Non-local Color Image Denoising with Convolutional Neural Networks

Stamatios Lefkimmiatis

Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia

s.lefkimmiatis@skoltech.ru

Abstract

We propose a novel deep network architecture for grayscale and color image denoising that is based on a non-local image model. Our motivation for the overall design of the proposed network stems from variational methods that exploit the inherent non-local self-similarity property of natural images. We build on this concept and introduce deep networks that perform non-local processing and at the same time they significantly benefit from discriminative learning. Experiments on the Berkeley segmentation dataset, comparing several state-of-the-art methods, show that the proposed non-local models achieve the best reported denoising performance both for grayscale and color images for all the tested noise levels. It is also worth noting that this increase in performance comes at no extra cost on the capacity of the network compared to existing alternative deep network architectures. In addition, we highlight a direct link of the proposed non-local models to convolutional neural networks. This connection is of significant importance since it allows our models to take full advantage of the latest advances on GPU computing in deep learning and makes them amenable to efficient implementations through their inherent parallelism.

1. Introduction

Deep learning methods have been successfully applied in various computer vision tasks, including image classification [16, 20] and object detection [11, 29], and have dramatically improved the performance of these systems, setting the new state-of-the-art. Recently, very promising results have also been reported for image processing applications such as image restoration [5, 39], super-resolution [18] and optical flow [1].

The significant boost in performance achieved by deep networks can be mainly attributed to their advanced modeling capabilities, thanks to their deep structure and the presence of non-linearities that are combined with discriminative learning on large training datasets. However, most of the current deep learning methods developed for im-

(a) (b)

Figure 1. Image denoising with the proposed deep non-local CNN model. (a) Noisy image corrupted with additive Gaussian noise ($\sigma = 25$) ; PSNR = 20.16 dB. (b) Denoised image using the 5-stage feed-forward network described in Sec. 3.3 ; PSNR = 29.53 dB.

age restoration tasks are based on general network architectures that do not fully exploit problem-specific knowledge. It is thus reasonable to expect that incorporating such information could lead to further improvements in performance. Only very recently, Schmidt and Roth [34] and Chen and Pock [6] introduced deep networks whose architecture is specifically tailored to certain image restoration problems. However, even in these cases, the resulting models are local ones and do not take into account the inherent *non-local self-similarity* property of natural images. On the other hand, conventional methods that have exploited this property have been shown to gain significant improvements compared to standard local approaches. A notable example is the Block Matching and 3D Collaborative Filtering (BM3D) method [7] which is a very efficient and highly engineered approach that held the state-of-the-art record in image denoising for almost a decade.

In this work, motivated by the recent advances in deep learning and relying on the rich body of algorithmic ideas

1

that have been developed in the past for tackling image reconstruction problems, we study deep network architectures for image denoising. Inspired by non-local variational methods and other related approaches, we design a network that performs non-local processing and at the same time it significantly benefits from discriminative learning. Specifically, our strategy is instead of manually designing a non-local regularization functional, to learn the non-local regularization operator and the potential function following a loss-based training approach.

Our contributions in this work can be summarized as follows: **(1)** We propose a novel deep network architecture that is discriminatively trained for image denoising. As opposed to the existing deep-learning methods for image restoration, which are based on local models, our network explicitly models the non-local self-similarity property of natural images through a grouping operation of similar patches and a joint filtering. **(2)** We unroll a proximal gradient method into a deep network and learn the relevant parameters using a simple yet effective back-propagation strategy. **(3)** In contrast to the majority of recent denoising methods that are designed for processing single-channel images, we introduce a variation of our network that applies to color images and leads to state-of-the-art results. **(4)** We highlight a direct link of our proposed non-local networks with convolutional neural networks (CNNs). This connection allows our models to take full advantage of the latest advances on GPU computing in deep learning and makes them amenable to efficient implementations through their inherent parallelism.

2. Variational Image Restoration Revisited

The goal of image denoising is the restoration of a grayscale or color image $\mathbf{X}$ from a corrupted observation $\mathbf{Y}$, with the later obtained according to the observation model

$$\mathbf{y} = \mathbf{x} + \mathbf{n}. \qquad (1)$$

In this setting, $\mathbf{y}, \mathbf{x} \in \mathbb{R}^{N \cdot C}$ are the vectorized versions of the observed and latent images, respectively, N is the number of pixels, C the number of image channels, and $\mathbf{n}$ is assumed to be i.i.d Gaussian noise with variance σ^2.

