Preconditioned Plug-and-Play ADMM with Locally Adjustable Denoiser for Image Restoration Mikael

Mikael Le Pendu, Christine Guillemot

Preconditioned Plug-and-Play ADMM with Locally Adjustable Denoiser for Image Restoration*

Mikael Le Pendu† and Christine Guillemot†

Abstract. Plug-and-Play priors recently emerged as a powerful technique for solving inverse problems by plugging a denoiser into a classical optimization algorithm. The denoiser accounts for the regularization and therefore implicitly determines the prior knowledge on the data, hence replacing typical hand-crafted priors. In this paper, we extend the concept of Plug-and-Play priors to use denoisers that can be parameterized for non-constant noise variance. In that aim, we introduce a preconditioning of the ADMM algorithm, which mathematically justifies the use of such an adjustable denoiser. We additionally propose a procedure for training a convolutional neural network for high quality non-blind image denoising that also allows for pixel-wise control of the noise standard deviation. We show that our pixel-wise adjustable denoiser, along with a suitable preconditioning strategy, can further improve the Plug-and-Play ADMM approach for several applications, including image completion, interpolation, demosaicing and Poisson denoising.

Key words. Plug-and-Play Prior, Inverse Problems, Preconditioning, ADMM, Denoising, Interpolation

MSC codes. 62H35, 68U10, 94A08, 68T99

1. Introduction.

Inverse problems arising in image restoration require the use of prior knowledge on images in order to determine the most likely solutions among an infinity of possibilities. Traditional optimization methods rely on priors modeled as convex regularization functions such as the total variation, encouraging smoothness, or the l_1 norm for sparsity. In this context, convex optimization algorithms have played an important role and their convergence properties have been well-established. However, the methods based on "hand-crafted" convex priors are now significantly outperformed by deep neural networks that directly learn an inverse mapping from the degraded measurements to the solution space. Here, the prior knowledge on images and the degradation to recover from do not need a formal mathematical definition. Instead, they are implicitly taken into account when training the network from a large dataset of degraded images (i.e. network's input) along with their original versions (i.e. network's output). This approach has enabled a significant leap in performance for common image restoration problems including denoising, demosaicing, super-resolution, etc. However, different neural networks must be carefully designed and trained for each problem specifically. Furthermore, it is usually impossible to interpret what task is performed by the different layers or sub-modules of the trained network which acts as a black box.

In order to increase the interpretability and genericity of deep neural networks, the field is evolving towards methods that combine them with traditional optimization algorithms.

*Submitted to the editors November 17, 2022

Funding: This work was supported in part by the french ANR research agency in the context of the artificial intelligence project DeepCIM, and in part by the EU H2020 Research and Innovation Programme under grant agreement No 694122 (ERC advanced grant CLIM).

†Inria Rennes – Bretagne-Atlantique, 263 Avenue Général Leclerc, 35042 Rennes Cedex, France (e-mail: firstname.lastname@inria.fr)

approach, seen for example in [1–8], consists in defining so-called "unrolled" neural networks that simulate several iterations of an optimization algorithm. For instance, in unrolled gradient descent, the gradient of the data-fidelity term can be computed knowing its closed form expression, while a convolutional neural network (CNN) is used to simulate the gradient computation of a regularization term. End-to-end training of this regularizing network is then performed by "unrolling" a given number of iterations of the optimization algorithm. Here, the trained CNN is meant to represent only prior knowledge on images, which does not depend on a specific image restoration task. However, in practice these networks require re-training for each task, meaning that task-specific features are also learnt in the end-to-end unrolled training.

A more universal approach referred to as "Plug-and-Play" (PnP), relies on proximal algorithms such as the Alternating Direction Method of Multipliers (ADMM) [9] or Half Quadratic Splitting (HQS) [10], where the proximal operator of the regularization term is usually replaced by a denoiser as in [11–15]. While Venkatakrishnan et al. [11] introduced this approach using traditional denoising techniques including BM3D [16] or K-SVD [17], more recent methods attain higher performance thanks to neural networks trained for denoising. The advantage compared to unrolled methods is that the neural network component is only trained for Gaussian denoising, and can then be used generically for solving several tasks. However, unrolled methods can reach even higher quality with fewer iterations by task-specific training, which highlights the necessity of embedding task-related features in the denoiser. Our aim is then to introduce a more general mathematical framework for Plug-and-Play that enables advanced task-specific tunable denoising, while preserving the interpretability, and thus the genericity of the approach.

More specifically, we propose in this paper a preconditioning of the ADMM in order to improve the algorithm's performance in the Plug-and-Play scenario. While the proposed formulation remains mathematically equivalent to the original problem, the preconditioning makes it possible to use a denoiser that takes into account spatially varying noise levels. This is particularly advantageous for applications where the variance of the approximation error made at each iteration varies spatially in a predictable way. For instance, in image demosaicing, completion, or interpolation, only the unknown pixels that require interpolation are expected to have an approximation error. Hence, the pattern of known and unknown pixels can be used in our scheme to define a sensible preconditioning matrix. Likewise, when an image is corrupted by Poisson noise, the noise variance at each pixel is known to be proportional to the noiseless pixel intensity. Note that the term "preconditioned ADMM" has also been used in the literature (e.g. [18, 19]) in reference to a different version of the ADMM designed for cases where the sub-problems are difficult to solve, which is not related to the proposed preconditioning scheme. In summary, the main contributions of our paper are as follows:

- We formulate a preconditioned ADMM, which enables the use of a denoiser that can take into account spatially varying standard deviation.
- We propose a procedure for training a denoising neural network that takes as input a map of pixel-wise standard deviation, along with the image to be denoised.
- We demonstrate the effectiveness of the approach and define suitable preconditioning schemes for several applications: image completion, interpolation, demosaicing and Poisson denoising. Note that for Poisson denoising, we present further derivations

accounting for the negative log-likelihood of the Poisson distribution, which replaces the conventional least square data fidelity term (only suitable for Gaussian noise).

2. Background: Plug-and-Play ADMM.

2.1. Plug-and-Play Prior. Let us consider the linear inverse problem of recovering a clean image $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ from its degraded measurements $\boldsymbol{b} \in \mathbb{R}^m$ obtained with the degradation model $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\nu}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is the degradation matrix, $\boldsymbol{\nu} \in \mathbb{R}^m$ is additive white Gaussian noise with standard deviation σ, and $\boldsymbol{x}$ is the unknown ground truth image arranged as a vector. The problem is generally formulated as the Maximum a Posteriori (MAP) estimation of $\boldsymbol{x}$, given its prior probability density function p. The MAP is given by:

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{b}) = \arg\max_{\boldsymbol{x}} p(\boldsymbol{b}|\boldsymbol{x}) \cdot p(\boldsymbol{x}), \tag{2.1}$$

$$= \arg\min_{\boldsymbol{x}} -\ln(p(\boldsymbol{b}|\boldsymbol{x})) - \ln(p(\boldsymbol{x})), \tag{2.2}$$

$$= \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \sigma^2 \mathcal{R}(\boldsymbol{x}). \tag{2.3}$$

In practice, the prior distribution p is not used directly, but is represented by the regularizer $\mathcal{R}(\boldsymbol{x}) = -\ln(p(\boldsymbol{x}))$, which penalizes unlikely solutions.

The PnP approach introduced in [11] goes one step further by removing the need for an explicit definition of the regularizer. This is made possible by the use of proximal algorithms such as ADMM in which the regularizer only appears in the evaluation of the proximal operator of $\gamma\mathcal{R}$ for some scalar factor γ. This operator is defined as:

$$\text{prox}_{\gamma\mathcal{R}}(\boldsymbol{u}) = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{u}\|_2^2 + \gamma \cdot \mathcal{R}(\boldsymbol{x}). \tag{2.4}$$

It can be noted that the expression of $\text{prox}_{\gamma \cdot \mathcal{R}}$ is a particular case of Eq. (2.3), where the degradation matrix $\boldsymbol{A}$ is the identity matrix, i.e. the degradation model only consists in the addition of white Gaussian noise. In other words $\text{prox}_{\gamma \cdot \mathcal{R}}$ is the MAP denoiser assuming additive white Gaussian noise of variance γ and given the underlying prior $p(\boldsymbol{x}) = e^{-\mathcal{R}(\boldsymbol{x})}$. Based on this observation, the authors of [11] replaced $\text{prox}_{\gamma \cdot \mathcal{R}}$ by a standard image denoiser which implicitly determines the prior distribution. In the following, we will note the denoiser as a function of the image and the noise standard deviation:

$$F_{\mathcal{R}}(\boldsymbol{u}, \sqrt{\gamma}) = \text{prox}_{\gamma\mathcal{R}}(\boldsymbol{u}) \tag{2.5}$$

More recent works use a trained CNN to represent the proximal operator. For instance, in [14], adversarial training is used to learn a projection operator. Here, the assumption is that the regularizer is the indicator function of the set of "natural images". Hence, the corresponding proximal operator projects the input image to the closest "natural image", which can be seen as a form of blind denoising. While this avoids the need for a parameter γ, it does not allow for controlling the denoising "strength" within the algorithm. In [12], a neural network called DRUNet, that combines Res-Net [20] and U-Net [21] architectures, is trained for the task of Gaussian denoising. The noise standard deviation is given as a constant

input map in addition to the noisy image. Hence, compared to approaches based on denoising CNNs trained for a single noise level [22], or for blind denoising [14,23,24], the method in [12] can control the algorithm's parameters more precisely at each iteration, while keeping the highest denoising performance. Since a single network is used, it also simplifies the method in [13], where 25 denoisers are trained for different standard deviations in order to cover a larger range of noise levels.

In this paper, we argue that a denoiser parameterized with an input map for standard deviation can even be trained to control the noise level at each pixel independently. Such a denoiser further improves the PnP scheme for several applications thanks to the proposed preconditioning.

2.2. ADMM Formulation. In this section, we present the classical ADMM formulation in the case of the least squares inverse problem in Eq. (2.3). The same notations will be used throughout the paper. In order to use the ADMM, the problem is first cast into a constrained optimization problem by splitting the two terms:

$$\begin{aligned} \hat{\boldsymbol{x}} = \underset{\boldsymbol{x},\boldsymbol{y}}{\arg\min} \quad & \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 + \sigma^2\mathcal{R}(\boldsymbol{y}) \\ \text{subject to} \quad & \boldsymbol{x} = \boldsymbol{y}, \end{aligned} \tag{2.6}$$

The constraint $\boldsymbol{x} = \boldsymbol{y}$ is included in the optimization by defining the augmented Lagrangian function $\mathcal{L}$, which introduces a dual variable $\boldsymbol{l} \in \mathbb{R}^n$ and a penalty parameter ρ:

$$\begin{aligned} \mathcal{L}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{l}) = & \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 + \sigma^2\mathcal{R}(\boldsymbol{y}) \\ & + \boldsymbol{l}^{\mathsf{T}}(\boldsymbol{x}-\boldsymbol{y}) + \frac{\rho}{2}\left\|\boldsymbol{x}-\boldsymbol{y}\right\|_2^2, \end{aligned} \tag{2.7}$$

$$= \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 + \sigma^2\mathcal{R}(\boldsymbol{y}) + \frac{\rho}{2}\left\|\boldsymbol{x}-\boldsymbol{y}+\frac{\boldsymbol{l}}{\rho}\right\|_2^2 - \frac{\left\|\boldsymbol{l}\right\|_2^2}{2\rho}. \tag{2.8}$$

The ADMM method consists in alternatively minimizing the augmented Lagrangian $\mathcal{L}$ for the variables $\boldsymbol{x}$ and $\boldsymbol{y}$ separately, and by updating the dual variable $\boldsymbol{l}$. In practice, the penalty parameter ρ may also be increased over the iterations. This strategy was shown in [25] to ensure convergence to a fixed-point in the PnP scenario with only mild assumptions on the denoiser. The ADMM iteration thus reads as:

$$\boldsymbol{x}^{\boldsymbol{k+1}} = \underset{\boldsymbol{x}}{\arg\min}\, \left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 + \rho^k \left\|\boldsymbol{x} - \left(\boldsymbol{y}^{\boldsymbol{k}} - \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k}\right)\right\|_2^2, \tag{2.9}$$

$$\boldsymbol{y}^{\boldsymbol{k+1}} = \underset{\boldsymbol{y}}{\arg\min}\, \frac{1}{2}\left\|\boldsymbol{y} - \left(\boldsymbol{x}^{\boldsymbol{k+1}} + \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k}\right)\right\|_2^2 + \frac{\sigma^2}{\rho^k}\mathcal{R}(\boldsymbol{y}), \tag{2.10}$$

$$\boldsymbol{l}^{\boldsymbol{k+1}} = \boldsymbol{l}^{\boldsymbol{k}} + \rho^k(\boldsymbol{x}^{\boldsymbol{k+1}} - \boldsymbol{y}^{\boldsymbol{k+1}}), \tag{2.11}$$

$$\rho^{k+1} = \rho^k \cdot \alpha, \quad \text{with} \quad \alpha > 1, \tag{2.12}$$

where the dual variable $\boldsymbol{l}$ is typically zero-initialized. An initialization of $\boldsymbol{x}$ is also required and is often performed as $\boldsymbol{x}^0 = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{b}$, but a finer initialization strategy may be used depending on the problem.

