Adrian Steinmann, BSc

A Duality-Based Approach for Learning Convex Regularizers in Image Denoising

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Mathematics

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Math. Dr. Kristian Bredies

University of Graz

Department of Mathematics and Scientific Computing

Graz, April 2026

Abstract

Image denoising is a fundamental challenge in image processing, aiming to recover clean images from noisy observations. Traditional variational methods, such as Total Variation (TV) denoising, rely on fixed linear operators like discrete gradients to enforce regularity. While these methods are interpretable, they lack adaptability to the specific statistics of natural images. Conversely, deep learning-based approaches achieve state-of-the-art results but often require large amounts of training data and lack transparency.

This thesis introduces a hybrid approach by learning linear filters for the penalty term in a variational denoising framework. The denoising problem is formulated as an optimization task, where the denoised image is obtained by minimizing a combination of a fidelity term and a penalty term. The penalty term is defined using a linear operator composed of convolutional filters and a mixed $\ell^{1,2}$-norm to promote sparsity. The objective is to learn these filters such that the denoised images closely match the ground truth.

A key contribution is the use of the primal objective gap (POG) as an objective function for learning the filters, which is reformulated using the primal-dual gap to avoid bi-level optimization. This approach is computationally efficient and leverages the Proximal Alternating Linearized Minimization (PALM) algorithm with inertia terms to handle non-convexity.

Empirical validation is conducted using the BSDS500 dataset, comparing the proposed method against TV denoising, a closely related method by Chen et al. that used bi-level learning for the filters, BM3D, and a deep learning-based method (DRUNet). The learned filters achieve competitive performance, particularly in preserving structural details while effectively removing noise. For example, with 80 filters of size 9×9, the method achieves an average PSNR of 28.03 dB and SSIM of 0.9683, outperforming TV denoising and approaching BM3D and DRUNet.

The learned filters effectively highlight irregularities, and visualizations in the dual space demonstrate how the penalty term suppresses noise while preserving edges. This work demonstrates that learning linear filters within a variational framework can achieve competitive denoising performance while maintaining interpretability. Future research could explore extending the denoiser to color images, refining the penalty term for better rotational symmetry, and investigating applications in related imaging problems. This thesis contributes to the development of faster, more stable, and reliable solutions in inverse problems, bridging the gap between traditional and modern data-driven approaches.

Kurzfassung

Bildentrauschung ist eine grundlegende Herausforderung in der Bildverarbeitung, die darauf abzielt, saubere Bilder aus verrauschten Beobachtungen wiederherzustellen. Traditionelle variationelle Methoden wie die Total-Variation-(TV)-Entrauschung basieren auf festen linearen Operatoren, etwa diskreten Gradienten, um Regularität zu erzwingen. Obwohl diese Methoden interpretierbar sind, fehlt ihnen die Anpassungsfähigkeit an die spezifischen Statistiken natürlicher Bilder. Im Gegensatz dazu erzielen Deep-Learning-basierte Ansätze zwar Spitzenleistungen, erfordern jedoch große Mengen an Trainingsdaten und sind oft intransparent in ihrer Arbeitsweise.

Diese Arbeit stellt einen hybriden Ansatz vor, bei dem lineare Filter für den Strafterm in einem variationellen Ansatz zum Entrauschen gelernt werden. Das Entrauschungsproblem wird als Optimierungsaufgabe formuliert, bei der das entrauschte Bild durch Minimierung einer Kombination aus einem Treueterm und einem Strafterm gewonnen wird. Der Strafterm wird mithilfe eines linearen Operators definiert, der aus Faltungsfiltern und einer gemischten $\ell^{1,2}$-Norm besteht, um dünn besetzte Lösungen zu fördern. Ziel ist es, diese Filter so zu lernen, dass die entrauschten Bilder möglichst nah an der Ground Truth liegen.

Ein zentraler Beitrag dieser Arbeit ist die Verwendung der primalen Diskrepanz im Zielfunktionswert (POG, Primal Objective Gap) als Kostenfunktion für das Lernen der Filter. Diese wird mithilfe der primal-dualen Lücke umformuliert, um ein zweistufiges Optimierungsschema zu vermeiden. Dieser Ansatz ist recheneffizient und nutzt den Proximal Alternating Linearized Minimization (PALM)-Algorithmus mit Trägheitstermen, um die nicht konvexe Kostenfunktion zu handhaben.

Die empirische Validierung erfolgt mit dem BSDS500-Datensatz. Dabei wird die vorgeschlagene Methode mit TV-Entrauschung, einer eng verwandten Methode von Chen et al., die zweistufiges Lernen für die Filter nutzt, BM3D und einer Deep-Learning-basierten Methode (DRUNet) verglichen. Die gelernten Filter erzielen eine wettbewerbsfähige Leistung, insbesondere bei der Erhaltung struktureller Details und der effektiven Rauschunterdrückung. Mit 80 Filtern der Größe 9×9 erreicht die Methode einen durchschnittlichen PSNR von 28,03 dB und einen SSIM von 0,9683 auf dem BSDS500 Datensatz, wodurch sie die TV-Entrauschung übertrifft und an die Leistung von BM3D und DRUNet heranreicht.

Die gelernten Filter heben Unregelmäßigkeiten effektiv hervor, und Visualisierungen im dualen Raum zeigen, wie der Regularisierungsterm Rauschen unterdrückt, während Kanten erhalten bleiben. Diese Arbeit demonstriert, dass das Lernen linearer Filter innerhalb eines variationellen Rahmens eine wettbewerbsfähige Entrauschungsleistung bei gleichzeitiger Interpretierbarkeit erreichen kann.

Zukünftige Forschung könnte die Erweiterung des Entrauschers auf Farbbilder, die

Verfeinerung des Strafterms für eine bessere Rotationssymmetrie und die Untersuchung von Anwendungen in verwandten Bildverarbeitungsproblemen umfassen. Diese Arbeit leistet einen Beitrag zur Entwicklung schnellerer, stabilerer und zuverlässigerer Lösungen in inversen Problemen und schließt die Lücke zwischen traditionellen und modernen datengetriebenen Ansätzen.

Acknowledgements

This endeavor would not have been possible without my supervisor, Prof. Bredies, who recognized my interest in image processing and decided to support and guide me through multi-year research journey. I am deeply grateful for the many in-depth discussions that expanded my knowledge and nurtured my passion in applied mathematics.

This work has been supported by the TraDE-OPT project which received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 861137.

Thanks should also go to Professors Clason, Haase and Pock for their willingness to assist me in overcoming challenges related to high-performance computing and optimization. My gratitude also goes to the Erasmus+ program for the opportunity to attend the International Ph.D. Summer School for Mathematics and Machine Learning for Image Analysis at the Università di Bologna, where I broadened my understanding of image processing and connected with like-minded researchers.

Finally, I offer my heartfelt thanks to my friends and family, especially my parents and spouse, for always believing in me and their constant support throughout this journey.

Contents

List of Figures

List of Tables

List of Acronyms and Symbols

MSE	mean squared error
PALM	proximal alternating linearized minimization
POG	primal objective gap
PSNR	peak signal-to-noise ratio
SSIM	structural similarity index
TV	total variation

1 Introduction

The term "image" carries significant ambiguity. While one person might envision a traditional oil painting, another might think of a picture shot with a digital camera. This broad term encompasses far more than just visual representations; it includes photographs (analog and digital), scans, drawings, sketches, and paintings. Furthermore, in technical contexts, "image" refers to the output of numerous imaging techniques that visualize data beyond simple visuals, such as X-rays, radio telescope observations, ultrasound, or sonar, all operating on different principles.

Therefore, before delving into image processing, we must first establish a clear definition of what we mean by "image", particularly in a mathematical context.

In mathematics, images are functions that map a two-dimensional spatial grid into a color space. In this sense, a digital grayscale image with $N \in \mathbb{N}$ pixels in height and $M \in \mathbb{N}$ pixels in width is given by a function:

$$x : \{0, ..., N-1\} \times \{0, ..., M-1\} \to \mathbb{R}, \tag{1.1}$$

where $x(i, j)$ denotes the intensity value of the image x at pixel (i, j). Equivalently, an image can also be represented as a matrix $x \in \mathbb{R}^{N \times M}$, where the entries $x_{i,j}$ of the matrix specify the intensity values of the image.

Irrespective of how a measurement is conducted and in what form it is represented as an image, a perfectly accurate measurement free of any errors does not exist in the real world. Defects occur in various forms, such as blurriness, chromatic aberration, low resolution, or noise. This work specifically deals with mathematical methods for noise reduction in images. Digital cameras capture images through a successive conversion process: photons-to-electrons, electrons-to-voltage, and voltage-to-digital. Imperfections inevitably arise at each of these conversion steps.

Furthermore, random deviations can begin before this process, for example, under extremely low light conditions or short exposure times, where the number of photons captured by the sensor is so small that the uneven arrival of these photons manifests as noise.

These measurements affected by noise are denoted here by x^δ, where the variable $\delta > 0$ represents the intensity of the noise. The corresponding noise-free counterpart, also known as the Ground Truth, is denoted by $x^\dagger$. In the real world, we only have access to the noisy measurements x^δ, and the algorithms for noise reduction should approximate the unknown Ground Truth as closely as possible.

However, to validate whether an algorithm actually works, we need the Ground Truth as a reference. Therefore, for the development of algorithms, clean images are assumed to serve as Ground Truth, which are subsequently corrupted artificially by adding noise to mimic the noisy measurements. Depending on the application in the real world for

which the algorithm is to be validated, the dataset for the Ground Truths and the model for the noise must be carefully selected.

Since this work does not target any specific specialized application, we will primarily use familiar motifs such as landscapes, people, animals, or architecture captured with a digital camera or smartphone.

For the same reason, we will not employ a particularly sophisticated model for the simulated noise and stick to additive Gaussian noise with a mean of zero.

Gaussian noise is likely the most extensively studied model because it is mathematically well-understood and produces realistically appearing defects, at least for grayscale images. The intensity for Gaussian noise is usually specified by the standard deviation. For grayscale images, whose pixel values are normalized between 0 and 1, standard deviation values δ between 0.01 and 0.10 are common. However, if the pixel values in an 8-bit representation range from 0 to 255, then the standard deviation must also be scaled accordingly by multiplying it by 255 to achieve the same effect.

The differences between two images can be measured in various ways. A frequently used method is the Mean Squared Error (MSE), which is defined as the arithmetic mean of the squared differences between the pixel values of image x and its corresponding Ground Truth $x^\dagger$:

$$\mathrm{MSE}(x, x^\dagger) := \frac{1}{NM}\|x - x^\dagger\|_2^2 = \frac{1}{NM}\sum_{i,j}(x_{i,j} - x^\dagger_{i,j})^2. \tag{1.2}$$

The MSE is particularly convenient due to its mathematical properties. For instance, the expected value of $\mathrm{MSE}(x^\delta, x^\dagger)$ for Gaussian noise with mean zero is exactly the variance of that distribution, which is δ^2 in our case. The MSE is a non-negative quantity and is exactly zero only if the image x exactly matches the Ground Truth $x^\dagger$.

The goal of any algorithm for noise reduction is to modify the input x^δ such that the output is as close as possible to the Ground Truth $x^\dagger$. This means minimizing the error, for example, measured using the MSE. Since the noise is random, the algorithms operate using statistics and heuristics to reduce the noise.

The simplest example of such a method is the Arithmetic Mean Filter. This filter calculates the local average over a small rectangular window. Assuming $n \in \mathbb{N}$ and $m \in \mathbb{N}$ specify how many neighboring pixels in height and width should be included in the averaging process, the filtered image $\bar{x}$ is given pixel-wise by:

$$\bar{x}_{i,j} = \frac{1}{(2n+1)(2m+1)}\sum_{k=-n}^{n}\sum_{l=-m}^{m} x_{i+k,j+l}. \tag{1.3}$$

This computation operation also accesses pixels outside the defined boundary, which can be handled in different ways. In the simplest case, the pixel values in the invalid region are assumed to be 0, although this leaves a dark border around the filtered image.

The Arithmetic Mean Filter smooths the image at the cost of sharpness. Averaging over neighboring pixels cancels out the independently occurring pixel-wise noise on average, and the larger the area over which the average is taken, the better the noise is

reduced. Figure 1 shows an example. On the left is the Ground Truth, serving as the reference. The image in the middle has been artificially corrupted with additive Gaussian noise with mean zero and standard deviation $\delta = 0.1$. The MSE of the middle image is 0.01003, which is close to the variance of the Gaussian distribution, $\delta^2 = 0.01$. The blurred image on the right has a smaller MSE of 0.00729 and is closer to the Ground Truth in this sense. However, the perceived image quality due to the blurring filter has not necessarily improved subjectively. Therefore, alongside statistics, example images are also needed to evaluate a noise reduction method.

Figure 1: The image of the flower field has a resolution of 600×400 and the grayscale values have been scaled to the interval $[0, 1]$. The clean image on the left serves as the ground truth, and the image in the middle has been artificially corrupted with Gaussian noise with standard deviation 0.1.
The image on the right has been blurred by an Arithmetic Mean Filter with kernel size 5×5. The blurred image is a bit less noisy, but at the cost of the image quality.

The example of the flower field in Figure 1 also shows that it is not clear in advance how strong the smoothing effect should be chosen. Prominent deviations on an otherwise constant sky are noticeable even with small noise, requiring strong smoothing, while the same noise in the flower field is comparatively insignificant and can remain almost unchanged.

In essence, image processing algorithm development, whether focused on creating new solutions or refining current ones, is predicated on two core requirements: access to suitable datasets and the selection of an appropriate quality assessment criterion. A single denoiser can never be optimal for all possible images and criteria at the same time. Furthermore, the characteristics of the dataset used for evaluation significantly influence the perceived performance and applicability of the developed algorithm. While synthetic data offers controlled environments for testing specific hypotheses, its relevance to real-world scenarios depends heavily on how closely it mimics practical conditions.

Therefore, robust algorithm development requires consideration of both synthetic benchmarks and representative real-world data.

This thesis is structured as follows. In Chapter 1, we address the fundamental problem statement and introduce some denoising methods from the field of sparse programming. Subsequently, in Chapter 2, we focus on the total variation denoising model in particular and how solvers have developed over time. We will highlight one algorithm, which is based on the theory for primal and dual problems from convex analysis. Based on

this theory, we will also introduce a new error measure called "primal objective gap" or shortly POG, which has received little attention in image processing so far. TV denoising has only one free parameter, which can be optimized with relatively little effort regarding various error metrics. However, models with a vast quantity of free parameters are interesting due to their adaptability. We introduce such a model in Chapter 3, which is computationally expensive to optimize for most error metrics but works well with the POG measure. Both the error metric and the denoising model are already known, but, to the best of my knowledge, have not yet been combined in this form, which constitutes the main innovation of this work. Empirical tests using publicly available and standardized datasets in Chapter 4 serve as the basis for comparison with other noise reduction methods. Finally, the main findings are summarized in Chapter 5, along with further research questions that arose during this work.

1.1 Sparse Image Processing

The challenge in image denoising lies in the fact that, even when the probability distribution of the noise is known, due to the random nature of the disturbance, there are typically infinitely many images that can explain the given measurement data. However, the pixel values in a "clean" image are not entirely random. They form connected structures, segments, or sections. Within such image segments, the pixel values do not vary arbitrarily but are related in strict neighborhood relationships. Thus, the pixel values might change gradually or periodically. In both cases, the segment can be described almost completely with relatively little information. This observation about "clean" images forms the basis for effective image compression. Pixel-wise random noise does not adhere to these neighborhood relationships and is also much harder to compress effectively. Accordingly, it is natural to restrict the vast number of images that could potentially explain the measurement data by searching for images that can be described with as little information as possible. Although algorithms for compressing or denoising images have different goals, they share remarkably similar mathematical foundations.

One speaks of a sparse image, or more generally, a sparse signal, when most of its entries are zero, thus containing only necessary information, which is also reflected in lower memory requirements. Sparse signals find surprisingly diverse applications, and are more relevant today than ever before.

The following section explains the key concepts from this research field and aims to motivate the models for image denoising studied in this work. Key foundational texts in this area include, for example, [19] and [31].

An image is rarely sparse itself, and it requires appropriate linear transformations to find a sparse representation for the image. Generally, the signal $x \in \mathbb{R}^{N \times M}$ is modeled as a linear combination of $S \in \mathbb{N}$ distinct signal atoms:

$$x = Dy = \sum_{i=1}^{S} d^i y_i \ , \tag{1.4}$$

where y_i are the coefficients representing the image x with respect to the dictionary

$d^1, ..., d^S$. Thus, the image is given as an superposition of the atoms, and consequently, the atoms d^i are elements of $\mathbb{R}^{N\times M}$ too. The coefficient vector y, however, lies in $\mathbb{R}^S$. Whether the equation system $Dy = x$ is solvable depends on both D and x. For the system to be solvable for any image x, there must be at least as many atoms as degrees of freedom, i.e., $S \geq NM$. If the dictionary D is chosen such that the linear transformation is invertible, then the system always has a unique solution. However, invertible linear transformations have a fairly limited sparsifying effect. Over complete dictionary correspond with under determined systems of linear equations and allow for the selection of the sparsest solution among many. This leads us to the following optimization problem:

$$\min_{y\in\mathbb{R}^S} \#\{i : y_i \neq 0\} \quad \text{s.t. } Dy = x \ . \tag{1.5}$$

The number of nonzero entries in a vector is sometimes referred to as the ℓ^0-norm, thus $\|y\|_0 := \#\{i : y_i \neq 0\}$. This is motivated by the limit $\lim_{p\to 0} \|y\|_p^p = \#\{i : y_i \neq 0\}$, although strictly speaking, $\|\cdot\|_0$ is not a norm because the absolute homogeneity condition,

$$\|\alpha y\|_0 = |\alpha|\|y\|_0 \ \forall \alpha \in \mathbb{R}, y \in Y,$$

does not hold for this mapping. Optimization problems aiming to describe the image using atoms and sparse coefficient vectors are also referred to as *sparse synthesis models* in the literature.

Using the number of non-zero entries as the objective function in the minimization problem serves the purpose of compressing images or signals. However, it also presents its challenges. This function maps to the natural numbers, i.e. $\|\cdot\|_0 : \mathbb{R}^S \mapsto \mathbb{N}$, which restricts us to methods from discrete optimization. Nevertheless, there is also a wide range of continuous functions $\phi : \mathbb{R}^S \mapsto \mathbb{R}$ that can lead to similarly sparse solutions. A precise mathematical definition for "sparsity-promoting" functions does not exist, since the very concept of "sparse" does not have a precise definition either.

Some important examples of functions, that are typically referred to as sparsity promoting, are presented in [19, p. 11] and repeated here:

$$y \mapsto \sum_{i=1}^{S} |y_i|^p \text{ with } 0 < p \leq 1 \ , \tag{1.6}$$

$$y \mapsto \sum_{i=1}^{S} \ln\left(1 + |y_i|\right) \ , \tag{1.7}$$

$$y \mapsto \sum_{i=1}^{S} 1 - \exp\left(-|y_i|\right) \ , \tag{1.8}$$

$$y \mapsto \sum_{i=1}^{S} \frac{|y_i|}{1 + |y_i|} \ . \tag{1.9}$$

All the objective functions mentioned above are separable, meaning they can be ex-

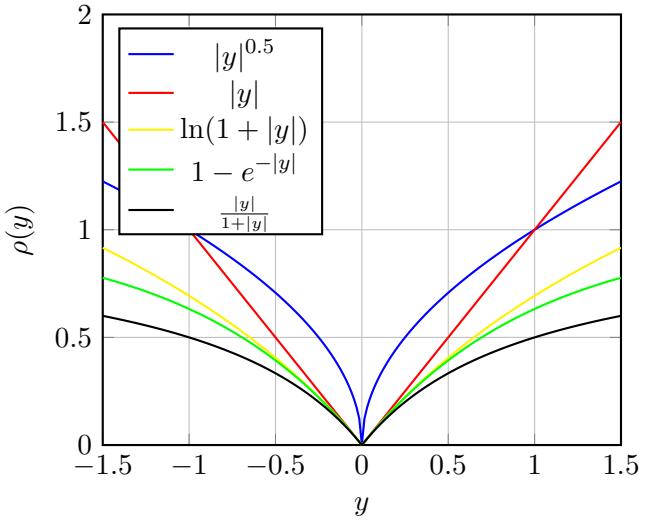


Figure 2: A selection of commonly used objective functions for promoting sparse coefficients. In all cases zero is the minimizer and the slope is steepest around zero.

pressed in the form

$$\phi(y) = \sum_{i=1}^{S} \rho(y_i),$$

where $\rho : \mathbb{R} \to \mathbb{R}$. Using a separable objective function is useful, because the partial derivatives of ϕ only require the knowledge about the derivative of ρ, i.e. $\dfrac{\partial}{\partial y_i}\phi(y) = \rho'(y_i)$. The examples for the functions $\rho : \mathbb{R} \to \mathbb{R}$ share a combination of properties that typically lead to sparse solutions. Specifically, they are continuous, symmetric about zero, strictly increasing and concave on the positive half-axis, and zero is always a minimizer for the function.

These properties play an important role in signal recovery. In particular, under suitable conditions on the dictionary D and the signal x, minimizing such sparsity-promoting functions can recover the same solution as the original combinatorial problem.

More precisely, suppose that for a given image x the problem (1.5) admits a sparse solution $y^\dagger$, meaning that $y^\dagger$ contains only a small number of nonzero entries. Then, for certain dictionaries, it can be shown that $y^\dagger$ is not only the sparsest solution, but often also a minimizer of the relaxed optimization problem

$$\min_{y \in \mathbb{R}^S} \sum_{i=1}^{S} \rho(y_i) \quad \text{s.t. } Dy = x \ . \tag{1.10}$$

In other words, although the original ℓ^0-minimization problem is combinatorial and difficult to solve, it can often be replaced by a continuous optimization problem without changing the solution. This observation forms the theoretical foundation for many practical sparse recovery methods.

Particularly interesting is the ℓ^1-norm as an objective function, as it leads to sparse solutions, although it is convex. The convexity allows for methods from convex optimization, which we will examine in detail in the next chapter.

Now, let us return to the context of image processing, specifically image compression. Allowing for a certain deviation from the exact solution to the system of equations $Dy = x$ improves the compression rate significantly. The constraint $\frac{1}{NM}\|x - Dy\|_2^2 = 0$ is relaxed to $\frac{1}{NM}\|x - Dy\|_2^2 \leq \varepsilon$, where the limit to the loss of quality is specified by $\varepsilon \geq 0$. The optimization problem for a lossy compression using a sparsity promoting objective function $\phi : \mathbb{R}^S \mapsto \mathbb{R}$ is therefore:

$$\min_{y \in \mathbb{R}^S} \phi(y) \quad \text{s.t. } \frac{1}{NM}\|x - Dy\|_2^2 \leq \varepsilon \ . \tag{1.11}$$

A larger tolerance ε allows for stronger compression of the signal. At this point, we find the connection between compression and denoising. Because coarse structures in the image can be described with few atoms and (in absolute value) large coefficients when the dictionary is chosen appropriately. Fine details in the image, but also noise, typically require the superposition of many atoms with small coefficients, which are the first to be "spared" in the optimization. Assuming that it involves normally distributed noise with variance $\delta^2 > 0$, one would set the tolerance ε to δ^2. If we substitute the measured data x^δ for the image x and replace the compressed image Dy with the ground truth $x^\dagger$, then the term in the constraint becomes $\frac{1}{NM}\|x^\delta - x^\dagger\|_2^2$ and this has an expected value of δ^2, which explains the choice of ε. Applying this change to the sparse synthesis model gives us the following optimization problem:

$$\min_{y \in \mathbb{R}^S} \phi(y) \quad \text{s.t. } \frac{1}{NM}\|x^\delta - Dy\|_2^2 \leq \delta^2 \ . \tag{1.12}$$

The choice of the dictionary in combination with the objective function is decisive for the behavior of these sparse synthesis models. In the literature, one finds many different dictionaries for various applications and signals. The atoms are roughly named according to their shapes, for example, there are wedgelets [18], wavelets [33], platelets [35], surflets [12], ridgelets [7] or curvelets [32]. For an overview and explanation of these dictionaries, see [27]. However, alongside the development of dictionaries, a second related approach has also been pursued, namely that of the co-sparse analysis models. The corresponding optimization problem is:

$$\min_{x \in \mathbb{R}^{N \times M}} \phi(Ax) \quad \text{s.t. } \frac{1}{NM}\|x^\delta - x\|_2^2 \leq \delta^2 \ . \tag{1.13}$$

Instead of the linear operator in the form of a dictionary D, one uses a so-called analysis operator A. This model aims for the sparsest possible representation of the analyzed signal Ax in the analysis domain, under the condition that the deviation from the measured data x^δ does not exceed the specified tolerance. The analysis domain is

also known as the co-domain and sparsity in the co-domain is therefore known as co-sparsity. Sometimes, in modeling the problem, the constraint on the deviation from the measurement is enforced using a penalty function:

$$\min_{x \in \mathbb{R}^{N \times M}} \frac{1}{2} \|x^\delta - x\|_2^2 + \lambda \phi(Ax) \ . \tag{1.14}$$

Here, $\lambda > 0$ now takes on the role of ε, balancing the denoising effect with fidelity to the measured data.

If one uses the discrete gradient operator for A, one obtains the well-known model for total variation denoising, details of which are discussed in Chapter 2.

The sparse synthesis models and the co-sparse analysis models are equivalent to each other if the operators D or A are invertible. By the substitutions $y = D^{-1}x = Ax$ or $x = A^{-1}y = Dy$, one can switch between the models without the solution changing. However, in practice, the operators are rarely invertible.

Regardless of whether dictionaries or analysis operators are used, in both cases, instead of manually designing the operators one can use machine learning for data driven designs. Thus, the operators A or D are tuned to given datasets, which leads to visibly improved denoised images.

Whether sparse synthesis models or co-sparse analysis models are better suited for machine learning cannot be answered clearly, but analysis models are less thoroughly researched and therefore of particular interest to us.

1.2 Machine Learning for Co-Sparse Analysis Models

In the field of machine learning, one fundamentally distinguishes between supervised and unsupervised learning. Both variants require a training dataset in the form of reference images/ground truth images $(x^\dagger[i])_{i=1}^T$, where $T \in \mathbb{N}$ denotes the number of data points. For supervised learning, an additional set of corrupted images $(x^\delta[i])_{i=1}^T$ is also needed.

The goal of the unsupervised learning process is to find a linear operator A that maps the image space X to the analysis space Y, such that the individual analyzed signals $(Ax^\dagger[i])_{i=1}^T$ are as sparse as possible. However, this optimization goal requires additional constraints to be useful for image denoising, because the trivial zero operator yields the sparsest signals. In [36], the authors Yaghoobi et al. constrain the operator A to the set of Uniform Normalized Tight Frames (UNTF). In that work the operator A is represented as a matrix of size $S \times NM$ with $S \geq NM$. Here, "Uniform Normalized" refers to the condition that all row norms must be equal, and "Tight Frames" requires the relationship $A^T A = I$, where I is the identity operator on $\mathbb{R}^{NM}$. The optimization problem thus becomes:

$$\min_A \sum_{i=1}^T \|Ax^\dagger[i]\|_1 \quad \text{s.t.} \quad A \in \text{UNTF}. \tag{1.15}$$

In [36] a projected subgradient method is employed to solve (1.15). In a later work [37] the denoising is also tested on synthetic data and real face images. However, the analysis

operator was trained on 8×8 patches and not on full images, because the operator A is fully populated and increasing the patch size drastically increases the compute cost. The degrees of freedom in the optimization problem scales with the square of the pixel count in a patch, limiting this method to small patches. This approach unfortunately also introduces visible transitions between the individual patches of the image.

Since then, more than a dozen papers have been published, further developing the concept of Uniform Normalized Tight Frames as well as employing entirely different objective functions and constraints. In particular, two relatively recent papers by Chun et al. [15], [16] address the question of how convolutional operators (instead of matrices) can be learned, in order to avoid the known limitations of patch-based operators.

However, all unsupervised learning methods are limited by the fact that they do not incorporate any information about the noise model into the training process, and the constraints on the analysis operator only have an indirect and difficult to control influence on the denoising process.

For supervised machine learning, in addition to the reference images/Ground Truth $(x^\dagger[i])_{i=1}^T$, also corrupted images $(x^\delta[i])_{i=1}^T$ are used. The denoised images $(x^A[i])_{i=1}^T$ are then computed according to a co-sparse analysis model such as in Equation (1.14):

$$x^A[i] \in \operatorname*{argmin}_{x \in \mathbb{R}^{N \times M}} \frac{1}{2}\|x^\delta[i] - x\|_2^2 + \lambda\phi(Ax). \tag{1.16}$$

Notice that the solutions x^A depend directly on the analysis operator A.

The goal of the training process is to find an analysis operator that minimizes the cumulative error between the denoised images x^A and the ground truths $x^\dagger$, thereby making the reconstruction quality the objective of the learning process. Again, conditions are imposed on the operator, and Ω denotes the set over which the optimization is performed. The distance between the denoised images and the ground truth is measured by a distance function $d : X \times X \to \mathbb{R}$ (such as the MSE). The training process is formulated here as a minimization problem:

$$\min_{A \in \Omega} \sum_{i=1}^{T} d(x^\dagger[i], x^A[i]) \text{ s.t. } \quad x^A[i] \in \operatorname*{argmin}_{x \in \mathbb{R}^{N \times M}} \frac{1}{2}\|x^\delta[i] - x\|_2^2 + \lambda\phi(Ax) \tag{1.17}$$

Since the denoised images themselves are computed as solutions to a co-sparse analysis model, this involves the nesting of two optimization problems. Such bilevel optimization problems are computationally expensive to solve because each step updating the upper problem requires fully solving the lower problem.

The works of Samuel et al. [29] and Chen et al. [13] address the question of efficiently solving the bilevel problem (1.17). Chen et al.'s work adopts most of the modeling approach of Samuel et al., but employs a different algorithm for solving the problem. In both cases, analysis operators were implemented using multiple convolutional operators, rather than, as often seen in earlier works, dividing the image into small patches. As the distance function d, the squared ℓ^2-norm was chosen, which, up to scaling, corresponds to the Mean Squared Error (MSE). For sparsity promotion in the lower level problem,

the function $\phi : y \mapsto \sum_i \log(1 + |y_i|^2)$ was selected. Chen et al. additionally investigated the functions:

$$y \mapsto \sum_i |y_i| \ , \quad y \mapsto \sum_i \log(1 + |y_i|), \tag{1.18}$$

but these are not differentiable at zero. Consequently, a small smoothing parameter $\varepsilon > 0$ was introduced to restore differentiability:

$$y \mapsto \sum_i \sqrt{y_i^2 + \varepsilon^2} \ , \quad y \mapsto \sum_i \log\left(1 - \varepsilon + \sqrt{y_i^2 + \varepsilon^2}\right). \tag{1.19}$$

The differentiability of the terms in the inner problem was subsequently utilized by the authors to replace the minimization constraint with the equality constraint:

$$(x^\delta[i] - x^A[i]) + \lambda A^* \phi'(Ax^A[i]) = 0 \quad \forall i \in \{1, \ldots, T\} \tag{1.20}$$

in the outer optimization problem. This new equality constraint is weaker than the full minimization constraint, because non-convex problems can have stationary points that are not minimizers. However, this substitution is used because guaranteeing global minimizers for this non-convex problem is not possible in a practical way.