Due to the *ill-posedness* of the studied problem [38], Eq. (1) that relates the latent image to the observation cannot uniquely characterize the solution. This implies that in order to obtain a physically or statistically meaningful solution, the image evidence must be combined with suitable image priors.

Among the most popular and powerful strategies available in the literature for combining the observation and prior information is the variational approach. In this framework the recovery of $\mathbf{x}$ from $\mathbf{y}$ heavily relies on the formation of an objective function

$$E(\mathbf{x}) = D(\mathbf{x}, \mathbf{y}) + \lambda J(\mathbf{x}), \qquad (2)$$

whose role is to quantify the quality of the solution. Typically the objective function consists of two terms, namely the *data fidelity term* $D(\mathbf{x}, \mathbf{y})$, which measures the proximity of the solution to the observation, and the *regularizer* $J(\mathbf{x})$ which constrains the set of plausible solutions by penalizing those that do not exhibit the desired properties. The regularization parameter $\lambda \geq 0$ balances the contributions of the two terms. Then, the restoration task is cast as the minimization of this objective function and the minimizer corresponds to the restored image. Note that for the problem under consideration and since the noise corrupting the observation is i.i.d Gaussian, the data term should be equal to $\frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2$. This variational restoration approach has also direct links to Bayesian estimation methods and can be interpreted either as a penalized maximum likelihood or a maximum a posteriori (MAP) estimation problem [2, 13].

2.1. Image Regularization

The choice of an appropriate regularizer is very important, since it is one of the main factors that determine the quality of the restored image. For this reason, a lot of effort has been made to design novel regularization functionals that can model important image properties and consequently lead to improved reconstruction results. Most of the existing regularization methods are based either on a synthesis- or an analysis-based approach. Synthesis-based regularization takes place in a sparsifying-domain, such as the wavelet basis, and the restored image is obtained by applying an inverse transform [13]. On the other hand, analysis-based regularization involves regularizers that are directly applied on the image one aims to restore. For general inverse problems, the latter regularization strategy has been reported to lead to better reconstruction results [9, 35] and therefore is mostly preferred.

The analysis-based regularizers are typically defined as:

$$J(\mathbf{x}) = \sum_{r=1}^{R} \phi(\mathbf{L}_r \mathbf{x}), \qquad (3)$$

where $\mathbf{L} : \mathbb{R}^N \mapsto \mathbb{R}^{R \times D}$ is the regularization operator ($\mathbf{L}_r \mathbf{x}$ denotes the D-dimensional r-th entry of the result obtained by applying $\mathbf{L}$ to the image $\mathbf{x}$) and $\phi : \mathbb{R}^D \mapsto \mathbb{R}$ is the potential function. Common choices for $\mathbf{L}$ are differential operators of the first or of higher orders such as the gradient [3, 31], the structure tensor [23], the Laplacian and the Hessian [21, 24], or wavelet-like operators such as wavelets, curvelets and ridgelets (see [13] and references therein). For the potential function ϕ the most popular choices are vector and matrix norms, but other type of functions are also frequently used such as the ℓ_0 pseudo-norm and the logarithm. Combinations of the above regularization operators and potential functions lead to existing regularization functionals that have been proven very effective in several inverse problems, including image denoising. A notable representative

of the above regularizers is the Total Variation (TV) [31], where the regularization operator corresponds to the gradient and the potential function to the ℓ_2 vector norm.

TV regularization and similar methods that penalize derivatives are essentially local methods, since they involve operators that act on a restricted region of the image domain. More recently, a different regularization paradigm has been introduced where non-local operators are employed to define new regularization functionals [10, 14, 19, 22, 40]. The resulting non-local methods are well-suited for image processing and computer-vision applications and produce very competitive results. The reason is that they allow long-range dependencies between image points and are able to exploit the inherent *non-local self-similarity* property of natural images. This property implies that images often consist of localized patterns that tend to repeat themselves possibly at distant locations in the image domain.