The y-update in Eq. (2.10) can be equivalently written using a proximal operator:

$$\boldsymbol{y^{k+1}} = \text{prox}_{\gamma\mathcal{R}}(\boldsymbol{x^{k+1}} + \boldsymbol{l^k}/\rho^k), \tag{2.13}$$

where $\gamma = \sigma^2/\rho^k$. As seen in Section 2.1, this step can be performed in the PnP approach by denoising the image $\boldsymbol{x^{k+1}} + \boldsymbol{l^k}/\rho^k$ using a Gaussian denoiser for a noise standard deviation $\sqrt{\gamma} = \sigma/\sqrt{\rho^k}$.

Note that other PnP methods such as [12] use the HQS algorithm which is similar to ADMM, but without the dual variable $\boldsymbol{l}$ in the two sub-problems (Eqs. (2.9) and (2.10)). Hence, the dual-update step in Eq. (2.11) is also removed. While we present our preconditioning method in the next section for the ADMM, it would apply similarly to the HQS algorithm.

3. Preconditioned Plug-and-Play ADMM.

In several problems, we have additional knowledge about locally varying noise level. For instance, in image completion or interpolation problems, the pixels already sampled in $\boldsymbol{b}$ are known without error. Thus, at a given iteration, only the pixels at unknown positions need to be denoised. However, in the original ADMM formulation the denoising step considers the same noise level $\sigma/\sqrt{\rho^k}$ at each pixel. Hence, in order to finely tune the denoising effect and improve the performance of the PnP-ADMM, we reformulate the problem with a preconditioner that takes into account the knowledge of a variable noise level.

3.1. ADMM Reformulation.

Let us consider the following problem, mathematically equivalent to Eq. (2.3):

$$\widehat{\boldsymbol{x_p}} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{APx} - \boldsymbol{b}\|_2^2 + \sigma^2\mathcal{R}(\boldsymbol{Px}), \tag{3.1}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{P}\widehat{\boldsymbol{x_p}}, \tag{3.2}$$

where $\boldsymbol{P}$ is a diagonal preconditioning matrix.

Similarly to the Section. 2.2, we can express the ADMM algorithm for solving Eq. (3.1) by replacing the updates of $\boldsymbol{x}$ and $\boldsymbol{y}$ in Eqs. (2.9) and (2.10) with respectively:

$$\boldsymbol{x^{k+1}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{APx} - \boldsymbol{b}\|_2^2 + \rho^k \left\|\boldsymbol{x} - \left(\boldsymbol{y^k} - \frac{\boldsymbol{l^k}}{\rho^k}\right)\right\|_2^2, \tag{3.3}$$

$$\boldsymbol{y^{k+1}} = \arg\min_{\boldsymbol{y}} \frac{1}{2}\left\|\boldsymbol{y} - \left(\boldsymbol{x^{k+1}} + \frac{\boldsymbol{l^k}}{\rho^k}\right)\right\|_2^2 + \frac{\sigma^2}{\rho^k}\mathcal{R}(\boldsymbol{Py}). \tag{3.4}$$

The variable $\boldsymbol{x}$ can be updated directly using the well-known closed form solution of Eq. (3.3):

$$\boldsymbol{x^{k+1}} = (\boldsymbol{P}^\top\boldsymbol{A}^\top\boldsymbol{AP} + \rho^k\boldsymbol{I})^{-1}(\boldsymbol{P}^\top\boldsymbol{A}^\top\boldsymbol{b} + \rho^k\boldsymbol{y^k} - \boldsymbol{l^k}). \tag{3.5}$$

In this paper we will focus on the case of diagonal matrices $\boldsymbol{A}$ and $\boldsymbol{P}$ where Eq. (3.5) can be implemented with simple pixel-wise operations. Note that for other problems such as deblurring and super-resolution where $\boldsymbol{A}$ includes a convolution, practical implementations

are given in [12] and remain applicable when $\boldsymbol{P}$ applies a deconvolution of the form $\boldsymbol{\Phi}^\top \boldsymbol{D}\boldsymbol{\Phi}$, where $\boldsymbol{\Phi}$ is the Fourier transform and $\boldsymbol{D}$ is diagonal.

However, for updating $\boldsymbol{y}$, the denoiser $F_{\mathcal{R}}$ from Eq. (2.5), is no longer suitable because of the matrix $\boldsymbol{P}$ within the regularization term. The next section presents how the y-update can still be performed using a more general denoiser that considers Gaussian noise with variable variance.

3.2. New Prior Term Sub-Problem.

3.2.1. General Gaussian Denoiser. Let us first define the general expression of a denoiser for Gaussian noise with covariance matrix $\boldsymbol{\Sigma}$. The MAP estimate from a noisy image $\boldsymbol{u}$ can be derived similarly to Eqs. (2.1)-(2.3), where the likelihood $p(\boldsymbol{u}|\boldsymbol{x})$ is a multivariate Gaussian distribution, i.e. $p(\boldsymbol{u}|\boldsymbol{x}) \propto e^{-\frac{1}{2}\left\|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{u}-\boldsymbol{x})\right\|_2^2}$. Including the prior and taking the negative logarithm gives the expression of the denoiser:

$$G_{\mathcal{R}}(\boldsymbol{u}, \boldsymbol{\Sigma}^{1/2}) = \arg\min_{\boldsymbol{x}} \frac{1}{2} \left\|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{u}-\boldsymbol{x})\right\|_2^2 + \mathcal{R}(\boldsymbol{x}), \tag{3.6}$$

Here, we will only consider the case of a diagonal matrix $\boldsymbol{\Sigma}$. Hence, we still assume independent noise at each pixel (but not identically distributed), and the diagonal of $\boldsymbol{\Sigma}$ corresponds to pixel-wise variance. Note that the denoiser $F_{\mathcal{R}}$ is equivalently defined as $F_{\mathcal{R}}(\boldsymbol{x}, \sigma) = G_{\mathcal{R}}(\boldsymbol{x}, \sigma \boldsymbol{I})$ for a constant standard deviation σ. A practical image denoiser $G_{\mathcal{R}}$ can be obtained by training a CNN that takes as input a map of the pixel-wise noise levels in addition to the noisy image. A training procedure for such a denoiser is presented in Section 4.

3.2.2. Sub-Problem Solution. Now, let us rewrite Eq. (3.4) using the change of variable $\tilde{\boldsymbol{y}} = \boldsymbol{P}\boldsymbol{y}$, which gives $\boldsymbol{y} = \boldsymbol{P}^{-1}\tilde{\boldsymbol{y}}$. Thus, we have $\boldsymbol{y}^{\boldsymbol{k+1}} = \boldsymbol{P}^{-1}\tilde{\boldsymbol{y}}^{\boldsymbol{k+1}}$ and $\tilde{\boldsymbol{y}}^{\boldsymbol{k+1}} = \arg\min_{\tilde{\boldsymbol{y}}} f(\tilde{\boldsymbol{y}})$ with:

$$f(\tilde{\boldsymbol{y}}) = \frac{\rho^k}{2\sigma^2} \left\| \boldsymbol{P}^{-1}\tilde{\boldsymbol{y}} - \left(\boldsymbol{x}^{\boldsymbol{k+1}} + \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k}\right)\right\|_2^2 + \mathcal{R}(\tilde{\boldsymbol{y}}), \tag{3.7}$$

$$= \frac{1}{2} \left\| \frac{\sqrt{\rho^k}}{\sigma} \boldsymbol{P}^{-1} \left(\tilde{\boldsymbol{y}} - \boldsymbol{P}\left(\boldsymbol{x}^{\boldsymbol{k+1}} + \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k}\right)\right)\right\|_2^2 + \mathcal{R}(\tilde{\boldsymbol{y}}). \tag{3.8}$$

Given the denoiser $G_{\mathcal{R}}$ in Eq. (3.6) we can then rewrite the y-update step as:

$$\boldsymbol{y}^{\boldsymbol{k+1}} = \boldsymbol{P}^{-1} G_{\mathcal{R}} \left(\boldsymbol{P}\left(\boldsymbol{x}^{\boldsymbol{k+1}} + \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k}\right), \frac{\sigma}{\sqrt{\rho^k}} \boldsymbol{P} \right) \tag{3.9}$$

Therefore, the preconditioned ADMM still solves the original problem but introduces a denoising step that assumes noise with spatially varying standard deviation. The standard deviation can be adjusted per pixel via the diagonal matrix $\boldsymbol{P}$ and the global parameters σ and ρ^k. A suitable choice of the preconditioning matrix can be made depending on the problem in order to improve the performance of the PnP-ADMM.

3.3. Variables Interpretation and Initialization. It should be noted that in the preconditioned problem, an image $\boldsymbol{x^k}$ only estimates the intermediate variable $\widehat{\boldsymbol{x_p}}$, but the final result is $\hat{\boldsymbol{x}} = \boldsymbol{P}\widehat{\boldsymbol{x_p}}$ (see Eq. (3.2)). Hence, $\boldsymbol{x^k}$ (similarly $\boldsymbol{y^k}$) may not be a plausible image. However, the input of the denoiser is $\boldsymbol{P}(\boldsymbol{x^{k+1}} + \boldsymbol{l^k}/\rho^k)$, where the multiplication by $\boldsymbol{P}$ rescales the image pixels consistently with the "natural image" $\hat{\boldsymbol{x}}$, as expected by the denoiser.

Inversely, for the initialization, the inverse scaling must be performed. Given an initial estimate $\hat{\boldsymbol{x}}^{\boldsymbol{0}}$ of the true image $\hat{\boldsymbol{x}}$, a good initialization for our problem is thus $\boldsymbol{x^0} = \boldsymbol{P}^{-1}\hat{\boldsymbol{x}}^{\boldsymbol{0}}$.

3.4. Computational complexity. The complexity of the x-update and y-update steps in Eqs. (3.5) and (3.9), is not significantly increased by the preconditioning in the case of diagonal matrices $\boldsymbol{A}$ and $\boldsymbol{P}$ since the multiplication by $\boldsymbol{P}$ (or equivalently $\boldsymbol{P}^\top$) and by $\boldsymbol{P}^{-1}$ only consists of a pixel-wise multiplication and division respectively. Hence, the overall per-iteration complexity of the algorithm is essentially that of the deep denoiser network $G_\mathcal{R}$. As detailed in the next Section, we use the same network architecture for $G_\mathcal{R}$ as in the non-preconditioned PnP method in [12]. Hence, the complexity of one iteration is similar in the preconditioned and non-preconditioned cases. Furthermore, our experiments in Section 6 show that the preconditioning accelerates the convergence, as expected by our analysis in Section 5.4. Since the overall complexity scales linearly with the number of iterations, significant computing time can be saved with the proposed approach.

4. Deep Locally Adjustable Denoiser. Our algorithm relies on a deep neural network for solving the prior term sub-problem which corresponds to a Gaussian denoising problem where the noise level (i.e. standard deviation) is known and can be adjusted for each pixel. Although recently proposed denoisers (e.g. FFDNet [26], DRUNet [12]) can take into account a spatially varying noise level to some extent, these methods are only suitable for smooth noise level patterns. It is also worth noting that very recent improvements have been achieved in image denoising by the Restormer network [27] that uses a transformer-based architecture to overcome the issue of limited receptive field of traditional convolution layers. However, although the Restormer network has demonstrated higher quality Gaussian denoising than DRUNet for a similar complexity and over a large range of noise standard deviations, it was trained in [27] as a blind denoiser. As shown in [12], blind denoisers are not suitable for the PnP framework which requires adjusting the noise level parameter within the algorithm. Furthermore, the images given as input to the denoiser within the PnP framework typically have different types of degradation than i.i.d. Gaussian noise.

In this section, we propose a method to train a locally adjustable denoiser for arbitrary noise level patterns.

4.1. Denoising Network for Known Noise Level. Here, we use the DRUNet architecture proposed in [12]. The overall structure consists of a U-Net where each level contains sequences of 4 residual-blocks and either a down-sampling layer (first branch of the U) or an upsampling layer (second branch).

The DRUNet network structure in [12] conforms to the definition of the denoisers $F_\mathcal{R}$ (Eq. (2.5)) or $G_\mathcal{R}$ (Eq. (3.6)) by taking as input the concatenation (in the channel dimension) of the noisy image and a noise level map whose pixels' values are equal to the noise standard deviation. The advantage of using an input noise level map is that a single generic model

can be trained to perform optimally for a large range of noise levels. However, in [12] the model is only trained considering constant noise level maps since their algorithm only uses the constant denoiser $F_{\mathcal{R}}$. In the rest of the paper, we refer to this network as *DRUNet-cst* and we re-trained it for fair comparisons with our locally adjustable versions.