To the end, we are back to a single-level (mono-level) optimization problem with non-linear constraints. The gradient of the cost function with respect to the parameter A can be transformed using the chain rule as follows:

$$\nabla_A \left(\sum_{i=1}^{T} \frac{1}{2} \|x^A[i] - x^\dagger[i]\|_2^2 \right) = \sum_{i=1}^{T} (\nabla_A x^A[i])(x^A[i] - x^\dagger[i]) \ . \tag{1.21}$$

The challenge here is calculating the gradient $\nabla_A x^A[i]$, which requires implicit differentiation of the constraint (1.20). For details, see [29]. The main difference between the works of Samuel et al. and Chen et al. is the choice of algorithms. Samuel et al. used a simple gradient descent method, while Chen et al. employed a quasi-Newton method. These, along with other minor improvements, ultimately led to significantly better results.

Since then, a lot of attention has shifted from learning analysis operators and similar approaches to deep machine learning and neural networks.

However, the potential of analysis operators in convex optimization has not yet been fully exhausted and can still deliver competitive results today.

2 Fitting a single parameter for optimal denoising

A cornerstone of image denoising is the Rudin-Osher-Fatemi (ROF) model, which introduces total variation (TV) regularization as a means to preserve sharp edges while removing noise [28]. Despite its conceptual simplicity, it is highly effective and the ROF model has had a profound impact on both theory and practice, inspiring a wide range of subsequent methods and extensions. The first part of this chapter is devoted to revisiting the ROF model: its mathematical formulation, continuous and discrete settings, isotropic and anisotropic discretizations, and the numerical strategies used to solve the associated optimization problem.

However, the study of denoising is not complete without addressing the question of how to assess the quality of a reconstruction. We, as humans, subconsciously judge the quality of images by many different standards and it often comes down to taste which image is preferred. A metric is an attempt at an impossible task: assigning a single score to the quality of reconstruction. That is why in practice, different application contexts may favor different notions of optimal reconstruction. Widely used evaluation metrics such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural SIMilarity index (SSIM [34]) offer quantitative perspectives on reconstruction quality.

In the second part of this chapter, we therefore stay in the already well-understood ROF setting, and investigate how different distance functions measuring the reconstruction error shape the notion of optimal denoising. Building on this motivation, we propose a new objective function based on the Bregman distance of the penalty term, and we explore its impact on the selection of optimal parameters for a given dataset. This framework aims to provide deeper insight into the relation between model design, parameter tuning, and the resulting image quality.

Taken together, this chapter serves two purposes: first, to review the foundations of TV-based denoising with an emphasis on the ROF model, and second, to develop and analyze new criteria for defining and computing optimal parameters for image denoising.

2.1 The Rudin-Osher-Fatemi Model

The Rudin-Osher-Fatemi (ROF) model was introduced in 1992 as the foundational approach to total variation (TV) based image denoising [28]. Even though better and more complex denoising methods have been developed since then, it remains a useful tool for image processing and has influenced a wide range of methods for inverse problems and image reconstruction.

As already briefly mentioned in the previous chapter, the ROF model belongs to the wider family of co-sparse analysis models of the form:

$$\min_{x\in\mathbb{R}^{N\times M}} \frac{1}{2}\|x - x^\delta\|_2^2 + \lambda\phi(Ax), \tag{2.1}$$

where the Lagrange multiplier $\lambda > 0$ balances the fidelity term $\frac{1}{2}\|\cdot - x^\delta\|_2^2$ with the regularizer $\phi \circ A$.

The ROF model specifically uses the discrete gradient operator D for analyzing the signal x. The operator D is formally defined as:

$$D : \mathbb{R}^{N\times M} \mapsto \mathbb{R}^{N\times M\times 2} \tag{2.2}$$

$$\forall (i,j) \in \{1,...,N\}\times\{1,...,M\} : \begin{cases} (Dx)_{i,j,1} = x_{i+1,j} - x_{i,j} \\ (Dx)_{i,j,2} = x_{i,j+1} - x_{i,j} \end{cases} \tag{2.3}$$

Notice, that this definition breaks down at the border of the image where $i = N$ or $j = M$. This is repaired by extending the range of definition by one row and column and setting the new values to that of its neighbor:

$$x_{N+1,j} = x_{N,j}\ \forall j \in \{1,...,M\} \text{ and } x_{i,M+1} = x_{i,M}\ \forall i \in \{1,...,N\}.$$

For the sparsity promoting objective function $\phi : \mathbb{R}^{N\times M\times 2} \mapsto \mathbb{R}$ there are two competing standards. Both a pure ℓ^1-norm and a mixed $\ell^{1,2}$-norm are commonly used. The ℓ^1-norm directly sums up the absolute values in the vector $Dx = y$, while the mixed $\ell^{1,2}$-norm first computes the magnitude of the discretized gradient via the ℓ^2-norm and then accumulates these values for all pixels:

$$\forall y \in \mathbb{R}^{N\times M\times 2} : \quad \|y\|_1 := \sum_{i,j,k} |y_{i,j,k}|, \tag{2.4}$$

$$\|y\|_{1,2} := \sum_{i,j} \sqrt{\sum_k |y_{i,j,k}|^2}. \tag{2.5}$$

Combining the analysis operator D with the ℓ^1-norm results in the so called anisotropic total variation:

$$TV_{aniso}(x) = \sum_{i,j} |x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}| = \|Dx\|_1, \tag{2.6}$$

and with the mixed $\ell^{1,2}$-norm it is called isotropic total variation:

$$TV_{iso}(x) = \sum_{i,j} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2} = \|Dx\|_{1,2}. \tag{2.7}$$

In summary, the models for isotropic and anisotropic total variation denoising are respectively:

$$\text{isotropic:} \quad \min_{x \in \mathbb{R}^{N \times M}} \frac{1}{2}\|x - x^\delta\|_2^2 + \lambda \|Dx\|_{1,2} \tag{2.8}$$

$$\text{anisotropic:} \quad \min_{x \in \mathbb{R}^{N \times M}} \frac{1}{2}\|x - x^\delta\|_2^2 + \lambda \|Dx\|_{1}. \tag{2.9}$$

In order to understand the "isotropic" and "anisotropic" properties of the total variation, we must first take a closer look at images with a continuous range of definition. The discrete grid $\{1, ..., N\} \times \{1, ..., M\}$ is replaced by an open rectangular domain $\Omega \subset \mathbb{R}^2$, and the image is represented as a function $u : \Omega \to \mathbb{R}$. The total variation of a differentiable function is defined via the gradient. The isotropic total variation uses the ℓ^2-norm of the gradient, leading to the functional $\int_\Omega \|\nabla u(x)\|_2 dx$, while the anisotropic version uses the ℓ^1-norm, giving $\int_\Omega \|\nabla u(x)\|_1 dx$. In the ROF model the ℓ^2-norm was used.

Isotropy refers in this context to the fact, that the magnitude of the gradient of u in a point x measured with respect to the ℓ^2-norm does not change under rotation. The ℓ^1-norm on the other hand, does not posses this rotational invariance. Since the fidelity term also uses the ℓ^2-norm, one can exchange the order of operations of denoising and rotating an image, provided the isotropic total variation term is used for regularization.

The rotational invariance does not transfer cleanly to the discrete setting, because arbitrary rotations of images on discrete grids require compromises such as interpolation. However, even in the discrete setting, is the isotropic total variation notably more consistent under rotation.

Rudin, Osher and Fatemi actually started with the continuous model for total variation denoising and derived the discrete model from the continuous one. This is highly unusual, because digital images are always discrete and therefore the development of image processing algorithms also tends to start in the discrete setting. In the case of the ROF model [28] the inspiration apparently came from studying shock wave theory in the field of partial differential equations. Regularizing with the total variation term does not overly smooth the system and allows for sharp transitions along the shock wave fronts. This property is also desirable for image processing, because clearly defined edges are fundamental to the perceived sharpness of an image.

Studying non continuous functions with sharp edges, such as images, requires the notion of total variation to be extended to a wider class of functions than just the differentiable ones. However, this is a non trivial matter and would require a long detour from the topic on hand. Instead we recommend the interested reader [5, pp. 333-372] for a comprehensive discussion of the total variation model in the continuous setting.

One of the key observations of the ROF model is that TV regularization is edge-preserving. Large gradients, corresponding to object boundaries or sharp transitions, are not excessively penalized and thus remain visible in the reconstruction. This is in sharp contrast to quadratic regularizers, which are easier to work with but also tend to smear out edges.

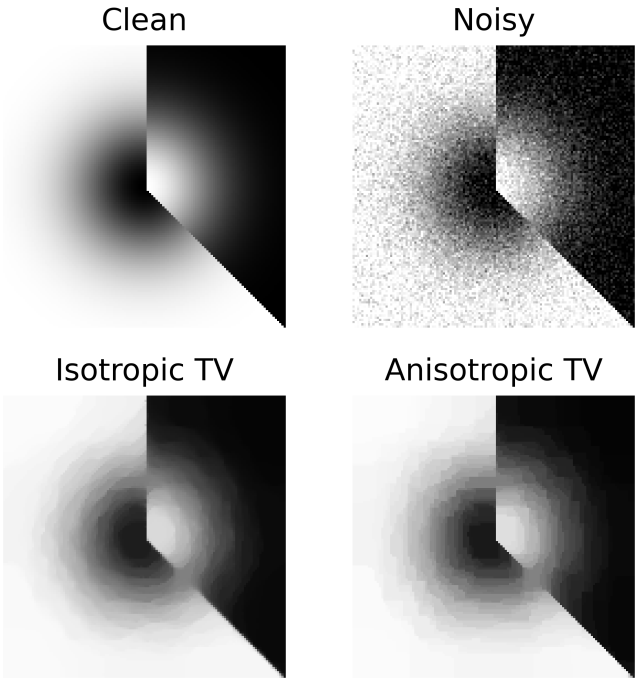


Figure 3: A direct comparison of the isotropic and anisotropic total variation denoising applied to an image corrupted by white Gaussian noise with standard deviation $\sigma = 0.1$. The denoising parameter λ was chosen as 0.3 for the isotropic TV and as $0.3/\sqrt{2}$ for the anisotropic TV. The solutions were computed using Algorithm 2, that will be presented later. The test image has a resolution of 128×128 and depicts a Gaussian blob that is color inverted along a vertical and diagonal axis. Both denoiser almost perfectly preserve the vertical and diagonal edges. On the flip side, they introduce edges radially around the bloom, where there should be none. This is known as the staircasing effect. The main difference between the isotropic and anisotropic TV is, that these artifacts are rounded in the left image while in the right they are aligned with the horizontal and vertical axes.

At the same time, the TV regularizer's preference for piecewise constant solutions leads to a well-known artifact: the staircasing effect. Instead of producing smooth transitions, the model often yields flat regions separated by sharp jumps. While edges are preserved, smooth intensity variations disturbed by noise may be incorrectly reconstructed as multiple small jumps, as demonstrated in Figure 3.

The ROF model's inherent non-differentiability at zero (due to the absolute value function) makes solving for a denoised image computationally challenging. Early methods, such as gradient descent, PDE solvers, and regularized TV, suffered from slow convergence and numerical instability. This instability stems from the fact that, as the algorithm converges to a sparse solution, more values approach zero, degrading the accuracy of gradient approximations due to the worsening lack of differentiability. This issue is common to many sparsity-promoting functions. To address this, different techniques for handling non-differentiable functions were explored. A key advancement was the introduction of primal-dual formulations to the denoising problems. These reformulate the problem using a dual variable for the calculation of the gradient, enabling efficient algorithms that can manage the non-differentiability in a stable manner.

Chambolle's algorithm [8] was in 2004 one of the first to demonstrate the effectiveness of the dual formulation for TV minimization, followed by the 2008 primal-dual hybrid gradient (PDHG [38]) method of Zhu and Chan. Later, a unified primal-dual solver for saddle-point problems was developed by Chambolle and Pock and became a widely adopted method for minimizing the sum of two convex, not necessarily differentiable functions [9]. Further refinements, such as the introduction of extrapolation and inertial terms, made the solver even quicker in practice, however the underlying idea stayed the same [10]. Today, primal-dual solvers are considered among the state of the art for TV denoising, combining convergence guarantees with practical efficiency.

2.2 Primal and dual problems in convex optimization

The total variation (TV) model fits into a broad class of convex optimization problems of the form

$$\inf_{x\in\mathbb{R}^n} \; f(x) + g(Ax), \tag{2.10}$$

where $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ and $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ are convex functions, $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear operator and $\overline{\mathbb{R}} := \mathbb{R} \cup \{+\infty\}$ denotes the real values extended to include infinity. The minimization problem (2.10) is referred to as the primal problem.

One should be aware, that any convex objective function $\phi : \mathbb{R}^n \to \overline{\mathbb{R}}$ can be expressed in infinitely many different ways as $f + g \circ A$. However, for a specific problem, there is typically only one natural decomposition. In practice, models are often designed to fit the $f + g \circ A$ structure, precisely to enable the use of primal-dual theory.

The structure $f + g \circ A$ actually encompasses a wide range of problems, such as the sparse synthesis problem or co-sparse analysis problem, provided the sparsity promoting function is convex such as the ℓ^1-norm. This decomposition is also common for other inverse problems in imaging besides denoising, such as super-resolution, deblurring or inpainting.

Allowing $+\infty$ as a function value enables elegantly incorporating constraints into the objective function, but also requires some additional attention to detail, because the symbolic operations "$0 \cdot \infty$" and "$+\infty - \infty$" are undefined. This and many other conventions were established by Rockafellar in the 70s in the foundational work *Convex Analysis* [25], which will be used as a reference for most definitions here.

We restrict ourselves to studying only functions defined on the whole space $\mathbb{R}^n$, because any function f defined on a subset $S \subset \mathbb{R}^n$ can be extended by setting $f(x) = +\infty \; \forall x \notin S$ [25, p. 23]. What previously was the domain of the function is now replaced by the *effective domain*.

Definition 1 (Effective Domain)**.** *[25, p. 23]*
The effective domain *of a function $f : \mathbb{R}^n \mapsto \overline{\mathbb{R}}$ is the set*

$$dom\ f := \{x \in \mathbb{R}^n : f(x) < +\infty\}.$$

For multiple definitions and theorems it is necessary to exclude the degenerate function with constant value $+\infty$. This is done by introducing the notion of a *proper* function.

Definition 2 (Proper functions)**.** *[25, p. 24]*
A convex function $f : \mathbb{R}^n \mapsto \overline{\mathbb{R}}$ is proper*, if and only if*

$$dom\ f \neq \emptyset.$$

A key concept from convex analysis is the *convex conjugate* of a function, also known as the Fenchel conjugate, of a proper function f.

Definition 3 (Convex Conjugate)**.** *[25, p. 104]*
Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper. Then the (convex) conjugate *of f is defined as*

$$f^* : \mathbb{R}^n \mapsto \overline{\mathbb{R}}, \tag{2.11}$$

$$f^*(x^*) := \sup_{x \in \mathbb{R}^n} \langle x, x^* \rangle - f(x). \tag{2.12}$$

The first step towards obtaining the dual problem makes use of the *Fenchel-Young inequality*. This inequality is a direct consequence of Definition 3 and states for every proper function f and its convex conjugate f^*:

$$\langle x, x^* \rangle \leq f(x) + f^*(x^*), \quad \forall x \in \mathbb{R}^n \; \forall x^* \in \mathbb{R}^n. \tag{2.13}$$

Rearranging yields

$$f(x) \geq \langle x, x^* \rangle - f^*(x^*), \quad \forall x \in \mathbb{R}^n \; \forall x^* \in \mathbb{R}^n. \tag{2.14}$$

Applying the Fenchel-Young inequality in the form $g(Ax) \geq \langle Ax, y \rangle - g^*(y) \; \forall y \in \text{dom } g^*$ to the primal problem (2.10) gives rise to the *saddle-point problem*

$$\inf_{x \in \mathbb{R}^n} \sup_{y \in \text{dom } g^*} f(x) + \langle Ax, y \rangle - g^*(y). \tag{2.15}$$

This problem has a saddle structure in the sense that it is convex in x and concave in y. Interchanging the order of infimum and supremum only makes the value smaller:

$$\inf_{x\in\mathbb{R}^n}\sup_{y\in\text{dom } g^*} f(x)+\langle Ax,y\rangle - g^*(y) \geq \sup_{y\in\text{dom } g^*}\inf_{x\in\mathbb{R}^n} f(x)+\langle Ax,y\rangle - g^*(y). \quad (2.16)$$

Applying the Fenchel-Young inequality to f and expanding the set dom g^* to $\mathbb{R}^m$, one obtains the dual problem:

$$\sup_{y\in\mathbb{R}^m} -f^*(-A^*y) - g^*(y). \quad (2.17)$$

So far this only shows weak duality, i.e., that the primal objective is greater than or equal to the dual objective.

$$\begin{aligned}&\forall x \in \text{dom } f,\ \forall y \in \text{dom } g^* :\\ &f(x)+g(Ax) \geq f(x)+\langle Ax,y\rangle - g^*(y) \geq -f^*(-A^*y)-g^*(y). \end{aligned} \quad (2.18)$$

The non-negative difference between the primal and dual objective values is known as the primal-dual gap, which is characterized by the gap function G:

$$G:\mathbb{R}^n\times\mathbb{R}^m,\ G(x,y) := f(x)+g(Ax)+f^*(-A^*y)+g^*(y). \quad (2.19)$$

The key for connecting the primal with the dual problem is Fenchel's Duality Theorem. It states, that under some mild assumptions strong duality holds, that means the primal (2.10), dual (2.17), and saddle-point (2.15) problems share the same optimal objective. Furthermore, if one knows a solution to either the primal or dual problem, then the other one can be recovered with the help of the primal-dual optimality conditions. Another consequence of that theorem is, that $G(x,y)$ is only zero, when (x,y) is optimal and motivates $G(x,y)<\varepsilon$ as a stopping criterion. In the context of computing solutions to the ROF model, we will make use of the saddle-point problem in particular.

Before the exact statement of Fenchel's Duality Theorem can be presented, we need a few more definitions.

The concept of a *subgradient* can be seen as a generalization for the gradient of a convex function, because a gradient is also a subgradient while the inverse implication is not true in general.

Definition 4 (Subgradient)**.** *[25, p. 216]*
Let f be a convex function and let $x \in$ dom f. Then $x^*\in\mathbb{R}^n$ is a ***subgradient*** *of f at x if*

$$\langle x^*,\tilde{x}-x\rangle \leq f(\tilde{x})-f(x)\ \forall\tilde{x}\in\mathbb{R}^n.$$

The set of all subgradients of f at x is denoted as

$$\partial f(x) := \{x^*\in\mathbb{R}^n : \langle x^*,\tilde{x}-x\rangle \leq f(\tilde{x})-f(x)\ \forall\tilde{x}\in\mathbb{R}^n\}.$$

Subgradients provide a simple optimality condition for minimization problems that is both necessary and sufficient. For a given proper and convex function f the condition $0 \in \partial f(x)$ is indeed equivalent to x being a global minimizer of f.

$$f(x) \leq f(\tilde{x}) \ \forall \tilde{x} \in \mathbb{R}^n \tag{2.20}$$

$$\langle 0, \tilde{x} - x \rangle \leq f(\tilde{x}) - f(x) \ \forall \tilde{x} \in \mathbb{R}^n. \tag{2.21}$$

Writing down the definition of a global minimizer (2.20) next to zero being a subgradient at x (2.21) reveals, that both conditions actually coincide.

Stating the obvious, finding a global minimizer is only possible if a global minimizer exists. A proper and convex function f does not have to have a minimizer, even if f is bounded from below and the effective domain is bounded. One common way to address this issue is to additionally require f to be *lower semi-continuous*, which is a less restrictive requirement than sequential continuity.

Definition 5 (Lower Semi-Continuity)**.** *[25, p. 51]*
A function $f : \mathbb{R}^n \mapsto \overline{\mathbb{R}}$ is lower semi-continuous in $x \in dom\ f$ if

$$f(x) \leq \liminf_{n \to \infty} f(x_n)$$

for every sequence $(x_n)_{n=1}^{\infty}$ in $dom\ f$ converging to x.

Lower semi-continuity ensures that convergent minimizing sequences of f converge to actual minimizers.

Theorem 6 (Fenchel's Duality Theorem)**.** *[25, p. 332]*
Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}, g : \mathbb{R}^m \to \overline{\mathbb{R}}$ be proper, convex, lower semi-continuous functions and A a linear operator from $\mathbb{R}^n$ to $\mathbb{R}^m$. Further, let the following conditions be satisfied:

1. *the primal problem **(P)** admits a solution $\bar{x} \in \mathbb{R}^n$,*
2. *there exists a $x_0 \in dom(f) \cap dom(g \circ A)$ with $Ax_0 \in int(dom(g))$.*

*Then, the dual problem **(D)** also admits a solution $\bar{y} \in \mathbb{R}^m$ and one has*

$$\textbf{(P)} \ \min_{x \in \mathbb{R}^n} f(x) + g(Ax) = \max_{y \in \mathbb{R}^m} -f^*(-A^*y) - g^*(y). \ \ \textbf{(D)} \tag{2.22}$$

In order for $\bar{x}$ and $\bar{y}$ to be solutions to the primal and dual problems respectively, it is necessary and sufficient that $\bar{x}$ and $\bar{y}$ satisfy the Fenchel-extremality conditions:

$$\begin{cases} -A^*\bar{y} & \in \partial f(\bar{x}), \\ \bar{y} & \in \partial g(A\bar{x}). \end{cases} \tag{2.23}$$

Proving Theorem 6 requires multiple definitions and lemmas we have not covered here, thus for a proof please refer to Rockafellar's *Convex Analysis* [25, p. 327].

Theorem 6 does not mention the saddle-point problem explicitly, but by weak duality (2.18), equation (2.22) can be extended to:

$$\min_{x\in\mathbb{R}^n} f(x)+g(Ax) = \min_{x\in\mathbb{R}^n}\max_{y\in\mathbb{R}^m} f(x)+\langle Ax,y\rangle - g^*(y) = \max_{y\in\mathbb{R}^m} -f^*(-A^*y)-g^*(y). \quad (2.24)$$

Which one of these three problems is the easiest to solve is a question without a clear answer. In the case of total variation denoising all of them have been used in the past and compared to each other. While the solvers have become substantially faster over time, it is unclear how much of that progress can be attributed to the problem formulation and how much to the algorithmic gains, because those two things are deeply intertwined with each other.

2.3 Duality theory applied to total variation denoising

This chapter focuses on the dual problem associated with total variation denoising. We explore how the saddle-point structure allows us to formulate a problem that circumvents the non-differentiability of the sparsity-promoting function. We then verify the conditions required for Theorem 6 to guarantee the equivalence between the primal and dual problems. Finally, we discuss how to recover the primal solution from the dual optimum.

Converting the generic primal problem (2.10) into that for isotropic total variation denoising comes naturally with the following choices:

$$f : \mathbb{R}^{N\times M} \to \mathbb{R}, \quad f(x) = \frac{1}{2}\|x - x^\delta\|_2^2, \quad (2.25)$$

$$g : \mathbb{R}^{N\times M\times 2} \to \mathbb{R}, \quad g(y) = \|y\|_{1,2} = \sum_{i,j}\sqrt{(y_{i,j,1})^2 + (y_{i,j,2})^2}, \quad (2.26)$$

$$A : \mathbb{R}^{N\times M} \to \mathbb{R}^{N\times M\times 2}, \quad (2.27)$$

$$\forall (i,j) \in \{1,...,N\}\times\{1,...,M\} : \begin{cases} (Ax)_{i,j,1} = \lambda(x_{i+1,j} - x_{i,j}), \\ (Ax)_{i,j,2} = \lambda(x_{i,j+1} - x_{i,j}), \end{cases} \quad (2.28)$$

$$\text{where } x_{N+1,j} = x_{N,j}\ \forall j \in \{1,...,M\} \text{ and } x_{i,M+1} = x_{i,M}\ \forall i \in \{1,...,N\}. \quad (2.29)$$

Here, the linear operator A is the same as the discrete gradient operator defined in equation (2.2) multiplied with the denoising parameter $\lambda > 0$. Alternatively to putting the parameter λ into A we could define the function g as $\lambda\|\cdot\|_{1,2}$. That would not change the underlying model, because the $\ell^{1,2}$-norm is positively homogeneous.

In case of total variation denoising, the convex conjugates of f and g can be determined by some basic calculations. An explicit form for the convex conjugate can sometimes be hard to come by and that has a major deciding factor, whether to go with an primal-dual formulation or a completely different approach.

The convex conjugate of f is by definition

$$f^*(x^*) = \sup_{x\in\mathbb{R}^{N\times M}} \langle x, x^*\rangle - \frac{1}{2}\|x - x^\delta\|_2^2. \quad (2.30)$$

The supremum can be calculated by differentiating with respect to x and setting the derivative to zero. The critical point is a maximizer, because the function is differentiable and strictly concave:

$$\frac{d}{dx}\left(\langle x, x^*\rangle - \frac{1}{2}\|x - x^\delta\|_2^2\right) = 0, \tag{2.31}$$

$$\Rightarrow x^* - x + x^\delta = 0, \Rightarrow x = x^* + x^\delta. \tag{2.32}$$

Plugging the maximizer into $\langle \cdot, x^*\rangle - \frac{1}{2}\|\cdot - x^\delta\|_2^2$ gives us the function value of $f^*(x^*)$.

$$\Rightarrow f^*(x^*) = \langle x^* + x^\delta, x^*\rangle - \frac{1}{2}\|x^*\|_2^2 \tag{2.33}$$

$$= \frac{1}{2}\|x^*\|_2^2 + \langle x^\delta, x^*\rangle = \frac{1}{2}\|x^* + x^\delta\|_2^2 - \frac{1}{2}\|x^\delta\|_2^2. \tag{2.34}$$

This calculation shows, that the convex conjugate of the squared ℓ^2-norm is again the squared ℓ^2-norm and the shift by x^δ becomes the linear function $\langle x^\delta, .\rangle$.

Calculating the convex conjugate of $g(y) = \|y\|_{1,2}$ needs a different approach, because the function is not differentiable everywhere. Instead, we are going to make use of the definition of the dual norm:

$$\|y^*\|_* := \sup_{\|y\|_{1,2}\leq 1} \langle y, y^*\rangle. \tag{2.35}$$

We want to determine:

$$g^*(y^*) = \sup_{y\in\mathbb{R}^{N\times M\times 2}} \langle y, y^*\rangle - \|y\|_{1,2}. \tag{2.36}$$

For any fixed y, note that for $t \geq 0$,

$$\langle y, ty^*\rangle - \|ty^*\|_{1,2} = t(\langle y, y^*\rangle - \|y^*\|_{1,2}). \tag{2.37}$$

If there exists a y such that $\langle y, y^*\rangle > \|y\|$, then scaling $t \to \infty$ drives the supremum to $+\infty$. So for the supremum to remain finite, $\langle y, y^*\rangle \leq \|y\|$ must be true for all y, which is equivalent to $\|y\|_* \leq 1$. Taken together, this shows that g^* is the indicator function over the closed unit ball in the dual norm:

$$g^*(y) = I_{\{\|\cdot\|_*\leq 1\}}(y) := \begin{cases} 0, \text{ if } \|y\|_* \leq 1, \\ +\infty, \text{ otherwise.} \end{cases} \tag{2.38}$$

The task has now shifted to determining the dual norm of $\|y\|_{1,2}$.

Lemma 7 (Dual norm)**.** *Let $p, q \geq 1$ and let $p^*, q^* \geq 1$ be conjugate scalars, i.e. $\frac{1}{p} + \frac{1}{p^*} = 1$ and $\frac{1}{q} + \frac{1}{q^*} = 1$. Then the dual of the $\ell^{p,q}$-norm is the ℓ^{p^*,q^*}-norm.*

A complete proof to Lemma 7 can be found in the paper [30]. However, here we are only going to prove the special case $p = 1, q = 2$.

Proof. We need to verify the claim

$$\sup_{\|y\|_{1,2}\leq 1} \langle y, y^* \rangle = \|y^*\|_{\infty,2} = \max_{i,j} \sqrt{(y^*_{i,j,1})^2 + (y^*_{i,j,2})^2}.$$

First, find an upper limit to the supremum

$$\forall y, y^* \in \mathbb{R}^{N\times M\times 2} : \|y\|_{1,2} \leq 1 \Rightarrow \langle y, y^* \rangle \leq \|y^*\|_{\infty,2},$$

and then construct a specific $y \in \mathbb{R}^{N\times M\times 2}$ that attains the supremum.

In the following paragraph the notation $y_{i,j} = \begin{pmatrix} y_{i,j,1} \\ y_{i,j,2} \end{pmatrix} \in \mathbb{R}^2$ is used.

