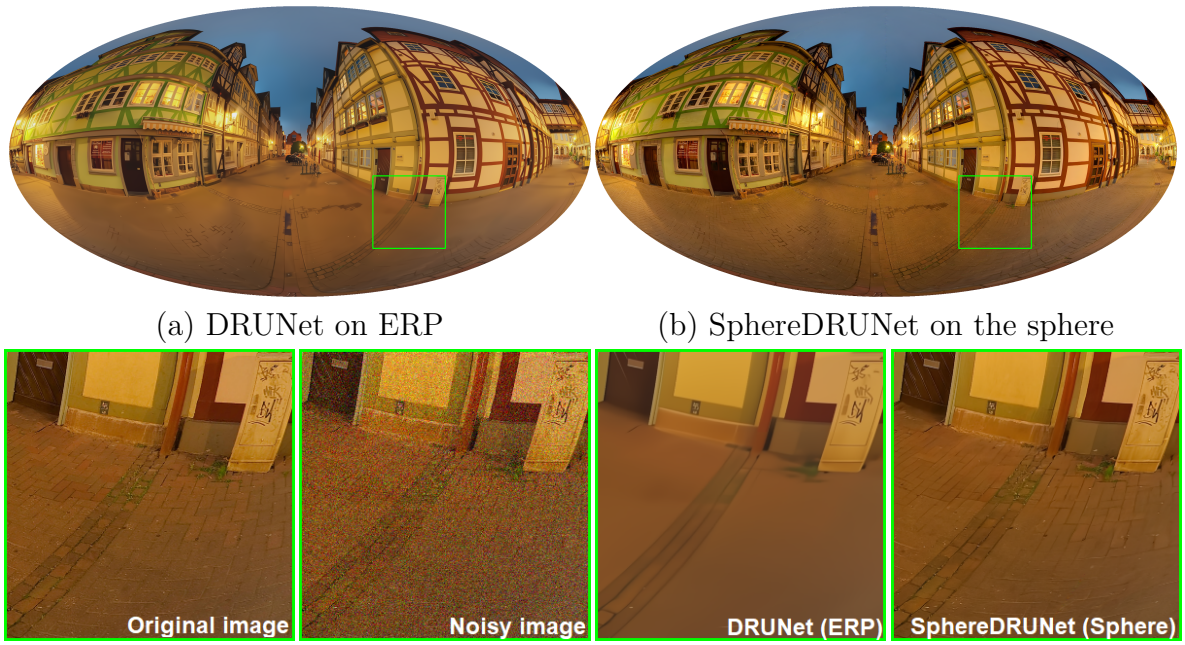


(a) DRUNet on ERP (b) SphereDRUNet on the sphere

Figure 6.6 – Mollweide projection of the (a) equirectangular image denoised by the DRUNet that was re-trained on ERP images and the (b) spherical image denoised by our SphereDRUNet. AWGN of $\sigma = 30/255$ was added on the original image. The zoomed versions of the red and green rectangular regions are shown in the bottom rows.

SphereDRUNet architecture with graph-based convolutions in the same training framework as our OSLO-based SphereDRUNet. For computational reasons, both networks are trained and evaluated on HEALPix images of resolution 8 (786,432 pixels) in this case.

Figure 6.9 shows denoising results of spherical images restored with the OSLO-based SphereDRUNet and the graph-based SphereDRUNet. We can see that the graph-based convolution approach fails to recover the clean image from the noisy measurement. This limitation arises from the inherent nature of the graph, where the lack of directional information restricts the learning process to a single weight per neighborhood. By contrast, our OSLO-based approach demonstrates superior performance, successfully recovering the clean image from the noisy measurement. This highlights the importance of anisotropic filters and the effectiveness of the directional information embedded in the OSLO-based convolution, leading to enhanced denoising capabilities.



(a) DRUNet on ERP (b) SphereDRUNet on the sphere

Figure 6.7 – Mollweide projection of the (a) equirectangular image denoised by the DRUNet that was re-trained on ERP images and the (b) spherical image denoised by our SphereDRUNet. AWGN of $\sigma = 50/255$ was added on the original image. The zoomed versions of the green rectangular region is shown in the bottom row.

Ablation study: training a denoiser on the equirectangular projection

Mapping-based solutions are a widely adopted approach when it comes to spherical image processing. The reason is simple: 2D processing tools can be easily applied to these projections. However, traditional 2D operations (such as CNN tools) can not effectively process omnidirectional images, because these mappings display a significant distortion. In this section, we further demonstrate the limits of mapping-based learning. Table 6.3

Table 6.1 – Quantitative results of denoising equirectangular, cubemap and spherical data in terms of WS-PSNR [dB].

	Noise level σ*255				
	10	20	30	40	50
Sphere + SphereDRUNet	**40.78**	**37.33**	**35.43**	**34.14**	**33.16**
ERP + DRUNet (trained on ERP)	39.85	35.38	32.46	30.15	28.16
CMP + DRUNet (trained on ERP)	30.30	29.65	28.73	27.61	26.38

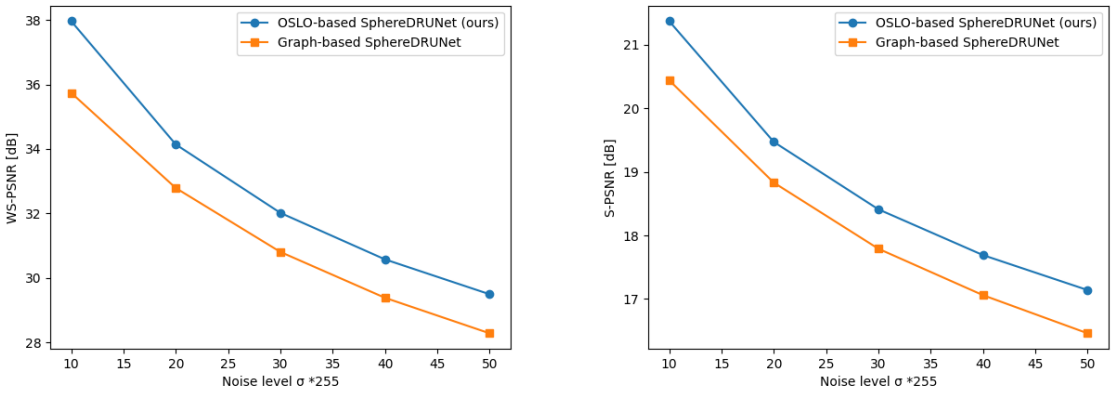
Table 6.2 – Quantitative results of denoising equirectangular, cubemap and spherical data in terms of S-PSNR [dB].