In order to use our preconditioning approach, a locally adjustable denoiser $G_{\mathcal{R}}$ is required. Therefore, the training process must be adapted so that each input sample consists of an arbitrary noise level map along with an image corrupted with the corresponding Gaussian noise level for each pixel. We describe in the next section how to randomly generate suitable patterns so that the trained model generalizes well for any arbitrary noise level map.

4.2. Noise Level Map Generation. Let us first consider the case of a constant noise level map as in [12]. Here, all the pixels are equal to the same random variable S that can be simply defined as:

$$S = 2\mu \cdot X, \tag{4.1}$$

where X is a uniformly distributed random variable in the range $[0, 1]$. The parameter μ is equal to the expectation of the random variable S. It can be selected to train a denoiser that is sufficiently generic for all the noise levels σ in the range $[0,2\mu]$.

Now, in order to train a locally adjustable denoiser $G_{\mathcal{R}}$, we generate a random map for each training image using the following random process:

$$S_i = 2\mu \cdot (X_i \cdot (1 - W) + O \cdot W), \tag{4.2}$$

where i is the pixel index, O, W and X_i (for each pixel i) are independent random variables with uniform distributions in the range $[0, 1]$. One can verify that μ is the expectation of S_i, and the range of possible values of S_i is $[0,2\mu]$ similarly to the previous case with constant noise level.

The random weight W allows us to adjust the variance of the noise level map (i.e. lower weight corresponding to higher variance). The random offset O is also necessary to reduce the correlation between the variance and the mean of the generated maps. In particular, the offset enables the generation of maps with a low variance but a high average level, so that the trained denoiser generalizes well even in the constant noise level case. Hence, the method makes it possible to train the network with noise level maps covering a wide range of first and second order statistics, which prevents overfitting for a specific type of pattern. In the paper, we refer to the network trained with this method as *DRUNet-var*.

For some applications, the noise standard deviation may also need to be adjusted depending on the color component. For these applications, we train another network called *DRUNet-var-RBG* that takes an input noise level map for each color component. For the training, the noise level maps are generated for each component separately using the random process in Eq. (4.2).

4.3. Training Details. The training is performed by generating a noise level map randomly for each input image, as described in Section 4.2. Considering pixel data in the range $[0, 1]$, we use the parameter $\mu = 25/255$ in Eq. (4.1) (for the *DRUNet-cst* denoiser) or Eq. (4.2) (for the *DRUNet-var* and *DRUNet-var-RBG* denoisers). The maximum noise level is thus

Table 1

Denoising performance (average PSNR over each dataset) for Gaussian noise with constant noise level (standard deviation $\sigma = 10$, 30 or 50).

Dataset	CBSD68			Set5		
noise level σ	10	30	50	10	30	50
CBM3D [16]	35.90	29.91	27.51	36.02	30.97	28.75
FFDNet [26]	36.14	30.32	27.97	36.16	31.35	29.24
RNAN [33]	36.43	30.65	28.30	36.62	31.77	29.61
DRUNet-cst [12]	36.51	30.79	28.48	36.67	31.90	29.80
DRUNet-var	36.51	30.80	28.48	36.67	31.91	29.79
DRUNet-var-RBG	36.47	30.78	28.46	36.61	31.90	29.77

$2\mu = 50/255$. Gaussian noise with standard deviation defined by the noise level map is then added to the input image.

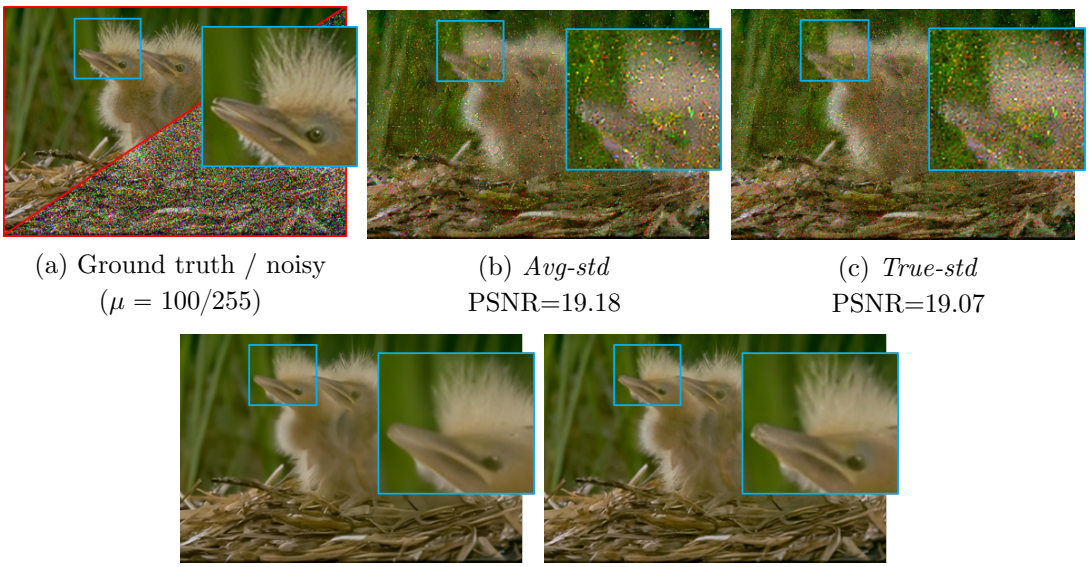
The remaining details of the training procedure are the same as described in [12]. We use the same large dataset of 8694 images, including the Berkeley Segmentation Dataset BSDS500 (without the validation images) [28], the Waterloo Exploration Database [29], the DIV2K dataset [30], and the Flickr2K dataset [31]. We minimize the l_1 loss between the denoiser's output and the ground truth image using the ADAM optimizer [32] with a learning rate initialized at $1e-4$ and decreased by half every 100000 iterations. The training is stopped when the learning rate is smaller than $5e-7$. Finally, each iteration of the training uses a batch of 16 patches of size 128x128 randomly selected from the images of the training dataset.

4.4. Denoising Performance. Here, we evaluate the performance of our denoisers *DRUNet-var* and *DRUNet-var-RBG* generalized for variable standard deviation, in comparison with *DRUNet-cst* [12] as well as other state-of-the-art denoisers that assume a constant noise level. These include the BM3D denoiser (noted CBM3D for color images) [16] and the two recent CNN-based methods FFDNet [26] and RNAN [33]. Similarly to *DRUNet-cst*, the FFDNet denoiser can be parameterized at inference time knowing the noise standard deviation. On the other hand, RNAN requires a separate training for each noise level, but has shown higher denoising performance thanks to a non-local attention module.

We perform our evaluations with the widely used Set5 [34] and CBSD68 [35] test datasets.

4.4.1. Constant Noise Level. First, let us consider input images corrupted with constant noise level. The results in Table 1 show that our locally adjustable denoisers *DRUNet-var* and *DRUNet-var-RBG* do not have significantly degraded performance compared to the reference network *DRUNet-cst*, despite being trained for more generic noise level maps. Only a very moderate loss is observed for the *DRUNet-var-RBG* denoiser compared to *DRUNet-cst*, mostly for the lowest and highest noise levels ($\sigma = 10$ or 50). Nevertheless, *DRUNet-var-RBG* still outperforms the other state-of-the-art methods, even the RNAN denoiser trained specifically for each noise level.

4.4.2. Spatially Varying Noise Level. Now, let us show the advantage of the proposed training procedure by comparing the performances of our *DRUNet-var* denoiser with the *DRUNet-cst* version when the input noise standard deviation varies spatially. For these tests,



(a) Ground truth / noisy ($\mu = 100/255$)

(b) *Avg-std* PSNR=19.18

(c) *True-std* PSNR=19.07

(d) *PnP-ADMM* PSNR=28.94

(e) *DRUNet-var* PSNR=29.28

Figure 1. *Denoising results with pixel-wise standard deviation randomly selected in* $[0, 2\mu]$. *The average noise standard deviation is* $\mu = 100/255$. *For the Avg-std, True-std and PnP-ADMM schemes in (b-d), the network DRUNet-cst is used. In (e), the network DRUNet-var is used with the True-std scheme.*

we randomly select a standard deviation in the range $[0, 2\mu]$ for each pixel. For a complete comparison with *DRUNet-cst*, we have tested this network in three different ways:

- *Avg-std*: using the average noise level μ as input.
- *True-std*: using the true noise level map as input.
- *PnP-ADMM*: solving the problem of denoising with variable noise level (Eq.(3.6)) using PnP-ADMM without preconditioning based on the *DRUNet-cst* network.

For the *PnP-ADMM*, we set the parameters using Eq. (6.1) so that the noise of highest standard deviation $\sigma^0_{den} = 2\mu$ is removed at the first iteration, while successive iterations refine the result by decreasing the noise level of the denoiser down to σ^N_{den}. Here, we use $\sigma^N_{den} = \frac{2\mu}{3}$ for $N = 25$ iterations, which we found to give the best results for this application.

Table 2 and Fig. 1 present the results of *DRUNet-cst* in each of the three schemes, and *DRUNet-var* in the *True-std* scheme. As expected, the results using only the average standard deviation are unsatisfying, since the pixels with above average noise level are not sufficiently denoised, while details are not optimally preserved in less noisy regions. The same behavior is observed when using the true noise level map as input to *DRUNet-cst*. This confirms that a denoiser trained for constant noise level does not generalize when the noise standard deviation strongly varies between pixels. More satisfying results are obtained when using *DRUNet-cst* in the PnP-ADMM scheme. However, our *DRUNet-var* denoiser trained directly for this task can still retrieve more details, which indicates that the PnP-ADMM without preconditioning is sub-optimal. More realistic applications are presented in the rest of the paper to demonstrate that our locally adjustable denoisers also allow for improved performances in practical

Table 2

Denoising performance (average PSNR over each dataset) for Gaussian noise with pixel-wise variable standard deviation in the range $[0, 2\mu]$.

		DRUNet-cst [12]			*DRUNet-var*
	Dataset	*Avg-std*	*True-std*	*PnP-ADMM*	*True-std*
$\mu = \frac{25}{255}$	Set5	29.01	29.03	34.35	34.48
	BSD68	28.14	28.22	32.75	32.90
$\mu = \frac{50}{255}$	Set5	24.45	24.37	32.03	32.24
	BSD68	23.59	23.60	30.09	30.27
$\mu = \frac{100}{255}$	Set5	19.82	19.58	29.69	30.06
	BSD68	19.03	18.88	27.73	28.01

scenarios thanks to the preconditioned PnP-ADMM.

5. Applications. In this section we present several examples of applications in image restoration. Although the proposed denoiser allows us to adjust of the variance locally, it still assumes an independent noise distribution for each pixel. Hence, in order to demonstrate the advantage of our preconditioning for the PnP-ADMM scheme, we restrict our study to applications where the error variance is expected to vary only in the pixel domain. While this excludes deblurring, or super-resolution from an anti-aliased downsampled image, many applications of high practical interest are still concerned. These include image completion, interpolation, demosaicing and Poisson denoising.

5.1. Image Completion and Interpolation. In the problems of image completion and interpolation, a subset Ω of the original image pixels are sampled in the input vector $\boldsymbol{b}$, while the remaining ones are unknown and must be determined. In both problems, the sampling can be expressed in the matrix form $\boldsymbol{Ax} = \boldsymbol{b}$ with a sampling matrix $\boldsymbol{A}$ such that each row contains only zeros, except one element equal to 1 at the index of a sampled pixel.

When solving the problem with ADMM, all the unknown pixels are expected to have the highest estimation error at the initialization stage. However, over iterations, more reliable estimates should be obtained especially for pixels close to a sampled pixel. Hence, the denoiser within our preconditioned PnP-ADMM should be adjusted locally depending on the proximity to a sampled pixel. Following this intuition, we define a preconditioning matrix $\boldsymbol{P}$ that varies over iterations according to the following formula for each iteration k:

$$\left[\boldsymbol{P}^{\boldsymbol{k}}\right]_{i,i} = \frac{\max(\boldsymbol{m}^{\boldsymbol{k}}) + \epsilon}{\left[\boldsymbol{m}^{\boldsymbol{k}}\right]_i + \epsilon} \tag{5.1}$$

where ϵ is a scalar parameter and $\boldsymbol{m}^{\boldsymbol{k}}$ is defined recursively by blurring $\boldsymbol{m}^{\boldsymbol{k-1}}$, and where $\boldsymbol{m}^{\boldsymbol{0}}$ is the mask indicating the known pixels:

$$\boldsymbol{m}^{\boldsymbol{k}} = \boldsymbol{m}^{\boldsymbol{k-1}} * g(\sigma_f), \quad \text{with} \quad \left[\boldsymbol{m}^{\boldsymbol{0}}\right]_i = \begin{cases} 1 & \text{if} \quad i \in \Omega \\ 0 & \text{otherwise} \end{cases} \quad \forall i \tag{5.2}$$

Here, * is the 2D convolution operation and g is a 2D Gaussian filter of parameter σ_f. Note that it is equivalent to have $\boldsymbol{m}^{\boldsymbol{k}} = \boldsymbol{m}^{\boldsymbol{0}} * g(\sigma_f\sqrt{k})$. We can thus determine σ_f so that the

preconditioning at the last iteration N does not depend on N, but only on a fixed parameter σ_f^{last} at the last iteration by taking $\sigma_f = \sigma_f^{last}/\sqrt{N}$.