Let $y, y^* \in \mathbb{R}^{N\times M\times 2}$ be two arbitrary vectors with $\|y\|_{1,2} \leq 1$. Using the Cauchy-Schwarz inequality in $\mathbb{R}^2$ for each pixel and then taking the maximum over all pixels, we obtain:

$$\langle y, y^* \rangle = \sum_{i,j} \langle y_{i,j}, y^*_{i,j} \rangle \leq \sum_{i,j} \|y_{i,j}\|_2 \, \|y^*_{i,j}\|_2 \tag{2.39}$$

$$\leq \max_{i,j} \|y^*_{i,j}\|_2 \sum_{i,j} \|y_{i,j}\|_2 = \|y^*\|_{\infty,2} \, \|y\|_{1,2} \leq \|y^*\|_{\infty,2} \, . \tag{2.40}$$

To construct a maximizer y for a given $y^* \in \mathbb{R}^{N\times M\times 2}$, let (i_0, j_0) be any pair of indices such that

$$\|y^*_{i_0,j_0}\|_2 = \max_{i,j} \|y^*_{i,j}\|_2 \, . \tag{2.41}$$

If $y^* = 0$, set $y = 0$. Otherwise, set

$$y_{i,j} = \begin{cases} \dfrac{y^*_{i,j}}{\|y^*_{i,j}\|_2}, & \text{if } (i,j) = (i_0, j_0), \\ 0, & \text{else.} \end{cases} \tag{2.42}$$

Clearly, this satisfies $\|y\|_{1,2} = 1$, and the inner product attains the supremum:

$$\langle y, y^* \rangle = \left\langle \frac{y^*_{i_0,j_0}}{\|y^*_{i_0,j_0}\|_2}, y^*_{i_0,j_0} \right\rangle = \|y^*_{i_0,j_0}\|_2 = \max_{i,j} \|y^*_{i,j}\|_2 = \|y^*\|_{\infty,2}. \tag{2.43}$$

This completes the proof that the dual norm of the $\ell^{1,2}$-norm is the $\ell^{\infty,2}$-norm. □

The last missing piece, before we can write down the dual problem for isotropic total variation, is the adjoint of the scaled discrete gradient operator $A : \mathbb{R}^{N\times M} \mapsto \mathbb{R}^{N\times M\times 2}$. A^* is defined through the identity

$$\langle Ax, y \rangle_{\mathbb{R}^{N\times M\times 2}} = \langle x, A^* y \rangle_{\mathbb{R}^{N\times M}} \ \forall x \in \mathbb{R}^{N\times M} \ \forall y \in \mathbb{R}^{N\times M\times 2}. \tag{2.44}$$

Since $\lambda > 0$ is just a scalar, we know that $A^* = \lambda D^*$. The adjoint of the discrete gradient operator is known to be the negative discrete divergence operator. Deriving D^* from the definition requires mostly shifting around of the indices in the sums involved, but one has to be careful at the boundary:

$$\begin{aligned}
\langle Dx, y\rangle &= \sum_{i=1}^{N-1}\sum_{j=1}^{M}(x_{i+1,j} - x_{i,j})y_{i,j,1} + \sum_{i=1}^{N}\sum_{j=1}^{M-1}(x_{i,j+1} - x_{i,j})y_{i,j,2} && (2.45)\\
&= \sum_{i=2}^{N}\sum_{j=1}^{M} x_{i,j}y_{i-1,j,1} - \sum_{i=1}^{N-1}\sum_{j=1}^{M} x_{i,j}y_{i,j,1} && (2.46)\\
&+ \sum_{i=1}^{N}\sum_{j=2}^{M} x_{i,j}y_{i,j-1,2} - \sum_{i=1}^{N}\sum_{j=1}^{M-1} x_{i,j}y_{i,j,2}\\
&= \sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j}((\tilde{y}_{i-1,j,1} - \tilde{y}_{i,j,1}) + (\tilde{y}_{i,j-1,2} - \tilde{y}_{i,j,2})) = \langle x, D^*y\rangle, && (2.47)
\end{aligned}$$

where $\tilde{y}$ defined on the extended grid $\{0,\dots,N\}\times\{0,\dots,M\}\times\{1,2\}$ accounts for the boundary condition via:

$$\tilde{y}_{i,j,1} = \begin{cases} y_{i,j,1}, \text{ if } (i,j) \in \{1,\dots,N-1\}\times\{1,\dots,M\} \\ 0, \text{ else} \end{cases} \quad (2.48)$$

$$\tilde{y}_{i,j,2} = \begin{cases} y_{i,j,2}, \text{ if } (i,j) \in \{1,\dots,N\}\times\{1,\dots,M-1\} \\ 0, \text{ else} \end{cases} . \quad (2.49)$$

With this extension the adjoint of the discrete divergence operator can be written in an explicit form as:

$$D^* : \mathbb{R}^{N\times M\times 2} \to \mathbb{R}^{N\times M}, \quad (2.50)$$

$$\forall (i,j) \in \{1,...,N\}\times\{1,...,M\} : (D^*y)_{i,j} = (\tilde{y}_{i-1,j,1} - \tilde{y}_{i,j,1}) + (\tilde{y}_{i,j-1,2} - \tilde{y}_{i,j,2}). \quad (2.51)$$

This completes deriving f^*, g^* and A^* for the isotropic total variation denoising model. Written side by side, the primal **(P)**, saddle-point **(S)** and dual **(D)** problem are:

$$\textbf{(P)} \quad \min_{x\in\mathbb{R}^{N\times M}} \frac{1}{2}\|x - x^\delta\|_2^2 + \lambda\|Dx\|_{1,2}, \quad (2.52)$$

$$\textbf{(S)} \quad \min_{x\in\mathbb{R}^{N\times M}} \max_{y\in\mathbb{R}^{N\times M\times 2}} \frac{1}{2}\|x - x^\delta\|_2^2 + \lambda\langle Dx, y\rangle - I_{\{\|.\|_{\infty,2}\le 1\}}(y), \quad (2.53)$$

$$\textbf{(D)} \quad \max_{y\in\mathbb{R}^{N\times M\times 2}} -\frac{1}{2}\|\lambda D^*y\|_2^2 + \lambda\langle x^\delta, D^*y\rangle - I_{\{\|.\|_{\infty,2}\le 1\}}(y). \quad (2.54)$$

For the sake of completeness, let us check if all necessary assumptions for Theorem 6 are met.

The functions $f(x) = \frac{1}{2}\|x - x^\delta\|_2^2$ and $g(y) = \|y\|_{1,2}$ are indeed proper, because they are finite everywhere. Furthermore, f is not only convex, it is even strictly convex and g is a norm and thus convex by the triangle inequality. Similarly, both functions are lower semi-continuous, because they are continuous. The other prerequisites for strong duality are:

1. the primal problem **(P)** admits a solution $\bar{x} \in \mathbb{R}^n$,
2. there exists a $x_0 \in \mathrm{dom} f \cap \mathrm{dom}(g \circ A)$ with $Ax_0 \in \mathrm{int}(\mathrm{dom}(g))$.

The objective of the primal problem $f + g \circ A$ is coercive and continuous, its level sets are bounded and closed, hence compact, ensuring the existence of a minimizer.

The second requirement is also immediate, because dom $f = \mathbb{R}^n$ and dom $(g \circ A) = \mathbb{R}^n$. Thus any $x_0 \in \mathbb{R}^n$ can be picked for the requirement $Ax_0 \in \mathrm{int}(\mathrm{dom}\ g)$. These requirements become far more interesting in the continuous setting where one has to deal with infinite-dimensional vector spaces, as laid out in [5].

This means every primal solution $\bar{x}$ and dual solution $\bar{y}$ satisfies the Fenchel-extremality conditions from Theorem 6:

$$\begin{cases} -A^*\bar{y} & \in \partial f(\bar{x}), \\ \bar{y} & \in \partial g(A\bar{x}). \end{cases} \tag{2.55}$$

A vector y is a subgradient of g at $z = \lambda Dx$ if it satisfies:

$$\langle y, \tilde{z} - z \rangle \leq g(\tilde{z}) - g(z)\ \forall \tilde{z} \in \mathbb{R}^n. \tag{2.56}$$

By expanding the inner product and substituting the definition of the $\ell_{1,2}$ norm, $g = \|\cdot\|_{1,2}$, the separable structure with respect to the indices (i, j) becomes evident:

$$\sum_{i,j} y_{i,j,1}(\tilde{z}_{i,j,1} - z_{i,j,1}) + y_{i,j,2}(\tilde{z}_{i,j,2} - z_{i,j,2}) \leq \sum_{i,j} \sqrt{\tilde{z}_{i,j,1}^2 + \tilde{z}_{i,j,2}^2} - \sqrt{z_{i,j,1}^2 + z_{i,j,2}^2}\ \forall \tilde{z} \in \mathbb{R}^n. \tag{2.57}$$

Letting $y_{i,j} = \begin{pmatrix} y_{i,j,1} \\ y_{i,j,2} \end{pmatrix}$ and $z_{i,j} = \begin{pmatrix} z_{i,j,1} \\ z_{i,j,2} \end{pmatrix}$, the condition above can be equivalently expressed as:

$$\forall (i,j) \in \{1, ..., N\} \times \{1, ..., M\}, \tilde{z}_{i,j} \in \mathbb{R}^2 : \tag{2.58}$$

$$\langle y_{i,j}, \tilde{z}_{i,j} - z_{i,j} \rangle \leq \|\tilde{z}_{i,j}\|_2 - \|z_{i,j}\|_2. \tag{2.59}$$

Consider a fixed index pair (i, j). If $z_{i,j} \neq 0$, the ℓ^2-norm is differentiable, and the condition reduces to:

$$y_{i,j} = \nabla \|z_{i,j}\|_2 = \frac{z_{i,j}}{\|z_{i,j}\|_2}.$$

If $z_{i,j} = 0$, the inequality simplifies to:

$$\forall \tilde{z}_{i,j} \in \mathbb{R}^2 : \langle y_{i,j}, \tilde{z}_{i,j} \rangle \leq \|\tilde{z}_{i,j}\|_2. \tag{2.60}$$

which is precisely the definition of the dual norm:

$$\sup_{\|\tilde{z}_{i,j}\|_2 = 1} \langle y_{i,j}, \tilde{z}_{i,j} \rangle \leq 1. \tag{2.61}$$

Since the ℓ^2-norm is self-dual, the subdifferential of $\|\cdot\|_2$ at zero is the unit ball:

$$\{w \in \mathbb{R}^2 : \|w\|_2 \leq 1\}.$$

Consequently, the subgradients of g at z are characterized as:

$$y_{i,j} \in \begin{cases} \left\{ \dfrac{z_{i,j}}{\|z_{i,j}\|_2} \right\}, & z_{i,j} \neq 0, \\ \{w \in \mathbb{R}^2 : \|w\|_2 \leq 1\}, & z_{i,j} = 0. \end{cases} \tag{2.62}$$

This, however, only gives us the solution to the dual problem, when we already have the solution to the primal problem. The other direction is far more interesting, because the primal solution is the denoised image. For this, we need the following result:

Theorem 8 (Characterization of the subgradient)**.** *[25, p. 218]*
Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper, convex and lower semi-continuous. Then, for all $x, x^* \in \mathbb{R}^n$ the following three conditions are equivalent:

1. $\langle x, x^* \rangle = f(x) + f^*(x^*)$,
2. $x^* \in \partial f(x)$,
3. $x \in \partial f^*(x^*)$.

Proof. If $x \notin \operatorname{dom} f$ or $x^* \notin \operatorname{dom} f^*$, then the set of subgradients is empty and there is nothing to show.

If 1. holds for some $x \in \operatorname{dom} f$ and $x^* \in \operatorname{dom} f^*$, then:

$$\begin{aligned} & \langle x^*, x \rangle - f(x) = f^*(x^*) && (2.63) \\ \Rightarrow\ & \langle x^*, x \rangle - f(x) \geq \langle x^*, \tilde{x} \rangle - f(\tilde{x}) \ \forall \tilde{x} \in \mathbb{R}^n && (2.64) \\ \Leftrightarrow\ & \langle x^*, \tilde{x} - x \rangle \leq f(\tilde{x}) - f(x) \ \forall \tilde{x} \in \mathbb{R}^n && (2.65) \\ \Leftrightarrow\ & x^* \in \partial f(x). && (2.66) \end{aligned}$$

If 2. holds, taking the supremum over all $\tilde{x} \in \mathbb{R}^n$ in (2.64) shows:

$$\begin{aligned} & \langle x^*, x \rangle - f(x) \geq \sup_{\tilde{x} \in \mathbb{R}^n} \langle x^*, \tilde{x} \rangle - f(\tilde{x}) && (2.67) \\ \Rightarrow\ & \langle x^*, x \rangle - f(x) \geq f^*(x^*) && (2.68) \\ \Rightarrow\ & \langle x^*, x \rangle \geq f^*(x^*) + f(x) && (2.69) \\ \text{Fenchel-Young inequality}\ \Rightarrow\ & \langle x^*, x \rangle \leq f^*(x^*) + f(x) && (2.70) \\ \Rightarrow\ & \langle x^*, x \rangle = f^*(x^*) + f(x). && (2.71) \end{aligned}$$

The equivalence between 1. and 2. requires f only to be proper and convex, but for the equivalence to 3. we need $f = f^{**}$, i.e., f is its own bi-conjugate defined as $f^{**}(x) = \sup_{x^* \in \mathbb{R}^n} \langle x^*, x \rangle - f^*(x^*)$. This result is true for proper, convex, and lower semi-continuous functions, but will not be proven here, as it requires additional results not presented. Instead we refer to [25, p. 104].

Defining $h = f^*$, we obtain $h^* = f^{**} = f$. We can apply the already proven equivalence between 1. and 2. to h, because h is proper and convex:

$$\langle x, x^* \rangle = h(x^*) + h^*(x) \Leftrightarrow x \in \partial h(x^*). \tag{2.72}$$

Replacing h by f^* and h^* by f^{**} gives the desired equivalence between 1. and 3. concluding the proof. □

Applying Theorem 8 to the Fenchel-extremality conditions gives:

$$\begin{cases} -A^*\bar{y} & \in \partial f(\bar{x}) \\ \bar{y} & \in \partial g(A\bar{x}) \end{cases} \Leftrightarrow \begin{cases} \bar{x} & \in \partial f^*(-A^*\bar{y}) \\ A\bar{x} & \in \partial g^*(\bar{y}). \end{cases} \tag{2.73}$$

Especially, the relation $\bar{x} \in \partial f^*(-A^*\bar{y})$ is useful, because this is how a solution to the primal problem can be recovered from the solution to the dual problem.

Since in our case $f^*(x^*) = \frac{1}{2}\|x^*\|_2^2 + \langle x^\delta, x^* \rangle$ is differentiable the optimality condition $\bar{x} \in \partial f^*(-A^*\bar{y})$ reads as:

$$\bar{x} = x^\delta - \lambda D^*\bar{y}. \tag{2.74}$$

2.4 Algorithm for efficient total variation denoising

Chan, Golub and Mulet [11] were the first authors to propose a total variation denoising algorithm based on the saddle-point formulation. The downside of the primal formulation is the non-smooth function $g(y) = \|y\|_{1,2}$, which requires artificial smoothing before derivative-based methods can be used. The tradeoffs between the dual and the saddle-point formulation are less clear. Both have to deal with the constraint on the dual variable $\|y\|_{\infty,2} \leq 1$ and the possibility of non-unique solutions, because the discrete gradient operator D is rank deficient.

The main downside of the dual formulation is its spatial stiffness. The operator D couples the variables $y_{i,j,k}$ with each other in the term $y \mapsto \|\lambda D^* y\|_2^2$. Meanwhile, in the saddle-point formulation, the variables are decoupled. The function f and the conjugate g^* appear in separable form, connected only by the bilinear term $\langle Dx, y \rangle$. This separation allows alternating updates in x and y in the form of a gradient ascent for x and a projected gradient descent in y. One such algorithm for solving saddle-point problems is the Arrow-Hurwicz method [2], which was used by Zhu and Chan [38] for TV denoising. The (projected) gradient descent is often combined in literature with proximity operators, also known as proximal maps or simply "prox". Using a proximal update is preferable, because it remains well-defined even for non-differentiable functions.

Definition 9 (Proximal map)**.** *[25, p. 339]*
Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper, lower semi-continuous function and $\tau > 0$. Then, the multi-valued map $prox_\tau^f : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is called the proximity operator corresponding to f.

$$prox_\tau^f(x) := \operatorname{argmin}\left\{ f(y) + \frac{1}{2\tau}\|y - x\|_2^2 : y \in \mathbb{R}^n \right\} \tag{2.75}$$

In the general case, the proximity operator is a set-valued mapping that returns the set of all minimizers. However, when f is convex, the minimizer is unique. Since the functions considered in this work are assumed to be convex, we treat $\operatorname{prox}_\tau^f$ as a single-valued function.

In the case of TV denoising, the proximity operators for f and g^* have an explicit form and can be calculated by hand. Let $\sigma, \tau > 0$ be the step sizes. Then,

$$\operatorname{prox}_\sigma^f(x) := \operatorname*{argmin}_y \frac{1}{2}\|y - x^\delta\|_2^2 + \frac{1}{2\sigma}\|y - x\|_2^2 \tag{2.76}$$

$$\Leftrightarrow y - x^\delta + \frac{1}{\sigma}(y - x) = 0 \text{ and } y = \operatorname{prox}_\sigma^f(x) \tag{2.77}$$

$$\Leftrightarrow (1 + \sigma)y = x + \sigma x^\delta \text{ and } y = \operatorname{prox}_\sigma^f(x) \tag{2.78}$$

$$\Leftrightarrow \operatorname{prox}_\sigma^f(x) = \frac{x + \sigma x^\delta}{1 + \sigma}. \tag{2.79}$$

For the quadratic fidelity term, the proximal map is a weighted averaging step between the input x and the minimizer of f given as x^δ. It coincides with a gradient descent step for f with step size $\dfrac{\sigma}{1+\sigma}$:

$$\frac{x + \sigma x^\delta}{1 + \sigma} = x - \frac{\sigma}{1 + \sigma}(x - x^\delta).$$

For a small step size σ, the proximation stays close to the input x and as the step size increases, the proximation moves closer to x^δ.

The proximal map of g^* is the projection onto the (closed) unit ball defined by the mixed $\ell^{\infty,2}$-norm. In this case, the parameter τ has no influence on the minimizer.

$$\operatorname{prox}_\tau^{g^*}(y) = \operatorname{argmin}\left\{ \frac{1}{2\tau}\|z - y\|_2^2 : \|z\|_{\infty,2} \le 1 \right\} \tag{2.80}$$

$$= \operatorname{argmin}\left\{ \frac{1}{2\tau}\|z - y\|_2^2 : \max_{i,j} \sqrt{z_{i,j,1}^2 + z_{i,j,2}^2} \le 1 \right\}. \tag{2.81}$$

By separability of the max-operator, we can also calculate the proximity operator separately for each index pair (i, j).

$$\left[\operatorname{prox}_\tau^{g^*}(y)\right]_{i,j} = \operatorname{argmin}\left\{\frac{1}{2\tau}\left\|\begin{pmatrix}y_{i,j,1}\\ y_{i,j,2}\end{pmatrix} - \begin{pmatrix}z_{i,j,1}\\ z_{i,j,2}\end{pmatrix}\right\|_2^2 : \sqrt{z_{i,j,1}^2 + z_{i,j,2}^2} \le 1\right\} \tag{2.82}$$

$$= \begin{pmatrix}y_{i,j,1}\\ y_{i,j,2}\end{pmatrix}\left(\frac{1}{\max\left\{1, \sqrt{y_{i,j,1}^2 + y_{i,j,2}^2}\right\}}\right). \tag{2.83}$$

Thus, $\operatorname{prox}_\tau^{g^*}(y) = \operatorname{proj}_{\{\|z\|_{\infty,2}\le 1\}}(y)$ projects all the vectors $y_{i,j} \in \mathbb{R}^2$ onto their respective closed unit balls measured in the ℓ^2-norm.

The Arrow-Hurwicz method features alternating updates on the primal variable x and the dual variable y for solving the saddle-point problem:

$$\min_{x\in\operatorname{dom} f}\max_{y\in\operatorname{dom} g^*} f(x) + \langle Ax, y\rangle - g^*(y). \tag{2.84}$$

For the primal update, we want to fix y and minimize $f(x) + \langle Ax, y\rangle$. The second term is linear in x and has gradient A^*y, which we will use for a gradient descent. Since $f(x)$ is not necessarily differentiable, we apply its proximal map instead, with a step size $\sigma > 0$:

$$x \mapsto \operatorname{prox}_\sigma^f(x - \sigma A^* y). \tag{2.85}$$

Similarly, the dual update is performed by fixing x and maximizing $\langle Ax, y\rangle - g^*(y)$ with respect to the variable y. Notice that we now use a gradient ascent, which means the sign for the linear term changes. Also for the same reason, the proximal map for g^* instead of $-g^*$ is used:

$$y \mapsto \operatorname{prox}_\tau^{g^*}(y + \tau A x). \tag{2.86}$$

For a guarantee on convergence, as proven in [9], the step sizes σ and τ must satisfy $\sigma\tau L^2 < 1$, where $L \ge \|A\|$ denotes an upper bound on the operator norm of A. In the case of total variation denoising, one can estimate $L \ge \lambda\sqrt{8}$. For the proof of this estimate, recall that the linear operator $A = \lambda D$ maps from $\mathbb{R}^{N\times M}$ to $\mathbb{R}^{N\times M\times 2}$. We also need the elementary inequality $(a-b)^2 \le 2(a^2+b^2)$:

$$\|Dx\|_2^2 = \sum_{i,j}(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2 \tag{2.87}$$

$$\le 2\sum_{i,j}(x_{i+1,j})^2 + (x_{i,j})^2 + (x_{i,j+1})^2 + (x_{i,j})^2 \le 8\sum_{i,j}(x_{i,j})^2 = 8\|x\|_2^2. \tag{2.88}$$

Each interior pixel $x_{i,j}$ appears at most twice in the vertical and the horizontal differences, so that gives the factor of 4. The pixels at the boundary appear less often than that. Combining those two estimates yields the following upper bound for the operator norm of D:

$$\sup_{x\neq 0}\frac{\|Ax\|_2}{\|x\|_2}=\sup_{x\neq 0}\frac{\|\lambda Dx\|_2}{\|x\|_2}\leq\lambda\sup_{x\neq 0}\frac{\|Dx\|_2}{\|x\|_2}\leq\lambda\sqrt{8}=L. \tag{2.89}$$

A checkerboard pattern $x_{i,j}=(-1)^{i+j}$ shows that this estimate is sharp, up to boundary effects. We have $\|x\|_2=\sqrt{NM}$ and $\|Dx\|_2=\sqrt{8NM}$. In practice, the exact constant L is not crucial, but a tight estimate allows for bigger step sizes. Having established the scaling of L for the discrete gradient operator, we could use it for the classical Arrow-Hurwicz method. However, we instead adopt a primal-dual method inspired by the Arrow-Hurwicz method, which utilizes the uniform convexity of f. In the following, $\gamma>0$ denotes the modulus of uniform convexity of f. Specifically, γ is a constant such that the following inequality holds:

$$f(x')\geq f(x)+\langle p,x'-x\rangle+\frac{\gamma}{2}\|x-x'\|_2^2,\quad\forall p\in\partial f(x)\;\forall x,x'\in X. \tag{2.90}$$

In case of $f(x)=\frac{1}{2}\|x-x^\delta\|_2^2$, we have $\gamma=1$.

This algorithm was initially proposed by Chambolle and Pock in [9, p. 15] and later improved in [10, p. 12].

Algorithm 1 Primal-Dual Iterative Scheme

1: **Initialization:** Choose $\theta_0,\tau_0,\sigma_0>0$, $(x^0,y^0)\in X\times Y$, and set $\hat{x}^0=x^0$
2: **for** $k=0,1,2,\ldots$ **do**
3: $\quad y^{k+1}=\operatorname{prox}_{\tau_k}^{g^*}(y^k+\tau_k A\hat{x}^k)$
4: $\quad x^{k+1}=\operatorname{prox}_{\sigma_k}^{f}(x^k-\sigma_k A^*y^{k+1})$
5: $\quad \theta_k=1/\sqrt{1+\gamma\sigma_k}$
6: $\quad \sigma_{k+1}=\theta_k\sigma_k$
7: $\quad \tau_{k+1}=\tau_k/\theta_k$
8: $\quad \hat{x}^{k+1}=x^{k+1}+\theta_k(x^{k+1}-x^k)$
9: **end for**

This algorithm still uses proximal steps alternating on the primal and dual variables, but it features variable step sizes and an extrapolation of the primal variable.

Line 3 performs the dual update, where instead of x, its extrapolated counterpart $\hat{x}$ is used for the gradient ascent. In contrast, line 4 performs the primal update without extrapolation. The extrapolation factor θ is updated in line 5, where σ denotes the primal step size.

The primal and dual step sizes are updated in lines 6 and 7 in such a way that $\sigma\xrightarrow{k\to\infty}0$ and $\tau\xrightarrow{k\to\infty}+\infty$, since $\theta\in(0,1)$. The last update in the loop is line 8, which uses θ times the change of x between this and the previous iteration as the extrapolation step.

If strong duality holds according to Theorem 6, then the objective in form of the primal-dual gap converges to zero at a rate of $\mathcal{O}(1/k^2)$ [10]. In the next chapter we will

show that the primal-dual gap of the denoising problem

$$G(x,y) = \frac{1}{2}\|x - x^\delta\|_2^2 + \lambda\|Dx\|_{1,2} + \frac{1}{2}\|x^\delta - \lambda D^* y\|_2^2 - \frac{1}{2}\|x^\delta\|_2^2 + I_{\{\|\cdot\|_{1,2}\leq 1\}}(y)$$

is always greater than or equal to $\frac{1}{2}\|x - \bar{x}\|_2^2$, where x is the current iterate and $\bar{x}$ is the solution. Thus, a typical stopping criterion is $G(x,y) < \varepsilon$ for some $\varepsilon > 0$.

An image with a mean squared error of 10^{-4} is usually visually indistinguishable from the solution, provided the images are scaled to the interval $[0,1]$. Therefore, choosing

$$\varepsilon = 10^{-4}\left(\frac{NM}{2}\right)$$

where NM is the number of pixels, is a reasonable stopping criterion.

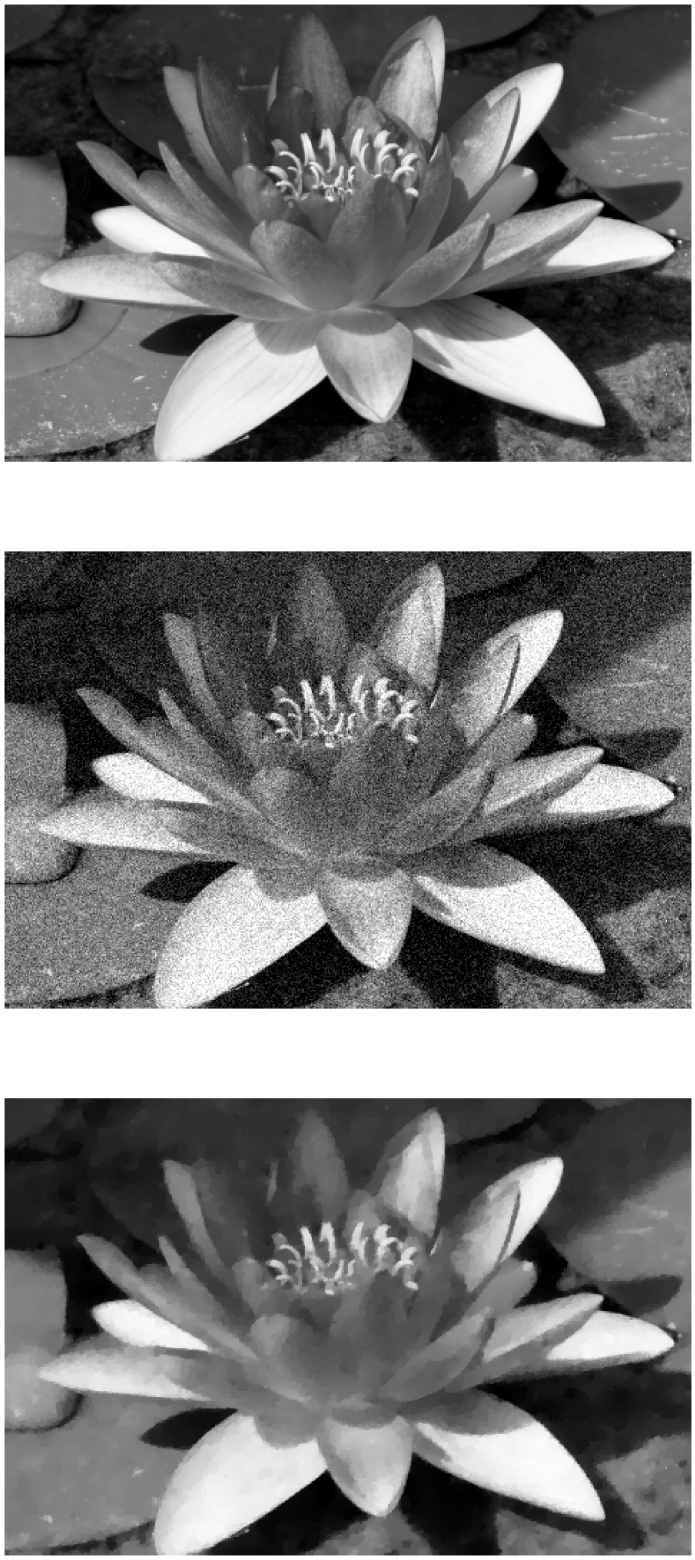
The requirements are that f and g are proper, convex, and lower semi-continuous, with either f or g^* being uniformly convex with modulus $\gamma > 0$. The initial step sizes σ_0, τ_0 must be chosen, such that they are positive and $\sigma_0\tau_0\|A\|^2 < 1$. In [9, p. 17], the recommendation is to pick $\sigma_0\gamma$ significantly larger than $\|\bar{x} - x^0\|_2$ and $\tau_0 = 1/(\sigma_0 L^2)$, where x^0 is the initial guess for the solution $\bar{x}$.

The quantity $\|\bar{x} - x^0\|_2$ is generally not known until convergence, so recommending a value for the initialization based on that seems odd, however we can obtain a good enough estimate by replacing the solution $\bar{x}$ by the ground truth $x^\dagger$ and setting the initialization x^0 to x^δ. This is useful, because we modeled x^δ as $x^\dagger + \eta$, where $\eta \sim \mathcal{N}(0, \delta^2 I_{NM})$. In conclusion, $\|\bar{x} - x^0\|_2 \approx \mathbb{E}(\|x^\dagger - x^\delta\|_2) = \mathbb{E}(\|\eta\|_2) = \delta\sqrt{NM}$ is a more practical starting point for choosing σ_0.

Applying all those observations about the total variation problem to the general primal-dual iterative scheme yields the following algorithm. It takes as input the noisy image x^δ (with noise level $\delta > 0$) and the regularization parameter $\lambda > 0$, and returns the denoised image.

Algorithm 2 Total variation denoising

Initialization: Set $L \leftarrow \sqrt{8}, \sigma \leftarrow \delta\sqrt{NM}, \tau \leftarrow (\sigma L^2)^{-1}$, $(x,y) \leftarrow (x^\delta, 0)$, $\hat{x} \leftarrow x^\delta$
while $\frac{1}{2}\|x - x^\delta\|_2^2 + \lambda\|Dx\|_{1,2} + \frac{1}{2}\|x^\delta - \lambda D^* y\|_2^2 - \frac{1}{2}\|x^\delta\|_2^2 \geq \varepsilon$ **do**
    $y \leftarrow \mathrm{proj}_{\{\|\cdot\|_{1,2}\leq 1\}}(y + \tau\lambda D\hat{x})$
    $s \leftarrow (x - \sigma\lambda D^* y + \sigma x^\delta)/(1+\sigma) - x$
    $x \leftarrow x + s$
    $\theta \leftarrow 1/\sqrt{1+\sigma}$
    $\sigma \leftarrow \sigma\theta$
    $\tau \leftarrow \tau/\theta$
    $\hat{x} \leftarrow x + \theta s$
end while
return x



Isotropic total variation denoising applied to an image of a water lily. The resolution of the grayscale image is 480×320 pixels with values scaled to the interval $[0, 1]$. The ground truth (top) was corrupted by Gaussian noise with standard deviation 0.1, this corresponds to a PSNR of 20dB (middle). The denoising parameter was set to $\lambda = 0.1$ for a reasonable balance between noise removal and details. The PSNR of the denoised image is 29,71dB (bottom). It took 40 iterations until the primal-dual gap dropped below the tolerance ε chosen as $10^{-4}(\frac{NM}{2})$.

Figure 4: Picture of a water lily from a botanical garden in Bologna.

Figure 4 shows TV denoising applied to an image of a water lily. The image features both flat regions with sharp corners along the petals and unordered fine particles floating in the water. The edges are preserved in the denoising process, but some of the finer details like the stripes on the petals are washed out. The fine particles in the water have merged locally to bigger blobs and became almost unrecognizable in the process.

2.5 Measuring reconstruction quality

The quality of an image restoration method is typically assessed by comparing the reconstructed image $\bar{x}$ to the ground truth image $x^\dagger$. Several quantitative metrics have been established for this purpose, the most common being the mean squared error (MSE), the peak signal-to-noise ratio (PSNR), and the structural similarity index (SSIM). Each metric captures a different notion of "closeness" between images.

The mean squared error is the most fundamental and mathematically motivated measure. It is defined as the squared euclidean distance between the images, normalized by the number of pixels NM:

$$\mathrm{MSE}(\bar{x}, x^\dagger) = \frac{1}{NM}\|\bar{x} - x^\dagger\|_2^2 = \frac{1}{NM}\sum_{i,j}(\bar{x}_{i,j} - x^\dagger_{i,j})^2. \tag{2.91}$$

It arises naturally from least-squares estimation and is directly related to the ℓ^2-norm used in many variational formulations. Although it provides a clear and mathematically interpretable error measure, it correlates poorly with human perception of visual quality.