	Noise level σ*255				
	10	20	30	40	50
Sphere + SphereDRUNet	**22.28**	**20.47**	**19.49**	**18.83**	**18.32**
ERP + DRUNet (trained on ERP)	21.65	19.78	18.73	17.98	17.40
CMP + DRUNet (trained on ERP)	17.43	17.08	16.78	16.50	16.25

(a) Original projection (b) Converted projection

Figure 6.8 – Zoomed versions of (a) a portion of an equirectangular image and (b) it's corresponding re-conversion from cubemap projection (i.e. the ERP image is converted to CMP, and re-converted back directly to ERP). Distortion and shifting are produced by the simple conversion between the mappings.

shows the results (in terms of WS-PSNR [dB]) of denoising ERP images with the initial DRUNet network trained to denoise 2D images and the DRUNet that we re-trained on equirectangular projections. We observe that the performance of the DRUNet network does not greatly enhance when we re-train it over spherical projections compared to the initial DRUNet that was trained to denoise 2D images. For instance, the gain is only 0.09 dB for $\sigma = 50/255$, vs. a gain of 5.09 dB when we consider a processing on the sphere. This comparison further confirms that two-dimensional convolutions are not able to successfully learn spherical features. Due to their inherent flat nature, they cannot capture the curvatures and non-planar characteristics of spherical surfaces. Thus, 2D learning tools do not generalize well on 360 mappings in the task of denoising.



(a) (b)

Figure 6.9 – Denoising results of spherical images restored with the OSLO-based SphereDRUNet (proposed network) and graph-based SphereDRUNet. Results are in terms of (a) WS-PSNR and (b) S-PSNR [dB].

Table 6.3 – Quantitative results of denoising equirectangular images with the initial DRUNet network trained to denoise perspective 2D images and the DRUNet that we re-trained on ERP inputs. Results are in terms of WS-PSNR [dB].

	Noise level σ*255				
	10	20	30	40	50
ERP + DRUNet (trained on ERP)	**39.85**	**35.38**	**32.46**	**30.15**	**28.16**
ERP + DRUNet (trained on 2D images)	39.53	35.19	32.33	30.03	28.07
Gain	0.32	0.19	0.13	0.12	0.09

6.5 Conclusion

In this last chapter, we have addressed the task of omnidirectional image denoising. Our study investigates the effectiveness of performing denoising directly on the sphere, as opposed to employing a projection. To address this, we have proposed a novel spherical denoising CNN (SphereDRUNet), by transferring a state-of-the-art Gaussian denoiser to the sphere, using the HEALPix spherical sampling and the OSLO-based convolution. Our network has shown promising denoising results on the sphere for different noise levels.

Our research demonstrates that denoising the spherical image gives superior results compared to denoising a corresponding mapping such as the equirectangular or the cube-map projections. This can be explained by the fact that a network needs less weights to recover the same high frequency components on the sphere compared to a projection.

Ultimately, this work does not only advance the field of omnidirectional image denoising, but also opens doors for solving other inverse problems. In fact, our proposed spherical regularizer can be leveraged in a Plug-and-play algorithm in order to solve a variety of inverse problems.

Chapter 7

Conclusion

The landscape of inverse problems solutions has undergone different transformations over time, notably pivoting with the advent of deep learning. In particular, research in regularization techniques has evolved to embrace more effective approaches to address the ill-posed nature of inverse problems. In this thesis, we focused on contributing to the research in inverse problem regularization through deep learning-based methodologies.

We began by giving a detailed review on inverse problems in imaging in Chapter 2. We first introduced the problem of interest and illustrated different examples that we have addressed in this thesis. Then, we elaborated on existing approaches that have been developed to solve inverse problems, and concentrated on the literature of deep learning-based regularization methods. In particular, regularizing inverse problems by leveraging the power of denoisers in a Bayesian framework is the core of this thesis.

At the beginning of this thesis, the Deep Image Prior [1] was getting popular for solving inverse problems. The DIP, distinguished by its untrained generative network which only leverages its architecture along with the degraded image to tackle inverse problems, ignited our curiosity. We were motivated to explore the potential advantages of combining its strengths with those of a generic regularizer trained in a supervised manner on a large image dataset.

Chapter 3 presented our first contribution, focusing on regularizing the estimate generated by the Deep Image Prior [1] using the DRUNet [2] within an ADMM framework. Our method yielded improved results compared to DIP and other techniques employing hand-crafted regularizations with the DIP. The complexity of the parameter-tuning caused by the ADMM and the additional hyper-parameter of the denoiser inspired us to transfer the prior implicitly defined by a denoiser to a regularizing network, that can be used in a simple gradient descent algorithm.

Chapter 4 exposed our second contribution on a novel approach to train a regularizer that can be used in a Plug-and-play gradient descent algorithm. Based on the assumption that a denoiser represents the proximal operator of an underlying differentiable regular-

izer, we proved that there exists a relation between the denoiser and the gradient of the corresponding regularizer. We proposed to train a network modeling the gradient of a regularizer by using a pre-trained deep denoiser. We jointly trained both networks by also updating the denoiser, in order for it to be consistent with the definition of a proximal operator. We tested the efficiency of our approach by using our network in a Plug-and-play gradient descent algorithm, and observed that it outperforms state-of-the-art generic approaches. We also showed that our regularizing network also serves as a pre-trained network in unrolled gradient descent.

Chapter 5 marked the beginning of the second part of this thesis, which focused on omnidirectional images. We gave a short review on processing tools and challenges for omnidirectional images, and discussed the existing literature on inverse problems.

As we delved into this second part, we observed a lack of comprehensive research on omnidirectional image denoising. Given the significant role denoisers play in modern regularization strategies and the importance of image denoising in image processing, we devoted Chapter 6 to omnidirectional image denoising. Our contribution in this chapter was two-fold: we introduced a spherical denoiser network designed and trained specifically for omnidirectional images, which gave promising denoising performance for different noise levels. We also showed that image denoising gives better results when working directly on the sphere rather than when processing their underlying projections.

The contributions presented in this thesis allow several improvements and open perspectives for future work. For inverse problems in perspective imagery, we suggest:

- It would be interesting to compare the behaviour of the Deep Image Prior regularized by our regularizing network (ReG in chapter 4) in a gradient descent algorithm with the denoiser-based regularization that we propose in Chapter 3.
- One limitation of our regularizing network presented in Chapter 4 is, that we do not enforce the network to have a symmetric Jacobian matrix. Therefore, our training strategy does not guarantee that the network can be interpreted as a conservative vector field. One possible direction is to study whether such a constraint could improve the Plug-and-play gradient descent without sacrificing the computational complexity. This would be possible by using a network directly modeling the regularizer and explicitly computing its gradient with back-propagation.