It is worth noting that alternative image denoising methods that do not fall in the category of analysis-based regularization schemes but still exploit the self-similarity property have been developed and produce excellent results. A non-exhaustive list of these methods is the non-local means filter (NLM) [4], BM3D [7], the Learned Simultaneous Sparse Coding (LSSC) [25], and the Weighted Nuclear Norm Minimization (WNNM) [15].

2.2. Objective Function Minimization

Besides the formulation of the objective function and the proper selection of the regularizer, another important aspect in the variational approach is the minimization strategy that will be employed to obtain the solution. For the case under study, the solution to the image denoising problem can be mathematically formulated as:

$$\mathbf{x}^* = \underset{a \le x_n \le b}{\arg\min} \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda \sum_{r=1}^{R} \phi\left(\mathbf{L}_r \mathbf{x}\right)$$

$$= \underset{\mathbf{x}}{\arg\min} \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda \sum_{r=1}^{R} \phi\left(\mathbf{L}_r \mathbf{x}\right) + \iota_{\mathcal{C}}\left(\mathbf{x}\right) \quad (4)$$

where $\iota_{\mathcal{C}}$ is the indicator function of the convex set $\mathcal{C} = \left\{\mathbf{x} \in \mathbb{R}^N | x_n \in [a, b] \, \forall n = 1, \ldots N\right\}$. The indicator function $\iota_{\mathcal{C}}$ takes the value 0 if $\mathbf{x} \in \mathcal{C}$ and $+\infty$ otherwise. The presence of this additional term in Eq. (4) stems from the fact that these type of constraints on the image intensities arise naturally. For example it is reasonable to require that the intensity of the restored image should either be non-negative (non-negativity constraint with $a = 0, b = +\infty$) or its values should lie in a specific range (box-constraint).

2.3. Proximal Gradient Method

There is a variety of powerful optimization strategies for dealing with Eq. (4). The simplest approach however,

which we will follow in this work, is to directly use a gradient-descent algorithm. Since the indicator function $\iota_{\mathcal{C}}$ is non-smooth, instead of the classical gradient descent algorithm we employ the proximal gradient method [28]. According to this method, the objective function is split into two terms, one of which is differentiable. Here we assume that the potential function ϕ is smooth and therefore we can compute its partial derivatives. In this case, the splitting that we choose for the objective function has the form $E\left(\mathbf{x}\right) = f\left(\mathbf{x}\right) + \iota_{\mathcal{C}}\left(\mathbf{x}\right)$, where $f\left(\mathbf{x}\right)$ is defined as

$$f\left(\mathbf{x}\right) = \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda \sum_{r=1}^{R} \sum_{d=1}^{D} \phi_d\left(\left(\mathbf{L}_r \mathbf{x}\right)_d\right). \quad (5)$$

Note that in the above definition we have gone one step further and we have expressed the multivariable potential function ϕ as the sum of D single-variable functions,

$$\phi\left(\mathbf{z}\right) = \sum_{d=1}^{D} \phi_d\left(\mathbf{z}_d\right). \quad (6)$$

As it will become clear later, this choice will allows us to reduce significantly the computational cost for training our network and will make the learning process feasible. It is also worth noting that this decoupled formulation of the potential function is met frequently in image regularization, as in wavelet regularization [13], anisotropic TV [12] and Field-of-Experts (FoE) [30].

After the splitting of the objective function, the proximal gradient method recovers the solution in an iterative fashion, using the updates

$$\mathbf{x}^t = \text{prox}_{\gamma^t \iota_C}\left(\mathbf{x}^{t-1} - \gamma^t \nabla_{\mathbf{x}} f\left(\mathbf{x}^{t-1}\right)\right), \quad (7)$$

where γ^t is a step size and $\text{prox}_{\gamma^t \iota_C}$ is the proximal operator [28] related to the indicator function $\iota_{\mathcal{C}}$. The proximal map in this case corresponds to the orthogonal projection of the input onto $\mathcal{C}$, and hereafter will be denoted as $P_{\mathcal{C}}$.