The PSNR is derived directly from the MSE, but expressed on a logarithmic decibel scale. It is defined as

$$\mathrm{PSNR}(\bar{x}, x^\dagger) = 10\log_{10}\left(\frac{\mathrm{M}^2}{\mathrm{MSE}(\bar{x}, x^\dagger)}\right), \tag{2.92}$$

where M denotes the maximum possible pixel value (e.g., 255 for 8-bit images). While PSNR does not introduce new information compared to MSE, it rescales the error into a range that has a more intuitive interpretation from an information-theoretic perspective: higher PSNR values correspond to higher signal fidelity relative to the background noise. Using either PSNR or MSE as an optimization objective makes no difference for a single image, since the two are monotonically related. The relation is inversely proportional, a small MSE corresponds to a big PSNR and vice versa. However, when averaging over multiple images, the logarithmic transformation causes PSNR to weigh low-error images more heavily than high-error ones.

Both MSE and PSNR measure pixel-wise deviations without accounting for the spatial structure of the image. The Structural Similarity Index (SSIM), introduced by Wang et al. [34], is based on the observation that the human visual system is highly sensitive to changes in local luminance, contrast, and structure rather than absolute pixel errors. Let $\mu : \mathbb{R}^{N\times M} \times \{1,\dots,N\} \times \{1,\dots,M\} \to \mathbb{R}$ be a function that measures the local mean of an image x centered around the pixel at position (i, j).

Similarly let $\sigma : \mathbb{R}^{N\times M} \times \mathbb{R}^{N\times M} \times \{1,\dots,N\} \times \{1,\dots,M\} \to \mathbb{R}$ be a function that measures the local covariance between two images x and y at pixel position (i,j).

The structural similarity of two images x and y at position (i,j) is defined as:

$$\mathrm{SSIM}(x,y,i,j) := \frac{(2\mu(x,i,j)\mu(y,i,j)+C_1)(2\sigma(x,y,i,j)+C_2)}{(\mu(x,i,j)^2+\mu(y,i,j)^2+C_1)(\sigma(x,x,i,j)+\sigma(y,y,i,j)+C_2)}. \tag{2.93}$$

The constants C_1, C_2 stabilize the division and are chosen as $C_1 = 0.01$ and $C_2 = 0.03$ in the original paper [34]. The local statistics use a 11×11 circularly symmetric Gaussian weights $\{w_{k,l}\}$ with standard deviation of 1.5, normalized to unit sum. The 11×11 grid is centered around zero, i.e. $\{-5,\dots,5\}\times\{-5,\dots,5\}$ and the weights on the grid position (i,j) are given by:

$$\tilde{w}_{i,j} = \exp\left(-\frac{i^2+j^2}{2\cdot(1.5)^2}\right), \tag{2.94}$$

$$w_{i,j} = \frac{\tilde{w}_{i,j}}{\sum_{k=-5}^{5}\sum_{l=-5}^{5}\tilde{w}_{k,l}}. \tag{2.95}$$

The weighted sum for the local mean of an image x is:

$$\mu(x,i,j) = \sum_{k=-5}^{5}\sum_{l=-5}^{5} w_{k,l}x_{i+k,j+l}, \tag{2.96}$$

where the image x is extended by reflection at the boundary for statistics close to the boundary. The weighted local covariance is defined as:

$$\sigma(x,y,i,j) = \sum_{k=-5}^{5}\sum_{l=-5}^{5} w_{k,l}(x_{i+k,j+l}-\mu(x,i+k,j+l))(y_{i+k,j+l}-\mu(y,i+k,j+l)). \tag{2.97}$$

The SSIM is defined pixel-wise, however confusingly, the same name is commonly used for its spatial mean given by:

$$\mathrm{SSIM}(x,y) = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\mathrm{SSIM}(x,y,i,j). \tag{2.98}$$

Unlike MSE and PSNR, SSIM captures perceptual quality by combining luminance, contrast, and structure comparisons. Its values range from -1 to 1, where 1 indicates perfect similarity.

Whilst MSE, PSNR and SSIM are certainly the most commonly used metrics for measuring the differences and similarities between two images, they are not the only sensible objectives for parameter fine tuning in the context of supervised machine learning.

2.6 Eliminating the Bi-Level Optimization Structure

The main downside of the SSIM is its complexity. The convoluted way of computing the score is not a problem, while we are dealing with only one parameter, but it is unfit for fine tuning denoising models with significantly more degrees of freedom. Even optimizing the MSE can be challenging, which motivates changing the objective function to something that is closer connected to the denoising process.

If the denoised image $\bar{x}$ is close to the ground truth $x^\dagger$, then the non-negative difference between the objectives

$$f(x^\dagger) + g(Ax^\dagger) - f(\bar{x}) - g(\bar{x}) = f(x^\dagger) + g(Ax^\dagger) - \left(\min_{x \in \mathbb{R}^{N \times M}} f(x) + g(Ax) \right) \geq 0 \quad (2.99)$$

is also small by the continuity of f and g. Crucially, we are going to show that the inverse implication is also true. Actually, we already used the reverse implication for determining the stopping criterion of the denoising algorithm. If the difference between the primal objective of the ground truth and the primal objective of the denoised image, from here on simply *primal objective gap (POG)*, is small then this implies that $\bar{x}$ is close to $x^\dagger$ and thus a good reconstruction.

Minimizing the POG in dependence of the parameter λ in the operator $A = \lambda D$ is in this form impractical, because the solution $\bar{x}$ depends directly on A. This issue only becomes worse, when later we take the whole operator A as a parameter with high degrees of freedom. However, we can decouple the solution $\bar{x}$ from the POG by utilizing the primal-dual gap:

$$G(x, y) = f(x) + g(Ax) + f^*(-A^*y) + g^*(y), \quad (2.100)$$

and this gap is zero at the point $(\bar{x}, \bar{y})$ by Theorem 6.

The primal-dual gap can be introduced into the POG term by adding and subtracting the optimal dual objective:

$$\begin{aligned}
\mathrm{POG}(x^\dagger, \bar{x}) &= f(x^\dagger) + g(Ax^\dagger) - f(\bar{x}) - g(\bar{x}) && (2.101)\\
&= f(x^\dagger) + g(Ax^\dagger) + f^*(-A^*\bar{y}) + g^*(\bar{y}) && (2.102)\\
&\underbrace{-f(\bar{x}) - g(\bar{x}) - f^*(-A^*\bar{y}) - g^*(\bar{y})}_{=0} && (2.103)\\
&= f(x^\dagger) + g(Ax^\dagger) + \min_{y \in \mathbb{R}^{N \times M \times 2}} f^*(-A^*y) + g^*(y) && (2.104)\\
&= \min_{y \in \mathbb{R}^{N \times M \times 2}} G(x^\dagger, y). && (2.105)
\end{aligned}$$

Thus instead of solving the bi-level problem

$$\min_A \mathrm{POG}(x^\dagger, \bar{x}), \text{ s.t. } \bar{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^{N \times M}} f(x) + g(Ax), \quad (2.106)$$

one can instead solve the mono-level problem

$$\min_{A} \min_{y \in \mathbb{R}^{N \times M \times 2}} f(x^\dagger) + g(Ax^\dagger) + f^*(-A^*y) + g^*(y). \tag{2.107}$$

This is, in contrast to the bi-level problems, a single level optimization scheme. Instead of calculating a denoised image $\bar{x}$ or equivalently $\bar{y}$ for every possible parameter A, we can update A and y at the same time. This idea to use the primal-dual gap in order to avoid bi-level optimization schemes has already been used for example by Chenchene et al. [14]. The paper unfortunately does not contain the proof, but they used Bregman divergences to show that minimizing the POG ensures the "closeness" between the denoised image $\bar{x}$ and the ground truth $x^\dagger$.

Definition 10 (Bregman divergence)**.** *[6]*
Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper and convex, $x, x' \in dom(f)$ and $p \in \partial f(x')$. Then, the Bregman divergence $D_p^f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined as

$$D_p^f(x, x') = f(x) - f(x') - \langle p, x - x' \rangle. \tag{2.108}$$

The Bregman divergence is always non-negative by Definition 4. One could also define the Bregman divergence for non-convex differentiable functions, using the gradient instead of the subgradient. However, this would also change some properties of the Bregman divergence. In the case of non-convex functions, a gradient is not necessarily also a subgradient and therefore, the Bregman divergence could also be negative.

Bregman divergences can be used to rephrase the primal-dual gap in another useful form.

Corollary 11 (Duality gap in terms of Bregman divergences)**.** *In the situation of Theorem 6, one has*

$$G(x, y) = f(x) + g(Ax) + f^*(-A^*y) + g^*(y) \tag{2.109}$$

$$= D^f_{-A^*\bar{y}}(x, \bar{x}) + D^g_{\bar{y}}(Ax, A\bar{x}) + D^{f^*}_{\bar{x}}(-A^*y, -A^*\bar{y}) + D^{g^*}_{A\bar{x}}(y, \bar{y}). \tag{2.110}$$

In particular, this implies $D^f_{-A^*\bar{y}}(x, \bar{x}) \leq G(x, y)$.

Proof. First of all, Theorem 6 directly implies that $G(\bar{x}, \bar{y}) = 0$. Further, the Bregman divergences are well defined, because of Theorem 8 and

- $\bar{x} \in \partial f^*(-A^*\bar{y})$,
- $A\bar{x} \in \partial g^*(\bar{y})$.

The rest of the proof consists of just plugging in the definitions and double checking that the unwanted terms indeed cancel:

$$
\begin{aligned}
&D^f_{-A^*\bar{y}}(x,\bar{x}) + D^g_{\bar{y}}(Ax, A\bar{x}) + D^{f^*}_{\bar{x}}(-A^*y, -A^*\bar{y}) + D^{g^*}_{A\bar{x}}(y,\bar{y}) && (2.111)\\
&\quad = f(x) - f(\bar{x}) - \langle -A^*\bar{y}, x - \bar{x}\rangle && (2.112)\\
&\qquad + g(Ax) - g(A\bar{x}) - \langle \bar{y}, Ax - A\bar{x}\rangle && (2.113)\\
&\qquad + f^*(-A^*y) - f^*(-A\bar{y}) - \langle \bar{x}, -A^*y + A^*\bar{y}\rangle && (2.114)\\
&\qquad + g^*(y) - g^*(\bar{y}) - \langle A\bar{x}, y - \bar{y}\rangle && (2.115)\\
&\quad = G(x,y) - G(\bar{x},\bar{y}) + \langle A^*\bar{y} - A^*\bar{y}, x - \bar{x}\rangle + \langle A\bar{x} - A\bar{x}, y - \bar{y}\rangle && (2.116)\\
&\quad = G(x,y). && (2.117)
\end{aligned}
$$

All the terms in equation (2.111) are non-negative, therefore the sum gets smaller if terms are omitted. Thus, the inequality $D^f_{-A^*\bar{y}}(x,\bar{x}) \leq G(x,y)$ follows immediately.

□

The Bregman divergence $D^f_p(x,x')$ quantifies how much the function f deviates at x from its first-order (linear) approximation taken at x'. It is always nonnegative but generally not symmetric, i.e. $D^f_p(x,x') \neq D^f_{p'}(x',x)$, and does not satisfy the triangle inequality. Also, while $x = x' \Rightarrow D^f_p(x,x') = 0$ holds, the reverse implication is not true. Hence, Bregman divergences are not metrics in the mathematical sense, though, interestingly, the same is true for perceptual measures such as the SSIM [34].

The Bregman divergence of $f(x) = \frac{1}{2}\|x - x^\delta\|_2^2$ has a particularly nice form:

$$
\begin{aligned}
D^f_{\nabla f}(x^\dagger, \bar{x}) =& \frac{1}{2}\|x^\dagger - x^\delta\|_2^2 - \frac{1}{2}\|\bar{x} - x^\delta\|_2^2 - \langle \bar{x} - x^\delta, x^\dagger - \bar{x}\rangle && (2.118)\\
=& \frac{1}{2}\|x^\dagger\|_2^2 - \langle x^\dagger, x^\delta\rangle + \frac{1}{2}\|x^\delta\|_2^2 && (2.119)\\
&- \frac{1}{2}\|\bar{x}\|_2^2 + \langle \bar{x}, x^\delta\rangle - \frac{1}{2}\|x^\delta\|_2^2\\
&+ \langle x^\dagger, x^\delta\rangle - \langle \bar{x}, x^\delta\rangle - \langle x^\dagger, \bar{x}\rangle + \|\bar{x}\|_2^2\\
=& \frac{1}{2}\|x^\dagger\|_2^2 - \langle x^\dagger, \bar{x}\rangle + \frac{1}{2}\|\bar{x}\|_2^2 && (2.120)\\
=& \frac{1}{2}\|x^\dagger - \bar{x}\|_2^2. && (2.121)
\end{aligned}
$$

This means that $D^f_{\nabla f}(x^\dagger, \bar{x})$ is proportional to the MSE, up to the factor $\frac{2}{NM}$. The Bregman divergence of $g(Ax) = \|Ax\|_{1,2}$ does not admit such a simple closed form. However, we can use the optimality condition $\bar{x} = x^\delta - A^*\bar{y}$, established in (2.74), to eliminate $\bar{y}$:

$$
\begin{aligned}
D^g_{\bar{y}}(Ax^\dagger, A\bar{x}) =& \|Ax^\dagger\|_{1,2} - \|A\bar{x}\|_{1,2} - \langle \bar{y}, Ax^\dagger - A\bar{x}\rangle && (2.122)\\
=& \|Ax^\dagger\|_{1,2} - \|A\bar{x}\|_{1,2} - \langle A^*\bar{y}, x^\dagger - \bar{x}\rangle && (2.123)\\
=& \lambda(\|Dx^\dagger\|_{1,2} - \|D\bar{x}\|_{1,2}) - \langle x^\delta - \bar{x}, x^\dagger - \bar{x}\rangle. && (2.124)
\end{aligned}
$$

The function $\bar{x} \mapsto D_{\tilde{y}}^{g}(Ax^{\dagger}, A\bar{x})$ is a somewhat unusual objective function, because it depends explicitly on the parameter λ and only works for the solution $\bar{x}$ and not any x. Optimizing with respect to λ, while the same parameter simultaneously alters the objective function may appear circular at first glance, but the function still has an interpretable structure.
The term $\lambda(\|Dx^{\dagger}\|_{1,2} - \|D\bar{x}\|_{1,2})$ measures the difference in total variation between the ground truth $x^{\dagger}$ and the reconstruction $\bar{x}$, scaled by λ. The term $-\langle x^{\delta} - \bar{x}, x^{\dagger} - \bar{x}\rangle$ serves as a correction ensuring that D^{g} remains non-negative and vanishes when $x^{\dagger} = \bar{x}$.

The objective $\frac{1}{2}\|x^{\dagger} - \bar{x}\|_2^2 + \lambda(\|Dx^{\dagger}\|_{1,2} - \|D\bar{x}\|_{1,2}) - \langle x^{\delta} - \bar{x}, x^{\dagger} - \bar{x}\rangle$ combines a pixel-wise discrepancy term and a total variation discrepancy. For small λ the objective is almost proportional to the MSE, while for bigger λ the priority shifts to minimizing the Bregman divergence of the total variation $D_{\tilde{y}}^{g}(Ax^{\dagger}, A\bar{x})$.

In summary, the primal objective gap $(f + g \circ A)(x^{\dagger}) - (f + g \circ A)(\bar{x})$ (POG) pursues a smaller λ than the MSE or PSNR. Which one of these two criteria is preferable is up to debate, because neither MSE or POG are actually based on the human visual system, but rather purely mathematically motivated. The major advantage of POG is its connection to the primal-dual gap, which means that a bi-level optimization scheme can be avoided.

We will conclude this chapter with a small empirical test on a single image corrupted by Gaussian noise with $\sigma = 0.1$, which can be seen in Figure 5. The goal here is, to determine the optimal parameter λ for isotropic total variation denoising with respect visual inspection and then by the four different metrics MSE, PSNR, SSIM and POG.

The side by side comparison in Figure 5 shows the ground truth, the noisy image and four denoised images with $\lambda \in \{0.05, 0.10, 0.15, 0.20\}$. The image with $\lambda = 0.05$ remains visibly noisy, with the noise exhibiting a degree of correlation across pixels (unlike the original noise). This correlation negatively impacts the appearance of metallic surfaces, dulling their specular highlights. However, this setting fails to adequately remove the noise, making it the worst performing parameter. Increasing λ to 0.10 reduces the noise to an acceptable level, though the flat surfaces (e.g., the background) still exhibit some textural artifacts or residual noise.

The two higher settings, $\lambda = 0.15$ and $\lambda = 0.20$, perform best for this image. This can be explained by the relatively small total variation of the test image and thus denoising by minimizing said total variation is highly effective. There is virtually no visual difference between the results for $\lambda = 0.15$ and $\lambda = 0.20$, indicating a plateau in performance where further increases in λ do not yield significant improvements in noise reduction, resulting in a visual tie between these two parameter values.

The scores given by PSNR, MSE, SSIM and POG with respect to λ are plotted in Figure 6. The PSNR is the MSE on a logarithmic scale, thus both have the same optimal value of $\lambda = 0.13$. While the MSE is almost flat for values ranging from 0.1 to 0.2, the PSNR amplifies those small changes and clearly peaks at $\lambda = 0.13$.

Meanwhile, the SSIM starts to plateau somewhere around $\lambda = 0.09$ and has an almost imperceptible peak at $\lambda = 0.16$.

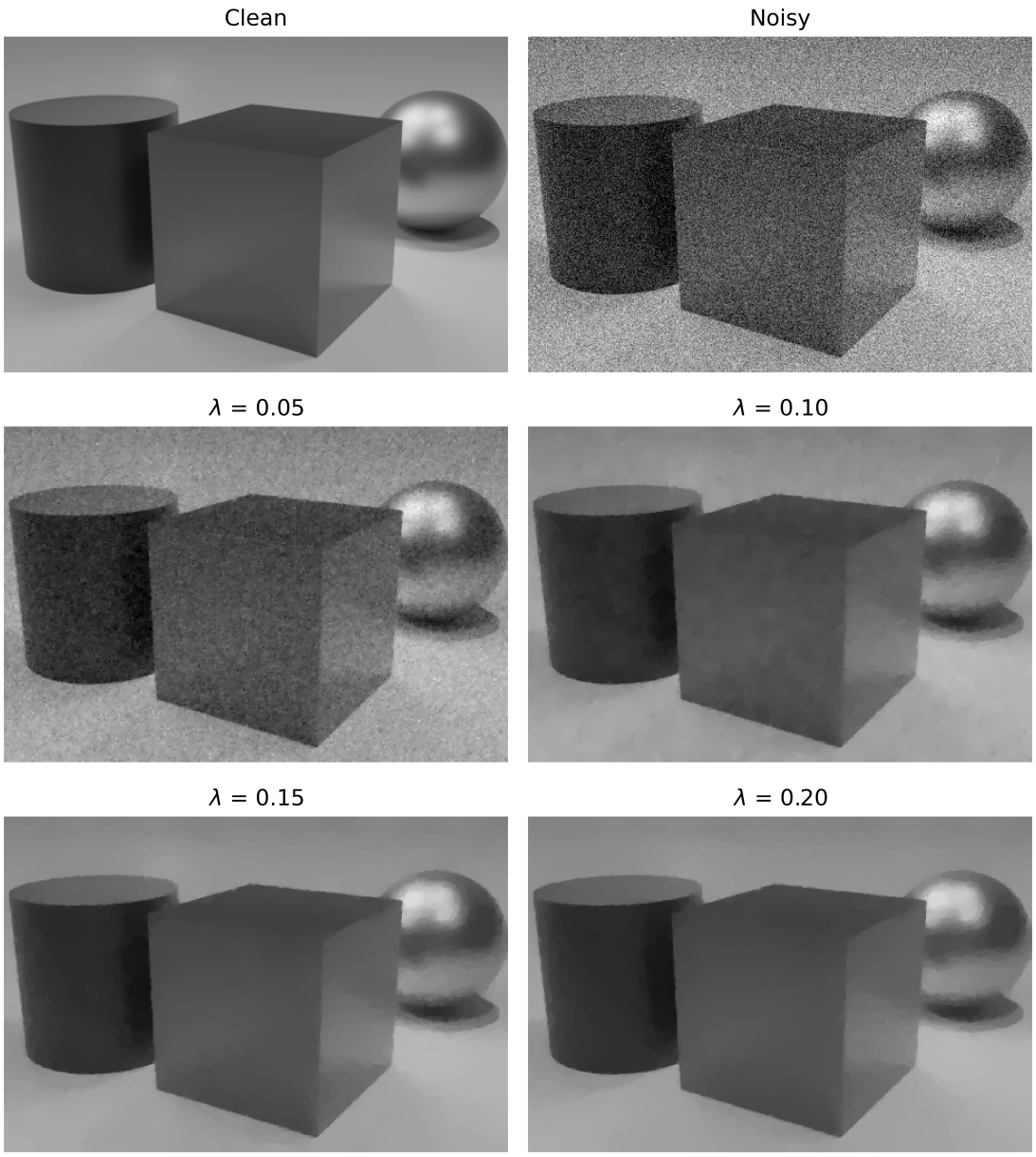


Figure 5: This image features three primitives rendered in Blender. The grayscale image has a resolution of 480×320 pixels. The noise follows a Gaussian distribution with $\sigma = 0.1$. The denoising parameter λ ranges from 0.05 to 0.2. This image is an almost ideal use case for total variation denoising, because it consists mostly smooth textures separated by sharp edges and almost no details and thus a rather aggressive denoising parameter of $\lambda = 0.15$ or $\lambda = 0.2$ is ideal.

The graph for POG was rescaled by $\frac{2}{NM}$ in order to make it comparable to the MSE. For small λ the curve is basically identical to the MSE, but for bigger λ, the additional term $D_{\tilde{y}}^{g}(Ax^{\dagger}, A\bar{x})$ becomes noticeable. This results in the overall smallest recommended parameter of only $\lambda = 0.09$. By visually inspecting the results in Figure 5 this value would not align with the personal preference and seems rather small. This shows, that the POG is rather mathematically motivated and not based on the human visual system.

However, since this bias towards smaller λ, than the MSE or SSIM would suggest, this can be compensated for. A simple way to compensate for this underestimation would be to multiply the optimal λ given by POG with a fixed value of e.g. 1.3. This enables us to use the POG as a more convenient objective to optimize for than the other metrics without a compromise in quality.

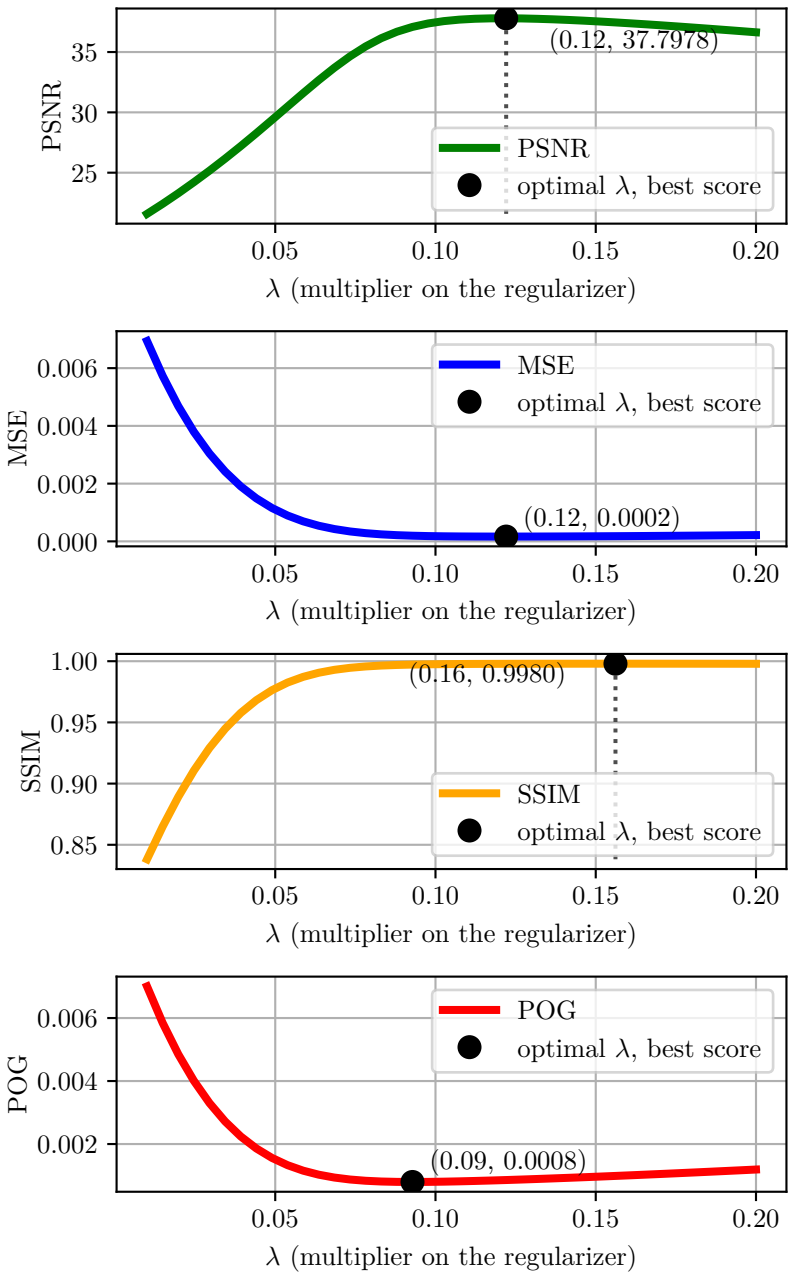


Figure 6: Scores for denoising the image from Figure 5 with $\lambda \in [0.01, 2]$.

3 Training linear filters for the denoising problem

The classical total variation (TV) denoising model relies on the discrete gradient operator D to penalize spatial variations in the image. In its discrete form, D consists of horizontal and vertical difference operators, scaled by a single regularization parameter λ. This formulation is highly rigid, as there is only a single tuneable parameter.

There is a vast design space inherent to the more general co-sparse analysis model of the form:

$$\min_{x \in \mathbb{R}^{N \times M}} \frac{1}{2}\|x^\delta - x\|_2^2 + \phi(Ax). \tag{3.1}$$

The degrees of freedom in this model can be expanded through several distinct approaches. First, the fixed linear operator A can be generalized. For instance, the static finite differences of the TV model can be replaced by a learnable matrix or convolutional kernels. Second, the sparsity-promoting function ϕ can be augmented by learnable parameters. Rather than a simple ℓ_1 norm, one could employ structured group-sparsity penalties, such as a mixed $\ell^{p,q}$-norm, where p and q are adjustable. Finally, the architecture can be further enriched by non-convex functions and introducing learnable weights to balance different components of the regularization term. Ultimately, the interplay between the choice of the operator A, the structure of ϕ, and the optimization strategy determines the representational power of the model.

In this chapter, we investigate a specific configuration of this framework: an $\ell^{1,2}$-norm sparsity-promoting function paired with a set of learnable convolutional filters. This approach can be viewed as a specialized instance of the "Fields of Experts" (FoE) framework [26], providing an established benchmark against which our proposed training algorithm can be evaluated.

3.1 Linear Image Filters

In the following, we clarify several notational and conceptual conventions regarding linear filters and discrete convolutions. The terms filter and convolution are here used interchangeably, but in some image processing literature, they are not identical. Depending on the chosen convention, the convolution operation may or may not include a flipping of the kernel. While this distinction is often inconsequential when one consistently follows a single definition, it can influence theoretical properties such as commutativity, associativity and the expression of adjoint operators. For the sake of precision, we make all definitions explicit below.

The classical definition for discrete convolution requires an unbounded domain.

Definition 12. *Discrete Convolution*
Let $x : \mathbb{Z}^2 \to \mathbb{R}$ be an image and $K : \mathbb{Z}^2 \to \mathbb{R}$ a kernel with compact support defined on the full integer lattice. The (discrete) convolution of K and x is defined by

$$(K * x)_{i,j} = \sum_{(u,v)\in\mathbb{Z}^2} K_{u,v} x_{i-u,j-v}. \tag{3.2}$$

This corresponds to the standard definition of convolution, including kernel flipping. The sum in the definition is actually finite, because of the compact support of the kernel. The correlation operator denoted by $\star$ does not flip the kernel and can be obtained by replacing $K_{u,v}$ with $K_{-u,-v}$ in (3.2). Differentiating between convolution and correlation is important, because convolution is commutative and associative, while correlation is neither. Furthermore, the correlation operator is the adjoint to the convolution operator. This can be seen, by using the commutativity of the convolution and rearranging the order of summation as shown here:

$$\begin{aligned}
\langle K * x, y\rangle &= \sum_{i,j}\left(\sum_{u,v} K_{u,v} x_{i-u,j-v}\right) y_{i,j} = \sum_{i,j}\left(\sum_{u,v} K_{i-u,j-v} x_{u,v}\right) y_{i,j} && (3.3)\\
&= \sum_{u,v}\left(\sum_{i,j} K_{i-u,j-v} y_{i,j}\right) x_{u,v} = \sum_{u,v}\left(\sum_{i,j} K_{-i,-j} y_{u-i,v-j}\right) x_{u,v} && (3.4)\\
&= \langle x, K \star y\rangle = \langle x, \tilde{K} * y\rangle, && (3.5)
\end{aligned}$$

where $\tilde{K}$ denotes the spatially flipped version of K.

In practical applications, the image x and kernel K are defined on finite rectangular grids, i.e. $x : \{0, ..., N-1\} \times \{0, ..., M-1\} \to \mathbb{R}$ and $K : \{0, ..., n-1\} \times \{0, ..., m-1\}$ which is identified with $x \in \mathbb{R}^{N\times M}$ and $K \in \mathbb{R}^{n\times m}$. In most textbook literature such as [20] only kernels with odd size are considered, because they can be centered around zero and the notation simplifies considerably. Definition 12 can be adapted to the finite setting by restricting the indices to $n-1 \leq i < N$ and $m-1 \leq j < M$, since x can not be evaluated at indices outside the valid range. The filtered image is therefore $n-1$ pixels in height and $m-1$ pixels in width smaller than the input. This has also implications for the adjoint operator, because it has to map back to the original space by extending the image by $n-1$ pixels vertically and $m-1$ pixels horizontally. The adjoint of the convolution in the finite case can therefore no longer be represented by convolving with the flipped kernel. Instead of inventing new definitions for convolution and correlation in the finite case it is preferable to use extensions to the unbounded domain and then restricting the convolved image back to the valid range.

Commonly used extensions (definitions adapted from [5, p. 78]) from a finite grid $G = \{0, ..., N-1\} \times \{0, ..., M-1\}$ to the whole plane $E : \ell^\infty(G) \to \ell^\infty(\mathbb{Z}^2)$ are:

- **Zero extension:** set all values to zero outside the domain.
  $(E_0 x)_{i,j} = 0$ for all $(i, j) \notin G$.

- **Constant extension:** repeat the same grayscale value of the boundary.
  $(E_C x)_{i,j} = x_{\sigma(i,j)}$ with $\sigma(i,j) = (\max\{\min\{N-1, i\}, 0\}, \max\{\min\{M-1, j\}, 0\})$.
- **Periodic extension:** tile the plane with copies of the original image.
  $(E_P x)_{i,j} = x_{\tau(i,j)}$ with $\tau(i,j) = (i \mod N, j \mod M)$.
- **Reflective extension:** mirror the image vertically and horizontally and then tile the plane with copies.
  $(E_R x)_{i,j} = x_{\pi(\varphi(i,j))}$ with $\pi(i,j) = (\min\{i, 2N-1-i\}, \min\{j, 2M-1-j\})$ and $\varphi(i,j) = (i \mod 2N, j \mod 2M)$

The restriction operator R_p is the linear map that extracts the values of an image defined on $\mathbb{Z}^2$ and returns only those on a finite grid. The restriction depends on the grid for the image $\{0, ..., N-1\} \times \{0, ..., M-1\}$, the grid for the kernel $\{0, ..., n-1\} \times \{0, ..., m-1\}$ and four non-negative parameters $p = (p_u, p_d, p_l, p_r) \in \{0, 1, 2, ...\}^4$ representing additional rows/columns above, below, to the left and to the right of the original domain. By convention, if only a single padding parameter is specified, it is applied uniformly to all sides.