For solving inverse problems in omnidirectional imagery, the following directions are possible:

- The spherical denoiser introduced in Chapter 6 is a first step towards an omnidirectional Plug-and-play on the sphere. Although this can be easily done for some applications such as pixel-wise inpainting, the tasks of super-resolution and deblurring require further implementations as they require to perform convolution on the sphere. In fact, the current implementation of convolution in OSLO supports 1-hop neighborhood, and uses iterative computation of the 1-hop convolution with a proper aggregation method in order to increase the size of kernels. This works well for training, as the weights of the filters are learned and not pre-determined. One initial step would involve re-implementing the convolution to access a pixel's secondary neighbors, enabling the use of filters with higher dimensions. Then, we could implement Plug-and-play ADMM on the sphere to solve different inverse problems using our SphereDRUNet.
- We also suggest to extend the work presented in Chapter 4 to the sphere. A network modeling the gradient of a regularizer can be trained using the spherical denoising network introduced in Chapter 6, and then be used in a Plug-and-play gradient descent algorithm on the sphere.

Appendix A

Parameter settings of Chapter 4

A.1 Plug-and-play ADMM: formulation and parameter settings

While the Plug-and-play ADMM can provide high quality image reconstruction, the parameter setting is not trivial and strongly influences the results. We detail in this section how we have parameterized the Plug-and-play ADMM method for our comparisons. First, let us remind the ADMM equations for solving linear inverse problems as detailed in 2.4.2.1:

$$\boldsymbol{x}^{\boldsymbol{k+1}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \rho^k \left\| \boldsymbol{x} - \left(\boldsymbol{z}^{\boldsymbol{k}} - \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k} \right) \right\|_2^2, \tag{A.1}$$

$$\boldsymbol{z}^{\boldsymbol{k+1}} = \mathcal{D}_s(\boldsymbol{x}^{\boldsymbol{k+1}} + \boldsymbol{l}^{\boldsymbol{k}}/\rho), \tag{A.2}$$

$$\boldsymbol{l}^{\boldsymbol{k+1}} = \boldsymbol{l}^{\boldsymbol{k}} + \rho(\boldsymbol{x}^{\boldsymbol{k+1}} - \boldsymbol{z}^{\boldsymbol{k+1}}), \tag{A.3}$$

where l is the dual variable (typically zero-initialized), ρ is the penalty parameter and $s = \sqrt{\sigma^2/\rho} = \sigma/\sqrt{\rho}$.

Note that σ is a parameter of the problem to be solved and should not depend on the algorithm. As discussed in 4.4.1.1, σ must be equal to the true noise level σ_n added in the degradation. However, in the noiseless scenario, using $\sigma = \sigma_n = 0$ removes the regularization term. Hence similarly to our method, we choose a very small non-zero value in this case. For our experiments, we have used $\sigma = \max(\sigma_n, 0.001/255)$.

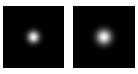
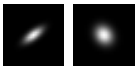
Note that the PnP-ADMM may thus parameterize the denoiser with a different standard deviation s than the noise level σ by setting the penalty parameter $\rho \neq 1$. In practice, the results of PnP-ADMM strongly depend on the setting of ρ. A common strategy in ADMM is to increase the value of ρ at each iteration with an update of the form $\rho^{k+1} = \rho^k \cdot \alpha$ for some fixed parameter $\alpha \geq 1$. Hence, this requires setting two parameters ρ_0 and α. A more intuitive parameterization typically used in the Plug-and-play context (e.g. [2], [146]) considers instead the denoising standard deviations s^0 and s^N respectively at the first and last iterations. Knowing that $s = \sigma/\sqrt{\rho^k}$ at each iteration,

ρ^0 and α can be set accordingly as:

$$\rho^0 = \left(\frac{\sigma}{s^0}\right)^2 \quad \text{and} \quad \alpha = \left(\frac{s^0}{s^N}\right)^{2/N} \tag{A.4}$$

Therefore, the main parameters that we need to set in PnP-ADMM are the number of iterations N and the initial and final noise standard deviation of the denoiser (i.e. s^0 and s^N). For our experiments, we have tested two configurations: the first configuration is similar to [2] and [146] with s varying between a sufficiently high value s^0 and the true noise standard deviation $s^N = \sigma_n$ (or a small value when $\sigma_n = 0$). In the second configuration, s is kept constant (i.e. $s^0 = s^N$). For both configurations, we have tuned the parameters to obtain the best results on the Set5 dataset, and we kept the best configuration. The parameters are given in Table A.1.

Table A.1 – Parameters used for the PnP-ADMM with the denoiser of [2]. σ_n: standard deviation of the AWGN added on the degraded image, s^0 noise standard deviation of the denoiser at the first iteration, s^N: noise standard deviation of the denoiser at the last iteration, N: number of iterations.

		$\sigma_n * 255$	$s^0 * 255$	$s^N * 255$	N
Super-Resolution x2	Bicubic	0	50	0.1	25
		$\sqrt{2}$	20	20	30
		2.55	30	30	30
	Gaussian	0	50	0.1	25
		$\sqrt{2}$	30	30	30
		2.55	45	45	30
Super-Resolution x3	Bicubic	0	50	0.1	40
		$\sqrt{2}$	60	60	30
		2.55	100	100	30
	Gaussian	0	50	0.1	25
		$\sqrt{2}$	30	30	30
		2.55	45	45	30
Deblurring		$\sqrt{2}$	25	25	20
		2.55	30	30	20
		7.65	35	35	20
		$\sqrt{2}$	25	25	30
		2.55	30	30	30
		7.65	35	35	30
Pixel-wise inpainting	0.1	0	255	1	118
	0.2	0	255	1	200

A.2 Parameter settings for the other methods

Table A.2 – Parameters used for the projection operator (One-Net) [105]. σ_n: standard deviation of the AWGN added on the degraded image, ρ: penalty parameter of the ADMM, n: number of iterations

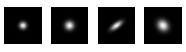
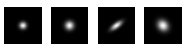
		$\sigma_n * 255$	ρ	n
Super-Resolution x2	Bicubic	0	0.05	1
		$\sqrt{2}$	0.06	12
		2.55	0.06	12
	Gaussian	0	0.02	1
		$\sqrt{2}$	0.06	10
		2.55	0.06	10
Super-Resolution x3	Bicubic	0	0.01	5
		$\sqrt{2}$	0.06	10
		2.55	0.05	12
	Gaussian	0	0.01	5
		$\sqrt{2}$	0.07	10
		2.55	0.07	10
Deblurring		$\sqrt{2}$	0.01	15
		2.55	0.02	15
		7.65	0.02	5
Pixel-wise inpainting	0.1	0	0.004	300
	0.2	0	0.01	250

Table A.3 – Parameters used for the RED [4] in a Gradient Descent framework using DRUNet. σ_n: standard deviation of the AWGN added on the degraded image, w: weight of the regularization, σ_f: noise standard deviation of the denoiser, μ: gradient step size.