Given that the gradient of f is computed as

$$\nabla_{\mathbf{x}} f\left(\mathbf{x}\right) = \mathbf{x} - \mathbf{y} + \lambda \sum_{r=1}^{R} \mathbf{L}_r^{\mathsf{T}} \psi\left(\mathbf{L}_r \mathbf{x}\right), \quad (8)$$

where $\psi\left(\mathbf{z}\right) = \begin{bmatrix} \psi_1\left(\mathbf{z}_1\right) & \psi_2\left(\mathbf{z}_2\right) & \ldots & \psi_D\left(\mathbf{z}_D\right) \end{bmatrix}^{\mathsf{T}}$ and $\psi_d\left(z\right) = \frac{d\phi_d}{dz}\left(z\right)$, each proximal gradient iteration can be finally re-written as

$$\mathbf{x}^t = P_{\mathcal{C}}\left(\mathbf{x}^{t-1}\left(1 - \gamma^t\right) + \gamma^t \mathbf{y} - \alpha^t \sum_{r=1}^{R} \mathbf{L}_r^{\mathsf{T}} \psi\left(\mathbf{L}_r \mathbf{x}^{t-1}\right)\right), \quad (9)$$

where $\alpha^t = \lambda \gamma^t$.

In order to obtain the solution of the minimization problem in Eq. (4) using this iterative scheme, a large number of
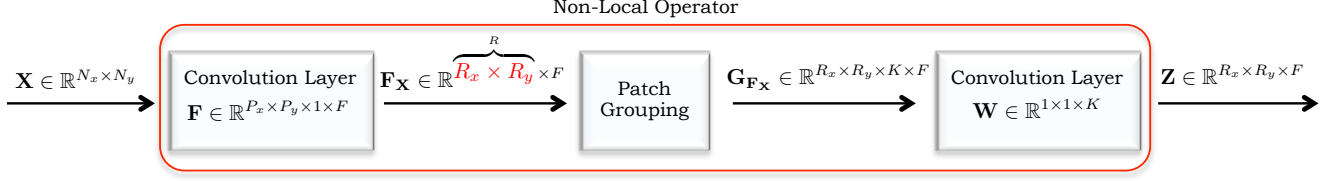
Figure 2. Convolutional implementation of the non-local operator of Eq. (12).

iterations is required. In addition, the exact form of the operator $\mathbf{L}$ and the potential function ϕ must be specified. Determining appropriate values for these quantities is in general a very difficult task. This has generated increased research interest and a lot of effort has been made for designing regularization functionals that can lead to good reconstruction results.

3. Proposed Non-Local Network

In this work, we pursue a different approach than conventional regularization methods and instead of hand-picking the exact forms of the potential function and the regularization operator, we design a network that has the capacity to learn these quantities directly from training data. The core idea is to unroll the proximal gradient method and use a limited number of the iterations derived in Eq. (9) to construct the graph of the network. Then, we learn the relevant parameters by training the network using pairs of corrupted and ground-truth data.

Next, we describe in detail the overall architecture of the proposed network, which is trained discriminatively for image denoising. First we motivate and derive its structure for processing grayscale images, and then we explain the necessary modifications for processing color images.

3.1. Non-Local Regularization Operator

As mentioned earlier, non-local regularization methods have been shown to produce superior reconstruction results than their local counterparts [14, 22] for several inverse problems, including image denoising. Their superiority in performance is mainly attributed to their ability of modeling complex image structures by allowing long-range dependencies between points in the image domain. This fact highly motivates us to explore the design of a network that will exhibit a similar behavior. To this end, our starting point is the definition of a non-local operator that will serve as the backbone of our network structure.

Let us consider a single-channel image $\mathbf{X}$ of size $N_x \times N_y$ and let $\mathbf{x} \in \mathbb{R}^N$, where $N = N_x \cdot N_y$, be the vector that is formed by stacking together the columns of $\mathbf{X}$. Further, we consider image patches of size $P_x \times P_y$ and we denote by $\mathbf{x}_r \in \mathbb{R}^P$, with $P = P_x \cdot P_y$, the vector whose elements correspond to the pixels of the r-th image patch extracted from $\mathbf{X}$. The vector $\mathbf{x}_r$ is derived from $\mathbf{x}$ as $\mathbf{x}_r = \mathbf{P}_r \mathbf{x}$,