Using this definition, the preconditioning values are always equal to 1 for the known pixels (i.e. $\left[\boldsymbol{P}^{\boldsymbol{k}}\right]_{i,i} = 1 \ \forall i \in \Omega$), and higher than 1 for the other pixels, which results in a stronger denoising for the unknown pixels in Eq. (3.9). The parameter ϵ prevents too high preconditioning values that would make the denoising impractical. The highest preconditioning value is $p_{max} = (1+\epsilon)/\epsilon$. Additionally, thanks to the Gaussian blur, the preconditioning values of unknown pixels will depend on their distance with sampled pixels. Therefore, the least reliable pixels (i.e. that are far from sampled pixels) will be denoised more.

Note that we only use non-zero preconditioning values (and thus non-zero standard deviation in the denoiser) so that the matrix $\boldsymbol{P}$ remains invertible. However, ideally, no denoising should be performed at the sampled pixel positions to prevent unnecessary loss of information. Therefore, we force the denoiser's output to be equal to the input for these pixels.

5.2. Demosaicing. Digital cameras typically capture colors using a sensor with a color filter array (CFA). Thanks to the mosaic formed by the CFA, neighbor pixels on the sensor record a different color information. Knowing the CFA pattern, the full color information can then be retrieved by an inverse problem called demosaicing.

Red, green and blue filters are generally used so that each pixel directly records one of the RGB components. Demosaicing can then be seen as an interpolation problem where a pixel R, G or B value is either known or unknown. Therefore, we use the same preconditioning strategy described in Section 5.1. However, in a realistic scenario, sensor data may also contain noise which is preferably removed jointly with the demosaicing step [36]. In the case of noisy data, our method applies similarly, but we do not force the denoiser to keep the sampled pixels unchanged as explained in Section 5.1.

Furthermore, unlike the problems of completion and interpolation, the demosaicing uses different masks indicating the sampled pixels for the R, G and B components. The preconditioning matrix thus takes different values for each component of the same pixel, which requires using a denoiser parameterized with a RGB noise level map. Our *DRUNet-var-RBG* network trained for arbitrary noise level patterns is thus suitable for this application.

However, in order to illustrate the tradeoff between the genericity of the denoiser and the optimal performance, we also trained a more specialized denoiser for demosaicing that we call *DRUNet-dem*. For training this network, instead of generating noise level maps with the generic random process in Eq. (4.2), we generate the patterns from the CFA mask, that is also used in the preconditioned ADMM. These patterns are defined by Eqs. (5.1) and (5.2), where a mask $\boldsymbol{m^0}$ is defined for each color component by the CFA pattern. Experimental results using either the generic network *DRUNet-var-RBG* or the specialized one *DRUNet-dem*, are given in Section 6.4.

5.3. Poisson Denoising. In many practical scenarios, the assumption that images are corrupted by additive white Gaussian noise is inaccurate. Camera noise is typically better modelled by a Poisson process. In order to formulate a MAP optimization for the problem of denoising an image corrupted with Poisson noise, the least squares data term in Eq. (2.3) must be modified. Here, we re-derive our preconditioned ADMM algorithm for Poisson denoising.

5.3.1. Preconditioned ADMM for Poisson denoising. Applying Poisson noise to an original image $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$ gives a noisy image $\boldsymbol{b} \in \mathbb{N}^n$ such that the likelihood distribution is:

$$p(\boldsymbol{b}|\boldsymbol{x}) = \prod_{i=1}^{n} \frac{\boldsymbol{x}_i^{\boldsymbol{b}_i} e^{-\boldsymbol{x}_i}}{\boldsymbol{b}_i!}. \tag{5.3}$$

As seen in Eqs. (2.1)-(2.3), the data term is the negative log-likelihood which can be derived as

$$-\ln(p(\boldsymbol{b}|\boldsymbol{x})) = -\boldsymbol{b}^{\mathsf{T}} \ln(\boldsymbol{x}) + \mathbb{1}^{\mathsf{T}}\boldsymbol{x} + c, \tag{5.4}$$

where c is a constant term that can be ignored for the minimization. The preconditioned Poisson denoising problem is then formulated by substituting $\boldsymbol{x}$ with $\boldsymbol{P}\boldsymbol{x}$ and by adding the regularization term as in Eq. (3.1):

$$\widehat{\boldsymbol{x}_{\boldsymbol{p}}} = \arg\min_{\boldsymbol{x}} -\boldsymbol{b}^{\mathsf{T}} \ln(\boldsymbol{P}\boldsymbol{x}) + \mathbb{1}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{x} + \mathcal{R}(\boldsymbol{P}\boldsymbol{x}), \tag{5.5}$$

The problem is solved using the ADMM, by splitting the data and regularization terms following the methodology in Section 2.2. Since the regularization term remains the same as in Section 3, the y-update in Eq. (3.9) still applies, and is performed using the same Gaussian denoiser $G_{\mathcal{R}}$. However, given the new data term, the x-update becomes:

$$\boldsymbol{x}^{\boldsymbol{k+1}} = \arg\min_{\boldsymbol{x}} -\boldsymbol{b}^{\mathsf{T}} \ln(\boldsymbol{P}\boldsymbol{x}) + \mathbb{1}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{x} + \frac{\rho^k}{2} \left\| \boldsymbol{x} - \boldsymbol{u}^{\boldsymbol{k}} \right\|_2^2, \tag{5.6}$$

where $\boldsymbol{u}^{\boldsymbol{k}} = \boldsymbol{y}^{\boldsymbol{k}} - \boldsymbol{l}^{\boldsymbol{k}}/\rho^k$.

Knowing that $\boldsymbol{P}$ is a diagonal matrix, this problem can be solved independently for each pixel. The following closed form solution is derived in Appendix A (similar derivations are also found in [15] for the case $\boldsymbol{P} = \boldsymbol{I}$):

$$\left[\boldsymbol{x}^{\boldsymbol{k+1}}\right]_i = \frac{\rho^k \boldsymbol{u}_i - \boldsymbol{P}_{i,i} + \sqrt{(\rho^k \boldsymbol{u}_i - \boldsymbol{P}_{i,i})^2 + 4\rho^k \boldsymbol{b}_i}}{2\rho^k}. \tag{5.7}$$

It is worth noting that unlike the Gaussian case, the noise level only depends on the data and is not controlled by a parameter σ. However, in photography the amount of Poisson noise can be reduced by increasing the exposure. This effect can be simulated by applying Poisson noise to a ground truth image rescaled in the range $[0, \lambda]$. A high peak value λ thus simulates a high exposure and less noise. However, for an input image $\boldsymbol{b}$ with values in the range $[0, \lambda]$, the algorithm must be modified since the denoiser $G_{\mathcal{R}}$ in Eq. (3.9) assumes data in the range $[0, 1]$. The y-update should then be performed by replacing $G_{\mathcal{R}}$ with the denoiser $G_{\mathcal{R}}^{\lambda}$ adapted to the range $[0, \lambda]$ by rescaling the input and output as:

$$G_{\mathcal{R}}^{\lambda}(\boldsymbol{u}, \boldsymbol{\Sigma}^{1/2}) = \lambda G_{\mathcal{R}}(\boldsymbol{u}/\lambda, \boldsymbol{\Sigma}^{1/2}/\lambda). \tag{5.8}$$

Similarly, the final ADMM result should be divided by λ to recover an image in the range $[0, 1]$.

5.3.2. Choice of Preconditioning Matrix. It is well known that the variance and the expected value of a random variable with the Poisson distribution are both equal to the distribution's parameter, i.e. the noiseless image pixel. This means that for each pixel of the image, the noise standard deviation is equal to the square root of the pixel value in the ground truth image. Although this image is unknown, the noisy image provides a first estimate of the noise variance, which can be refined at the next iterations, as we obtain a better estimate of the noiseless image.

Therefore, in this problem, we use $\rho_k = 1$ for each iteration, so that the denoiser is only controlled using the preconditioning matrix which is initialized as $\boldsymbol{P^0} = \text{diag}(\sqrt{\boldsymbol{b}})$. Here, the square root is applied element-wise and the notation diag forms a diagonal matrix from the vector. Following the initialization strategy in Section 3.3, the variable $\boldsymbol{x}$ is then initialized such that $\hat{\boldsymbol{x}}^{\boldsymbol{0}} = \boldsymbol{P^0 x^0} = \boldsymbol{b}$, and thus $\boldsymbol{x^0} = \left[\boldsymbol{P^0}\right]^{-1} \boldsymbol{b} = \sqrt{\boldsymbol{b}}$.

Then, at each iteration, a more accurate estimate of the noiseless image is given by $\boldsymbol{P^k y^k}$ obtained after the denoising step in Eq. (3.9). Note that $\boldsymbol{P^k y^k}$ is directly the output of the denoiser (before applying $\left[\boldsymbol{P^k}\right]^{-1}$) and is thus obtained without further matrix multiplication. Hence, the preconditioning matrix can be updated to better estimate the noise standard deviation using $\boldsymbol{P^{k+1}} = \text{diag}(\sqrt{\boldsymbol{P^k y^k}})$.

5.4. Analysis for noise-free sampling applications. In this section, we provide further analysis to illustrate the benefit of the proposed preconditioning approach for the sampling problems such as image completion, interpolation or demosaicing, in the noise-free scenario.

For these problems, we can define the sets $\mathcal{S}$ and $\bar{\mathcal{S}}$ of indices of respectively the known and unknown pixel's. The sampling matrix $\boldsymbol{A}$ is diagonal and such that $\boldsymbol{A}_{i,i} = 1$ if $i \in \mathcal{S}$ and $\boldsymbol{A}_{i,i} = 0$ if $i \in \bar{\mathcal{S}}$. In Sections 5.1 and 5.2, we have also defined a diagonal preconditioning matrix $\boldsymbol{P}$ such that $\boldsymbol{P}_{i,i} = 1$ if $i \in \mathcal{S}$ (and for $i \in \bar{\mathcal{S}}$, we have $\boldsymbol{P}_{i,i} > 1$ in the preconditioned case and $\boldsymbol{P}_{i,i} = 1$ if no preconditioning is used). The closed form solution to the data-fidelity sub-problem in Eq. (3.5) can thus be expressed per pixel as:

$$\left[\boldsymbol{x^{k+1}}\right]_i = \begin{cases} \frac{\boldsymbol{b}_i + \rho^k \cdot \left(\left[\boldsymbol{y^k}\right]_i - \left[\boldsymbol{l^k}\right]_i\right)}{1+\rho^k}, & \text{if } i \in \mathcal{S}, \\ \left[\boldsymbol{y^k}\right]_i - \left[\boldsymbol{l^k}\right]_i / \rho^k, & \text{if } i \in \bar{\mathcal{S}}. \end{cases} \tag{5.9}$$

First, let us show that for the noise-free sampling applications, the dual variable in the ADMM (either with or without preconditioning) has no effect, making the algorithm equivalent to the HQS, which simplifies the analysis.

For all the known pixels of indices $i \in \mathcal{S}$, the initialization is such that $\left[\boldsymbol{y^0}\right]_i = \boldsymbol{b}_i$ and $\left[\boldsymbol{l^0}\right]_i = 0$. Hence, at the first iteration, Eq. (5.9) gives $\left[\boldsymbol{x^1}\right]_i = \boldsymbol{b}_i$. Furthermore, since we prevent the denoising of the known pixels for noise-free sampling problems by forcing the denoiser's output to be equal to its inputs for these pixels, the prior term sub-problem in Eq. (3.9) becomes $\left[\boldsymbol{y^{k+1}}\right]_i = \left[\boldsymbol{x^{k+1}}\right]_i + \left[\boldsymbol{l^k}\right]_i / \rho^k$, which gives at the first iteration $\left[\boldsymbol{y^1}\right]_i = \left[\boldsymbol{x^1}\right]_i = \boldsymbol{b}_i, \forall i \in \mathcal{S}$. The dual variable update in Eq. (2.11) then gives $\left[\boldsymbol{l^1}\right]_i = \left[\boldsymbol{l^0}\right]_i + \rho^k \left(\left[\boldsymbol{x^0}\right]_i - \left[\boldsymbol{y^0}\right]_i\right) = 0$. Thus, both $\boldsymbol{y}_i$ and $\boldsymbol{l}_i$ are unchanged after 1 iteration, and by recurrence we obtain $\left[\boldsymbol{x^k}\right]_i = \left[\boldsymbol{y^k}\right]_i = \boldsymbol{b}_i$ and $\left[\boldsymbol{l^k}\right]_i = 0, \forall k \in \mathbb{Z}, \forall i \in \mathcal{S}$. The

data-fidelity sub-problem is then:

$$\left[\boldsymbol{x}^{\boldsymbol{k+1}}\right]_i = \begin{cases} \boldsymbol{b}_i, & \text{if } i \in \mathcal{S}, \\ \left[\boldsymbol{y}^{\boldsymbol{k}}\right]_i - \left[\boldsymbol{l}^{\boldsymbol{k}}\right]_i / \rho^k, & \text{if } i \in \bar{\mathcal{S}}. \end{cases} \tag{5.10}$$

For the unknown pixels of indices $i \in \bar{\mathcal{S}}$, on the other hand, the dual variable $\boldsymbol{l}$ may be non-zero. However, the y-update step of the ADMM in Eq. (3.4) only uses this variable to compute the input $\boldsymbol{x}^{\boldsymbol{k+1}} + \boldsymbol{l}_{\boldsymbol{k}}/\rho^k$ of the regularization term's proximal operator. Furthermore, from Eq. (5.10), for $i \in \bar{\mathcal{S}}$, we have $\left[\boldsymbol{x}^{\boldsymbol{k+1}}\right]_i + \left[\boldsymbol{l}^{\boldsymbol{k}}\right]_i / \rho^k = \left[\boldsymbol{y}^{\boldsymbol{k}}\right]_i$. Therefore, the y-update can be directly computed from the previous estimate $\boldsymbol{y}^k$ for the unknown pixels independently of the values $\left[\boldsymbol{l}^{\boldsymbol{k}}\right]_i$. Hence the ADMM and HQS are equivalent in this particular case. Note that for the noisy case, we provide comparisons between HQS and ADMM (either with or without preconditioning) in Appendix B.