		$\sigma_n * 255$	w	$\sigma_f * 255$	μ
Super-Resolution x2 (Bicubic and Gaussian)		0	0.005	7	0.08
		$\sqrt{2}$	0.03	7	0.08
		2.55	0.07	7	0.08
Super-Resolution x3 (Bicubic and Gaussian)		0	0.005	10	0.08
		$\sqrt{2}$	0.01	10	0.08
		2.55	0.03	10	0.08
Deblurring		$\sqrt{2}$	0.01	10	0.01
		2.55	0.03	16	0.01
		7.65	0.03	16	0.01
		$\sqrt{2}$	0.01	16	0.01
		2.55	0.01	16	0.01
		7.65	0.03	16	0.01
Pixel-wise inpainting	0.1	0	0.001	27	0.006
	0.2	0	0.001	24	0.008

BIBLIOGRAPHY

[1] D. Ulyanov, A. Vedaldi, and V. Lempitsky, « Deep image prior », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9446–9454. DOI: 10.1109/CVPR.2018.00984.

[2] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, « Plug-and-play image restoration with deep denoiser prior », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, *10*, pp. 6360–6376, 2021.

[3] S. Hurault, A. Leclaire, and N. Papadakis, « Gradient step denoiser for convergent plug-and-play », *in International Conference on Learning Representations (ICLR'22)*, 2022.

[4] Y. Romano, M. Elad, and P. Milanfar, « The little engine that could: regularization by denoising (red) », *SIAM Journal on Imaging Sciences*, vol. 10, *4*, pp. 1804–1844, 2017. DOI: 10.1137/16M1102884.

[5] B. Guo, Y. Han, and J. Wen, « Agem: solving linear inverse problems via deep priors and sampling », *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[6] Y. Song and S. Ermon, « Generative modeling by estimating gradients of the data distribution », *Advances in neural information processing systems*, vol. 32, 2019.

[7] Z. Kadkhodaie and E. P. Simoncelli, « Solving linear inverse problems using the prior implicit in a denoiser », *arXiv preprint arXiv:2007.13640*, 2020. DOI: 10.48550/ARXIV.2007.13640.

[8] B. Kawar, G. Vaksman, and M. Elad, « Stochastic image denoising by sampling from the posterior distribution », *in Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1866–1875.

[9] J. P. Snyder, *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

[10] L. I. Rudin, S. Osher, and E. Fatemi, « Nonlinear total variation based noise removal algorithms », *Physica D: nonlinear phenomena*, vol. 60, *1-4*, pp. 259–268, 1992.

[11] A. Beck and M. Teboulle, « Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems », *IEEE transactions on image processing*, vol. 18, *11*, pp. 2419–2434, 2009.

[12] B. Shi, L. Huang, and Z.-F. Pang, « Fast algorithm for multiplicative noise removal », *Journal of Visual Communication and Image Representation*, vol. 23, *1*, pp. 126–133, 2012.

[13] X. Liu and T. Sun, « Hybrid non-convex regularizers model for removing multiplicative noise », *Computers & Mathematics with Applications*, vol. 126, pp. 182–195, 2022.

[14] M. Lee and J. Park, « An optimized dynamic mode decomposition model robust to multiplicative noise », *SIAM Journal on Applied Dynamical Systems*, vol. 22, *1*, pp. 235–268, 2023.

[15] S. Baraha and A. K. Sahoo, « Restoration of speckle noise corrupted sar images using regularization by denoising », *Journal of Visual Communication and Image Representation*, vol. 86, p. 103 546, 2022.

[16] A. K. Shukla, S. K. Dwivedi, G. Chandra, and R. Shree, « Deep learning-based suppression of speckle-noise in synthetic aperture radar (sar) images: a comprehensive review », *in Proceedings of the International Conference on Cognitive and Intelligent Computing: ICCIC 2021, Volume 2*, Springer, 2023, pp. 693–705.

[17] W. Cheng and K. Hirakawa, « Minimum risk wavelet shrinkage operator for poisson image denoising », *IEEE Transactions on Image Processing*, vol. 24, *5*, pp. 1660–1671, 2015.

[18] W. Kumwilaisak, T. Piriyatharawet, P. Lasang, and N. Thatphithakkul, « Image denoising with deep convolutional neural and multi-directional long short-term memory networks under poisson noise environments », *IEEE Access*, vol. 8, pp. 86 998–87 010, 2020.

[19] M. Rahman Chowdhury, J. Zhang, J. Qin, and Y. Lou, « Poisson image denoising based on fractional-order total variation », *Inverse Problems & Imaging*, vol. 14, *1*, 2020.

[20] V. Göreke, « A novel method based on wiener filter for denoising poisson noise from medical x-ray images », *Biomedical Signal Processing and Control*, vol. 79, p. 104 031, 2023.

[21] B. Hughes, « On the error probability of signals in additive white gaussian noise », *IEEE Transactions on Information Theory*, vol. 37, *1*, pp. 151–155, 1991.

[22] I. W. Selesnick, « The estimation of laplace random vectors in additive white gaussian noise », *IEEE Transactions on Signal Processing*, vol. 56, *8*, pp. 3482–3496, 2008.

[23] F. Luisier, T. Blu, and M. Unser, « Image denoising in mixed poisson–gaussian noise », *IEEE Transactions on image processing*, vol. 20, *3*, pp. 696–708, 2010.

[24] A. Jezierska, C. Chaux, J.-C. Pesquet, and H. Talbot, « An em approach for poisson-gaussian noise modeling », *in 2011 19th European Signal Processing Conference*, IEEE, 2011, pp. 2244–2248.

[25] N. Bähler, M. El Helou, É. Objois, K. Okumuş, and S. Süsstrunk, « Pogain: poisson-gaussian image noise modeling from paired samples », *IEEE Signal Processing Letters*, vol. 29, pp. 2602–2606, 2022.