where $\mathbf{P}_r$ is a $P \times N$ binary matrix that indicates which elements of $\mathbf{x}$ belong to $\mathbf{x}_r$. For each one of the R extracted image patches, its K closest neighbors are selected. Let $i_r = \{i_{r,1}, i_{r,2}, \ldots, i_{r,K}\}$, with $r = 1, \ldots, R$, be the set of indices of the K most similar patches to the r-th patch $\mathbf{x}_r$[1]. Next, a two-dimensional transform is applied to every patch $\mathbf{x}_r$. The patch transform can be represented by a matrix-vector multiplication $\mathbf{f}_r = \mathbf{F}\mathbf{x}_r$ where $\mathbf{F} \in \mathbb{R}^{F \times P}$. Note that if $F > P$ then the patch representation in the transform domain is redundant. In this work, we focus on the non-redundant case where $F = P$. For the transformed patch $\mathbf{f}_r$, a group is formed using the K-closest patches. This is denoted as

$$\mathbf{f}_{i_r} = \begin{bmatrix} \mathbf{f}_{i_{r,1}}^\mathsf{T} & \mathbf{f}_{i_{r,2}}^\mathsf{T} & \cdots & \mathbf{f}_{i_{r,K}}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{F \cdot K}. \quad (10)$$

The final step of the non-local operator involves collaborating filtering among the group, which can be expressed as $\mathbf{z}_r = \mathbf{W}\mathbf{f}_{i_r}$, where $\mathbf{W} \in \mathbb{R}^{F \times (F \cdot K)}$ is a weighting matrix and is constructed by retaining the first F rows of a circulant matrix. The first row of this matrix corresponds to the vector $\mathbf{r} = \begin{bmatrix} \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_K \end{bmatrix} \in \mathbb{R}^{F \cdot K}$, where $\boldsymbol{w}_i = \begin{bmatrix} w_i & 0 & \ldots & 0 \end{bmatrix} \in \mathbb{R}^F$. This collaborative filtering amounts to performing a weighted sum of the K transformed patches in the group, *i.e.*

$$\mathbf{z}_r = \sum_{k=1}^{K} w_k \mathbf{f}_{i_{r,k}}. \quad (11)$$

Based on the above, the non-local operator acting on an image patch $\mathbf{x}_r$ can be expressed as the composition of three linear operators, that is

$$\mathbf{L}_r \mathbf{x} = \left(\mathbf{W}\tilde{\mathbf{F}}\mathbf{P}_{i_r} \right) \mathbf{x}, \quad (12)$$

where $\mathbf{P}_{i_r} = \begin{bmatrix} \mathbf{P}_{i_{r,1}}^\mathsf{T} & \mathbf{P}_{i_{r,2}}^\mathsf{T} & \cdots & \mathbf{P}_{i_{r,K}}^\mathsf{T} \end{bmatrix}^\mathsf{T}$ and $\tilde{\mathbf{F}} \in \mathbb{R}^{(F \cdot K) \times (P \cdot K)}$ is a block diagonal matrix whose diagonal elements correspond to the patch-transform matrix $\mathbf{F}$. The non-local operator $\mathbf{L} : \mathbb{R}^N \mapsto \mathbb{R}^{R \cdot F}$ described above bears strong resemblance to the BM3D analysis operator studied in [8]. The main difference between the two is that for the proposed operator in (12) a weighted average of the transformed patches in the group takes place, as described in