Consequently, the restriction operator is given by:

$$R_p : \mathbb{Z}^2 \to \{n-1-p_u, ..., N-1+p_d\} \times \{m-1-p_l, ..., M-1+p_r\}. \tag{3.6}$$

The operation $K * x$ for an image x and a kernel K on finite grids is defined by the composition of a zero extension on the kernel, a reflective extension on the image, the convolution as in Definition 12 followed by a restriction to the finite grid:

$$K * x := R_p(E_0 K * E_R x). \tag{3.7}$$

The reflection extension is the most complicated one, but also produces the least visual artifacts.

The importance of choosing the correct padding for the restriction operator depends on the image size $N \times M$ and the kernel size $n \times m$. The boundary effects are negligible, when working with high resolution images and small kernels, but they become visible in small images. Figure 7 shows four low resolution images of a glider with only 64×64 pixels. The first image is the original and the second image is corrupted by additive Gaussian noise with $\sigma = 0.05$. The third and fourth image are denoised with the method presented later in this chapter. 64 filters of size 8×8 were trained on 100 copies of the image, each corrupted with Gaussian noise. The third and fourth image show the result of the denoising process with 4 and with 0 pixels of reflective padding in the convolution process. The image without padding has noisy pixels close to the edge, while the padded image handles the boundary properly. Notably, because the filters were trained on a single image, artifacts in the form of streaks (corresponding to the diagonal wings) appear in the background. Furthermore, as the evaluation is performed on the training data itself, these results should be interpreted as a theoretical upper bound on performance rather than a measure of generalizable accuracy for real-world applications.

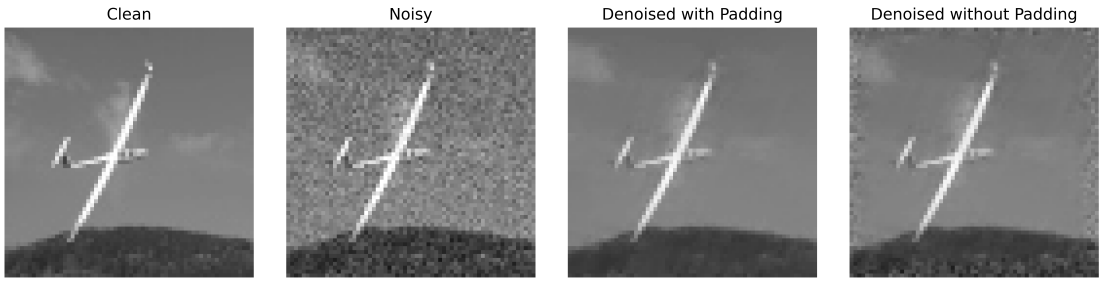


Figure 7: Comparison of convolution padding strategies for the regularizer. A clean 64×64 image is corrupted with additive Gaussian noise ($\sigma = 0.05$). The denoising method, utilizing 64 kernels of size 8×8, is evaluated with and without padding. The absence of padding results in visible noise artifacts at the image boundaries.

We now generalize from a single filter to a set of filters $\mathcal{K} = \{K^1, K^2, ..., K^c\}$, each of the $c \in \mathbb{N}$ filters of size $n \times m$. The associated linear operator

$$A = \mathcal{K} * (\cdot) : \mathbb{R}^{N \times M} \to \mathbb{R}^{\tilde{N} \times \tilde{M} \times c},$$

convolves each extended image independently with the help of the extensions and restriction

$$(\mathcal{K} * x)_l := R_p(E_0 K^l * E_R x), \qquad l = 1, \dots, c. \tag{3.8}$$

The filtered images are of size $\tilde{N} \times \tilde{M}$, where $\tilde{N} = N - n + 1 + 2p$ and $\tilde{M} = M - m + 1 + 2p$.

Thus, Ax can be interpreted as a c-channel feature map, and each map $(Ax)_l$ is computed according to the discrete convolution definition via the extension and restriction explained above.

The adjoint operator A^* of a multichannel convolution on finite grids is composed of the adjoints of the extension operators E_0 and E_R, the adjoint of the restriction operator R_p, and the adjoint of the convolution on unbounded domains.

Since the zero extension operator depends on the underlying grid, its adjoint is likewise dependent on the domain. The action of E_0 on a kernel K^l is defined as:

$$E_0 : \{0, \dots, n-1\} \times \{0, \dots, m-1\} \to \mathbb{Z}^2, \; (E_0 x)_{i,j} = \begin{cases} x_{i,j}, & \text{if } 0 \le i < n \text{ and } 0 \le j < m, \\ 0, & \text{else.} \end{cases} \tag{3.9}$$

Its adjoint is a restriction mapping:

$$E_0^* : \mathbb{Z}^2 \to \{0, \dots, n-1\} \times \{0, \dots, m-1\}. \tag{3.10}$$

By contrast, the adjoint of the restriction operator R_p, as defined in (3.6), is given by:

$$R_p^* : \{n-1-p_u, ..., N-1+p_d\} \times \{m-1-p_l, ..., M-1+p_r\} \to \mathbb{Z}^2, \tag{3.11}$$

$$(R_p^* x)_{i,j} = \begin{cases} x_{i,j}, \text{ if } n-p < i < N+p \text{ and } m-p \leq j < M+p, \\ 0, \text{ else.} \end{cases} \tag{3.12}$$

While the adjoint of the restriction operator is itself a zero extension, its domain may not coincide with that of other extension operators. For notational convenience, we shall identify $E_0^* = R_p$ and $R_p^* = E_0$, recognizing that these operators depend on the context.

For the other kind of extensions the adjoint involves the preimage of the index mappings. By the definition of the adjoint we can deduce for the reflective extension:

$$\langle E_R x, y \rangle = \sum_{(i,j)\in\mathbb{Z}^2} x_{(\pi\circ\varphi)(i,j))} y_{i,j} = \sum_{(i,j)\in G} x_{i,j} \sum_{(k,l)\in(\pi\circ\varphi)^{-1}(i,j)} y_{k,l} = \langle x, E_R^* y \rangle. \tag{3.13}$$

Therefore, $E_R^* : \ell^1(\mathbb{Z}^2) \to \ell^1(G)$ is given component wise by

$$(E_R^* y)_{i,j} = \sum_{(k,l)\in(\pi\circ\varphi)^{-1}(i,j)} y_{k,l}. \tag{3.14}$$

The preimage of $(\pi \circ \varphi)^{-1}(i,j) = (\varphi^{-1}(\pi^{-1}(i,j))$ is the set

$$\{(k,l) \in \mathbb{Z}^2 : (\pi \circ \varphi)(k,l) = (i,j)\}. \tag{3.15}$$

The mapping $\pi : \{0, ..., 2N-1\} \times \{0, ..., 2M-1\} \to \{0, ..., N-1\} \times \{0, ..., M-1\}$ reflects the pixels horizontally and vertically. The preimage of (i,j) contains therefore exactly four distinct pairs of indices:

$$\pi^{-1}(i,j) = \{(i,j), (2N-1-i,j), (i, 2M-1-j), (2N-1-i, 2M-1-j)\}. \tag{3.16}$$

The preimage of (i,j) with respect to the $(2N, 2M)$ modulo operation in $\varphi : \mathbb{Z}^2 \to \{0, ..., 2N-1\} \times \{0, ..., 2M-1\}$ is:

$$\varphi^{-1}(i,j) = \{(i+2Ns, j+2Mt) \mid s,t \in \mathbb{Z}\} \tag{3.17}$$

Combining those two preimages gives us an explicit form for the adjoint of the reflective extension operator.

$$(E_R^* y)_{i,j} = \sum_{\substack{s\in 2N\mathbb{Z} \\ t\in 2M\mathbb{Z}}} y_{s+i,t+j} + y_{s-i-1,t+j} + y_{s+i,t-j-1} + y_{s-i-1,t-j-1}. \tag{3.18}$$

Last, but not least, the adjoint of a multichannel convolution corresponds to a correlation on each channel, followed by summation over the channel dimension. This operation is also known as cross-correlation.

The adjoint operator $A^* : \mathbb{R}^{\tilde{N}\times\tilde{M}\times c} \to \mathbb{R}^{N\times M}$ can now be written down explicitly. Let $y_l \in \mathbb{R}^{\tilde{N}\times\tilde{M}}$ denote the l-th channel of $y \in \mathbb{R}^{\tilde{N}\times\tilde{M}\times c}$. Then:

$$\langle Ax, y\rangle = \sum_{l=1}^{c} \langle R_p(E_0 K^l * E_R x), y_l\rangle = \sum_{l=1}^{c} \langle E_0 K^l * E_R x, E_0 y_l\rangle \tag{3.19}$$

$$= \sum_{l=1}^{c} \langle E_R x, E_0 \tilde{K}^l * E_0 y_l\rangle = \sum_{l=1}^{c} \langle x, E_R^*(E_0 \tilde{K}^l * E_0 y_l)\rangle. \tag{3.20}$$

In summary, the adjoint to the multichannel convolution on a finite grid with reflective padding is the following composition of linear operators:

$$A^* y = \mathcal{K} \star y = \sum_{l=1}^{c} E_R^*(E_0 K^l \star E_0 y_l) = \sum_{l=1}^{c} E_R^*(E_0 \tilde{K}^l * E_0 y_l). \tag{3.21}$$

Thus, in practice, A^* performs a convolution of each channel y_l with the flipped kernel $\tilde{K}^l$, and the results are summed over all channels. In the case of TV denoising, this adjoint operation corresponds to applying the discrete divergence operator, which is the adjoint of the discrete gradient.

To construct the regularizer, we group the filters in pairs. This is not strictly necessary, but enables a true generalization of the isotropic total variation regularizer. Since each pair represents a two-dimensional feature vector (analogous to horizontal and vertical gradients in the TV model), we require the number of channels c to be even. The regularization functional is given by the mixed $\ell^{1,2}$-norm:

$$g(Ax) = \|Ax\|_{1,2} = \sum_{i,j} \sum_{\substack{l=1 \\ l \text{ odd}}}^{c-1} \sqrt{(\mathcal{K} * x)_{i,j,l}^2 + (\mathcal{K} * x)_{i,j,l+1}^2}. \tag{3.22}$$

Alternatively, if one wants to generalize the anisotropic total variation, one could define the regularizer as:

$$g(Ax) = \|Ax\|_1 = \sum_{i,j,l} |(\mathcal{K} * x)_{i,j,l}|. \tag{3.23}$$

In case of the mixed $\ell^{1,2}$-norm the inner ℓ^2-norm couples two filters at each spatial location (i, j), while the outer ℓ^1-norm aggregates these magnitudes across the entire image. Grouping the filters pairwise was not the first choice. Initially the ℓ^2-norm was applied over all channels, but that norm did not promote sparsity in the co-domain and resulted in significantly worse performance. Although there is no theoretical upper bound on the number of channels used in $\mathcal{K}$, empirical evidence suggests that using more than nm filters (where $n \times m$ is the kernel size) yields little additional benefit in practice.

Assuming additive Gaussian noise with zero mean implies that the noisy and noise-free images have, on average, the same global luminance. Consequently, the denoised image should preserve this overall luminance level. While this property is not explicitly enforced in the learning objective, in practice it is reflected in the learned filters by the

fact that each kernel sums to zero (up to a small error), $\sum_{i,j} K^l_{i,j} = 0$, ensuring that constant image regions (i.e., uniform luminance) produce zero response.

3.2 Algorithm for efficient denoising using the learned filters

In the special case of TV denoising we have two filters represented by the kernels $K^1 = (\lambda, -\lambda)$ and $K^2 = (\lambda, -\lambda)^\top$. Adapting the denoising scheme 1 to a more general setting with filters of size $m \times n \times c$ is straight forward, except for obtaining a tight estimate for the operator norm of A.

One way for obtaining an $L > 0$ such that $\|Ax\|_2 \leq L\|x\|_2 \ \forall x \in \mathbb{R}^{N\times M}$ is through a power iteration. This is fine if it only has to be done once, but we need also need an estimate for training process, where $\|A\|$ changes with every iteration. For this task we need Young's convolution inequality.

Theorem 13 (Young's convolution inequality (discrete case))**.** *Let $K \in L^p(\mathbb{Z}^2)$ and $x \in L^q(\mathbb{Z}^2)$ and*

$$\frac{1}{p} + \frac{1}{q} = \frac{1}{r} + 1$$

with $1 \leq p, q, r \leq \infty$. Then

$$\|K * x\|_r \leq \|K\|_p \|x\|_q.$$

We are interested in the special case $p = 1$ and $q = r = 2$, which only requires the Cauchy-Schwarz inequality:

Proof.

$$\|K * x\|_2^2 = \sum_{i,j} \left(\sum_{u,v} K_{u,v} x_{i-u,j-v} \right)^2 \tag{3.24}$$

$$\leq \sum_{i,j} \left(\sum_{u,v} \underbrace{|K_{u,v}|}_{=|K_{u,v}|^{\frac{1}{2}}|K_{u,v}|^{\frac{1}{2}}} |x_{i-u,j-v}| \right)^2 \tag{3.25}$$

$$\leq \sum_{i,j} \left((\sum_{u,v} |K_{u,v}|)^{\frac{1}{2}} (\sum_{u,v} |K_{u,v}||x_{i-u,j-v}|^2)^{\frac{1}{2}} \right)^2 \tag{3.26}$$

$$= \|K\|_1 \sum_{i,j} \sum_{u,v} |K_{u,v}||x_{i-u,j-v}|^2 \tag{3.27}$$

$$= \|K\|_1 \sum_{u,v} |K_{u,v}| \sum_{i,j} |x_{i-u,j-v}|^2 = \|K\|_1^2 \|x\|_2^2 \tag{3.28}$$

$$\Rightarrow \|K * x\|_2 \leq \|K\|_1 \|x\|_2 \tag{3.29}$$

□

The main challenge with calculating the operator norm is the reflective extension mapping from the finite grid to the whole integer grid $E_R : \ell^\infty(G) \mapsto \ell^\infty(\mathbb{Z}^2)$. This operator is unbounded with respect to the ℓ^2-norm, i.e. for $x \neq 0$ we have $\|E_R x\|_2 = \infty$.

Instead, we have to take a closer look at the convolution as it is actually implemented, avoiding unbounded operators. First the image x defined on the grid $\{0, \dots, N-1\} \times \{0, \dots, M-1\}$ is extended to the padded grid $\{-p, \dots, N+p-1\} \times \{-p, \dots, M+p-1\}$ and convolving with a kernel defined on $\{0, \dots, n-1\} \times \{0, \dots, m-1\}$ reduces the valid grid size to $\{n-1-p, \dots, N-1+p\} \times \{m-1-p, \dots, M-1+p\}$. This implies, that the operator A does not change, if the full extension E_R is replaced by a limited extension $\tilde{E}_R$, that sets all values to zero outside the padded grid:

$$\tilde{E}_R : \ell^2(G) \to \ell^2(\mathbb{Z}^2) \tag{3.30}$$

$$\tilde{E}_R x_{i,j} = \begin{cases} E_R x_{i,j}, & \text{if } (i,j) \in \{-p, \dots, N+p-1\} \times \{-p, \dots, M+p-1\} \\ 0, & \text{else} \end{cases}. \tag{3.31}$$

The operator norm of the limited reflection extension $\tilde{E}_R$ grows with the amount of padded pixels p to infinity, but for the given use case it is enough to only consider $p \leq \frac{1}{2} \min\{M, N\}$. This ensures that each pixel is at most mirrored once horizontally and once vertically, implying $\|\tilde{E}_R x\|_2^2 \leq 4\|x\|_2^2$.

The restriction operator R_p and the zero extension operator E_0 have both an operator norm of 1. Combining these estimates with Theorem 13 yields:

$$\|Ax\|_2^2 = \sum_{l=1}^{c} \|R_p(E_0 K^l * \tilde{E}_R x)\|_2^2 \leq \sum_{l=1}^{c} \|(E_0 K^l * \tilde{E}_R x)\|_2^2 \tag{3.32}$$

$$\leq \sum_{l=1}^{c} \|E_0 K^l\|_1^2 \|\tilde{E}_R x\|_2^2 \leq 4\|x\|_2^2 \sum_{l=1}^{c} \|K^l\|_1^2 \tag{3.33}$$

$$\leq 4\|x\|_2^2 \left(\sum_{l=1}^{c} \|K^l\|_1 \right)^2 = 4\|x\|_2^2 \|\mathcal{K}\|_1^2. \tag{3.34}$$

The last inequality shows, that Young's inequality even extends to multichannel convolution, but using the tighter upper bound of the operator norm is better for the performance of the denoising algorithm:

$$L = 2\sqrt{\sum_{i=1}^{c} \|K^l\|_1^2}. \tag{3.35}$$

This estimate is tight, meaning there is an operator A such that $\|A\| = L$. Consider a kernel $\mathcal{K}$ of size $n \times m$ and an image x of size $N \times M$ that are constant 1. Further assume that the image is square, i.e. $N = M$, and N is even and set the padding to the maximum assumed: $p = \frac{N}{2}$. Then Ax is an image of size $(N+2p+1-n) \times (M+2p+1-m) \times c =$

$(2N+1-n)\times(2M+1-m)\times c$ with the constant value of nm. The operator norm of A is therefore bounded from below by:

$$\|A\| \geq \frac{\|Ax\|_2}{\|x\|_2} = \frac{\sqrt{(2N+1-n)(2M+1-m)c(nm)^2}}{\sqrt{NM}} \tag{3.36}$$

$$= nm\sqrt{c}\sqrt{\frac{(2N+1-n)(2M+1-m)}{NM}}. \tag{3.37}$$

Comparing this to the estimate given in Equation (3.35):

$$L = 2\sqrt{c(nm)^2}, \tag{3.38}$$

shows that the operator norm coincides with the estimate if $(n,m) = (1,1)$, but convolution with a 1×1 convolution is simply a scalar multiplication and not interesting. However, in the more realistic case where N and M are in the thousands and n and m are single digit numbers, the term $\sqrt{\frac{(2N+1-n)(2M+1-m)}{NM}}$ is only slightly smaller than 2. Furthermore, if p is chosen close to $\frac{1}{2}\max\{n,m\}$ instead of $\frac{1}{2}\max\{N,M\}$, then there is in practice no harm in setting $L = \sqrt{\sum_{i=1}^{c}\|K^l\|_1^2}$ for the purpose of the denoising algorithm, despite loosing the guarantee of convergence.

The Chambolle-Pock iterative procedure adapted for denoising is shown in Algorithm 3.

Algorithm 3 Denoising with general filters

Initialization: Set $L \leftarrow 2\sqrt{\sum_{i=1}^{c}\|K^l\|_1^2}, \sigma \leftarrow \delta\sqrt{NM}, \tau \leftarrow (\sigma L^2)^{-1}$, $(x,y) \leftarrow (x^\delta, 0)$, $\hat{x} \leftarrow x^\delta$
while $\frac{1}{2}\|x-x^\delta\|_2^2 + \|\mathcal{K}*x\|_{1,2} + \frac{1}{2}\|x^\delta - \mathcal{K}\star y\|_2^2 - \frac{1}{2}\|x^\delta\|_2^2 \geq \varepsilon$ **do**
    $y \leftarrow \mathrm{proj}_{\{\|\cdot\|_{1,2}\leq 1\}}(y + \tau(\mathcal{K}*\hat{x}))$
    $s \leftarrow (x - \sigma(\mathcal{K}\star y) + \sigma x^\delta)/(1+\sigma) - x$
    $x \leftarrow x + s$
    $\theta \leftarrow 1/\sqrt{1+\sigma}$
    $\sigma \leftarrow \sigma\theta$
    $\tau \leftarrow \tau/\theta$
    $\hat{x} \leftarrow x + \theta s$
end while
return x

3.3 The objective for the training process

Formally, we can express the general bi-level learning problem as

$$\min_{\mathcal{K}} \sum_{i=1}^{T} d(x^\dagger[i], \bar{x}[i]), \tag{3.39}$$

$$\text{s.t.} \quad \bar{x}[i] = \operatorname*{argmin}_{x[i]} \; f(x[i]) + g(\mathcal{K} * x[i]), \tag{3.40}$$

where $\{x^\dagger[i], x^\delta[i]\}_{i=1}^T$ is the training set, d is a chosen discrepancy measure (such as MSE, PSNR, or SSIM), and $\mathcal{K}$ denotes the convolution operator defined by the learned filters. The inner variational problem produces the reconstruction $\bar{x}[i]$ for each noisy input $x^\delta[i]$, while the outer optimization seeks filters $\mathcal{K}$ that minimize the total discrepancy over all training samples.

Instead of solving this nested bi-level problem directly for one of the discrepancy measures mentioned above, we use the measure based on the gap of the primal objective value of $x^\dagger$ and $\bar{x}$ (POG), as already discussed in the context of parameter fine tuning for TV denoising. That formulation exploits the primal-dual structure of the variational formulation and defines a single-level (mono-level) training criterion based on the primal-dual gap. Using this objective connects the filter optimization directly to the variational problem and ensures that updates are consistent with the reconstruction process.

The discrepancy measure in question has multiple forms, as shown in the previous chapter and summarized here:

$$d(x^\dagger, \bar{x}) = f(x^\dagger) + g(Ax^\dagger) - f(\bar{x}) - g(A\bar{x}) \tag{3.41}$$

$$= D^f_{-A^*\bar{y}}(x^\dagger, \bar{x}) + D^g_{\bar{y}}(Ax^\dagger, A\bar{x}) \tag{3.42}$$

$$= f(x^\dagger) + g(Ax^\dagger) + f^*(-A^*\bar{y}) + g^*(\bar{y}) = G(x^\dagger, \bar{y}). \tag{3.43}$$

While all three expressions are equal, they differ in interpretation. The primal objective gap (3.41) shows, that this discrepancy measure is always non negative by optimality of $\bar{x}$.

Expressing the POG in terms of Bregman divergences (3.42) allows for a comparison to the MSE. Let $\mathcal{J}$ be an optimal set of filters with respect to the MSE and $\mathcal{K}$ optimal with respect to POG. Further let $x^{\mathcal{J}}[i]$ and $x^{\mathcal{K}}[i]$ denote the denoised images. Then,

$$\sum_{i=1}^{T} \frac{1}{2}\|x^\dagger[i] - x^{\mathcal{J}}[i]\|_2^2 \leq \sum_{i=1}^{T} \frac{1}{2}\|x^\dagger[i] - x^{\mathcal{K}}[i]\|_2^2. \tag{3.44}$$

Each term in the sum is equal to $D^f_{-A^*\bar{y}[i]}(x^\dagger[i], x^{\mathcal{K}}[i])$, as calculated in (2.118). If we add to this the non-negative terms $D^g_{\bar{y}}(\mathcal{K} * x^\dagger[i], \mathcal{K} * x^{\mathcal{K}}[i])$, we obtain the POG. By using

the calculations from (2.122) and some basic inequalities we can find an upper bound:

$$\sum_{i=1}^{T} D_{\bar{y}}^{g}(\mathcal{K} * x^{\dagger}[i], \mathcal{K} * x^{\mathcal{K}}[i]) \tag{3.45}$$

$$= \sum_{i=1}^{T} \|\mathcal{K} * x^{\dagger}[i]\|_{1,2} - \|\mathcal{K} * x^{\mathcal{K}}[i]\|_{1,2} - \langle x^{\delta}[i] - x^{\mathcal{K}}[i], x^{\dagger}[i] - x^{\mathcal{K}}[i] \rangle \tag{3.46}$$

$$\leq \sum_{i=1}^{T} \|\mathcal{K} * (x^{\dagger}[i] - x^{\mathcal{K}}[i])\|_{1,2} + \|x^{\dagger}[i] - x^{\mathcal{K}}[i]\|_2 \|x^{\delta}[i] - x^{\mathcal{K}}[i]\|_2 \tag{3.47}$$

$$\leq \sum_{i=1}^{T} c_1 \|\mathcal{K} * (x^{\dagger}[i] - x^{\mathcal{K}}[i])\|_2 + \|x^{\dagger}[i] - x^{\mathcal{K}}[i]\|_2 \|x^{\delta}[i] - x^{\mathcal{K}}[i]\|_2 \tag{3.48}$$

$$\leq \sum_{i=1}^{T} \|x^{\dagger}[i] - x^{\mathcal{K}}[i]\|_2 \left(2c_1 \|\mathcal{K}\|_1 + \|x^{\delta}[i] - x^{\mathcal{K}}[i]\|_2 \right), \tag{3.49}$$

that is proportional to the mean error and the ℓ^1-norm of $\mathcal{K}$. The constant c_1 depends on the size of the dual image and is given by $\sqrt{(N - 2p + 1 - n)(M - 2p + 1 - m)\frac{c}{2}}$. A low noise level requires little regularization and therefore $\|\mathcal{K}\|_1$ also expected to be small and POG is almost identical to MSE. For a high noise level the impact of the additional term becomes more pronounced and the values for POG and MSE start to deviate significantly from each other.

In the previous chapter at equation (2.106) it was already shown, that minimizing the POG objective:

$$\min_{A} \mathrm{POG}(x^{\dagger}, \bar{x}) \quad \text{s.t. } \bar{x} \in \operatorname*{argmin}_{x \in \mathbb{R}^{N \times M}} f(x) + g(Ax), \tag{3.50}$$

is equal to solving the mono-level problem

$$\min_{A} \min_{y \in \mathbb{R}^{N \times M \times c}} f(x^{\dagger}) + g(Ax^{\dagger}) + f^{*}(-A^{*}y) + g^{*}(y). \tag{3.51}$$

The constant terms can of course be left away when minimizing. The minimization problem for training the filters $\mathcal{K}$ is:

$$\boxed{\min_{\substack{\mathcal{K} \in \mathbb{R}^{n \times m \times c} \\ y[i] \in \mathbb{R}^{\tilde{N} \times \tilde{M} \times c}}} \sum_{i=1}^{T} \|\mathcal{K} * x^{\dagger}[i]\|_{1,2} + \frac{1}{2} \|x^{\delta}[i] - \mathcal{K} \star y[i]\|_2^2 + I_{\{\|\cdot\|_{\infty,2} \leq 1\}}(y[i]).} \tag{3.52}$$

The objective is convex in $\mathcal{K}$ for fixed $(y[i])_i$ and convex in each $y[i]$ for a fixed $\mathcal{K}$, but not jointly convex in $(\mathcal{K}, (y[i])_i)$. Indeed, since the functional is even in both arguments, if $(\bar{\mathcal{K}}, (\bar{y}[i])_i)$ is a minimizer, then $(-\bar{\mathcal{K}}, (-\bar{y}[i])_i)$ is also a minimizer. However, their midpoint $(0, 0)$ is in general not optimal, even though it lies between them. This shows

that the set of minimizers is not convex, and therefore the objective cannot be jointly convex.

Moreover, this sign symmetry applies independently to each layer $l = 1, ..., c$ resulting in at least 2^c distinct global minimizers.

3.4 PALM algorithm for training

The objective function to the training problem (3.52) consists of the smooth terms $(\mathcal{K}, y[i]) \mapsto \frac{1}{2}\|x[i]^\delta - \mathcal{K} \star y[i]\|_2^2$, the non differentiable term $\mathcal{K} \mapsto \|\mathcal{K} * x[i]^\dagger\|_{1,2}$ and the constraints $\|y[i]\|_{\infty,2} \leq 1$. Similar to the denoising problem, a stationary point can be found by performing gradient descent with respect to the differentiable terms and then projecting back into the feasible set. The problem is also non convex and therefore we can not directly rely on the primal-dual framework, but fortunately an algorithm for such non convex problems in the form of "Proximal Alternating Linearized Minimization" (PALM) was already proposed in 2014 by Jérôme Bolte, Shoham Sabach, and Marc Teboulle [3].

PALM was designed to effectively minimize objective functions of the form:

$$\psi(x_1, \ldots, x_n) = f_1(x_1) + \cdots + f_n(x_n) + H(x_1, \ldots, x_n), \tag{3.53}$$

where the $x_i \in \mathbb{R}^{n_i}$ are blocks of variables, that are coupled via a differentiable function H with Lipschitz continuous gradients and the f_i are proper, lower semi-continuous and convex functions depending on only a single block of variables. The x_i are updated cyclically by the proximal forward-backward steps:

$$x_i \leftarrow \operatorname{prox}_{\tau_i}^{f_i}(x_i - \tau_i \nabla_{x_i} H(x_1, \ldots, x_n)). \tag{3.54}$$

The "Inertial Proximal Alternating Linearized Minimization" (iPALM [24]) introduces extrapolation steps into the algorithm, which speed up the convergence and can prevent getting trapped in local extrema. The following describes the iPALM algorithm for two variables inclusive the necessary assumptions, as described in [24].

Suppose X_1, X_2 are Hilbert spaces and the functions f_1, f_2 and H satisfy assumptions I-V.

Assumptions:

I. $f_i : \mathbb{R}^{n_i} \to (-\infty, \infty]$ are proper and lower semi-continuous functions such that $\inf_{\mathbb{R}^{n_i}} f_i > -\infty$.

II. $H : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ is differentiable and $\inf_{\mathbb{R}^{n_1} \times \mathbb{R}^{n_2}} \psi > -\infty$.

III. For any fixed x_2 the function $x_1 \mapsto H(x_1, x_2)$ is continuously differentiable and the partial gradient $\nabla_{x_1} H(x_1, x_2)$ is globally Lipschitz, i.e. there exists a function $x_2 \mapsto L_1(x_2)$

$$\|\nabla_{x_1} H(x, x_2) - \nabla_{x_1} H(x', x_2)\|_2 \leq L_1(x_2)\|x - x'\|_2 \quad \forall x, x' \in \mathbb{R}^{n_1}. \tag{3.55}$$

The analogous assumption holds for the function $x_2 \mapsto H(x_1, x_2)$ with Lipschitz constant $L_2(x_1)$.

IV. The function $x_2 \mapsto L_1(x_2)$, which is an upper bound to the Lipschitz moduli, is locally bounded from above and below by strictly positive values, that is, for all compact sets $B_i \subset \mathbb{R}^{n_i}$ there exist $\lambda_i^-, \lambda_i^+ > 0$ such that

$$\inf\{L_1(x_2) : x_2 \in B_2\} \geq \lambda_1^- \text{ and } \sup\{L_1(x_2) : x_2 \in B_2\} \leq \lambda_1^+, \tag{3.56}$$

and the analogous assumption holds for $L_2(x_1)$ with constants λ_2^+, λ_2^-.

V. ∇H is Lipschitz continuous on bounded subsets of $\mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$. In other words, for each bounded subset $B_1 \times B_2$ of $\mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ there exists $M > 0$ such that:

$$\left\| \begin{pmatrix} \nabla_{x_1} H(x_1, x_2) - \nabla_{x_1} H(y_1, y_2) \\ \nabla_{x_2} H(x_1, x_2) - \nabla_{x_2} H(y_1, y_2) \end{pmatrix} \right\|_2 \leq M \left\| \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \end{pmatrix} \right\|_2 . \tag{3.57}$$

Under these assumptions the algorithm generates a sequence that converges to a critical point of ψ.