[26] V. Mannam, Y. Zhang, Y. Zhu, *et al.*, « Real-time image denoising of mixed poisson–gaussian noise in fluorescence microscopy images using imagej », *Optica*, vol. 9, *4*, pp. 335–345, 2022.

[27] R. Zeyde, M. Elad, and M. Protter, « On single image scale-up using sparse-representations », vol. 6920, Jun. 2010, pp. 711–730, ISBN: 978-3-642-27412-1. DOI: 10.1007/978-3-642-27413-8_47.

[28] C. Bouman and K. Sauer, « A generalized gaussian image model for edge-preserving map estimation », *IEEE Transactions on image processing*, vol. 2, *3*, pp. 296–310, 1993.

[29] F. J. Anscombe, « The transformation of poisson, binomial and negative-binomial data », *Biometrika*, vol. 35, *3/4*, pp. 246–254, 1948.

[30] J. Hadamard, « Sur les problèmes aux dérivés partielles et leur signification physique », *Princeton University Bulletin*, vol. 13, pp. 49–52, 1902.

[31] A. N. Tikhonov, « On the regularization of ill-posed problems », *in Doklady Akademii Nauk*, Russian Academy of Sciences, vol. 153, 1963, pp. 49–52.

[32] P. Saint-Marc, J.-S. Chen, and G. Medioni, « Adaptive smoothing: a general tool for early vision », *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, *06*, pp. 514–529, 1991.

[33] C. Chu, I. Glad, F. Godtliebsen, and J. Marron, « Edge-preserving smoothers for image processing », *Journal of the American Statistical Association*, vol. 93, *442*, pp. 526–541, 1998.

[34] H. K. Aggarwal and A. Majumdar, « Hyperspectral image denoising using spatio-spectral total variation », *IEEE Geoscience and Remote Sensing Letters*, vol. 13, *3*, pp. 442–446, 2016.

[35] D. L. Donoho and I. M. Johnstone, « Ideal spatial adaptation by wavelet shrinkage », *biometrika*, vol. 81, *3*, pp. 425–455, 1994.

[36] M. Elad and M. Aharon, « Image denoising via sparse and redundant representations over learned dictionaries », *IEEE Transactions on Image processing*, vol. 15, *12*, pp. 3736–3745, 2006.

[37] F.-X. Dupé, J. M. Fadili, and J.-L. Starck, « A proximal iteration for deconvolving poisson noisy images using sparse representations », *IEEE Transactions on Image Processing*, vol. 18, *2*, pp. 310–321, 2009.

[38] W. Dong, X. Li, L. Zhang, and G. Shi, « Sparsity-based image denoising via dictionary learning and structural clustering », *in CVPR 2011*, IEEE, 2011, pp. 457–464.

[39] G. Yu, G. Sapiro, and S. Mallat, « Solving inverse problems with piecewise linear estimators: from gaussian mixture models to structured sparsity », *IEEE Transactions on Image Processing*, vol. 21, *5*, pp. 2481–2499, 2011.

[40] K. Egiazarian and V. Katkovnik, « Single image super-resolution via bm3d sparse coding », *in 2015 23rd European signal processing conference (EUSIPCO)*, IEEE, 2015, pp. 2849–2853.

[41] S. Gu, L. Zhang, W. Zuo, and X. Feng, « Weighted nuclear norm minimization with application to image denoising », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2862–2869.

[42] N. Yair and T. Michaeli, « Multi-scale weighted nuclear norm image restoration », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3165–3174.

[43] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, « Beyond a gaussian denoiser: residual learning of deep cnn for image denoising », *IEEE transactions on image processing*, vol. 26, 7, pp. 3142–3155, 2017.

[44] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, « Non-local recurrent network for image restoration », *Advances in neural information processing systems*, vol. 31, 2018.

[45] X. Jia, S. Liu, X. Feng, and L. Zhang, « Focnet: a fractional optimal control network for image denoising », *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6054–6063.

[46] C. Dong, C. C. Loy, K. He, and X. Tang, « Learning a deep convolutional network for image super-resolution », *in European conf. on computer vision*, Springer, 2014, pp. 184–199. DOI: 10.1007/978-3-319-10593-2_13.

[47] C. Ledig, L. Theis, F. Huszar, *et al.*, « Photo-realistic single image super-resolution using a generative adversarial network », *in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4681–4690. DOI: 10.1109/cvpr.2017.19.

[48] X. Wang, K. Yu, S. Wu, *et al.*, « Esrgan: enhanced super-resolution generative adversarial networks », *in Proceedings of the European conference on computer vision (ECCV) workshops*, 2018.

[49] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, « Image super-resolution using very deep residual channel attention networks », *in Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 286–301.

[50] L. Xu, J. S. Ren, C. Liu, and J. Jia, « Deep convolutional neural network for image deconvolution », *Advances in neural information processing systems*, vol. 27, pp. 1790–1798, 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/1c1d4df596d01da60385f0bb17a4a9e0-Paper.pdf.

[51] E. T. Jaynes, *Probability theory: The logic of science*. Cambridge university press, 2003.

[52] D. P. Bertsekas, W. Hager, and O. Mangasarian, « Nonlinear programming. athena scientific belmont », *Massachusets, USA*, 1999.

[53] J. J. Moreau, « Fonctions convexes duales et points proximaux dans un espace hilbertien », *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, vol. 255, pp. 2897–2899, 1962.

[54] D. Geman and C. Yang, « Nonlinear image recovery with half-quadratic regularization », *IEEE transactions on Image Processing*, vol. 4, *7*, pp. 932–946, 1995.

[55] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, « Plug-and-play priors for model based reconstruction », *in 2013 IEEE Global Conference on Signal and Information Processing*, 2013, pp. 945–948. DOI: 10.1109/GlobalSIP.2013.6737048.

[56] S. H. Chan, X. Wang, and O. A. Elgendy, « Plug-and-play admm for image restoration: fixed-point convergence and applications », *IEEE Transactions on Computational Imaging*, vol. 3, *1*, pp. 84–98, 2016.

[57] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, *et al.*, « Plug-and-play priors for bright field electron tomography and sparse interpolation », *IEEE Transactions on Computational Imaging*, vol. 2, *4*, pp. 408–423, 2016.

[58] M. Terris, A. Repetti, J.-C. Pesquet, and Y. Wiaux, « Building firmly nonexpansive convolutional neural networks », *in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 8658–8662.

[59] E. T. Reehorst and P. Schniter, « Regularization by denoising: clarifications and new interpretations », *IEEE transactions on computational imaging*, vol. 5, *1*, pp. 52–67, 2018. DOI: 10.1109/TCI.2018.2880326.