In order to analyse the effect of the preconditioning, let us now rewrite the x and y-update steps as a single step by injecting the result of Eq. (5.10) into the y-update equation. Since the final result is obtained as $\tilde{\boldsymbol{y}} = \boldsymbol{P}\boldsymbol{y}$, let us directly consider the y-update with this change of variable as expressed in Eq. (3.7). We obtain:

$$\tilde{\boldsymbol{y}}^{\boldsymbol{k+1}} = \arg\min_{\tilde{\boldsymbol{y}}} \frac{1}{2} \left\| \boldsymbol{P}^{-1}\tilde{\boldsymbol{y}} - \left(\boldsymbol{x}^{\boldsymbol{k+1}} + \frac{\boldsymbol{l}^{\boldsymbol{k}}}{\rho^k} \right) \right\|_2^2 + \frac{\sigma^2}{\rho^k} \mathcal{R}(\tilde{\boldsymbol{y}}), \tag{5.11}$$

$$= \arg\min_{\tilde{\boldsymbol{y}}} \frac{1}{2} \sum_{i \in \mathcal{S}} (\tilde{\boldsymbol{y}}_i - \boldsymbol{b}_i)^2 + \frac{1}{2} \sum_{i \in \bar{\mathcal{S}}} \left(\frac{\tilde{\boldsymbol{y}}_i - \left[\tilde{\boldsymbol{y}}^k\right]_i}{\boldsymbol{P}_{i,i}} \right)^2 + \frac{\sigma^2}{\rho^k} \mathcal{R}(\tilde{\boldsymbol{y}}), \tag{5.12}$$

$$= \arg\min_{\tilde{\boldsymbol{y}}} \frac{1}{2} \left\| \boldsymbol{A}\tilde{\boldsymbol{y}} - \boldsymbol{b} \right\|_2^2 + \frac{\sigma^2}{\rho^k} \mathcal{R}(\tilde{\boldsymbol{y}}) + \frac{1}{2} \sum_{i \in \bar{\mathcal{S}}} \left(\frac{\tilde{\boldsymbol{y}}_i - \left[\tilde{\boldsymbol{y}}^k\right]_i}{\boldsymbol{P}_{i,i}} \right)^2. \tag{5.13}$$

One can note that by taking $\rho = 1$ and $\boldsymbol{P}_{i,i}$ arbitrarily large for all $i \in \bar{\mathcal{S}}$, Eq. (5.13) would directly solve the original problem of Eq. (2.3) in a single iteration. However, choosing arbitrarily large preconditioning values for the unknown pixels is impractical in the plug-and-play context. This would require parameterizing the denoiser using a standard deviation map with extreme variations between the known pixel positions and the unkown ones, while the denoiser can only be trained for a limited range of standard deviation values. In our implementation for sampling problems, we have constructed a preconditioning matrix with maximum values p_{max} as described in Section 5.1 in order to keep standard deviations in a reasonable range.

Nevertheless, we have $p_{max} \geqslant \boldsymbol{P}_{i,i} > 1$ for the unknown pixels, hence reducing the weight of the last term in Eq. (5.13) compared to the non-preconditioned algorithm (where $\boldsymbol{P}_{i,i} = 1 \, \forall i$). This term ensures proximity of the unknown pixels with their values at the previous iteration, which intuitively slows down the algorithm. Reducing the weight of this term with the proposed preconditioning is thus expected to accelerate the algorithm, as confirmed by the experimental results presented in the next section.

6. Experimental Results.

6.1. ADMM Parameters Setting.

In this section, we provide the main intuition behind the tuning of the ADMM parameters. Note that alternatively to a manual tuning, automatic

hyper-parameter tuning have also been developed recently in [37] for the PnP-ADMM, and would apply similarly in our preconditioned case.

The main hyper-parameters of the ADMM are the number of iterations N, the initial penalty parameter ρ^0, and it's increase factor α in Eq. (2.12). The additional parameter σ in Eq (3.1) depends on the problem and corresponds to the Gaussian noise standard deviation in the measurement vector $\boldsymbol{b}$. For noise-free applications (i.e. completion, interpolation, noise-free demosaicing), using the theoretical value $\sigma = 0$ would remove the regularization term. Therefore, in order to keep the benefit of the regularization in these applications, we use a small value $\sigma = 1/255$. Also note that this parameter does not appear in the case of Poisson denoising in Eq. (5.5) since the noise standard deviation only depends on the ground truth image.

6.1.1. Completion, Interpolation and Demosaicing. For the completion, interpolation and demosaicing (either with or without noise) the parameters setting is inspired from [12]: since the noise level of the denoiser is globally controlled at each iteration by $\sigma^k_{den} = \sigma/\sqrt{\rho^k}$, we choose ρ^0 such that σ^0_{den} is sufficiently large to remove initialization noise or artifacts at the first iteration (regardless of the loss in details). For example, in the completion problem, we use zero-initialisation and a large value of $\sigma^0_{den} = 1$. For the interpolation and demosaicing problems, more accurate initialisation can be performed, thus we use a smaller value of $\sigma^0_{den} = 50/255$. The parameter α is then determined so that σ^k_{den} decreases down to $\sigma^N_{den} = \sigma$, in order to best preserve the details in the final image. Therefore, we have $\sigma^k_{den} = \sigma^N_{den}/\sqrt{\rho^k}$. Using this equation for $k = 0$ and $k = N$, we can compute the parameters values:

$$\rho^0 = \left(\frac{\sigma^N_{den}}{\sigma^0_{den}}\right)^2 \quad \text{and} \quad \alpha = \left(\frac{1}{\rho^0}\right)^{1/N} \tag{6.1}$$

Note that the noise level assumed by the denoiser is also controlled locally by the preconditioning matrix $\boldsymbol{P}$ in Eq.(3.9). However, our preconditioning strategy for these problems is such that $\boldsymbol{P}_{i,i} = 1$ if i is the index of a sampled pixel (see Section 5.1). Hence at the last iteration, the image is denoised assuming the correct noise standard deviation $\sigma^N_{den} = \sigma$ for the sampled pixels. In our experiments, we can thus compare our preconditioned PnP-ADMM to the non-preconditioned version using the same parameterization of σ^0_{den} and σ^N_{den}.

The preconditioning matrix definition in Eqs. (5.1) and (5.2) introduces the additional parameters σ_f, controlling the blurring of the mask at each iteration, and ϵ, controlling the maximum preconditioning value $p_{max} = (1 + \epsilon)/\epsilon$. In all the experiments, we use $p_{max} = 10$ (i.e. $\epsilon = 1/9$). The parameter σ_f is set according to the number of iterations N using $\sigma_f = \sigma^{last}_f/\sqrt{N}$ as explained in Section 5.1, where σ^{last}_f is the blurring parameter at the last iteration that we set empirically for each application (see details in the respective sections and parameters summary in Table 3).

6.1.2. Poisson Denoising. For the problem of Poisson denoising, the noise level at each pixel and each iteration is fully controlled by the preconditioning matrix as explained in section 5.3, which removes the need for the penalty parameter. Therefore we simply use $\rho^0 = 1$ and $\alpha = 1$.

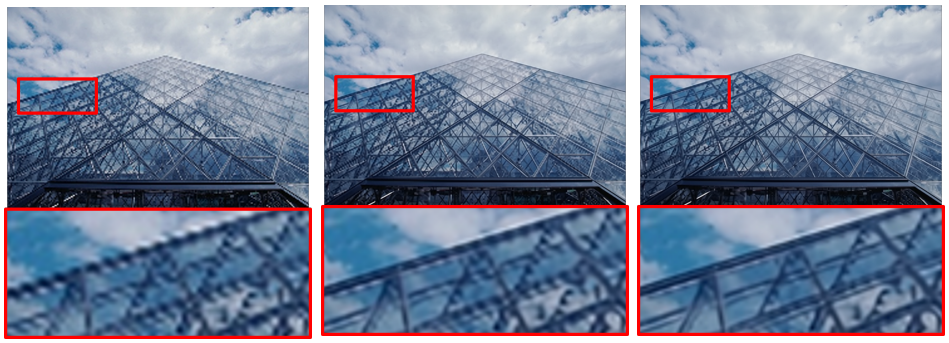
Nevertheless in order to evaluate the advantage of the preconditioning, we also implement the non-preconditioned version by taking $\boldsymbol{P} = \boldsymbol{I}$ and using the *DRUNet-cst* network. In this

Table 3

Summary of the parameters setting. For the demosaicing with preconditioning, we use either $\sigma_f^{last} = 0$ *with the DRUNet-dem network or* $\sigma_f^{last} = 0.3$ *with the DRUNet-var-RBG network.*

		without preco.		with preco.			
		σ_{den}^0	σ_{den}^N	σ_{den}^0	σ_{den}^N	σ_f^{last}	p_{max}
completion		1	$\frac{1}{255}$	1	$\frac{1}{255}$	0	10
interpolation		$\frac{50}{255}$	$\frac{1}{255}$	$\frac{50}{255}$	$\frac{1}{255}$	0.4	10
demosaicing	noise-free	$\frac{50}{255}$	$\frac{1}{255}$	$\frac{50}{255}$	$\frac{1}{255}$	0 or 0.3	10
	noise σ	$\frac{50}{255}$	σ	$\frac{50}{255}$	σ	0 or 0.3	10

	without preco.		with preco.	
Poisson denoising (peak λ)	ρ^0	α	ρ^0	α
	$\frac{1}{\lambda}$	$4^{1/N}$	1	1



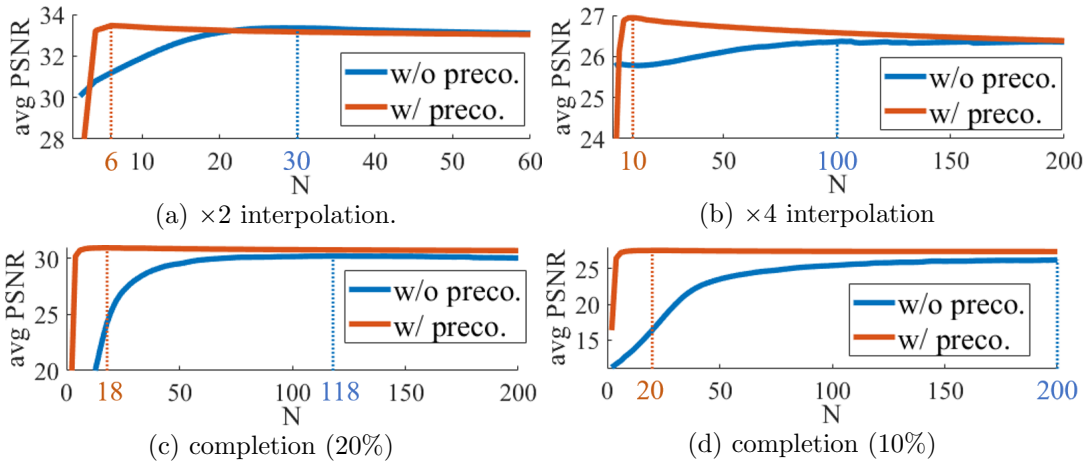
(a) Bicubic (b) Ours: $\sigma_f^{last} = 0$ (c) Ours: $\sigma_f^{last} = 0.4$

Figure 2. *Interpolation for x2 upsampling using either: (a) bicubic interpolation, (b) our method without* $\boldsymbol{P}$ *update (i.e.* $\sigma_f^{last} = 0$*), (c) our method with* $\sigma_f^{last} = 0.4$*. For (b) and (c), we used* $N = 6$ *iterations.*

case, the noise standard deviation assumed by the denoiser in Eq. (3.9) is $1/\sqrt{\rho^k}$ at each iteration k. Given an input image data in the range $[0, \lambda]$ and with Poisson noise, the highest possible noise standard deviation is $\sqrt{\lambda}$. Hence, we initialize ρ_0 so that $1/\sqrt{\rho^0} = \sqrt{\lambda}$. We found empirically that the best results are always obtained by decreasing the denoiser's standard deviation down to $1/\sqrt{\rho^N} = \sqrt{\lambda}/2$ at the last iteration N. Therefore in our experiments without preconditioning, we always use $\rho^0 = 1/\lambda$ and $\alpha = 4^{1/N}$.