Algorithm 4 Inertial Proximal Alternating Linearized Minimization (iPALM)

Initialization: $(x_1^0, x_2^0) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$
for k = 1,2,... **do**
  Take $\alpha_1^k, \beta_1^k \in [0, 1)$ and $\tau_1^k = (\gamma_1 L_1(x_2^k))^{-1} > 0$. Compute
  $y_1^k = x_1^k + \alpha_1^k(x_1^k - x_1^{k-1})$
  $z_1^k = x_1^k + \beta_1^k(x_1^k - x_1^{k-1})$
  $x_1^{k+1} = \text{prox}_{\tau_1^k}^{f_1}\left(y_1^k - \tau_1^k \nabla_{x_1} H(z_1^k, x_2^k)\right)$
  Take $\alpha_2^k, \beta_2^k \in [0, 1)$ and $\tau_2^k = (\gamma_2 L_2(x_1^k))^{-1} > 0$. Compute
  $y_2^k = x_2^k + \alpha_2^k(x_2^k - x_2^{k-1})$
  $z_2^k = x_2^k + \beta_2^k(x_2^k - x_2^{k-1})$
  $x_2^{k+1} = \text{prox}_{\tau_2^k}^{f_2}\left(y_2^k - \tau_2^k \nabla_{x_2} H(x_1^{k+1}, z_2^k)\right)$
end for

There are multiple ways for splitting the objective function up into the parts f_1, f_2 and H, but for the variables it only seems natural to choose $x_1 = \mathcal{K}$ in the vector space $X_1 = \mathbb{R}^{n \times m \times c}$ and $x_2 = (y[i])_{i=1}^T$ in $X_2 = \mathbb{R}^{\tilde{N} \times \tilde{M} \times c \times T}$. The training dataset consists of T images with $N \times M$ pixels and c is the number of channels in the convolution kernel $\mathcal{K}$. For convenience and readability, Y will sometimes be used in place of $(y[i])_{i=1}^T$:

$$\psi(\mathcal{K}, Y) = \sum_{i=1}^{T} \|\mathcal{K} * x^{\dagger}[i]\|_{1,2} + \frac{1}{2}\|\mathcal{K} \star y_i - x^{\delta}[i]\|_2^2 + I_{\{\|\cdot\|_{\infty,2}\leq 1\}}(y_i), \tag{3.58}$$

$$f_1(\mathcal{K}) = 0, \tag{3.59}$$

$$f_2(Y) = \sum_{i=1}^{T} I_{\{\|\cdot\|_{\infty,2}\leq 1\}}(y[i]), \tag{3.60}$$

$$H(\mathcal{K}, Y) = \sum_{i=1}^{T} \|\mathcal{K} * x^{\dagger}[i]\|_{\varepsilon} + \frac{1}{2}\|\mathcal{K} \star y[i] - x^{\delta}[i]\|_2^2. \tag{3.61}$$

Here, $\|\cdot\|_{\varepsilon}$ denotes the smoothed replacement for $\|\cdot\|_{1,2}$. The norm $\|\cdot\|_{1,2}$ is modified by adding a small positive term in the square roots, which allows us to avoid the lack of differentiability at zero:

$$\|y\|_{1,2} = \sum_{\substack{i,j \\ k \text{ odd}}} \sqrt{(y_{i,j,k})^2 + (y_{i,j,k+1})^2} = \sum_{\substack{i,j \\ k \text{ odd}}} \left\| \begin{pmatrix} y_{i,j,k} \\ y_{i,j,k+1} \end{pmatrix} \right\|_2, \tag{3.62}$$

$$\|y\|_{\varepsilon} = \sum_{\substack{i,j \\ k \text{ odd}}} \sqrt{(y_{i,j,k})^2 + (y_{i,j,k+1})^2 + \varepsilon^2}, \tag{3.63}$$

$$\text{for } k \text{ odd: } \frac{\partial}{\partial y_{i,j,k}}\|y\|_{\varepsilon} = \frac{y_{i,j,k}}{\sqrt{(y_{i,j,k})^2 + (y_{i,j,k+1})^2 + \varepsilon^2}}, \tag{3.64}$$

$$\text{for } k \text{ even: } \frac{\partial}{\partial y_{i,j,k}}\|y\|_{\varepsilon} = \frac{y_{i,j,k}}{\sqrt{(y_{i,j,k-1})^2 + (y_{i,j,k})^2 + \varepsilon^2}}. \tag{3.65}$$

This modification to the norm clearly fulfills $\forall y : \|y\|_{\varepsilon} \xrightarrow{\varepsilon \searrow 0} \|y\|_{1,2}$, which allows the recovery of the initial problem, and it is continuously differentiable with a Lipschitz continuous derivative. For calculating the Lipschitz constant of the derivative it is enough to consider the function

$$(z_1, z_2) \mapsto \frac{(z_1, z_2)}{\sqrt{z_1^2 + z_2^2 + \varepsilon^2}},$$

because each derivative of the norm with respect to any variable $y_{i,j,k}$ depends only on that variable and its paired variable (either $y_{i,j,k+1}$ or $y_{i,j,k-1}$). The Lipschitz constant of h is bounded by the spectral norm of its Jacobian matrix, which is:

$$\frac{1}{(z_1^2 + z_2^2 + \varepsilon^2)^{\frac{3}{2}}} \begin{pmatrix} z_2^2 + \varepsilon^2 & -z_1 z_2 \\ -z_1 z_2 & z_1^2 + \varepsilon^2 \end{pmatrix}. \tag{3.66}$$

The matrix is symmetric and positive definite with eigenvectors (z_1, z_2) and $(z_2, -z_1)$. The eigenvalues are

$$\lambda_1 = \frac{\varepsilon^2}{(z_1^2 + z_2^2 + \varepsilon^2)^{\frac{3}{2}}} \text{ and } \lambda_2 = \frac{z_1^2 + z_2^2 + \varepsilon^2}{(z_1^2 + z_2^2 + \varepsilon^2)^{\frac{3}{2}}},$$

and maximal for $(z_1, z_2) = (0, 0)$ with values $\lambda_1 = \lambda_2 = \varepsilon^{-1}$. Thus the Lipschitz constant for the derivative of $\|.\|_\varepsilon$ is ε^{-1}.

Alternatively to smoothing the norm one could also set $f_1(\mathcal{K}) = \sum_{i=1}^{T} \|\mathcal{K} * x^\dagger[i]\|_{1,2}$, but computing the proximal mapping of f_1 is then a non trivial subproblem

$$\min_{\mathcal{J}} \frac{1}{2\tau}\|\mathcal{K} - \mathcal{J}\|_2^2 + \sum_{i=1}^{T} \|\mathcal{J} * x^\dagger[i]\|_{1,2}, \tag{3.67}$$

which can be solved with a primal-dual algorithm up to a tolerance ε. Both approaches are compromises in their own way, but the smoothing approach avoids a bi-level structure and is less complicated.

The smoothing modification to the norm was necessary for the assumptions I-V under the PALM algorithm, so let us check carefully, if they are actually satisfied.
Assumption I is trivial for $f_1 \equiv 0$, but f_2 is a indicator function over the set

$$\{(y[i])_i : \|y[i]\|_{\infty,2} \leq 1\} \subset X_2. \tag{3.68}$$

Indicator functions I_C are proper, if and only if the set C is not empty and lower semi-continuous, if and only if C is closed. Therefore f_2 is proper and lower semi-continuous, since closed unit balls are indeed non empty and closed.

Assumption II is about the differentiability of the coupling function H. The derivative of H with respect to Y is straight forward, because it only involves the quadratic ℓ^2-norm, a constant shift and a linear operator in the form of the adjoint to the convolution $\mathcal{K} \star (\cdot)$. First of all, the term $\|\mathcal{K} * x^\dagger[i]\|_\varepsilon$ is constant with respect to Y and vanishes in the derivative. Second, if we know the derivative for the case $T = 1$, then the general case follows immediately by:

$$\nabla_Y H(\mathcal{K}, Y) = \nabla_Y \left(\sum_{i=1}^{T} \frac{1}{2}\|\mathcal{K} \star y[i] - x^\delta[i]\|_2^2 \right) = \frac{1}{2} \begin{pmatrix} \nabla_{y[1]} \|\mathcal{K} \star y[1] - x^\delta[1]\|_2^2 \\ \vdots \\ \nabla_{y[T]} \|\mathcal{K} \star y[T] - x^\delta[T]\|_2^2 \end{pmatrix}. \tag{3.69}$$

Thus we only need to consider the helper function $h : \mathbb{R}^{\tilde{N} \times \tilde{M} \times c} \to \mathbb{R}$ defined by $h(y) = \frac{1}{2}\|\mathcal{K} \star y - x^\delta\|_2^2$. The derivative of h with respect to y is by the chain rule:

$$\nabla_y h(y) = \mathcal{K} * (\mathcal{K} \star y - x^\delta). \tag{3.70}$$

Recall that the discrete multichannel convolution on a finite domain is defined in (3.7). This construction utilizes a sequence of operators, namely a reflective extension E_R, a zero extension E_0, the convolution on an unbounded domain, and a restriction operator R_p that accounts for the padded pixels:

$$\mathcal{K} * (\cdot) : \mathbb{R}^{N\times M} \to \mathbb{R}^{\tilde{N}\times\tilde{M}\times c},\ x \mapsto \Big(R_p(E_0 K^l * E_R x)\Big)_{l=1}^{c}, \tag{3.71}$$

$$\mathcal{K} \star (\cdot) : \mathbb{R}^{\tilde{N}\times\tilde{M}\times c} \to \mathbb{R}^{N\times M},\ y \mapsto \sum_{l=1}^{c} E_R^*(E_0 \tilde{K}^l * E_0 y_l), \tag{3.72}$$

where E_R^* denotes the adjoint of E_R and $\tilde{K}^l$ represents the spatially flipped kernel of K^l. This flipping operations also changes the domain from $\{0,\dots,n\}\times\{0,\dots,m\}$ to $\{-n,\dots,0\}\times\{-m,\dots,0\}$.

For the derivative of H with respect to $\mathcal{K}$ we need to consider a role reversal in the convolution operation. The images and dual images x and y also act as linear operators on $\mathcal{K}$.

$$(\cdot) * x : \mathbb{R}^{n\times m\times c} \to \mathbb{R}^{\tilde{N}\times\tilde{M}\times c},\ \mathcal{K} \mapsto \Big(R_p(E_0 K^l * E_R x)\Big)_{l=1}^{c}, \tag{3.73}$$

$$(\cdot) \star y : \mathbb{R}^{n\times m\times c} \to \mathbb{R}^{N\times M},\ \mathcal{K} \mapsto \sum_{l=1}^{c} E_R^*(E_0 \tilde{K}^l * E_0 y_l). \tag{3.74}$$

The adjoint operators of those two operator are required for the desired derivative. The linear operator x acts on the multiple kernels $\mathcal{K}$ in a parallel manner, i.e. $\mathcal{K} * x = (K^l * x)_{l=1}^{c}$.

Let $x \in \mathbb{R}^{N\times M}$ be a fixed operator acting on the multi-kernel $\mathcal{K}$ consisting of the individual kernels $K^l \in \mathbb{R}^{n\times m}$ and let $y \in \mathbb{R}^{\tilde{N}\times\tilde{M}}$ be an arbitrary vector. Then the adjoint to the linear operation $(\cdot) * x$ is given for a $l = 1,\dots,c$ by:

$$\langle K^l * x, y\rangle = \langle R_p(E_0 K^l * E_R x), y\rangle = \langle E_0 K^l * E_R x, E_0 y\rangle \tag{3.75}$$

$$= \langle E_0 K^l, E_0 y * E_R \tilde{x}\rangle = \langle K^l, R(E_0 y * E_R \tilde{x})\rangle, \tag{3.76}$$

where $\tilde{x}$ is the spatially flipped image x and $R : \mathbb{Z}^2 \to \{0,\dots,n\}\times\{0,\dots,m\}$ is a simple restriction of the domain. Thus the adjoint operator is:

$$(\cdot) \star x : \mathbb{R}^{\tilde{N}\times\tilde{M}\times c} \to \mathbb{R}^{n\times m\times c},\ y \mapsto \Big(R(E_0 y_l * E_R \tilde{x})\Big)_{l=1}^{c}. \tag{3.77}$$

Notice, even though same notation for $K \star y$ and $y \star x$ is used, these operations fundamentally differ in their order of the extension, flipping and restriction operations.

Similar computations are needed for the adjoint operator to $(\cdot) \star y$.

Let $y \in \mathbb{R}^{\tilde{N}\times\tilde{M}\times c}$ be a fixed operator acting on $\mathcal{K}$ and let $x \in \mathbb{R}^{N\times M}$ be an arbitrary vector. Then,

$$\langle \mathcal{K} \star y, x \rangle = \left\langle \sum_{l=1}^{c} E_R^*(E_0 \tilde{K}^l * E_0 y_l), x \right\rangle \tag{3.78}$$

$$= \sum_{l=1}^{c} \langle \tilde{K}^l, R(E_R x * E_0 \tilde{y}_l) \rangle \tag{3.79}$$

$$= \sum_{l=1}^{c} \langle K^l, R(E_R \tilde{x} * E_0 y_l) \rangle. \tag{3.80}$$

In the last equation we used that spatially flipping is its own inverse, i.e. $\widetilde{(\tilde{K})} = K$ and interacts with convolution by $\widetilde{(x * y)} = \tilde{x} * \tilde{y}$. While not visible in the notation, this flipping operation also requires changing the target domain of the restriction R.

The adjoint operator to $(\cdot) \star y$ is therefore:

$$(\cdot) * y : \mathbb{R}^{N \times M} \to \mathbb{R}^{n \times m \times c} \tag{3.81}$$

$$x \mapsto \Big(R(E_R \tilde{x} * E_0 y_l) \Big)_{l=1}^{c}. \tag{3.82}$$

An explicit expansion of the derivatives of H is notationally cumbersome. However, by accounting for the context-dependent definitions of $*$ and $\star$, the gradient of H can be expressed compactly as

$$\nabla_{y[i]} H(\mathcal{K}, Y) = \mathcal{K} * (\mathcal{K} \star y[i] - x^\delta[i]), \tag{3.83}$$

$$\nabla_{\mathcal{K}} H(\mathcal{K}, Y) = \sum_{i=1}^{T} \Big(\|\mathcal{K} * x^\dagger[i]\|_\varepsilon' \Big) \star x^\dagger[i] + (\mathcal{K} \star y[i] - x^\delta[i]) * y[i], \tag{3.84}$$

where $\|\cdot\|_\varepsilon'$ is defined pointwise according to (3.64) and (3.65).

The convolutions of the form $x * y$ and $y \star x$ appearing in $\nabla_{\mathcal{K}} H(\mathcal{K}, Y)$ are prohibitively expensive to compute if implemented naively. The computational complexity of the 2d discrete convolution is $\theta(N(N-n)M(M-m))$, if x is of size $N \times M$ and y of size $(N+n-1) \times (M+m-1)$. The output of this convolution is only of size $n \times m$. This can be improved by utilizing Fourier transformations (FFT) or by avoiding the convolution of two images altogether. Features like "autograd" in PyTorch are highly optimized for computing the gradients in convolutional neural networks and therefore also the preferred way for computing the gradient $\nabla_{\mathcal{K}} H$.

Regarding the assumptions made by the iPALM algorithm, the combinations of quadratic and linear terms are clearly differentiable and the differentiability of the regularizer has also been ensured by the smoothing parameter $\varepsilon > 0$. Furthermore, the objective function ψ involves only non negative terms and is thus bounded from below by zero.

Estimates for the Lipschitz moduli of the partial gradients are needed for assumption III and also for the choice of the step sizes. The step sizes τ_1 and τ_2 are directly

proportional to the Lipschitz constants of the block-wise gradients of H. Finding a good estimate for those constants is crucial for the convergence rate of the algorithm. The estimate for the Lipschitz constant $L_1(Y)$ for the derivative of H with respect to $\mathcal{K}$ is allowed to depend on Y and characterized by the inequality:

$$\|\nabla_{\mathcal{K}} H(\mathcal{K}, Y) - \nabla_{\mathcal{K}} H(\mathcal{K}', Y)\|_2 \leq L_1(Y)\|\mathcal{K} - \mathcal{K}'\|_2 \quad \forall \mathcal{K}, \mathcal{K}' \in \mathbb{R}^{n \times m \times c}. \tag{3.85}$$

Let $\mathcal{K}, \mathcal{K}'$ be two arbitrary kernels. Using the gradient calculated in equation (3.84) gives:

$$\left\| \sum_{i=1}^{T} \left(\|\mathcal{K} * x^\dagger[i]\|_\varepsilon' - \|\mathcal{K}' * x^\dagger[i]\|_\varepsilon' \right) \star x^\dagger[i] + ((\mathcal{K} - \mathcal{K}') \star y[i]) * y[i] \right\|_2 \tag{3.86}$$

$$\leq \sum_{i=1}^{T} \left\| \left(\|\mathcal{K} * x^\dagger[i]\|_\varepsilon' - \|\mathcal{K}' * x^\dagger[i]\|_\varepsilon' \right) \star x^\dagger[i] \right\|_2 + \left\| ((\mathcal{K} - \mathcal{K}') \star y[i]) * y[i] \right\|_2. \tag{3.87}$$

By previous estimates the Lipschitz constant of $\|\cdot\|_\varepsilon'$ is ε^{-1} which can be used to estimate the left term in the sum. Also, all the way back in equation (3.35) it was shown that the padding operator involved in the discrete convolution on a finite grid may introduce an additional multiplicative factor of two in Theorem 13. Recall the result:

$$\|\mathcal{K} * x\|_2 \leq 2\|\mathcal{K}\|_1 \|x\|_2. \tag{3.88}$$

Even though we have not explicitly proven it, it is trivial to extend the results from Theorem 13 the $*$ and $\star$ operations defined in (3.81) and (3.77). Applying Theorem 13 to (3.87) gives the following estimates:

$$(3.87) \leq 2 \sum_{i=1}^{T} \varepsilon^{-1} \|(\mathcal{K} - \mathcal{K}') * x^\dagger[i]\|_2 \|x^\dagger[i]\|_1 + \|(\mathcal{K} - \mathcal{K}') \star y[i]\|_2 \|y[i]\|_1 \tag{3.89}$$

$$\leq 4\|\mathcal{K} - \mathcal{K}'\|_2 \sum_{i=1}^{T} \varepsilon^{-1} \|x^\dagger[i]\|_1^2 + \|y[i]\|_1^2. \tag{3.90}$$

This is the most straightforward way to obtain an estimate for $L_1(Y)$, but this approach is also wasteful, because it does not scale favorably with the image resolution $N \times M$. The ℓ^1-norm grows faster than the ℓ^2-norm with additional pixels, provided that the values in the images are approximately evenly distributed and not concentrated in a few pixels. In the extreme case where x is a flat image containing only one value, we have $\|x\|_1 = \sqrt{NM}\|x\|_2$. Thus we have to get a bit creative with the use of Theorem 13. The kernel size $n \times m$ is expected to be significantly smaller than the full training image size of $N \times M$. If this is indeed the case, we can improve the estimate by using the

inequalities $\|\mathcal{K}\|_\infty \leq \sqrt{nmc}\|\mathcal{K}\|_2$ and $\|\mathcal{K}\|_1 \leq \sqrt{nmc}\|\mathcal{K}\|_2$. This allows Theorem 13 to be applied in the form $\|\mathcal{K} * x\|_\infty \leq \|\mathcal{K}\|_2\|x\|_2$. Starting again from line (3.87) we obtain:

$$(3.87) \leq \sqrt{nmc}\sum_{i=1}^{T}\left\|\left(\|\mathcal{K} * x^\dagger[i]\|_\varepsilon' - \|\mathcal{K}' * x^\dagger[i]\|_\varepsilon'\right) \star x^\dagger[i]\right\|_\infty + \left\|((\mathcal{K}-\mathcal{K}') \star y[i]) * y[i]\right\|_\infty \tag{3.91}$$

$$\leq 2\sqrt{nmc}\sum_{i=1}^{T}\varepsilon^{-1}\|(\mathcal{K}-\mathcal{K}') * x^\dagger[i]\|_2\|x^\dagger[i]\|_2 + \|(\mathcal{K}-\mathcal{K}') \star y[i]\|_2\|y[i]\|_2 \tag{3.92}$$

$$\leq 4\sqrt{nmc}\sum_{i=1}^{T}\varepsilon^{-1}\|\mathcal{K}-\mathcal{K}'\|_1\|x^\dagger[i]\|_2^2 + \|\mathcal{K}-\mathcal{K}'\|_1\|y[i]\|_2^2 \tag{3.93}$$

$$\leq 4nmc(\varepsilon^{-1}\|X^\dagger\|_2^2 + \|Y\|_2^2)\|\mathcal{K}-\mathcal{K}'\|_2. \tag{3.94}$$

This concludes the estimates regarding $L_1(Y)$. Moving on, $L_2(\mathcal{K})$ is characterized by:

$$\|\nabla_Y H(\mathcal{K}, Y) - \nabla_Y H(\mathcal{K}, Y')\|_2 \leq L_2(\mathcal{K})\|Y - Y'\|_2 \quad \forall Y, Y' \in \mathbb{R}^{\tilde{N}\times\tilde{M}\times c\times T}. \tag{3.95}$$

The function $Y \mapsto \nabla_Y H(\mathcal{K}, Y)$ is linear and therefore estimating $L_2(\mathcal{K})$ is comparatively simple. Plugging (3.83) into the inequality and using Theorem 13 repeatedly, with a multiplicative factor of two taking the padding into account, yields the following estimate for each $i = 1, \ldots, T$:

$$\|\mathcal{K} * (\mathcal{K} \star y[i] - x^\delta[i]) - \mathcal{K} * (\mathcal{K} \star y'[i] - x^\delta[i])\|_2 \tag{3.96}$$

$$= \|\mathcal{K} * (\mathcal{K} \star (y[i] - y'[i]))\|_2 \leq 2\|\mathcal{K}\|_1\|\mathcal{K} \star (y[i] - y'[i])\|_2 \tag{3.97}$$

$$\leq 4\|\mathcal{K}\|_1^2\|y[i] - y'[i]\|_2. \tag{3.98}$$

Finally, the estimates for the upper bounds on the Lipschitz constants of the gradients can be expressed as:

$$L_1(Y) = 4nmc(\varepsilon^{-1}\|X^\dagger\|_2^2 + \|Y\|_2^2), \tag{3.99}$$

$$L_2(\mathcal{K}) = \|\mathcal{K}\|_1^2. \tag{3.100}$$

Crucial for assumption IV, the functions $L_1(Y)$ and $L_2(\mathcal{K})$ are indeed bounded from above on compact sets, but $L_2(\mathcal{K})$ can get arbitrary close to zero and that would result unbounded step sizes. A simple fix, as recommended by the PALM authors [3], is to add a small positive constant to L_2. In L_1 one can see, that the smoothing parameter ε indirect proportional to the step sizes and should be chosen with care.

Finally, for assumption V the function ∇H has to be Lipschitz continuous on bounded subsets $B_1 \times B_2 \subset \mathbb{R}^{n\times m\times c} \times \mathbb{R}^{\tilde{N}\times\tilde{M}\times c\times T}$. In the general case, this does not follow directly from assumption III and IV, however in our case, we can use the estimates

for L_1 and L_2 to get a combined estimate. Let $\mathcal{K}, \mathcal{K}' \in B_1$ and $Y, Y' \in B_2$ be arbitrary vectors and let $r > 0$ be an upper bound to the radius of the bounded sets, i.e. $\max\{\|\mathcal{K}\|_2, \|\mathcal{K}'\|_2, \|Y\|_2, \|Y'\|_2\} \leq r$. Then, by expressing the Lipschitz constants L_1 and L_2 with respect to r:

$$L_1(Y) = 4nmc(\varepsilon^{-1}\|X^\dagger\|_2^2 + \|Y\|_2^2) \leq 4nmc(\varepsilon^{-1}\|X^\dagger\|_2^2 + r^2), \tag{3.101}$$

$$L_2(\mathcal{K}) \leq \|\mathcal{K}\|_1^2 \leq nmc\|\mathcal{K}\|_2^2 \leq nmcr^2, \tag{3.102}$$

and setting $M = 5nmc(\varepsilon^{-1}\|X^\dagger\|_2^2 + r^2)$ we obtain a finite $M > 0$ such that:

$$\left\| \begin{pmatrix} \nabla_{\mathcal{K}} H(\mathcal{K}, Y) - \nabla_{\mathcal{K}} H(\mathcal{K}', Y') \\ \nabla_Y H(\mathcal{K}, Y) - \nabla_Y H(\mathcal{K}', Y') \end{pmatrix} \right\|_2 \leq M \left\| \begin{pmatrix} \mathcal{K} - \mathcal{K}' \\ Y - Y' \end{pmatrix} \right\|_2 . \tag{3.103}$$

In summary, all assumptions I–V are satisfied, ensuring that the algorithm converges to a critical point of the training problem. However, verifying whether this critical point corresponds to a global minimum remains impractical. The consistency of the algorithm and the empirical likelihood of getting stuck in suboptimal local minima will be investigated in the next chapter, providing evidence for its robustness.

Regarding initialization, the dual variable Y can be safely initialized as all zeros or, if a reasonable kernel is already available (e.g., from a preliminary training phase on a smaller dataset), as $Y = \|\mathcal{K} * X^\dagger\|'_\varepsilon$.

For a fresh initialization one has to be careful, because if any two kernels of $\mathcal{K}$ coincide at the beginning of the training process, they stay the same throughout. In order to prevent this, $\mathcal{K}$ is first initialized as c random vectors and then orthonormalized, i.e. $\langle K^i, K^j \rangle$ for $i \neq j$. Naturally, this orthogonalization step is omitted if $c > nm$.

At last, we still have to talk about the inertia parameters α and β and the step size parameter γ. The parameters α and β can be either chosen as a specific constant in the interval $[0, 1)$ or change with every iteration. In [24] it was observed empirically, that $\alpha = \beta = \dfrac{k-1}{k+2}$, where k is the current iteration, produced better results than a static choice. The parameter γ for the step sizes should be chosen as $\dfrac{1+\beta}{2-\alpha}$, if the f_i are convex, as explained in [24, p. 15].
Therefore the iPALM algorithm applied to the training problem results in the following pseudo code.

One should note, that $(\hat{\mathcal{K}}, \hat{Y})$ represents $(\mathcal{K}, Y)$ from the previous iteration and can be initialized arbitrarily, since $\alpha = 0$ in the first iteration. The gradients $\nabla_{\mathcal{K}} H$ and $\nabla_Y H$ are calculated according to (3.84) and (3.83) respectively.

Using a combined early stopping criterion based on $(\mathcal{K}, Y)$ is impractical and instead we are going to check stopping criteria for $\mathcal{K}$ and Y separately. The algorithm only stops, if both are fulfilled.

If $\nabla_{\mathcal{K}} H = 0$, then $\mathcal{K}$ is a stationary point, because the objective is unconstrained and differentiable with respect to $\mathcal{K}$.

Algorithm 5 Training Algorithm

Initialization: $(\mathcal{K}, Y) \in \mathbb{R}^{n\times m\times c} \times \mathbb{R}^{\tilde{N}\times\tilde{M}\times c\times T}$, $\varepsilon > 0$
for k = 1,2,... **do**
  Set $\alpha \leftarrow \dfrac{k-1}{k+2}$ and $\gamma \leftarrow \dfrac{1+\alpha}{2-\alpha}$
  Set $\tau_1 \leftarrow (4\gamma nmc(\varepsilon^{-1}\|X^\dagger\|_2^2 + \|Y\|_2^2)^{-1}$. Compute
  $\mathcal{J} \leftarrow \mathcal{K} + \alpha(\mathcal{K} - \hat{\mathcal{K}})$
  $\hat{\mathcal{K}} \leftarrow \mathcal{K}$
  $\mathcal{K} \leftarrow \mathcal{J} - \tau_1 \nabla_{\mathcal{K}} H(\mathcal{J}, Y)$

  Set $\tau_2 \leftarrow (4\gamma\|\mathcal{K}\|_1^2)^{-1}$. Compute
  $Z \leftarrow Y + \alpha(Y - \hat{Y})$
  $\hat{Y} \leftarrow Y$
  $Y \leftarrow \mathrm{proj}_{\{\|\cdot\|_{\infty,2}\leq 1\}}\left(Z - \tau_2 \nabla_Y H(\mathcal{K}, Z)\right)$
end for

Minimizing with respect to $(y[i])_i$ coincides with the dual of the denoising problem. Therefore the same stopping criterion can be used. For solutions $x[i]$ and $y[i]$ to the denoising problem, the primal-dual gap is zero. The indicator function in the primal-dual gap always evaluates as zero, because the algorithm only produces iterates in the valid domain. Since the primal-dual gap is positive otherwise, the individual primal-dual gaps can be accumulated:

$$G(X,Y) = \sum_{i=1}^{T} \frac{1}{2}\|x[i] - x[i]^\delta\|_2^2 + \|\mathcal{K} * x[i]\|_{1,2} + \frac{1}{2}\|\mathcal{K} \star y[i]\|_2^2 - \langle x[i]^\delta, \mathcal{K} \star y[i]\rangle. \quad (3.104)$$

Furthermore, if $y[i]$ is optimal, then, by the Fenchel-extremality condition in Theorem 6, the denoised image is $x[i] = x[i]^\delta - \mathcal{K} \star y[i]$. By plugging $x[i]$ into the function above we obtain an optimality condition only depending on Y:

$$G(Y) = \sum_{i=1}^{T} \|\mathcal{K} \star y[i]\|_2^2 - \langle x[i]^\delta, \mathcal{K} \star y[i]\rangle + \|\mathcal{K} * (x[i]^\delta - \mathcal{K} \star y[i])\|_{1,2}. \quad (3.105)$$

Let $\mu_1, \mu_2 > 0$ be the tolerances for the stopping criterion. The algorithm is set to stop if:

$$\|H_{\mathcal{K}}(\mathcal{K}, Y)\|_2 < \mu_1 \text{ and } G(Y) < \mu_2. \quad (3.106)$$

The tolerance for the primal-dual gap is set to $\mu_2 = 10^{-5}NMT$. This is one order of magnitude tighter than the stopping criterion for the denoising problem, because a tight tolerance in the training process is crucial for the final performance. The tolerance for

the gradient with respect to $\mathcal{K}$ is set to $\mu_1 = 10^{-4}NMT$. The factor NMT adjusts the tolerance based on the training volume and the factor 10^{-4} was determined empirically. The norm of the gradient was also averaged over the last 100 iterations, to minimize the chance of prematurely stopping in unstable stationary points.

Additionally, the algorithm was stopped early, if the objective starts to worsen due to an accumulation of floating point errors.

It should also be noted, that there is potential to use faster algorithms than iPALM, because if either $\mathcal{K}$ or $(y[i])_i$ is fixed, we recover already well known minimization problems. Minimizing with respect to $(y[i])_i$ is the dual of the denoising problem and can be solved for each $i \in \{1, \dots, T\}$ independently. Despite this observation, the significantly faster primal-dual denoising Algorithm 3 can not be used, because it assumes the linear operator $\mathcal{K}$ to be constant.

On the other hand, minimizing with respect to $\mathcal{K}$ can be interpreted as a deconvolution problem, where $(y[i])_i$ are convolution kernels and the $(x^\dagger[i])_i$ act as regularizer. Again, primal-dual algorithm are the preferred over gradient descent on the smoothed functions, but deconvolving, while the kernels $(y[i])_i$ keep changing, is highly unstable.

In conclusion, while both subproblems could benefit from faster algorithm, the coupling between the variables poses a serious challenge that was solved by using the iPALM algorithm.