[60] R. G. Gavaskar and K. N. Chaudhury, « Plug-and-play ista converges with kernel denoisers », *IEEE Signal Processing Letters*, vol. 27, pp. 610–614, 2020.

[61] Y. Sun, B. Wohlberg, and U. S. Kamilov, « An online plug-and-play algorithm for regularized image reconstruction », *IEEE Transactions on Computational Imaging*, vol. 5, *3*, pp. 395–408, 2019.

[62] Y. Sun, Z. Wu, X. Xu, B. Wohlberg, and U. S. Kamilov, « Scalable plug-and-play admm with convergence guarantees », *IEEE Transactions on Computational Imaging*, vol. 7, pp. 849–863, 2021.

[63] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, « Plug-and-play methods provably converge with properly trained denoisers », *in International Conference on Machine Learning*, PMLR, 2019, pp. 5546–5557.

[64] R. Laumont, V. De Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, « On maximum a posteriori estimation with plug & play priors and stochastic gradient descent », *Journal of Mathematical Imaging and Vision*, vol. 65, *1*, pp. 140–163, 2023.

[65] U. S. Kamilov, C. A. Bouman, G. T. Buzzard, and B. Wohlberg, « Plug-and-play methods for integrating physical and learned models in computational imaging: theory, algorithms, and applications », *IEEE Signal Processing Magazine*, vol. 40, *1*, pp. 85–97, 2023.

[66] S. Mukherjee, A. Hauptmann, O. Öktem, M. Pereyra, and C.-B. Schönlieb, « Learned reconstruction methods with convergence guarantees: a survey of concepts and applications », *IEEE Signal Processing Magazine*, vol. 40, *1*, pp. 164–182, 2023.

[67] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, « Image denoising by sparse 3-d transform-domain collaborative filtering », *IEEE Transactions on image processing*, vol. 16, *8*, pp. 2080–2095, 2007. DOI: 10.1109/TIP.2007.901238.

[68] A. Buades, B. Coll, and J.-M. Morel, « A non-local algorithm for image denoising », *in 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Ieee, vol. 2, 2005, pp. 60–65.

[69] R. Cohen, M. Elad, and P. Milanfar, « Regularization by denoising via fixed-point projection (red-pro) », *SIAM Journal on Imaging Sciences*, vol. 14, *3*, pp. 1374–1406, 2021.

[70] J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux, « Learning maximally monotone operators for image recovery », *SIAM Journal on Imaging Sciences*, vol. 14, *3*, pp. 1206–1237, 2021.

[71] T. STATISTICS and H. ROBBINS, « An empirical bayes approach », *in Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, vol. 1, 1956, p. 157.

[72] B. Efron, « Tweedie's formula and selection bias », *Journal of the American Statistical Association*, vol. 106, *496*, pp. 1602–1614, 2011.

[73] C. M. Stein, « Estimation of the mean of a multivariate normal distribution », *The annals of Statistics*, pp. 1135–1151, 1981.

[74] K. Miyasawa *et al.*, « An empirical bayes estimator of the mean of a normal population », *Bull. Inst. Internat. Statist*, vol. 38, *181-188*, pp. 1–2, 1961.

[75] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, « Generative adversarial nets », *Advances in neural information processing systems*, vol. 27, 2014.

[76] D. P. Kingma and M. Welling, « Auto-encoding variational bayes », *arXiv preprint arXiv:1312.6114*, 2013.

[77] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, « Compressed sensing using generative models », *in International conference on machine learning*, PMLR, 2017, pp. 537–546.

[78] V. Shah and C. Hegde, « Solving linear inverse problems using gan priors: an algorithm with provable guarantees », *in 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, pp. 4609–4613.

[79] M. González, A. Almansa, and P. Tan, « Solving inverse problems by joint posterior maximization with autoencoding prior », *SIAM Journal on Imaging Sciences*, vol. 15, *2*, pp. 822–859, 2022.

[80] D. Im Im, S. Ahn, R. Memisevic, and Y. Bengio, « Denoising criterion for variational auto-encoding framework », *in Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[81] J. Prost, A. Houdard, A. Almansa, and N. Papadakis, « Inverse problem regularization with hierarchical variational autoencoders », *arXiv preprint arXiv:2303.11217*, 2023.

[82] K. Gregor and Y. LeCun, « Learning fast approximations of sparse coding », *in Proceedings of the 27th international conference on international conference on machine learning*, 2010, pp. 399–406. [Online]. Available: `https://dl.acm.org/doi/10.5555/3104322.3104374`.

[83] J. Sun, H. Li, Z. Xu, *et al.*, « Deep admm-net for compressive sensing mri », *Advances in neural information processing systems*, vol. 29, 2016. [Online]. Available: `https://proceedings.neurips.cc/paper/2016/file/1679091c5a880faf6fb5e6087eb1b2dc-Paper.pdf`.

[84] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein, « Unrolled optimization with deep priors », *arXiv preprint arXiv:1705.08041*, 2017. DOI: `10.48550/ARXIV.1705.08041`.

[85] H. K. Aggarwal, M. P. Mani, and M. Jacob, « Modl: model-based deep learning architecture for inverse problems », *IEEE transactions on medical imaging*, vol. 38, *2*, pp. 394–405, 2018.

[86] C. Yang, R. Liu, L. Ma, X. Fan, H. Li, and M. Zhang, « Unrolled optimization with deep priors for intrinsic image decomposition », *in 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*, IEEE, 2018, pp. 1–7. DOI: 10.1109/BigMM.2018.8499478.

[87] D. Gilton, G. Ongie, and R. Willett, « Neumann networks for linear inverse problems in imaging », *IEEE Transactions on Computational Imaging*, vol. 6, pp. 328–343, 2019. DOI: 10.1109/TCI.2019.2948732.

[88] E. Kobler, A. Effland, K. Kunisch, and T. Pock, « Total deep variation for linear inverse problems », *in Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7546–7555. DOI: 10.1109/cvpr42600.2020.00757.

[89] J. Liu, Y. Sun, X. Xu, and U. S. Kamilov, « Image restoration using total variation regularized deep image prior », *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP) pages = 7715–7719*, 2019.

[90] P. Cascarano, A. Sebastiani, and M. C. Comes, « Admm-diptv: combining total variation and deep image prior image restoration », *arXiv preprint arXiv:2009.11380*, 2001.

[91] P. Cascarano, A. Sebastiani, M. C. Comes, G. Franchini, and F. Porta, « Combining weighted total variation and deep image prior for natural and medical image restoration via admm », *arXiv preprint arXiv:2009.11380*, 2021.