A summary of the parameters setting can be found in Table 3. It does not include the number of iterations N which is studied more in detail for each application in the following sections.

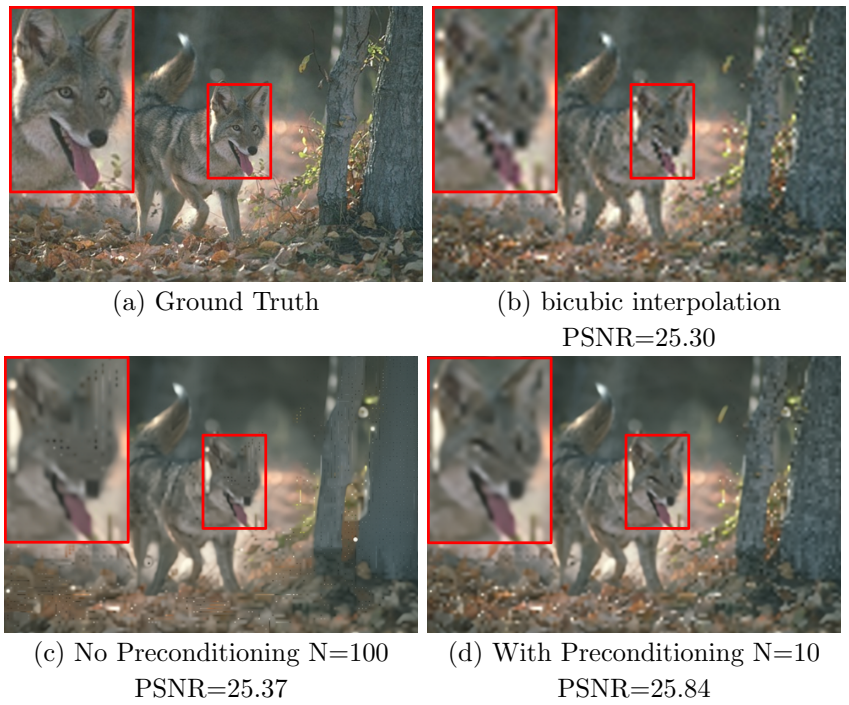
6.2. Interpolation Results. The interpolation problem can be seen as a particular case of super-resolution, where the low resolution input image was generated without applying an antialiasing filter before sub-sampling the ground truth image pixels. As a result, the main difficulty of the interpolation problem is to remove aliasing effects. First, we show in Fig. 2



(a) ×2 interpolation. (b) ×4 interpolation

(c) completion (20%) (d) completion (10%)

Figure 3. *Results of the PnP-ADMM either with or without preconditioning with respect to the number of iterations N for: (a) ×2 interpolation, (b) ×4 interpolation, (c) completion from 20% pixels, (d) completion from 10% pixels. The plots show the average PSNR over the Set5 Dataset.*



(a) Ground Truth (b) bicubic interpolation PSNR=25.30

(c) No Preconditioning N=100 PSNR=25.37 (d) With Preconditioning N=10 PSNR=25.84

Figure 4. *Example of ×4 interpolation results.*

that our method better removes aliasing when we update the preconditioning matrix $\boldsymbol{P}$ by blurring the mask of sampled pixels, as described in Section 5.1 (see Eqs. (5.1)-(5.2)). In this example, the results of the bicubic interpolation in Fig. 2(a) displays strong aliasing artifacts. In Fig. 2(b), using our preconditioned PnP-ADMM with $\sigma_f^{last} = 0$ (i.e. no update of $\boldsymbol{P}$ and no blurring of the mask) partially removes the aliasing. However, better results are obtained

Table 4

Interpolation results for ×2 and ×4 factors (average PSNR over each dataset). In each case, the number of iterations N giving the best results is used for the PnP-ADMM.

		Set5	CBSD68
×2	bicubic interpolation	31.63	27.85
	PnP-ADMM w/o preco. ($N = 30$)	33.34	28.85
	PnP-ADMM w/ preco. ($N = 6$)	33.47	29.00
×4	bicubic interpolation	25.66	23.20
	PnP-ADMM w/o preco. ($N = 100$)	26.36	23.65
	PnP-ADMM w/ preco. ($N = 10$)	26.94	23.90

in Fig. 2(c), when updating $\boldsymbol{P}$ using $\sigma_f^{last} = 0.4$.

Although many super-resolution algorithms exist and excellent performances have been obtained by deep learning techniques, these methods are generally trained assuming that the degradation already includes a given antialiasing filter. Therefore these methods are not suitable for the interpolation. Hence, we only compare our approach to the non-preconditioned PnP-ADMM, as well as the bicubic interpolation which we also use as an initialisation in the ADMM. The results of the PnP-ADMM either with or without preconditioning depends on the chosen number of iterations N. Fig. 3(a)-(b) shows that without preconditioning, the best results are obtained using respectively $N = 30$ and $N = 100$ iterations for x2 and x4 interpolation, while with our preconditioning, we only need to use $N = 6$ and $N = 10$ respectively. Note however that the results are sensitive to the rate of change of the ADMM parameters (controlled by N) both in the preconditioned and non-preconditioned cases, since a quality degradation can be seen in Fig. 3(a)-(b) for too high values of N (e.g. too slow increase of the parameter ρ). Nevertheless, from our experiments, the best setting of N does not significantly depend on the tested image. Using these values for the parameter N in each case, the results in Table 4 and Fig. 4 show that in addition to the faster convergence, our preconditioning improves the final results of the PnP-ADMM both in x2 and x4 interpolation.

6.3. Completion Results. For the completion problem, we evaluate the results using a rate of either 20% or 10% of known pixels. Unlike the interpolation problem, the known pixels do not form a regular grid and are instead selected randomly, which prevents the aliasing artifacts. For this reason, contrary to what we observed for the interpolation in Fig. 2, updating the preconditioning matrix $\boldsymbol{P}$ with a blurred version of the mask does not improve the completion results. Hence, we use $\sigma_f^{last} = 0$ in the following experiments.

Based on the results in Fig. 3(c)-(d), we use $N = 18$ and $N = 20$ iterations respectively for the 20% and 10% completion problems when using our preconditioning since these values gave the best results for the Set5 dataset. In this case, the PSNR remains stable for a higher number of iterations N. Hence, our method has little sensitivity to the rate of change of the ADMM parameters for the image completion task. Similarly to the interpolation problem, the convergence of the non-preconditioned PnP-ADMM is significantly slower: the best results for the Set5 dataset are obtained using respectively $N = 118$ and $N = 200$ for 20% and 10% completion.

For further comparisons, we also provide completion results obtained with the conven-

(a) Ground Truth (b) Input (10% pixels) (c) Total variation (d) Diffusion (EED) (e) PnP-ADMM (no preco.) (f) PnP-ADMM (preco.)

Figure 5. *Results of completion from* 10% *pixels for the butterfly image using (c) Total Variation regularization, (d) Edge-Enhancing Diffusion, (e) PnP-ADMM without preconditioning (using* $N = 200$ *iterations), (f) PnP-ADMM with our preconditioning (using* $N = 20$ *iterations).*

tional total variation (TV) regularization, using the implementation in [38]. We also compare our results to a state of the art diffusion based method. A review of diffusion methods in [39] have shown that the best completion performance was obtained with a non-linear anisotropic diffusion scheme referred to as edge enhancing diffusion (EED). The EED was also exploited within compression schemes in more recent works (e.g. [40], [41]), where only a small percentage of pixels are encoded and the remaining ones must be recovered by the decoder. Therefore, we use the EED scheme for our comparisons.

The results in Fig. 5 show that, while the EED diffusion better recovers large image structures than solving the inverse problem with TV regularization, the PnP-ADMM scheme retrieves finer details and sharper edges thanks to the advanced regularization provided by the trained denoiser. However, without our preconditioning, some important structures of the image may be lost as shown in Fig. 5(e) (e.g. white spots on the butterfly, zebras' stripes). On the other hand, our preconditioned version recovers all the structures that could be inferred from the input data, hence resulting in a large gain in PSNR. Note that this observation is consistent with the interpolation results in Fig. 4, where important details of the image are lost in the non-preconditioned PnP-ADMM but are preserved with our preconditioning. The average PSNR results in Table 5 confirm the superiority of the proposed preconditioned PnP-ADMM over the other approaches for image completion from either 10% or 20% pixels.

6.4. Demosaicing Results. For the demosaicing, we evaluate the results both in the noise-free scenario and in the presence of Gaussian noise of standard deviation $\sigma = 10$ or $\sigma = 20$. In each case, the input images are filtered by the traditional Bayer CFA, i.e. only one of the red green or blue components is known at each pixel. The Bayer CFA forms a repetitive pattern of 2x2 blocks, each containing one red, one blue and two green pixels. For our preconditioned PnP-ADMM method, we evaluate the results using either the generic denoising network *DRUNet-var-RBG* or the specialized network *DRUNet-dem* that was trained for noise level patterns generated from the Bayer CFA, as explained in Section 5.2.

Table 5

Completion results for 20% and 10% rates (average PSNR over each dataset). In each case, the number of iterations N giving the best results is used for the PnP-ADMM.

		Set5	CBSD68
20%	Total Variation regularization	26.11	24.80
	Edge Enhancing Diffusion	28.61	25.55
	PnP-ADMM w/o preco. ($N = 118$)	30.20	26.75
	PnP-ADMM w/ preco. ($N = 18$)	30.94	27.43
10%	Total Variation regularization	22.86	22.66
	Edge Enhancing Diffusion	25.85	23.43
	PnP-ADMM w/o preco. ($N = 200$)	26.20	24.06
	PnP-ADMM w/ preco. ($N = 20$)	27.52	24.95

Table 6

Demosaicing results (average PSNR over each dataset) for the Bayer pattern in the noise-free scenario ($\sigma = 0$). For our preconditioned PnP-ADMM method, we show the results using either the generic denoiser DRUNet-var-RBG or the specialized denoiser for demosaicing DRUNet-dem. The best and second best results are in red and blue respectively.

	Malvar (Matlab) [42]	Gharbi (DeepJoint) [36]	Henz [24]	Kokkinos (MMNet) [43]	PnP-ADMM		
					no preconditioning	with preconditioning	
					(*DRUNet-cst*)	*DRUNet-var-RBG*	*DRUNet-dem*
Kodak	34.68	41.85	42.05	40.26	41.45	42.37	42.32
McMaster	34.46	39.13	39.50	36.84	39.24	39.27	39.31

Similarly to the interpolation problem, the repetition of the 2x2 block pattern in the Bayer CFA may cause artifacts comparable to aliasing. When using the generic network *DRUNet-var-RBG* in our preconditioning scheme, these artifacts are better removed by updating the matrix $\boldsymbol{P}$ with a blurred version of the mask. In this case, we use $\sigma_f^{last} = 0.3$ to control the blur at each iteration. However, when using the denoiser *DRUNet-dem* specialized for such patterns, the aliasing artifacts from the Bayer CFA are naturally removed without the need for altering the preconditioning matrix. Thus we use $\sigma_f^{last} = 0$ in this case.

Note that for all the experiments, a fast initialisation is performed with Matlab's demosaicing method [42] that uses bilinear interpolation and gradient-based corrections. Given this initialisation, we fix the number of iterations N of the PnP-ADMM according to the preliminary experiment in Fig. 7. Using our preconditioning (Fig. 7(a)), close to optimal results are obtained using $N = 10$, either with or without noise, and for both the *DRUNet-var-RBG* and *DRUNet-dem* networks. For the non-preconditioned PnP-ADMM, more iterations are needed and the setting of N is also more sensitive to the noise level: the best performance is reached using $N = 40$ in the noise-free scenario ($\sigma = 0$), and using $N = 16$ in both noisy settings (see Fig. 7(b)).