4 Empirical results on natural images

The algorithm was implemented in Python and executed on consumer-grade hardware. By utilizing PyTorch with CUDA 13.0, the GPU-based implementation achieved more than double the computational speed compared to the CPU. However, numerical stability requirements necessitated the use of double-precision (FP64) arithmetic. This is unfortunate, because FP64 is only partially supported on consumer hardware. Consequently, the GPU's performance in FP64 operations dropped fourfold instead of the expected twofold slowdown in comparison to FP32. The complete hardware and software specifications are detailed in Table 1. The code is available via a GitHub repository[1].

Table 1: Test Bench Specifications

Category	Details
Operating System	Microsoft Windows 11 Education (Version 10.0.26200)
System Type	x64-based PC
Python Version	3.14.3
CUDA Version	13.1
CPU	AMD Ryzen 5 7600X (6-Core, 4.7 GHz)
GPU	NVIDIA GeForce RTX 4070 Super
Motherboard	PRO B650-S WIFI (MS-7E26)
BIOS	American Megatrends International, LLC. 1.90 (UEFI)
Installed RAM	32.0 GB
Memory Clock Speed	5600 MT/s
VRAM	12.0 GB

4.1 Obtaining training data

The images for the training process are taken from the "Berkeley Segmentation Data Set (BSDS500)" [1]. This dataset consists of 500 natural images with a resolution of 481×321 pixels. The dataset also contains a manually drawn segmentation for each image, because of its initial use case as a benchmark for image segmentation, but they are irrelevant to our use case. They are all color images, however they will be converted to grayscale images for this test. The denoising algorithm can be easily adapted to

[1] https://github.com/Nebelwaffel/ParameterLearning.git

work on color images too, by extending the input of the multichannel convolution from one to three channels. This means, the denoising method will show at least the same performance on color images, potentially even better.

As it turns out, BSDS500 successfully established itself in the visual computing community as the default basis for comparing all sorts of image processing tasks, such as denoising, deblurring, inpainting or edge detection. The images are already sorted into one of the three categories "training", "validation" and "test", as usual for machine learning datasets. The training dataset contains 200 images, and as the name suggests, are intended to be used for the training phase of the algorithm. The 100 validation images and the 200 test images are both intended for evaluating the performance of the algorithm, but good scientific conduct demands, that only the validation images can be used for adjusting parameters (such as kernel size or channel count). The test images are only used once, after the best possible combination of parameters has been determined, and that settles the final score and provides a fair comparison to other algorithms.

Additionally to the ground truths $x^\dagger$ provided by the dataset, one also needs noisy data x^δ. The most common way to generate noisy data, is to add some Gaussian noise with standard deviation $\delta > 0$ to the ground truth, i.e. $x^\delta = x^\dagger + \mu$ with $\mu \sim \mathcal{N}(0, \delta)$. This is also the approach that was taken here. A noise level of $\delta = 0.1$ corresponds to a PSNR (peak-signal-to-noise-ratio) of 20.00 dB, because all the images are scaled, such that their values are from 0 to 1.

One should keep in mind, that a normal distribution is only a rough approximation for real noise distribution on images taken with a camera. If one actually plans on deploying a machine learning algorithm into a real world application, one should also put effort into modeling the noise accurately. The algorithm will fit to that noise distribution from the training phase specifically and it will perform poorly, if the real measured data suddenly follows a different distribution. Alternatively, in special cases it is also possible to use so called variance-stabilizing transformations (VST) in order to change the noise distribution to a Gaussian [23].

4.2 Hyper-parameter fine tuning

In this chapter, we examine the numerical properties of the proposed algorithm and the impact of the hyper-parameters $\varepsilon, N, M, T, n, m, c$ on the learned parameters in the convolution $\mathcal{K}$. Before presenting the statistical analysis, we clarify our notation: the objective POG is scaled by a factor of $2/(NMT)$. This normalization ensures comparability with the MSE irrespective of the number of pixels or training images. The scaling factor of 2 is necessary to satisfy the inequality $\mathrm{MSE}(x, y) \leq \mathrm{POG}(x, y)$, as deduced in Equation (2.118).

Several parameters influence the computational runtime of the algorithm. Primarily, the smoothing parameter ε is inversely proportional to the step size and consequently, it dictates the number of iterations required for convergence. Second, the time consumed per iteration depends on the volume and resolution of the training dataset and the kernel dimensions.

To identify the optimal configuration, we conduct a series of six controlled experiments, systematically varying the parameters $\varepsilon, N, M, T, n, m, c$.

I. Consistency of random initialization

II. Smoothing parameter ε

III. Training data count T

IV. Training data resolution $N \times M$

V. Kernel size $n \times m$

VI. Channels c

A significant challenge arises from the interdependence of these parameters. For instance, larger kernels typically require more training data to achieve robust performance. While a comprehensive exploration of the entire parameter space is theoretically ideal, it is practically infeasible and computationally inefficient. Instead, we adopt an iterative approach: whenever a later test yields a critical insight, we revisit and refine earlier experiments using this new knowledge. Tests conducted with suboptimal parameters (in hindsight) are omitted from this analysis.

Table 2 summarizes the parameter configurations used across all experiments and the final choice of parameters. Here p denotes the padding applied during convolution.

Test	ε	N × M	T	n × m	c	p
I	10^{-4}	32×32	50	3×3	8	0
II	$10^{-6} - 10^{-3}$	32×32	200	5×5	16	2
III	10^{-4}	32×32	4-225	3×3	8	0
IV	10^{-4}	$(8 \times 8) - (196 \times 196)$	100	5×5	16	2
V	10^{-5}	48×48	200	$(2 \times 2) - (8 \times 8)$	32	1-4
VI	10^{-5}	48×48	200	6×6	4-40	3
FINAL	10^{-5}	96×96	200	9×9	80	4

Table 2: Summary of the parameters used in the empirical tests.

Test I serves as a baseline to evaluate the consistency of the proposed method. Provided that kernels are randomly initialized and stochastic noise is applied to the training data, results may exhibit variability across repeated runs. Quantifying this variability is critical, because the validity of all subsequent results depend on this robustness. This test enables us to distinguish between random fluctuations and the genuine effects of the parameters under investigation.

To minimize computational cost, we used a small kernel of size $3\times3\times8$ and a moderate dataset of 50 training images. These images consist 32×32 patches extracted from full-resolution images of 481×321. The selection of patches is randomized, but fixed, for all runs in this test. The ground truth remains identical across runs to isolate the impact

of random initialization and stochastic noise. The influence of random training data selection is addressed in Test III.

For this test, the smoothing parameter ε was set to 10^{-4}, and no padding was applied during convolution. While reflective padding improves denoising performance, it also increases computational overhead and necessitates more conservative step sizes. As the focus of this test is consistency rather than optimal denoising, padding was omitted.

The training procedure was repeated ten times. The number of iterations required for convergence varied between 3300 and 6900. The computation rate averaged approximately 430 iterations per second, with total training times ranging from 8 to 16 seconds. Upon termination, the objective ranged between 0.004333 and 0.00444 and the MSE on the training ranged between 0.002314 and 0.002404. Denoising performance was evaluated on all 100 validation images at full resolution from the BSDS500 dataset. Despite the big differences in the training time the average PSNR varied only from 27.095 dB to 27.154 dB.

Regarding perceptual significance, a study on JPEG-compressed images [4] reports that the Just Noticeable Difference (JND) varies significantly with image content and distortion type. However, they found that PSNR differences below approximately 0.6–0.8 dB are not reliably distinguished by human observers. This puts the range of 0.059dB well below the perceivable differences and demonstrates consistent results despite the random initialization of the kernels and the noisy images.

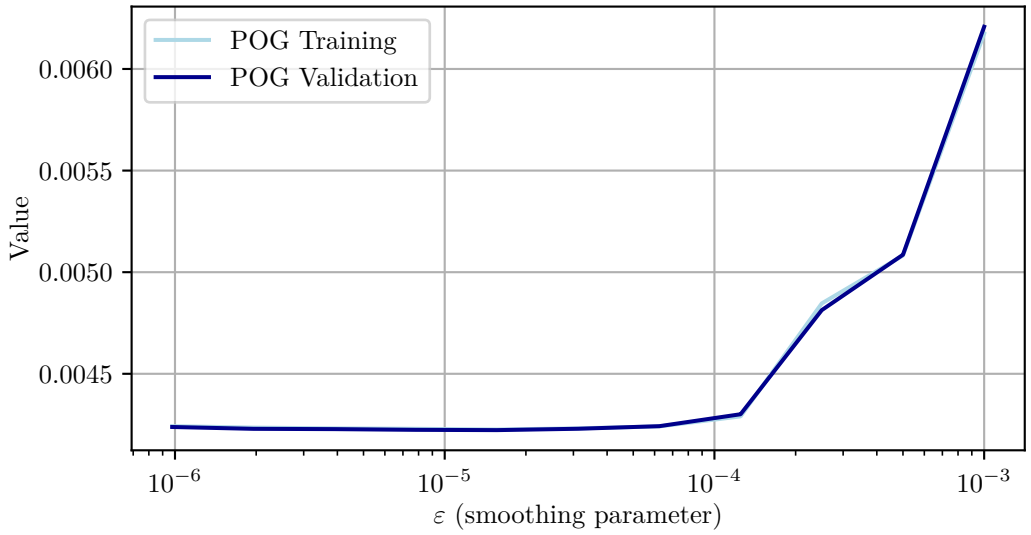


Figure 8: **Test II,** optimal objective (32×32 images) and validation POG (481×321 images) as a function of the smoothing parameter ε.

In **Test II** the point of diminishing returns for the smoothing parameter ε is investigated. While a smaller ε theoretically yields results closer to the original problem, it may introduce numerical instability or unnecessarily prolong training time without measurable improvements. To assess this, we train a kernel of size $5 \times 5 \times 16$ on 200

images of resolution 32×32. The evaluation started with $\varepsilon = 10^{-6}$ and was doubled ten times for a final value of $\varepsilon = 10^{-3}$.

Figure 8 presents the optimal objective, i.e. the POG on the training set, and the POG on 100 denoised validation images of resolution 481×321, all with respect to ε. Notably, the scores on the training data and the validation data are almost perfectly aligned, showing a good generalization from the training dataset to the validation set.

The scores only slightly improve for ε smaller than 10^{-4} and are the best at 10^{-5}. For values even smaller the numerical instability negates any potential benefits. Meanwhile the number of iterations increased from 3000 iterations at $\varepsilon = 10^{-3}$ to 4300 at $\varepsilon = 10^{-4}$ and all the way up to 30200 at $\varepsilon = 10^{-5}$. The number of iterations did not increase beyond that, because the gradients became unstable for even smaller ε and the algorithm was terminated early. This instability is significantly worse with FP32 precision, but even with FP64 the gradient does not always converge to zero. Going forward we will use $\varepsilon = 10^{-4}$ as our default value for testing and $\varepsilon = 10^{-5}$ is recommended for best results.

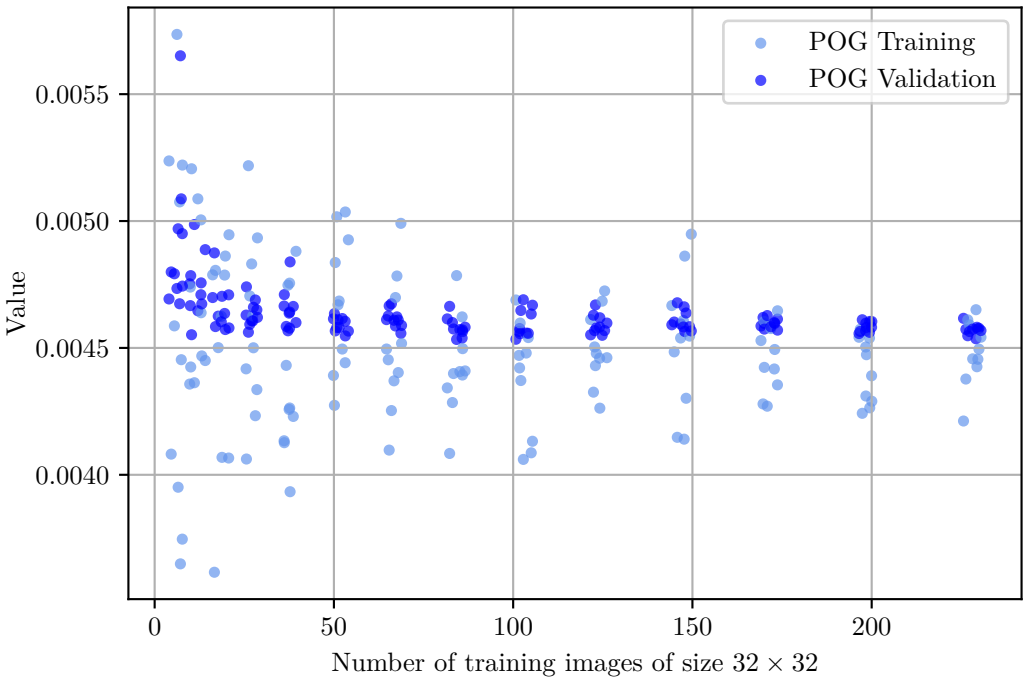


Figure 9: **Test III,** validation and training scores as a function of training data volume T. A majority of the variability between repeated runs can be attributed to the randomized selection of the training data and not the randomized initialization of the kernels or noise.

Test III examines the influence of additional training data, which is critical for two distinct reasons. First, sufficient training data is essential to mitigate overfitting. If only a limited number of images are used, the resulting model may perform exceptionally well on those specific images but fail to generalize to unseen data. Second, the training

data must be representative of the broader dataset. While one could manually curate the training set to ensure representativeness by using a validation set to guide selection, we instead chose to use a sufficiently large, randomly sampled training set to minimize the risk of selecting an unrepresentative subset.

In this test, the number of training images ranges from 4 to 225. For each run, the training data is renewed by randomly picking patches of size 32×32 from the images of the full dataset of 200 images at resolution of 481×321. The training process is repeated 10 times for each dataset size, and the results are visualized in the scatterplot of Figure 9.

The scatterplot reveals significant variability in training scores when only a small number of images are used. For instance, if the training data predominantly consists of flat or smooth regions, the model achieves better scores compared to a dataset with more complex textures. Both the training and validation scores are increasingly stable with additional training data, even though the training scores remain widely dispersed.

Furthermore, Figure 9 shows that some of the best validation scores are achieved with as few as 9 training images. This suggests that the small kernel $3 \times 3 \times 8$ requires barely any data to reach peak performance and overfitting is of little concern. However, by the wide spread of the validation scores one can deduce, that a random selection of 9 images is rarely representative of the entire dataset, causing unreliable results.

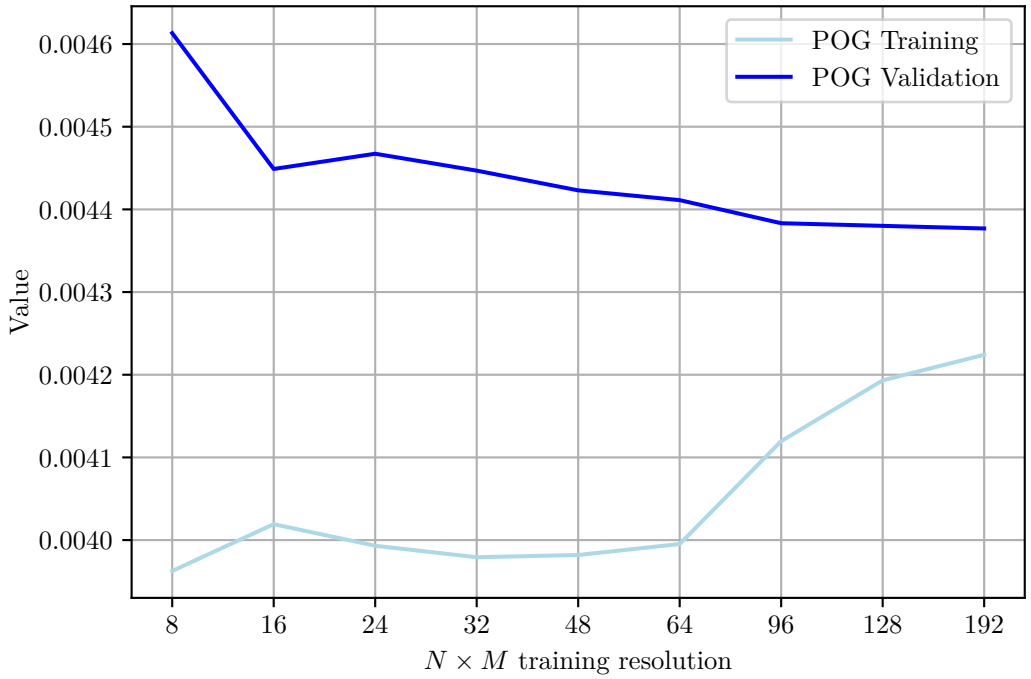


Figure 10: **Test IV,** validation and training scores as a function of training data resolution $N \times M$.

Test IV investigates the optimal resolution $N \times M$ for training images. While smaller

patches accelerate training, excessively small patches may fail to generalize to full-size images.

To ensure consistency across resolutions, the training process began with the highest-resolution data. All subsequent runs at lower resolutions used cropped versions of the same 100 images from the previous run. This approach eliminates variability introduced by selecting different image patches. The spread, as observed in Test III, would obscure the effect of the resolution on the performance.

The plot in Figure 10 demonstrates that using higher resolutions does indeed improve the outcome. However, using images of resolution 192×192 instead of 96×96 quadruples the training cost for only marginal improvements. For smaller resolutions, the reduced volume of training data leads to a noticeable decline in training scores, as the model has less information to fit. The kink at resolution 16×16, that breaks the monotonicity of the graph, contradicts the expectation. Repeating Test IV multiple times showed, that the scores for small resolutions exhibits strong variability and the monotonicity of the graph is not guaranteed. Increasing the amount of training images used for this experiment can reduce the chances of outliers, but the limiting factor was the memory requirements for the highest resolution images. The POG validation score remains remarkably flat across resolutions, despite spanning multiple orders of magnitude in input data.

The training time scales linearly with the amount of training data, meaning that doubling the resolution quadruples the computational cost. However, for the three smallest resolutions ($8 \times 8, 16 \times 16$, and 24×24), no measurable difference in training time was observed. This suggests that the GPU is not fully utilized for these small datasets, likely due to overhead or underutilization of parallel processing capabilities.

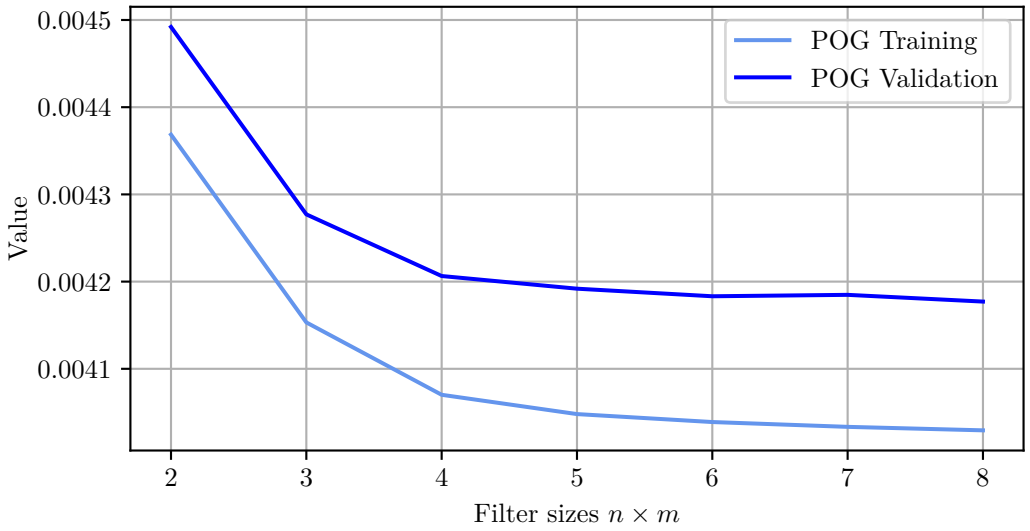


Figure 11: **Test V,** validation and training scores as a function of filter sizes $n \times m$.

Test V investigates the benefits of increasing the filter sizes. The tested sizes range from 2×2 to 8×8 and the padding p is set to half the filter size (rounded down) to

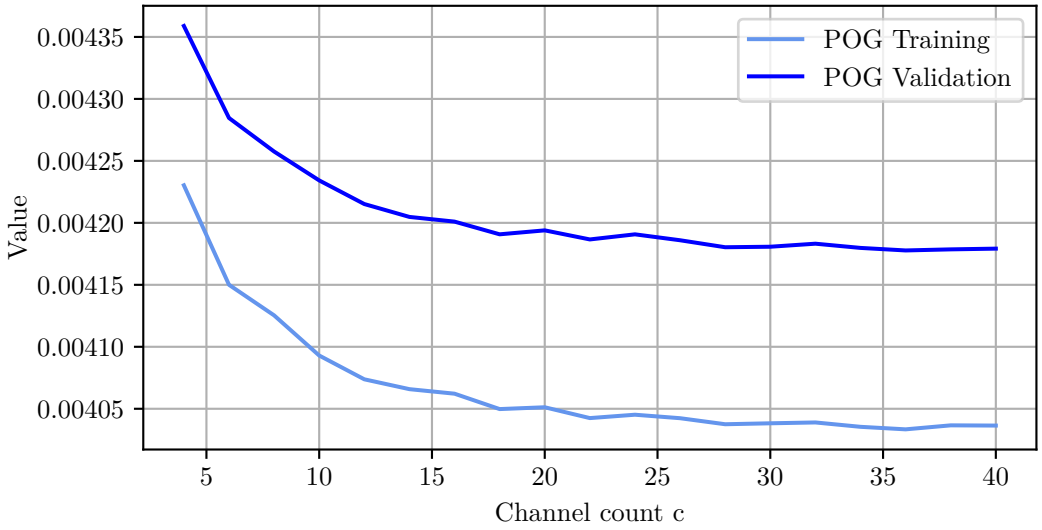


Figure 12: **Test VI**, validation and training scores as a function of the channel count c.

minimize boundary effects. The number of channels c is fixed at 32, isolating the effect of filter size. This choice contrasts with test VI, where the number of channels will be the primary focus.

The training data consists of 200 images of size of size 48×48 and ε was set to 10^{-5}. Using not enough training data would hide the benefits from using bigger kernels and result in almost the same validation score across all filter sizes.

Given the hierarchical nature of filter operations, a strictly monotonic relationship between filter size and performance is expected. Larger filters can emulate the behavior of smaller ones, but the converse is not true. However larger filter sizes come at the cost of increased computational cost and decreased numerical stability, which negates any potential gains beyond a certain size. The graph in Figure 11 shows that filters of size 6×6 are almost as good as filters of size 8×8 and no significant improvements for filters bigger than that should be expected.

Test VI investigates the impact of additional channels, ranging from 4 to 40. The filter size was fixed to 6×6 and 3 pixels of padding were used. Using more channels allows for more features to be analyzed by the convolution operator. Again a strictly monotonic behavior is expected with the scores improving with additional channels. However, the number of channels is one of the parameters that also increases the memory and compute requirements for inference.

The graph in Figure 12 confirms the expected monotonicity and using more channels improves the scores. The point of diminishing returns seems to be around 28 channels for filters of size 6×6.

In conclusion, maximizing the potential of this training method requires balancing the smoothing ε between accuracy and stability, maximizing the input in form of training data and maximizing the filter sizes and channels. The constraints are the memory

requirements, the compute and the numerical stability.

For our final configuration, we selected a smoothing parameter of $\varepsilon = 10^{-5}$, a training dataset size of $N \times M \times T = 96 \times 96 \times 200$ and a kernel size of $n \times m \times c = 9 \times 9 \times 80$. This setup required approximately 10GB of VRAM. The training process reached the self-imposed upper limit of 100,000 iterations after 14 hours and 15 minutes, achieving a final POG validation score of 0.004165.

Notably, a comparable score of 0.004166 was already attained at iteration 87,300 and the last 12,700 iterations improved the objective only in the last decimal place. While the stopping criterion could be adjusted to trigger earlier in future experiments, it was necessary to allow the training to run for an extended period at least once. This approach ensures a comprehensive understanding of the optimal stopping point, which might otherwise remain unclear.

As already observed for the TV denoising, the POG encourages less regularization than the MSE or PSNR. This behavior can be visualized by plotting the average PSNR and SSIM depending on a multiplicative factor applied to a kernel. Both plots in Figure 13 agree, that the maximum for the average PSNR and SSIM is actually attained around $\lambda = 1.3$, which results in a significant gain of 0.7 dB for the PSNR score. Note, that these plots are about linearly scaling all filters learned by POG by the same amount and even better results could be achieved by fine tuning each layer separately.

The analysis reveals that the trained penalty term is not inherently optimized for maximizing average PSNR and would require a multiplicative adjustment to enable a fair comparison with methods that directly target PSNR or SSIM. Notably, the presence of the term $\|Ax^\dagger[i]\|_{1,2}$ in the training objective not only promotes sparsity, but also acts as a penalty on $\|A\|_F$, provided that a sufficient amount of training data is available.

To elucidate this, consider the general case where the images $x^\dagger[i] \in \mathbb{R}^n$ are vectorized, and the linear operator $A \in \mathbb{R}^{m \times n}$ is represented as a matrix. The Frobenius norm of A is defined as $\|A\|_F = \sqrt{\langle A, A \rangle}$. Assume that the training dataset is sufficiently rich in linearly independent samples, such that the matrix $X^\dagger := \sum_{i=1}^{T} x^\dagger[i](x^\dagger[i])^* \in \mathbb{R}^{n \times n}$ is invertible. Under this assumption, the following chain of inequalities demonstrates the relationship between the penalty term and the Frobenius norm:

$$\sum_{i=1}^{T} \|Ax^\dagger[i]\|_{1,2} \geq \sum_{i=1}^{T} \|Ax^\dagger[i]\|_2 = \sum_{i=1}^{T} \sqrt{\langle Ax^\dagger[i], Ax^\dagger[i] \rangle} \tag{4.1}$$

$$= \sum_{i=1}^{T} \sqrt{\langle x^\dagger[i], A^*Ax^\dagger[i] \rangle} = \sum_{i=1}^{T} \sqrt{\langle x^\dagger[i](x^\dagger[i])^*, A^*A \rangle} \tag{4.2}$$

$$\geq \sqrt{\langle X^\dagger, A^*A \rangle} = \sqrt{\langle A(X^\dagger)^{\frac{1}{2}}, A(X^\dagger)^{\frac{1}{2}} \rangle} \tag{4.3}$$

$$= \|A(X^\dagger)^{\frac{1}{2}}(X^\dagger)^{-\frac{1}{2}}(X^\dagger)^{\frac{1}{2}}\|_F \geq \|A\|_F \|(X^\dagger)^{\frac{1}{2}}\|_F. \tag{4.4}$$

This derivation illustrates that the proposed objective inherently prevents the operator A from becoming unbounded. Thus, the penalty term not only regularizes the solution

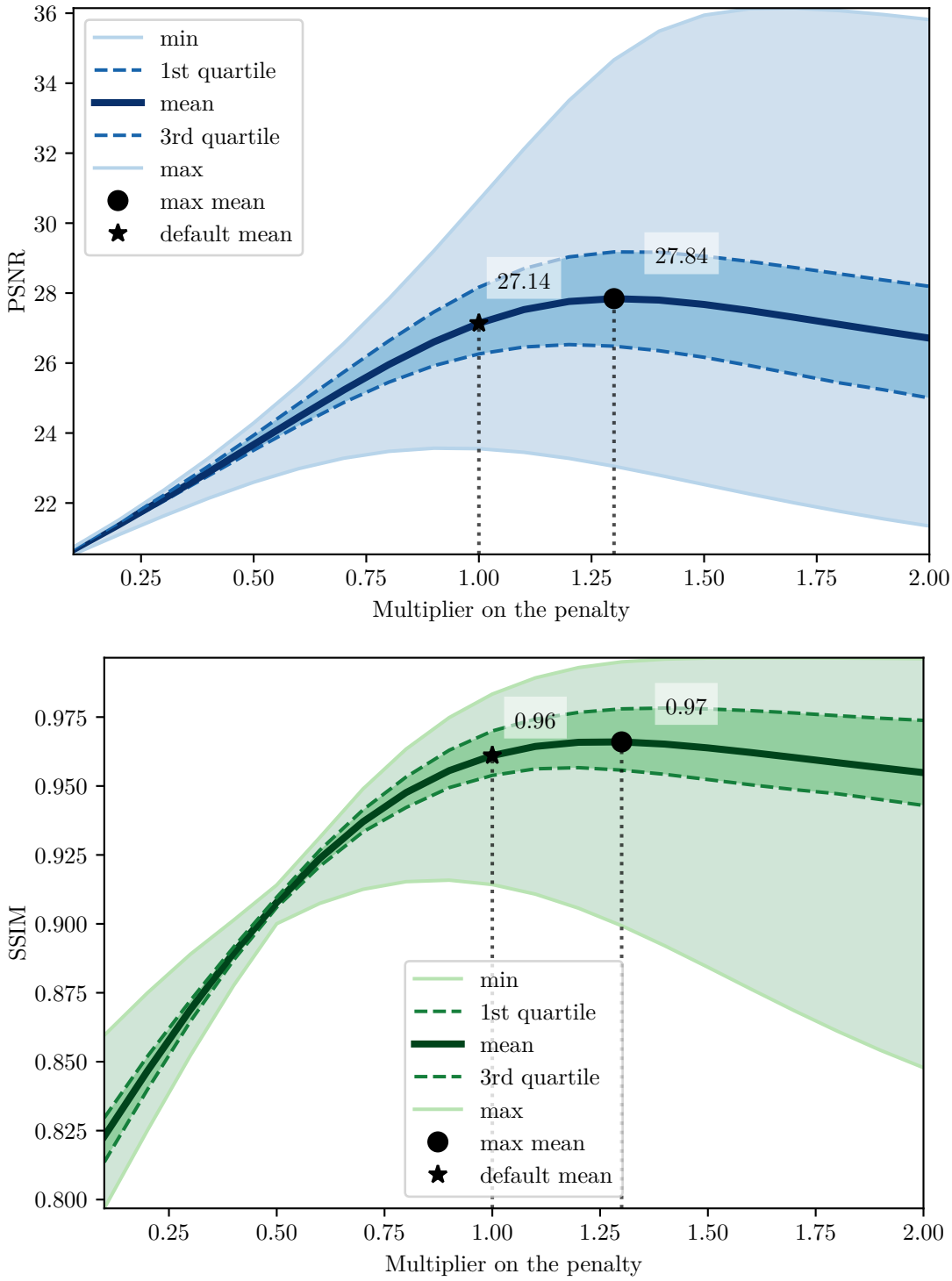


Figure 13: Statistics on the PSNR and SSIM of 100 denoised images from the validation set, depending on a multiplicative factor on the trained filters.

but also constrains the operator norm, ensuring stability in the training process.

4.3 Investigating the learned regularizer

Understanding the features encoded by the algorithm into the convolution kernel, and thus the regularizer, during training is of significant interest. To this end, we analyze our most effective kernel, which has a size of 9×9 with 80 channels and was trained on 200 images of resolution 96×96.

We begin by presenting a side-by-side comparison of the original image, the noisy image, and the denoised image in Figure 14. For this comparison, a multiplicative factor of $\lambda = 1.4$ was applied to the regularizer. The visual assessment demonstrates that the denoising process successfully suppresses a substantial portion of the noise. While the reconstruction does not achieve perfection, an outcome that is inherently unrealistic given the severity of the noise and the ill-posed nature of the problem, the results represent a meaningful improvement over the corrupted input.