[92] G. Mataev, M. Elady, and P. Milanfarz, « Deepred: deep image prior powered by red », *arXiv preprint arXiv:1903.10176*, 2019.

[93] K. Zhang, W. Zuo, and L. Zhang, « Ffdnet: toward a fast and flexible solution for cnn-based image denoising », *IEEE Transactions on Image Processing*, vol. 27, *9*, pp. 4608–4622, 2018. DOI: 10.1109/TIP.2018.2839891.

[94] A. H. Al-Shabili, H. Mansour, and P. T. Boufounos, « Learning plug-and-play proximal quasi-newton denoisers », *in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 8896–8900. DOI: 10.1109/ICASSP40776.2020.9054537.

[95] O. Ronneberger, P. Fischer, and T. Brox, « U-net: convolutional networks for biomedical image segmentation », *in International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[96] K. He, X. Zhang, S. Ren, and J. Sun, « Deep residual learning for image recognition », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90.

[97] S. Mohan, Z. Kadkhodaie, E. P. Simoncelli, and C. Fernandez-Granda, « Robust and interpretable blind image denoising via bias-free convolutional neural networks », *in International Conference on Learning Representations*, 2020.

[98] K. Ma, Z. Duanmu, Q. Wu, *et al.*, « Waterloo exploration database: new challenges for image quality assessment models », *IEEE Transactions on Image Processing*, vol. 26, *2*, pp. 1004–1016, 2016. DOI: 10.1109/TIP.2016.2631888.

[99] Y. Chen and T. Pock, « Trainable nonlinear reaction diffusion: a flexible framework for fast and effective image restoration », *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, *6*, pp. 1256–1272, 2016. DOI: 10.1109/TPAMI.2016.2596743.

[100] E. Agustsson and R. Timofte, « Ntire 2017 challenge on single image super-resolution: dataset and study », *in Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 126–135. DOI: 10.1109/CVPRW.2017.150.

[101] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, « Enhanced deep residual networks for single image super-resolution », *in Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144. DOI: 10.1109/CVPRW.2017.151.

[102] D. P. Kingma and J. Ba, « Adam: a method for stochastic optimization. », *in ICLR*, 2015.

[103] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-l. A. Morel, « Low-complexity single-image super-resolution based on nonnegative neighbor embedding », *in Proc. BMVC*, 2012, pp. 135.1–135.10. DOI: 10.5244/C.26.135.

[104] D. Martin, C. Fowlkes, D. Tal, and J. Malik, « A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics », *in Proc. ICCV*, vol. 2, 2001, pp. 416–423. DOI: 10.1109/ICCV.2001.937655.

[105] J. Rick Chang, C.-L. Li, B. Poczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, « One network to solve them all–solving linear inverse problems using deep projection models », *in Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5888–5897. DOI: 10.1109/iccv.2017.627.

[106] K. Zhang, L. V. Gool, and R. Timofte, « Deep unfolding network for image super-resolution », *in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3217–3226. DOI: 10.1109/CVPR42600.2020.00328.

[107] R. Cohen, Y. Blau, D. Freedman, and E. Rivlin, « It has potential: gradient-driven denoisers for convergent solutions to inverse problems », *in Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[108] T. Maugey, « Acquisition, representation, and rendering of omnidirectional videos », *in Immersive Video Technologies*, Elsevier, 2023, pp. 27–48.

[109] S. G. Tzafestas, *Introduction to mobile robot control*. Elsevier, 2013.

[110] D. Scaramuzza, « Omnidirectional vision: from calibration to robot motion estimation. eth zurich », Ph.D. dissertation, ETH Zurich, 2008.

[111] K.-T. Ng, S.-C. Chan, and H.-Y. Shum, « Data compression and transmission aspects of panoramic videos », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, *1*, pp. 82–95, 2005.

[112] C.-W. Fu, L. Wan, T.-T. Wong, and C.-S. Leung, « The rhombic dodecahedron map: an efficient scheme for encoding panoramic video », *IEEE Transactions on Multimedia*, vol. 11, *4*, pp. 634–644, 2009.

[113] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun, « Deep 360 pilot: learning a deep agent for piloting through 360 sports videos », *in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 1396–1405.

[114] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang, « Semantic-driven generation of hyperlapse from 360 degree video », *IEEE transactions on visualization and computer graphics*, vol. 24, *9*, pp. 2610–2621, 2017.

[115] Y.-C. Su, D. Jayaraman, and K. Grauman, « Pano2vid: automatic cinematography for watching 360 videos », *in Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part IV*, Springer, 2017, pp. 154–171.

[116] Y.-C. Su and K. Grauman, « Making 360 video watchable in 2d: learning videography for click free viewing », *in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 1368–1376.

[117] S.-H. Chou, Y.-C. Chen, K.-H. Zeng, H.-N. Hu, J. Fu, and M. Sun, « Self-view grounding given a narrated 360 video », *in Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[118] Y. Yu, S. Lee, J. Na, J. Kang, and G. Kim, « A deep ranking model for spatio-temporal highlight detection from a 360 ◦ video », *in Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[119] Y.-C. Su and K. Grauman, « Learning spherical convolution for fast features from 360 imagery », *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[120] Y.-C. Su and K. Grauman, « Kernel transformer networks for compact spherical convolution », *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9442–9451.

[121] Y.-C. Su and K. Grauman, « Learning spherical convolution for 360 recognition », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, *11*, pp. 8371–8386, 2021.

[122] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, « Spherical cnns », *arXiv preprint arXiv:1801.10130*, 2018.

[123] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, « Learning so (3) equivariant representations with spherical cnns », *in Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 52–68.

[124] C. Esteves, A. Makadia, and K. Daniilidis, « Spin-weighted spherical cnns », *Advances in Neural Information Processing Systems*, vol. 33, pp. 8614–8625, 2020.

[125] P. J. Roddy and J. D. McEwen, « Sifting convolution on the sphere », *IEEE Signal Processing Letters*, vol. 28, pp. 304–308, 2021.

[126] N. Perraudin, M. Defferrard, T. Kacprzak, and R. Sgier, « Deepsphere: efficient spherical convolutional neural network with healpix sampling for cosmological applications », *Astronomy and Computing*, vol. 27, pp. 130–146, 2019.

[127] Q. Yang, C. Li, W. Dai, J. Zou, G.-J. Qi, and H. Xiong, « Rotation equivariant graph convolutional network for spherical image classification », *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4303–4312.