For the evaluation, we use the Kodak [44] and McMaster [45] test datasets that are widely used for demosaicing benchmarks as they contain natural images which include challenging features for this application. We compare both versions of our preconditioned PnP-ADMM

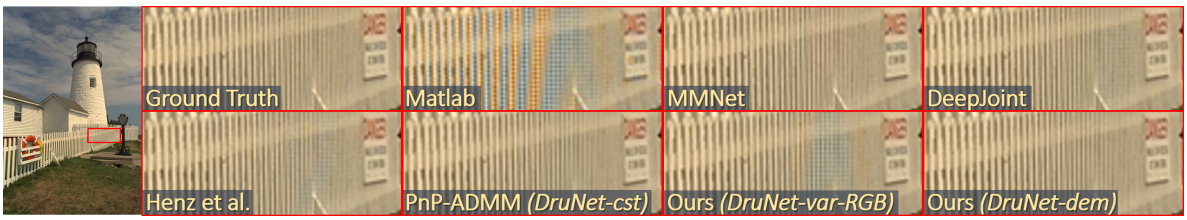


Figure 6. *Demosaicing results in the noise-free scenario. The "PnP-ADMM (DRUNet-cst)" result corresponds to the non-preconditioned version, while our method is preconditioned and uses either the generic denoiser DRUNet-var-RBG or the specialized one DRUNet-dem.*

Table 7

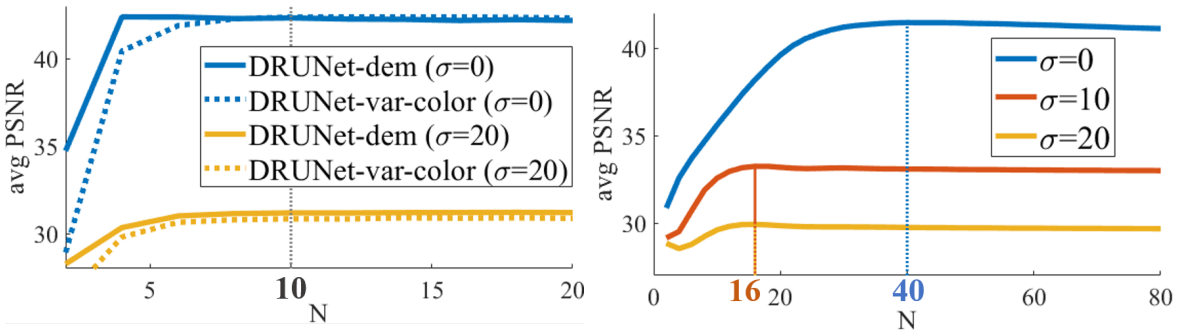
Demosaicing results (average PSNR over each dataset) for the Bayer pattern in the noisy scenario with noise of standard deviation $\sigma = 10/255$ or $\sigma = 20/255$. For our preconditioned PnP-ADMM method, we show the results using either the generic denoiser DRUNet-var-RBG or the specialized denoiser for demosaicing DRUNet-dem. The best and second best results are in red and blue respectively.

σ:		Malvar (Matlab) [42]	Gharbi (DeepJoint) [36]	Kokkinos (MMNet) [43]	PnP-ADMM		
					no preconditioning	with preconditioning	
					(*DRUNet-cst*)	*DRUNet-var-RBG*	*DRUNet-dem*
$\frac{10}{255}$	Kodak	27.54	33.27	30.96	33.24	33.90	34.18
	McMaster	27.69	33.18	30.19	33.35	33.89	34.06
$\frac{20}{255}$	Kodak	22.38	30.04	23.81	29.92	30.84	31.18
	McMaster	22.75	30.18	23.65	30.09	31.09	31.36

with the non-preconditioned version (based on the *DRUNet-cst* denoiser), as well as other reference methods including Matlab's demosaicing method by Malvar et al. [42] and the deep learning based methods of Gharbi et al. (DeepJoint) [36], Henz et al. [24] and Kokkinos et al. (MMNet) [43].

Table 6 and Fig. 6 present the results in the noise-free scenario. Our method performs among the best in average PSNR, indicating that it successfully recovers the fine details overall, and with greater accuracy than the non-preconditioned PnP-ADMM. However, in very challenging areas with high frequency patterns (see Fig. 6), our preconditioning based on the generic network *DRUNet-var-RBG* leaves some zipper and color artifacts. Note that, on closer inspection, similar artifacts also remain in the Henz et al. and DeepJoint methods. However, using the specialized denoiser *DRUNet-dem* in our scheme completely solves this issue. This shows that it was not a limitation of the preconditioning or the PnP-ADMM scheme itself, but rather a limitation of the *DRUNet-var-RBG* denoiser that does not perform optimally when the noise level map is a structured pattern constructed from the Bayer CFA.

Similar conclusions can be drawn for the noisy scenario in Table 7 and Fig. 8. Our preconditioned PnP-ADMM best recovers the image details, and using the specialized *DRUNet-dem* network prevents the color issues that may appear in challenging areas when using the generic *DRUNet-var-RBG* network.



(a) with preconditioning. (b) without preconditioning

Figure 7. *Results of the PnP-ADMM for demosaicing with respect to the number of iterations N: (a) with preconditioning, (b) without preconditioning. The plots show the average PSNR over the Kodak Dataset.*

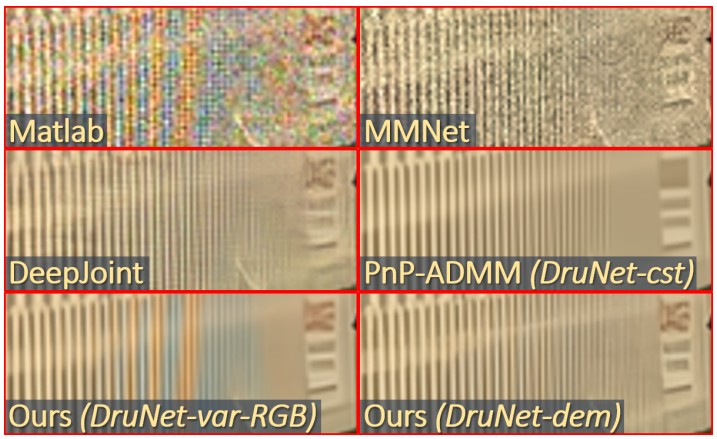


Figure 8. *Demosaicing results for the same detail as in Fig. 6, but in the noisy scenario with Gaussian noise of standard deviation $\sigma = 20$.*

6.5. Poisson Denoising. For Poisson denoising, we compare the results of the PnP-ADMM either with or without our preconditioning. Note that the non-preconditioned version was previously described in [15] for this application. However, the BM3D denoiser was used in [15]. For a fair evaluation of our approach, we use instead the *DRUNet-cst* network which gives the best performance when no preconditioning is required. For our preconditioned scheme, we use the *DRUNet-var-RBG* denoiser since the Poisson noise variance depends not only on pixels' positions, but also on the colour component.

First, we analyse the results for moderate to high levels of Poisson noise (using $\lambda \in \{255, 255/4, 255/8\}$). Table 8 and Fig. 10(a) show that the proposed preconditioning improves the results compared to the original PnP-ADMM, regardless of the fixed number of iterations N. Furthermore, a better convergence is obtained with our approach as shown in Fig. 10. While the results of the non-preconditioned PnP-ADMM start degrading when using more than $N = 6$ iterations, our method successfully converges to the best result. The improved convergence is confirmed in Fig. 10(b) showing that the equality constraint between the two

(a) Ground Truth (b) Noisy $\lambda = 255/8$ (c) No Preco. N=6 PSNR=30.15 (d) No Preco. N=100 PSNR=29.49 (e) With Preco. N=6 PSNR=30.28 (f) With Preco. N=100 PSNR=30.76

Figure 9. *Poisson Denoising given a peak value $\lambda = 255/8$. Results are shown for the PnP-ADMM either with preconditioning (e,f) or without (c,d) and using a number of iterations $N = 6$ (c,e) or $N = 100$ (d,f).*

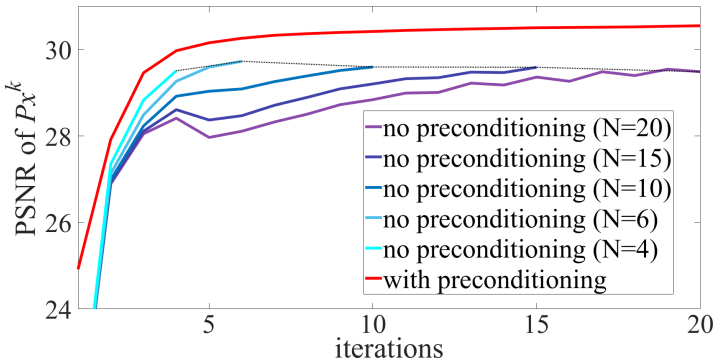
Table 8

Results of PnP-ADMM for Poisson denoising either with or without preconditioning (average PSNR over each dataset).

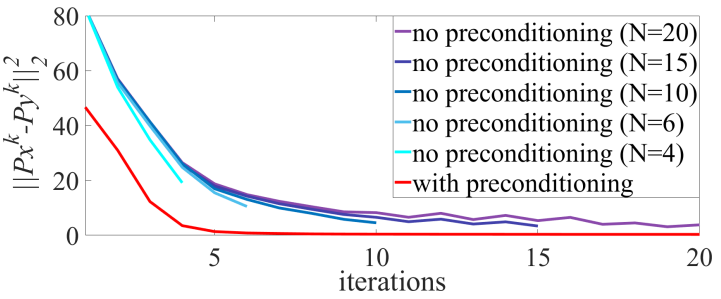
		Without Preconditioning		With Preconditioning	
	Dataset	$N = 6$	$N = 20$	$N = 6$	$N = 20$
$\lambda = 255$	Set5	36.54	36.33	36.82	36.81
	CBSD68	36.35	36.25	36.58	36.57
$\lambda = \frac{255}{4}$	Set5	33.34	33.13	33.65	33.69
	CBSD68	32.36	32.19	32.72	32.74
$\lambda = \frac{255}{8}$	Set5	31.73	31.41	31.94	32.06
	CBSD68	30.42	30.20	30.81	30.89

ADMM variables is better satisfied with our preconditioning. The visual results in Fig. 9 also show that our approach can recover finer details. Using a large number of iterations further improves the PSNR in this case, although the visual results are similar (see Fig. 9(e,f)). On the other hand, in the non-preconditioned PnP-ADMM, using a too large number of iterations visually degrades the results by removing details in the bright regions, while leaving more noise in the dark regions (see Fig. 9(c,d)).

Let us now evaluate our method for an extreme noise level, i.e. using $\lambda = 1$. In this situation, the noisy image is a very inaccurate estimate of the noise variance, and it can't be used to initialize the preconditioning matrix. Instead, we compute an initial estimation using variance stabilization and Gaussian denoising: the Anscombe transform [46] is first applied to obtain an image with approximately constant noise variance; our Gaussian denoiser is then used assuming a constant standard deviation; finally the inverse Anscombe transform is applied after denoising. The preconditioning matrix $\boldsymbol{P}$ is thus computed as the square root of this initial estimate. We also disabled the update of $\boldsymbol{P}$ which degraded our results in this case. The PSNR results using this initialization method are presented in Table 9, and visual comparisons are shown in Fig. 11. Here, the best results are obtained after convergence



(a) PSNR of the results at each iteration.



(b) Mean Square Error between the two ADMM variables

Figure 10. *Convegence plots of the PnP-ADMM for Poisson denoising with $\lambda = 255/8$ and for the image "head" (from Fig. 9). For, the non-preconditioned case, several parameters depend on the fixed number of iterations N, thus, different curves are obtained with respect to N. With our preconditioning, the curves for different values of N are superimposed on the red curve because no parameter depend on N.*

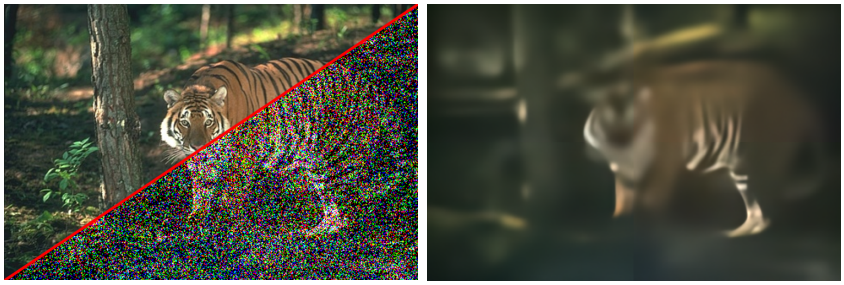
Table 9

Denoising Results for extreme Poisson noise with $\lambda = 1$ (average PSNR over each dataset).

	Anscombe initialization	PnP-ADMM (N=100) no preconditioning	PnP-ADMM (N=100) with preconditioning
Set5	17.50	22.04	23.82
CBSD68	17.30	20.50	23.05

either with or without the preconditioning. Hence the results are given for a large number of iterations $N = 100$. Our approach significantly outperforms both the non-preconditioned PnP-ADMM and the Anscombe variance stabilization method used for the initialization.