For instance, in the image depicting a historical warplane, the clear sky regions between the clouds appear almost perfectly homogeneous in the ground truth, whereas the denoised version retains some residual artifacts. Additionally, finer details around the mechanical components of the plane are not fully restored and appear blurred. This limitation is understandable when considering the severity of the Gaussian noise ($\sigma = 0.1$), which obliterates much of the fine detail. Restoring such intricate features would require prior knowledge of the expected structure, a capability inherent in many modern deep learning denoisers. However, these methods often lack transparency, leaving users uncertain about the origin of the restored information: whether it stems from the noisy input or is generated by the model. In contrast, our approach, based on learned linear filters, avoids such "deceptive" reconstructions due to its inherent simplicity.

Similarly, the image of the zebra in Figure 14 demonstrates that denoising comes at the cost of reduced contrast, shifting the black and white stipes of the zebra to light and dark gray. Furthermore, individual blades of grass cannot be restored, resulting in a blurry and somewhat washed-out appearance of the grassy field.

Conversely, the restoration of irregular and out-of-focus backgrounds, as seen in the image of a woman at a market, is relatively straightforward. Unfortunately, the same cannot be said for the woman's face. Due to the high sensitivity of human perception to even minor irregularities in facial features the denoised portrait appears significantly degraded compared to the other images.

Plotting all 80 learned kernels in Figure 15 reveals several notable patterns. Most prominently, the pairwise coupling via the ℓ^2-norm resulted in many visually similar filter pairs. This suggests that the ℓ^2-norm coupling introduces a degree of redundancy among the filters. While the ℓ^2-coupling is important for the isotropic properties in the total variation method, another form of rotational invariance appeared independent of the ℓ^2-norm. Specifically, similar filter structures appeared repeatedly in different orientations, indicating that the method implicitly encodes rotational symmetry anyway.

The learned filters in Figure 15 also include several well-known operators from liter-

Figure 14: The figure presents a direct comparison between the denoised image (right), the image corrupted with Gaussian noise ($\sigma = 0.1$, middle), and the clean reference image (left), which is sourced from the BSDS500 testing dataset. For detailed inspection, magnified highlights of the full-resolution images (481×321) are provided below.

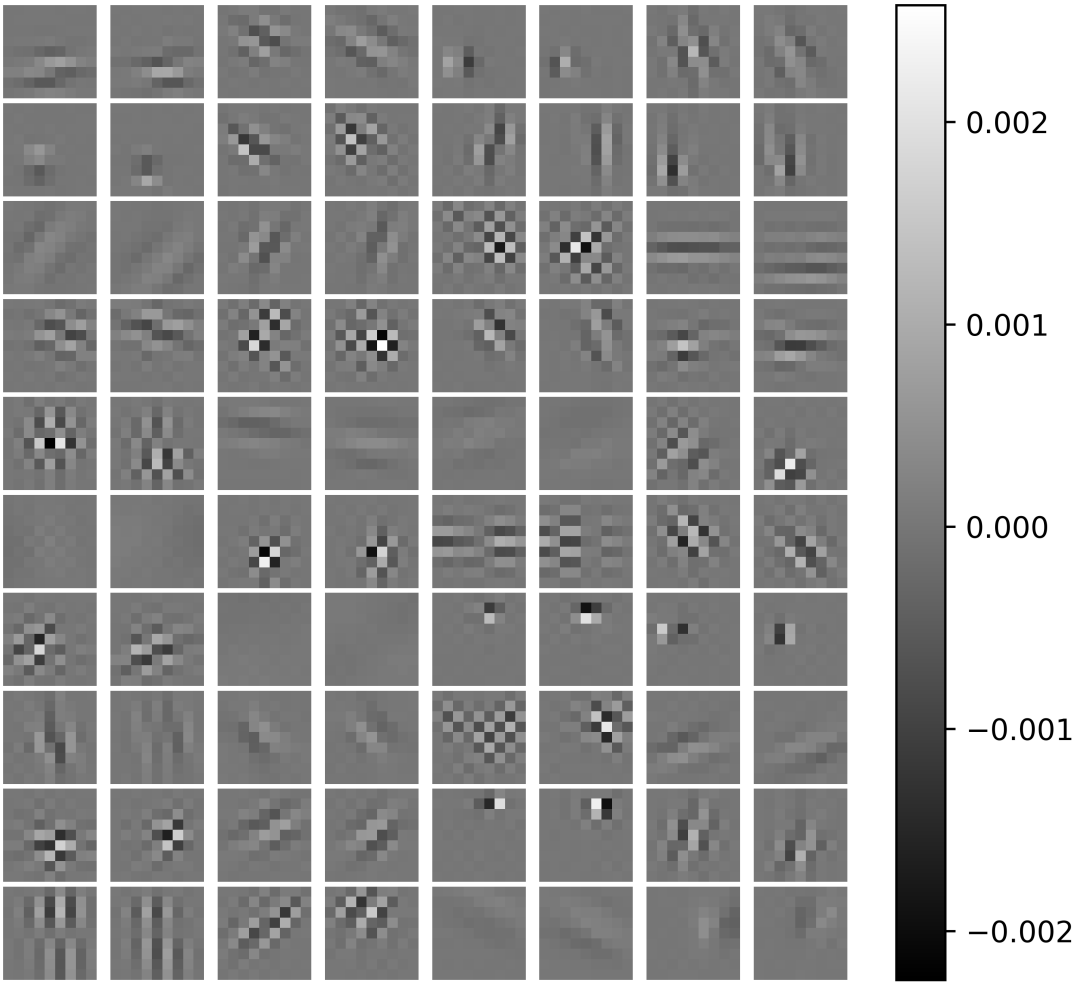


Figure 15: 80 convolution kernels of size 9×9 learned by our method.

ature. For instance, the discrete difference operator on which total variation denoising is based on, appears multiple times. These filters are characterized by a black pixel adjacent to a white pixel within an otherwise gray filter, reflecting their role in capturing local image gradients.

Some filters exhibit periodic or oscillatory patterns, reminiscent of basis functions in a Discrete Fourier Transform (DFT) or Discrete Cosine Transform (DCT). This observation suggests that the method has learned to decompose images into frequency components, a common and effective strategy in denoising applications.

Finally, a subset of the filters appears as uniform gray patches, effectively acting as null filters that contribute nothing to the regularizer. Their appearance is not entirely unexpected, as these null filters represent stationary points in the training problem. Once a filter's norm drops to zero during training, it remains inactive for the duration

of the process. To mitigate this issue, future work could explore re-initializing filters whose norms fall below a predefined threshold during training or pruning redundant filters post-training. Such adjustments could enhance both the diversity of the learned filters and the computational efficiency of the method.

Figure 16 illustrates the penalty computed for each pixel (i, j) of an image x. The penalty is derived by pairing the 80 filtered images to 40 images with non negative values. Finally, their contributions are accumulated into a single image:

$$\sum_{\substack{l=1 \\ l \text{ odd}}}^{c} \sqrt{(K^l * x)_{i,j}^2 + (K^{l+1} * x)_{i,j}^2}. \tag{4.5}$$

The total penalty is obtained by accumulating these values across all pixels. In the figure, the colors are inverted and normalized, such that white pixels correspond to a penalty of zero, while darker pixels indicate stronger penalties.

The noisy image in the center appears significantly darker than the other two, reflecting the detection of numerous irregularities by the penalty term. The blurriness in the dual images arises from the 9×9 filters, which diffuse sharp irregularities across neighboring pixels.

In an ideal scenario, if the penalty term were perfectly tailored to the ground truth image (left), it would appear completely white, as no penalty should be applied. However, achieving this with a simple set of linear filters is inherently impossible.

The denoised image (right) visualizes the sparsity induced by the $\ell^{1,2}$-norm in the co-domain. Most pixels remain white, with darker regions concentrated along edges. This sparsity-promoting behavior contrasts with penalties based on the ℓ^2-norm or other non-sparsity-inducing functions, which would distribute dark pixels more uniformly and thus reduce denoising effectiveness. Conversely, non-convex sparsity-promoting functions could further concentrate dark regions, enhancing contrast and sharpness.

4.4 Comparison to other methods

To evaluate the performance of our proposed method, we conduct a comparative study using the 200 test images from the BSDS500 dataset converted to grayscale with additive Gaussian noise of standard deviation 0.1 (coinciding with an expected PSNR of 20.00dB). We compare five different denoising approaches:

1. TV denoising with a manually chosen weight of $\lambda = 0.1$.

2. Our method with filters of size $9 \times 9 \times 80$ for a total of 6480 learned parameters and weight $\lambda = 1.3$ for better performance with respect to PSNR and SSIM.

3. Chen et al. (2014) Filters [13]: This method is closely related to ours and uses filters of size $7 \times 7 \times 48$ (= 2352 learned parameters), provided by the authors. The main difference is the non-convex $\log(1 + z^2)$ response function as described in the paper. These filters were trained on the BSDS300 dataset, which is a predecessor of the BSDS500.

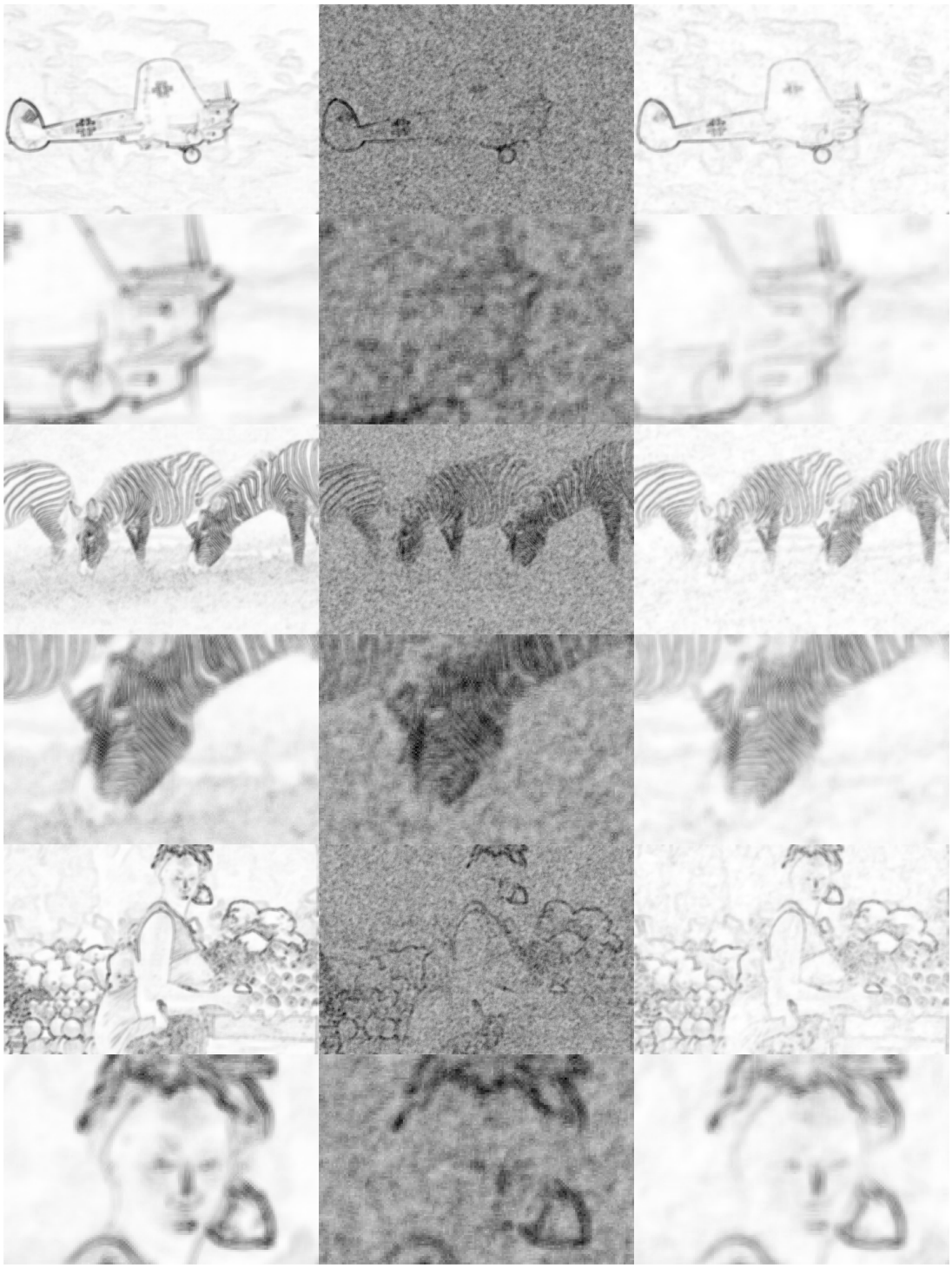


Figure 16: Dual representation of the denoised images. The figure visualizes the pixel-wise penalty for the ground truth (left), noisy (middle), and denoised (right) images, with white corresponding to no penalty and black to strong penalties highlighting irregularities.

4. BM3D [17]: Block-Matching and 3D Filtering (BM3D) is a state-of-the-art denoising algorithm that relies solely on the statistics of the image itself and does not require a training dataset. It exploits the non-local self-similarity of natural images by grouping similar 2D patches into 3D stacks and applying collaborative filtering to these stacks. BM3D is widely regarded as one of the best traditional denoising methods and serves as a strong baseline for comparison.

5. DRUNet [21]: A deep residual U-Net with ∼3 million parameters designed for image restoration tasks, including denoising. DRUNet leverages deep learning to learn complex mappings from noisy to clean images, achieving impressive results across a variety of noise levels and image types.

For methods 4 and 5, we use the implementations provided by the DeepInverse Python library. It is important to note that the deep learning-based method DRUNet would likely benefit from fine-tuning on the BSDS500 dataset. However, due to hardware limitations, we use the pre-trained model provided by the library without further refinements.

Method	PSNR	SSIM
TV	26.73	0.9552
Ours	28.03	0.9683
Chen	28.35	0.9692
BM3D	28.72	0.9731
DRUNet	29.66	0.9782

Table 3: Evaluation of 5 denoising techniques on the BSDS500 test dataset with Gaussian noise $\sigma = 0.1$.

Beyond quantitative metrics, we include a visual comparison of the denoising results, as these averaged scores cannot fully capture the nuances. Figures 17 and 18 show two complimentary images from the BSDS500 dataset, along with the denoised outputs of each method. The visual comparison highlights the strengths and weaknesses of each approach.

The image of the elephant in Figure 17 illustrates the challenges associated with denoising complex, unstructured textures, such as elephant skin and foliage. Aggressive region flattening proves counterproductive in this context, as it results in a loss of detail. The grass field becomes blurry and the elephant loses its characteristic rough, weathered texture. The TV denoiser, in particular, struggles with the grass and trees in the background, merging individual blades and leaves into indistinct blobs.

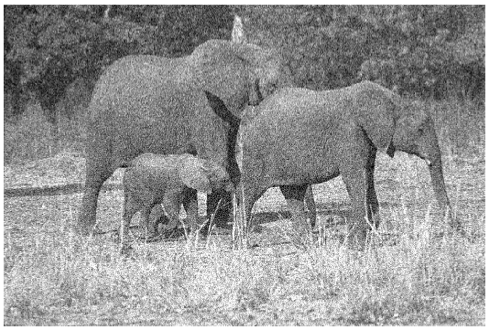
Despite achieving lower PSNR and SSIM scores, our method excels at preserving the overall roughness of the image while effectively removing spatially uncorrelated Gaussian noise. The approach proposed by Chen et al. not only shares theoretical similarities with our method but also produces comparable results. BM3D attains a similar quantitative score to our method but yields a noticeably softer and less textured appearance.

DRUNet, while achieving the highest score, produces an image so remarkably clean that it is difficult to reconcile with the heavily noise-corrupted input. However, this deep learning-based denoiser also introduces artifacts, such as an unnaturally clean and rosy depiction of the baby elephant, deviating from the ground truth.

The image of the Ferrari 288 GTO in Figure 18 provides an ideal case study for evaluating the edge-preserving properties of each denoising method, in combination with strongly warped reflections on curved body panels. While all methods are designed to preserve edges, subtle differences emerge in their treatment of fine details, such as the cooling fins composed of numerous small, parallel slots. For instance, TV denoising tends to merge some of these parallel lines into flat regions, compromising structural fidelity.

The three variational methods, TV denoising, our approach, and that of Chen et al., struggle to retain the reflective, glossy appearance of the wet body panels, resulting in a more matte finish. DRUNet, on the other hand, produces an overly refined output, lending the car an almost rendered, rather than photographed, appearance. Although the BM3D-denoised image may subjectively appear the most visually appealing, this preference is not reflected in quantitative metrics. DRUNet achieves the highest PSNR and SSIM scores. Finally, none of the methods successfully restore the granular texture of the asphalt or the raindrops on the body panels, as these features are structurally similar to the Gaussian noise and thus difficult to distinguish during denoising.

(a) Ground Truth



(b) Noisy ($\sigma = 0.1$)

(c) TV ($\lambda = 0.1$)

(d) Ours

(e) FoE (Chen 2014)

(f) BM3D

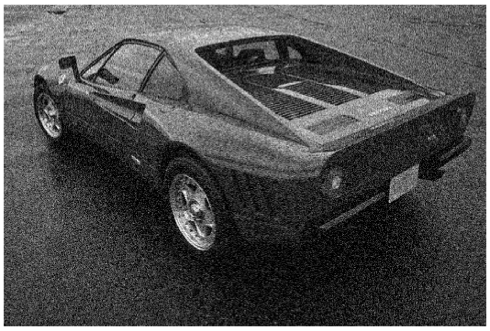
(g) DRUNet

Method	PSNR	SSIM
TV	25.45	0.9466
Ours	26.76	0.9594
Chen	26.97	0.9612
BM3D	26.81	0.9591
DRUNet	27.44	0.9653

(h) PSNR and SSIM scores

Figure 17: Comparison of five denoising techniques applied to an image of elephants. Note the difficulty in preserving the intricate texture of the elephant skin and the complex structures of foliage.

(a) Ground Truth



(b) Noisy ($\sigma = 0.1$)

(c) TV ($\lambda = 0.1$)

(d) Ours

(e) FoE (Chen 2014)

(f) BM3D

(g) DRUNet

Method	PSNR	SSIM
TV	28.08	0.9689
Ours	29.07	0.9745
Chen	29.65	0.9774
BM3D	30.21	0.9794
DRUNet	31.30	0.9837

(h) PSNR and SSIM scores

Figure 18: Comparison of five denoising applied to an image of a ferrari. This scenario presents a significant challenge for our method, particularly due to its difficulty in handling flat and smooth surfaces. While aggressive smoothing may enhance the appearance of the car, it often inadvertently removes critical details, such as raindrops on the body panels and the granular texture of the asphalt.

5 Conclusion

Image denoising is a fundamental problem in image processing, where the goal is to recover a clean image from noisy observations. Traditional variational methods, such as Total Variation (TV) denoising, rely on fixed linear operators like discrete gradients to enforce regularity. While these methods are interpretable, they lack adaptability to the specific statistics of natural images. On the other hand, deep learning-based approaches achieve state-of-the-art results but often require large amounts of training data and lack transparency in their decision-making process. This work uses a method in between by learning linear filters for the penalty term in a variational denoising framework, combining the strengths of both approaches.

The denoising problem is formulated as an optimization problem, where the denoised image is obtained by minimizing a combination of a fidelity term and a penalty term. The fidelity term ensures that the solution remains close to the noisy measurements, while the penalty term enforces regularity, such as sparsity in a transformed domain. Specifically, the penalty term is defined using a linear operator A, which consists of a set of convolutional filters, and a mixed $\ell^{1,2}$-norm that promotes sparsity. The objective is to learn the operator A such that the denoised images are as close as possible to the ground truth.

A key contribution of this work is the use of the primal objective gap (POG) as an objective function for learning the filters, which can also be expressed by the primal-dual gap. The primal-dual gap measures the difference between the primal objective evaluated at the ground truth and the denoised image, providing a direct connection to the denoising problem. This approach avoids the need for a bi-level optimization scheme, which can be computationally expensive and unstable. Instead, the problem is reformulated as a single-level optimization using the Proximal Alternating Linearized Minimization (PALM) algorithm enhanced by inertia terms. iPALM is well-suited for this task because it can handle non-convex problems and is computationally efficient.

Empirical validation is conducted using the BSDS500 dataset, a standard benchmark for image processing tasks. The dataset consists of natural images, which are corrupted with additive Gaussian noise to simulate real-world noise conditions. The proposed method is evaluated against several baseline approaches, including TV denoising, BM3D, and a deep learning-based method (DRUNet). The results show that the learned filters achieve competitive performance, particularly in preserving structural details while effectively removing noise. For example, with filters of size $7 \times 7 \times 48$, the method achieves an average PSNR of 27.79 dB and SSIM of 0.9656, outperforming TV denoising and approaching the performance of BM3D and DRUNet.

The learned filters resemble edge detectors, indicating that the model effectively captures important image structures. Visualizations in the dual space demonstrate how the

penalty term suppresses noise while preserving edges, highlighting the effectiveness of the $\ell^{1,2}$-norm in promoting sparsity.

In conclusion, this work demonstrates that learning linear filters within a variational framework can achieve competitive denoising performance while maintaining interpretability.

5.1 Future research

One of the central challenges in this research was the minimization of the non-convex and non-smooth objective (3.52) inherent in the training process. While replacing the $\|\cdot\|_{1,2}$ norm with $\|\cdot\|_\varepsilon$ rendered the objective differentiable and but still non-convex, this approach introduced trade-offs. The method performed well for small kernels but encountered limitations with larger kernels due to restricted step sizes and local convergence rates.

Furthermore, for certain configurations the numerical precision of the gradient calculation prevented convergence. This issue could be attacked on multiple fronts. For a starter, one could improve the implementation of the code by examining each and every operation in the code and quantifying the floating point operation errors. However, this would demand substantial time and effort, particularly if errors accumulate across operations. While such refinements might enhance the reliability of gradient convergence ($\nabla_K H \to 0$), the resulting improvements in image quality could remain marginal.

An alternative strategy involves further modifying the objective function. Drawing inspiration from denoising problems, where non-smooth terms are circumvented using convex conjugates, this technique could potentially be adapted to the non-smooth components of the training objective. Nevertheless, the coupling with the variable Y necessitates deeper investigation to ensure feasibility and effectiveness.

To address the issue of rotational symmetry and redundant filters, a potential avenue for future research could involve explicitly incorporating rotational invariance into the filter design. Specifically, each filter K could be applied twice: once in its original form and once rotated by 90 degrees, denoted as $\hat{K}$. The current pairwise coupling term

$$\sum_{\substack{i,j \\ l \text{ odd}}} \sqrt{(K^l * x)_{i,j}^2 + (K^{l+1} * x)_{i,j}^2}, \tag{5.1}$$

could be replaced by a rotationally symmetric variant:

$$\sum_{i,j,l} \sqrt{(K^l * x)_{i,j}^2 + (\hat{K}^l * x)_{i,j}^2}. \tag{5.2}$$

This modification has the potential to halve the number of required channels by inherently enforcing rotational invariance. However, it may also impact the final performance of the method, as the learned filters might lose some flexibility in adapting to anisotropic features. Further empirical testing would be necessary to evaluate the trade-offs between computational efficiency and denoising effectiveness.

Beyond refining the existing formulation, several avenues for expanding the denoiser's capabilities warrant exploration. These include extending the denoiser to handle color images or investigating the behavior of learned filters under varying noise models and strengths.

To further enhance the performance of the denoiser, particularly in extremely flat regions such as the sky, additional pre- or post-training stages could be introduced. The current training objective does not explicitly prioritize strong denoising in such regions which leads to distracting visual artifacts. One potential solution involves modifying the objective function during pre-training by incorporating a bias based on the local variance of the ground truth data or a similar metric.

Alternatively, a post-training approach could be employed to refine the learned filters using a different quality measure. While we have explored this concept in a one-dimensional setting, by analyzing the relationship between PSNR and SSIM with respect to a global multiplier applied to the regularizer (as illustrated in Figure 13), a more sophisticated and filter-specific adjustment could yield superior results.

Such post-training adjustments would also enable the replacement of the ℓ^1-norm in the regularizer with non-convex response functions that promote stronger sparsity, such as the ℓ^p-norm for $p \in (0, 1)$ or $\ln(1 + |z|)$. Emphasizing sparsity in this manner is expected to enhance edge contrast and sharpness while producing fully flattened regions. However, it may also introduce new types of artifacts, which would need to be carefully evaluated.

While convex formulations and linear filters offer significant advantages from an optimization perspective, they also come with inherent limitations in achieving peak performance in benchmarks. Nevertheless, their simplicity and computational efficiency make them highly attractive for addressing a wide range of inverse problems. In recent years, the plug-and-play framework [22] has emerged as a powerful paradigm, harvesting the work put into training denoisers and putting them to more complex tasks such as super-resolution and deblurring.

Exploring these and other applications, where learned filters can be leveraged in innovative and impactful ways, presents a promising and fruitful direction with ongoing research.

In summary, this work highlights the value of combining established optimization methods with modern data-driven approaches to push the boundaries of research in inverse problems. By thoroughly investigating the interplay between convex formulations, linear filters, and sparsifying transformations, we aim to contribute to the development of faster, more stable, and reliable solutions. While much remains to be explored, this thesis represents a step toward deeper understanding and practical advancements in the field.

Bibliography

[1] Pablo Arbelaez et al. “Contour Detection and Hierarchical Image Segmentation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.5 (May 2011), pp. 898–916. DOI: 10.1109/TPAMI.2010.161.

[2] Kenneth Joseph Arrow et al. *Studies in linear and non-linear programming*. Vol. 2. Stanford University Press Stanford, 1958.

[3] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. “Proximal alternating linearized minimization for nonconvex and nonsmooth problems”. In: *Mathematical Programming* 146.1 (2014), pp. 459–494.

[4] Boban Bondžulić et al. “Efficient prediction of the first just noticeable difference point for JPEG compressed images”. In: *Acta Polytechnica Hungarica* 18.8 (2021), pp. 201–220.

[5] Kristian Bredies and Dirk Lorenz. *Mathematische Bildverarbeitung*. 1st ed. Springer, 2011.

[6] Lev M Bregman. “The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming”. In: *USSR Computational Mathematics and Mathematical Physics* 7.3 (1967), pp. 200–217.

[7] Emmanuel J Candès and David L Donoho. “Ridgelets: A key to higher-dimensional intermittency?” In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 357.1760 (1999), pp. 2495–2509.

[8] Antonin Chambolle. “An algorithm for total variation minimization and applications”. In: *Journal of Mathematical Imaging and Vision* 20.1 (2004), pp. 89–97.

[9] Antonin Chambolle and Thomas Pock. “A first-order primal-dual algorithm for convex problems with applications to imaging”. In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145.

[10] Antonin Chambolle and Thomas Pock. “On the ergodic convergence rates of a first-order primal–dual algorithm”. In: *Mathematical Programming* 159.1 (2016), pp. 253–287.

[11] Tony F Chan, Gene H Golub, and Pep Mulet. “A nonlinear primal-dual method for total variation-based image restoration”. In: *SIAM Journal on Scientific Computing* 20.6 (1999), pp. 1964–1977.

[12] Venkat Chandrasekaran et al. “Representation and compression of multidimensional piecewise functions using surflets”. In: *IEEE Transactions on Information Theory* 55.1 (2008), pp. 374–400.

[13] Yunjin Chen, Rene Ranftl, and Thomas Pock. “Insights into analysis operator learning: From patch-based sparse models to higher order MRFs”. In: *IEEE Transactions on Image Processing* 23.3 (2014), pp. 1060–1072.

[14] Enis Chenchene, Alireza Hosseini, and Kristian Bredies. “A hybrid proximal generalized conditional gradient method and application to total variation parameter learning”. In: *2023 European Control Conference (ECC)*. IEEE. 2023, pp. 1–6.

[15] Il Yong Chun and Jeffrey A Fessler. “Convolutional analysis operator learning: Acceleration and convergence”. In: *IEEE Transactions on Image Processing* 29 (2019), pp. 2108–2122.

[16] Il Yong Chun et al. “Convolutional analysis operator learning: Dependence on training data”. In: *IEEE Signal Processing Letters* 26.8 (2019), pp. 1137–1141.

[17] Kostadin Dabov et al. “Image denoising by sparse 3-D transform-domain collaborative filtering”. In: *IEEE Transactions on Image Processing* 16.8 (2007), pp. 2080–2095.

[18] David L Donoho. “Wedgelets: Nearly minimax estimation of edges”. In: *the Annals of Statistics* 27.3 (1999), pp. 859–897.

[19] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

[20] Rafael C Gonzalez. *Digital image processing*. Pearson education india, 2009.

[21] Mina Jafari et al. “DRU-Net: an efficient deep convolutional neural network for medical image segmentation”. In: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE. 2020, pp. 1144–1148.

[22] Ulugbek S Kamilov et al. “Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications”. In: *IEEE Signal Processing Magazine* 40.1 (2023), pp. 85–97.

[23] Markku Makitalo and Alessandro Foi. “Optimal Inversion of the Generalized Anscombe Transformation for Poisson-Gaussian Noise”. In: *IEEE Transactions on Image Processing* 22.1 (2013), pp. 91–103. DOI: 10.1109/TIP.2012.2202675.

[24] Thomas Pock and Shoham Sabach. “Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems”. In: *SIAM Journal on Imaging Sciences* 9.4 (2016), pp. 1756–1787.

[25] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 1970.

[26] Stefan Roth and Michael J Black. “Fields of experts”. In: *International Journal of Computer Vision* 82.2 (2009), pp. 205–229.

[27] Ron Rubinstein, Alfred M Bruckstein, and Michael Elad. “Dictionaries for sparse representation modeling”. In: *Proceedings of the IEEE* 98.6 (2010), pp. 1045–1057.

[28] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: Nonlinear Phenomena* 60.1-4 (1992), pp. 259–268.

[29] Kegan GG Samuel and Marshall F Tappen. “Learning optimized MAP estimates in continuously-valued MRF models”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 477–484.

[30] Suvrit Sra. “Fast projections onto mixed-norm balls with applications”. In: *Data Mining and Knowledge Discovery* 25.2 (2012), pp. 358–377.

[31] J.L. Starck, F. Murtagh, and J.M. Fadili. *Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity*. Cambridge University Press, 2010. ISBN: 9780521119139.

[32] Jean-Luc Starck, Emmanuel J Candès, and David L Donoho. “The curvelet transform for image denoising”. In: *IEEE Transactions on Image Processing* 11.6 (2002), pp. 670–684.

[33] Michael B Wakin et al. “Wavelet-domain approximation and compression of piecewise smooth images”. In: *IEEE Transactions on Image Processing* 15.5 (2006), pp. 1071–1087.

[34] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[35] Rebecca M Willett and Robert D Nowak. “Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging”. In: *IEEE Transactions on Medical Imaging* 22.3 (2003), pp. 332–350.

[36] Mehrdad Yaghoobi et al. “Analysis operator learning for overcomplete cosparse representations”. In: *2011 19th European Signal Processing Conference*. IEEE. 2011, pp. 1470–1474.

[37] Mehrdad Yaghoobi et al. “Constrained overcomplete analysis operator learning for cosparse signal modelling”. In: *IEEE Transactions on Signal Processing* 61.9 (2013), pp. 2341–2355.

[38] Mingqiang Zhu and Tony Chan. “An efficient primal-dual hybrid gradient algorithm for total variation image restoration”. In: *UCLA Cam Report* 34.2 (2008).