[128] R. Khasanova and P. Frossard, « Geometry aware convolutional filters for omnidirectional images representation », *in International Conference on Machine Learning*, PMLR, 2019, pp. 3351–3359.

[129] N. M. Bidgoli, R. G. d. A. Azevedo, T. Maugey, A. Roumy, and P. Frossard, « Oslo: on-the-sphere learning for omnidirectional images and its application to 360-degree image compression », *IEEE Transactions on Image Processing*, vol. 31, pp. 5813–5827, 2022.

[130] K. M. Gorski, E. Hivon, A. J. Banday, *et al.*, « Healpix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere », *The Astrophysical Journal*, vol. 622, *2*, p. 759, 2005.

[131] H. Nagahara, Y. Yagi, and M. Yachida, « Super-resolution from an omnidirectional image sequence », *in 2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies*, IEEE, vol. 4, 2000, pp. 2559–2564.

[132] L. Bagnato, Y. Boursier, P. Frossard, and P. Vandergheynst, « Plenoptic based super-resolution for omnidirectional image sequences », *in 2010 IEEE International Conference on Image Processing*, IEEE, 2010, pp. 2829–2832.

[133] Z. Arican and P. Frossard, « Joint registration and super-resolution with omnidirectional images », *IEEE Transactions on Image Processing*, vol. 20, *11*, pp. 3151–3162, 2011.

[134] C. Ozcinar, A. Rana, and A. Smolic, « Super-resolution of omnidirectional images using adversarial learning », *in 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2019, pp. 1–6.

[135] X. Deng, H. Wang, M. Xu, L. Li, and Z. Wang, « Omnidirectional image super-resolution via latitude adaptive network », *IEEE Transactions on Multimedia*, 2022.

[136] X. Chai, F. Shao, Q. Jiang, and H. Ying, « Tccl-net: transformer-convolution collaborative learning network for omnidirectional image super-resolution », *Knowledge-Based Systems*, p. 110 625, 2023.

[137] F. Yu, X. Wang, M. Cao, G. Li, Y. Shan, and C. Dong, « Osrt: omnidirectional image super-resolution with distortion-aware transformer », *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 13 283–13 292.

[138] X. Sun, W. Li, Z. Zhang, *et al.*, « Opdn: omnidirectional position-aware deformable network for omnidirectional image super-resolution », *arXiv preprint arXiv:2304.13471*, 2023.

[139] S. Bigot, D. Kachi, S. Durand, and E. M. Mouaddib, « Spherical image denoising and its application to omnidirectional imaging. », *in VISAPP (1)*, 2007, pp. 101–108.

[140] C. Demonceaux and P. Vasseur, « Markov random fields for catadioptric image processing », *Pattern Recognition Letters*, vol. 27, *16*, pp. 1957–1967, 2006.

[141] A. Iazzi, A. Radgui, M. Rziza, *et al.*, « An adapted block thresholding method for omnidirectional image denoising », *Research Journal of Applied Sciences, Engineering and Technology*, vol. 8, *18*, pp. 1966–1972, 2014.

[142] T. D. K. Phan and T. H. Y. Tran, « A space-variant nonlinear algorithm for denoising omnidirectional images corrupted by poisson noise », *IEEE Signal Processing Letters*, vol. 27, pp. 535–539, 2020.

[143] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, « Recognizing scene viewpoint using panoramic place representation », *in 2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 2695–2702.

[144] M. Yu, H. Lakshman, and B. Girod, « A framework to evaluate omnidirectional video coding schemes », *in 2015 IEEE international symposium on mixed and augmented reality*, IEEE, 2015, pp. 31–36.

[145] Y. Sun, A. Lu, and L. Yu, « Weighted-to-spherically-uniform quality evaluation for omnidirectional video », *IEEE signal processing letters*, vol. 24, *9*, pp. 1408–1412, 2017.

[146] M. Le Pendu and C. Guillemot, « Preconditioned plug-and-play admm with locally adjustable denoiser for image restoration », *SIAM Journal on Imaging Sciences*, vol. 16, *1*, pp. 393–422, 2023.

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Apprentissage profond pour les problèmes inverses et application à l'imagerie omnidirectionnelle

Mot clés : problèmes inverses, régularisation, deep learning, images omnidirectionnelles, plug-and-play, débruitage

Résumé : Cette thèse est consacrée à contribuer à des solutions d'apprentissage profond pour régulariser des problèmes inverses en imagerie perspective et omnidirectionnelle. Dans la première partie, nous nous concentrons sur les images 2D et nous commençons par proposer de régulariser le Deep Image Prior avec le débruiteur DRUNet de pointe. Cette combinaison améliore les performances du DIP et se compare favorablement avec les méthodes qui ont été proposées précédemment pour le régulariser. Ensuite, nous proposons une nouvelle approche pour entraîner un réseau modélisant le gradient d'un régulariseur en utilisant un débruiteur appris. Nous utilisons ce réseau dans un algorithme de descente de gradient Plug-and-play et montrons qu'il surpasse les méthodes génériques existantes. Dans la deuxième partie de cette thèse, nous nous concentrons sur les images omnidirectionnelles et nous abordons le problème du débruitage, comme première étape vers des méthodes de régularisation basées sur le débruiteur. Nous introduisons un nouveau débruiteur sphérique, en transférant le débruiteur DRUNet de pointe sur la sphère. Nous montrons également que le débruitage d'images omnidirectionnelles est plus efficace lorsqu'il est effectué directement sur la sphère plutôt que débruiter sa projection.

Title: Deep learning for inverse problems and application to omnidirectional imaging

Keywords: Inverse problems, regularization, deep learning, omnidirectional images, plug-and-play, denoising

Abstract: This thesis is devoted to contribute to deep learning solutions to regularize inverse problems in perspective and omnidirectional imagery. In the first part of this manuscript, we focus on perspective imagery and we begin by proposing to regularize the Deep Image Prior with the state-of-the-art DRUNet denoiser. This combination enhances the performance of the DIP and compares favourably with the methods that have been proposed previously to regularize it. Then, we propose a novel approach to train a network modeling the gradient of a regularizer by using a deep denoiser. We use this network in a Plug-and-play gradient descent algorithm and show that it outperforms existing generic methods. In the second part of this thesis, we focus on omnidirectional images and we address the problem of omnidirectional image denoising, as a first step towards denoiser-based regularization methods. We introduce a novel spherical denoiser, by transferring the state-of-the-art DRUNet denoiser into the sphere. We also show that omnidirectional image denoising is more efficient when performed directly on the sphere rather than a corresponding mapping.