7. Conclusion. In this paper, we have proposed a new preconditioning approach for Plug-and-Play methods in the context of image restoration. Existing PnP schemes perform regularization of inverse problems using any conventional denoiser that assumes independent and identically distributed Gaussian noise. On the theoretical level, we have shown that our preconditioning removes this i.i.d. assumption by introducing in the algorithm a denoiser parameterized with a covariance matrix (directly related to the chosen preconditioning matrix) instead of a single standard deviation parameter. Greater control is thus granted for better adapting the algorithm to each task by accounting for the specific error distribution of the



(a) Ground Truth/Noisy $\lambda = 1$

(b) Anscombe (initialization) PSNR=17.89

(c) No Preconditioning N=100 PSNR=16.90

(d) With Preconditioning N=100 PSNR=22.30

Figure 11. *Poisson Denoising for very high noise level (using peak value $\lambda = 1$).*

current image estimate.

For our practical implementation, we have proposed a training procedure that generalizes a state-of-the-art CNN denoiser, enabling its parameterization with an arbitrary noise level map. While this restricts our study to diagonal covariance and preconditioning matrices, it is sufficient for many applications where the input image is degraded independently on each pixel. Such applications include image interpolation, completion, demosaicing and Poisson denoising. For each of these tasks, we have defined a suitable preconditioning scheme that significantly outperforms the non-preconditioned version both in convergence speed and image quality. The proposed method also preserves the genericity of the PnP approach since a denoiser is used to solve several inverse problems. Nevertheless, as we have shown with demosaicing, our method may be further specialized and improved for a given task by training the denoiser using the corresponding noise level maps.

A possible direction for future work would be to develop denoisers suitable for correlated noise (i.e. non-diagonal covariance matrix), hence extending the use of our preconditioned PnP scheme to a broader range of applications. Another promising direction for reducing the complexity of plug-and-play approaches would be to study efficient denoising network architectures (e.g. Transformer-based architectures as in Restormer [27]).

Appendix A. Poisson data term sub-problem.

Let us find the closed form solution of the minimization in Eq. (5.6) which we rewrite

here without the iteration numbers for simplicity of notation:

$$\arg\min_{\boldsymbol{x}} -\boldsymbol{b}^{\mathsf{T}} \ln(\boldsymbol{P}\boldsymbol{x}) + \mathbb{1}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{x} + \frac{\rho}{2}\left\|\boldsymbol{x}-\boldsymbol{u}\right\|_2^2, \tag{A.1}$$

where $\boldsymbol{P}$ is a diagonal matrix. Here, we additionally assume $\rho > 0$, $\boldsymbol{b}_i \geqslant 0$ and $\boldsymbol{P}_{i,i} > 0$, $\forall i$.

Since $\boldsymbol{P}$ is diagonal, the problem is equivalently expressed for each element i independently. Using the scalar notations $b = \boldsymbol{b}_i$, $p = \boldsymbol{P}_{i,i}$ and $u = \boldsymbol{u}_i$, the problem is to find the value x that minimizes the function h defined as:

$$h(x) = -b \cdot \ln(x \cdot p) + x \cdot p + \frac{\rho}{2}(x-u)^2 \tag{A.2}$$

The function is defined for $x > 0$ and is convex. Therefore, finding $x > 0$ such that $\frac{\mathrm{d}h}{\mathrm{d}x}(x) = 0$ gives the global minimum. Differentiating h gives:

$$\frac{\mathrm{d}h}{\mathrm{d}x} = \frac{-b}{x} + p + \rho \cdot (x-u) = 0, \tag{A.3}$$

$$= \frac{\rho}{x} \cdot \left(x^2 + x \cdot \left(\frac{p-\rho u}{\rho} \right) - \frac{b}{\rho} \right). \tag{A.4}$$

Therefore, if we can find $x > 0$ such that x is a root of the second order polynomial $x^2 + x \cdot \left(\frac{p-\rho u}{\rho}\right) - \frac{b}{\rho}$, then x minimizes the function h. For $b_i > 0$, the solution always exists and is:

$$x = \frac{1}{2}\left(\frac{\rho u - p}{\rho} + \sqrt{\left(\frac{\rho u - p}{\rho}\right)^2 + \frac{4b}{\rho}} \right), \tag{A.5}$$

$$= \frac{1}{2\rho}\left(\rho u - p + \sqrt{(\rho u - p)^2 + 4\rho b} \right). \tag{A.6}$$

Note that if $b = 0$ and $u \leqslant p/\rho$, then the highest root is $x = 0$, which is not in the domain of h. However, when $b = 0$, we can simply redefine h using $h(x) = x \cdot p + \frac{\rho}{2}(x-u)^2$ which is defined for $x = 0$. So in practice, we can extend the domain of h to the value 0 (negative values are not valid pixel values and should remain excluded). With this definition, Eq. (A.6) gives the solution in every case.

Rewritting Eq.(A.6) using the matrix notations directly gives the solution to Eq. (A.1):

$$\boldsymbol{x}_i = \frac{\rho\boldsymbol{u}_i - \boldsymbol{P}_{i,i} + \sqrt{(\rho\boldsymbol{u}_i - \boldsymbol{P}_{i,i})^2 + 4\rho\boldsymbol{b}_i}}{2\rho}. \tag{A.7}$$

Appendix B. Comparisons with the HQS algorithm.

We have established in Section 5.4 that our implementation of the PnP-ADMM is equivalent to the PnP-HQS in the particular case of noise-free sampling problems. Hence, for the interpolation, completion and noise-free demosaicing applications, the results of the PnP-ADMM either with or without preconditioning (see e.g. results in Tables 4, 5, 6) are not changed when replacing the ADMM with the HQS algorithm (i.e. by removing the dual variable $\boldsymbol{l}$).

Table 10

Demosaicing results with PnP-HQS and PnP-ADMM (average PSNR over each dataset) for the Bayer pattern in the noisy scenario with noise of standard deviation $\sigma = 10/255$ or $\sigma = 20/255$. For both algorithms the DRUNet-dem networks is used for the preconditioned version.

		PnP-HQS					PnP-ADMM	
		no preco. (*DRUNet-cst*)				preco. (*DRUNet-dem*)	no preco. (*DRUNet-cst*)	preco. (*DRUNet-dem*)
σ :		N=16	N=40	N=100	N=200	N=10	N=16	N=10
$\frac{10}{255}$	Kodak	31.38	32.08	32.40	32.52	34.23	33.24	34.18
	McMaster	31.80	32.42	32.71	32.81	34.09	33.35	34.06
$\frac{20}{255}$	Kodak	28.68	29.04	29.21	29.29	31.21	29.92	31.18
	McMaster	29.09	29.44	29.62	29.69	31.35	30.09	31.36

However, this equivalence is not true for any inverse problem. In particular, we present here additional results with the HQS algorithm for the demosaicing problem from noisy data. Note that when using the same parameter setting as in the ADMM, the HQS results in too strong denoising and lack of detail. Hence, for the experiments, instead of using the true noise variance σ^2 as the weight of the regularization term in Eq. 2.3, we use a small weight σ_{min}^2, with $\sigma_{min} = 1/255$ (i.e. σ is replaced with σ_{min} in the x-update and y-update equations). The other parameters are set as in the ADMM experiments (see Table 3).

The results are reported in Table 10, which also includes the PnP-ADMM results from Table 7 for the comparison. One can note that the preconditioned PnP-HQS yields similar performance as the preconditioned PnP-ADMM. Without the preconditioning, on the other hand, the PnP-HQS requires significantly more iterations to converge than the PnP-ADMM. Furthermore, the reconstruction quality remains lower, still due to too strong denoising despite the adjustment of the regularization weight.

REFERENCES

[1] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein, "Unrolled optimization with deep priors," *arXiv preprint arXiv:1705.08041*, 2017.

[2] M. Mardani, Q. Sun, S. Vasawanal, V. Papyan, H. Monajemi, J. Pauly, and D. Donoho, "Neural proximal gradient descent for compressive imaging," *arXiv preprintarXiv:1806.03963*, 2018.

[3] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep admm-net for compressive sensing mri," in *Proc. NeurIPS*, 2016, pp. 10–18.

[4] D. Gilton, G. Ongie, and R. Willett, "Neumann networks for linear inverse problems in imaging," *IEEE Trans. Comput. Imaging*, vol. 6, pp. 328–343, 2019.

[5] Q. Ning, W. Dong, G. Shi, L. Li, and X. Li, "Accurate and lightweight image super-resolution with model-guided deep unfolding network," *IEEE J. Sel. Topics Signal Process.*, 2020.

[6] S. Lefkimmiatis, "Universal denoising networks : A novel cnn architecture for image denoising," in *Proc. CVPR*, Jun. 2018, pp. 3204–3213.

[7] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2305–2318, 2019.

[8] A. H. Al-Shabili, H. Mansour, and P. T. Boufounos, "Learning plug-and-play proximal quasi-newton

denoisers," in *Proc. IEEE ICASSP*, 2020, pp. 8896–8900.
[9] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.
[10] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE Trans. Image Process.*, vol. 4, no. 7, pp. 932–946, 1995.
[11] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *IEEE Global Conference on Signal and Information Processing*, 2013, pp. 945–948.
[12] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.
[13] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proc. CVPR*, 2017, pp. 3929–3938.
[14] J. R. Chang, C.-L. Li, B. Póczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, "One network to solve them all — solving linear inverse problems using deep projection models," in *Proc. ICCV*, 2017, pp. 5889–5898.
[15] A. Rond, R. Giryes, and M. Elad, "Poisson inverse problems by the plug-and-play scheme," *J. Vis. Commun. Image. Represent.*, vol. 41, pp. 96–108, 2016.
[16] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, 2007.
[17] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.
[18] K. Bredies and H. Sun, "A Proximal Point Analysis of the Preconditioned Alternating Direction Method of Multipliers," *J. Optim. Theory. Appl.*, vol. 173, no. 3, pp. 878–907, Jun. 2017.
[19] Y. Jiao, Q. Jin, X. Lu, and W. Wang, "Preconditioned alternating direction method of multipliers for inverse problems with constraints," *Inverse Problems*, vol. 33, no. 2, p. 025004, Jan. 2017.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.
[21] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.
[22] T. Meinhardt, M. Moeller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems," in *Proc. ICCV*, Oct. 2017, pp. 1799–1808.
[23] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.
[24] B. Henz, E. S. Gastal, and M. M. Oliveira, "Deep joint design of color filter arrays and demosaicing," in *Computer Graphics Forum*, vol. 37, 2018, pp. 389–399.
[25] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play admm for image restoration: Fixed-point convergence and applications," *IEEE Trans. Comput. Imaging*, vol. 3, no. 1, pp. 84–98, 2017.
[26] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4608–4622, 2018.
[27] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5728–5739.
[28] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.
[29] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, "Waterloo exploration database: New challenges for image quality assessment models," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 1004–1016, 2017.
[30] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proc. CVPR Workshops*, Jul. 2017.
[31] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. CVPR Workshops*, 2017, pp. 1132–1140.
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
[33] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *Proc. ICLR*, May 2019.
[34] M. Bevilacqua, A. Roumy, C. Guillemot, and M. line Alberi Morel, "Low-complexity single-image super-

resolution based on nonnegative neighbor embedding," in *Proc. BMVC*, 2012, pp. 135.1–135.10.
[35] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. ICCV*, vol. 2, Jul. 2001, pp. 416–423.
[36] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–12, 2016.
[37] K. Wei, A. Aviles-Rivero, J. Liang, Y. Fu, C.-B. Schönlieb, and H. Huang, "Tuning-free plug-and-play proximal algorithm for inverse imaging problems," in *Proc. ICML*. PMLR, 2020, pp. 10 158–10 169.
[38] J. Dahl, P. C. Hansen, S. H. Jensen, and T. L. Jensen, "Algorithms and software for total variation image reconstruction via first-order methods," in *Numerical Algorithms*, vol. 53, no. 1, Jul. 2009, pp. 67–92.
[39] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel, "Image compression with anisotropic diffusion," *J. Math. Imaging Vis.*, vol. 31, no. 2, pp. 255–269, Apr. 2008.
[40] S. Andris, P. Peter, and J. Weickert, "A proof-of-concept framework for pde-based video compression," in *Proc. PCS*, 2016, pp. 1–5.
[41] C. Schmaltz, P. Peter, M. Mainberger, F. Ebel, J. Weickert, and A. Bruhn, "Understanding, optimising, and extending data compression with anisotropic diffusion," *Int. J. Comput. Vision*, vol. 108, no. 3, p. 222–240, Jul. 2014.
[42] H. S. Malvar, L.-W. He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *Proc. IEEE ICASSP*, 2004.
[43] F. Kokkinos and S. Lefkimmiatis, "Iterative joint image demosaicking and denoising using a residual denoising network," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4177–4188, 2019.
[44] R. Franzen, "Kodak lossless true color image suites," 1999, source: http://r0k.us/graphics/kodak.
[45] L. Zhang, X. Wu, A. Buades, and X. Li, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *J. Electron. Imaging*, vol. 20, no. 2, pp. 1 – 17, 2011.
[46] F. Anscombe, "The transformation of poisson, binomial and negative-binomial data," in *Biometrika*, vol. 35, no. 3-4, Dec. 1948, pp. 246–254.