---

[1]The convention used here is that the set i_r includes the reference patch, *i.e.* $i_{r,1} = r$.
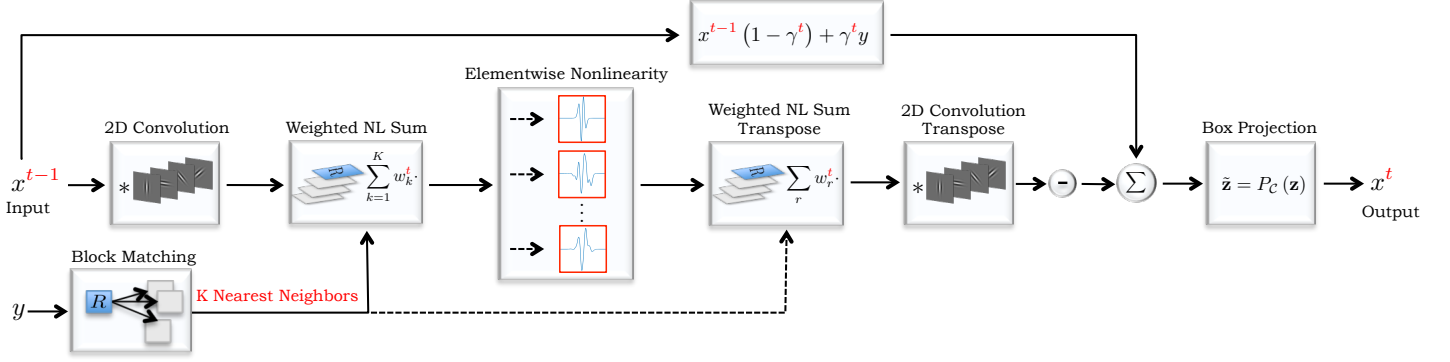
Figure 3. Architecture of a single stage of the proposed non-local convolutional network. Each stage of the network is symmetric and consists of both convolutional and de-convolutional layers. In between of these layers there is a layer of trainable non-linear functions.

Eq. (11), while for the operator of [8] a 1D Haar wavelet transform is applied on the group. Our decision for this particular set-up of the non-local operator was mainly based on computational considerations and for decreasing the memory requirements of the network that we propose next.

Due to the specific structure of the non-local operator $\mathbf{L}_r$ (composition of linear operators) it is now easy to derive its adjoint as

$$\mathbf{L}_r^\mathsf{T} = \mathbf{P}_{i_r}^\mathsf{T} \tilde{\mathbf{F}}^\mathsf{T} \mathbf{W}^\mathsf{T}. \tag{13}$$

The adjoint of the non-local operator is an important component of our network since it provides a reverse mapping from the transformed patch domain to the original image domain, that is $\mathbf{L}^\mathsf{T} : \mathbb{R}^{R \cdot F} \mapsto \mathbb{R}^N$.

3.1.1 Convolutional Implementation of the Non-Local Operator

As we explain next, both the non-local operator defined in (12) and its adjoint defined in (13) can be computed using convolution operations and their transpose. Therefore, they can be efficiently implemented using modern software libraries such as OMP and cuDNN that support multi-threaded CPU and parallel GPU implementations.

Concretely, the image patch extraction and the 2D patch transform, $\mathbf{f}_r = \mathbf{F}\mathbf{P}_r\mathbf{x}$, can be combined and computed by passing the image $\mathbf{X}$ from a convolutional layer. In order to obtain the desired output, the filterbank should consist of as many 2D filters as the number of coefficients in the transform domain. In addition, the support of these filters should match the size of the image patches. This implies that in our case F filters with a support of $P_x \times P_y$ should be used. Also note that based on the desired overlap between consecutive image patches, an appropriate stride for the convolution layer should be chosen. Finally, the non-local weighted sum operation of (11) can also be computed using convolutions. In particular, following the grouping operation of the similar transformed patches, which is completely defined

by the set $I = \{i_r : r = 1 \dots R\}$, the desired output can be obtained by convolving the grouped data with a single 3D filter of support $1 \times 1 \times K$. The necessary steps for computing the non-local operator using convolutional layers are illustrated in Fig. 2. To compute the adjoint of the non-local operator one simply has to follow the opposite direction of the graph shown in Fig. 2 and replace the convolution and patch grouping operations with their transpose operations.

3.2. Parameterization of the Potential Function

Besides the non-local operator $\mathbf{L}$, we further need to model the potential function ϕ. We do this indirectly by representing its partial derivatives ψ_i as a linear combination of Radial Basis Functions (RBFs), that is

$$\psi_i(x) = \sum_{j=1}^M \pi_{ij} \rho_j(|x - \mu_j|), \tag{14}$$

where π_{ij} are the expansion coefficients and μ_j are the centers of the basis functions ρ_j. There are a few radial functions to choose from [17], but in this work we use Gaussian RBFs, $\rho_j(r) = \exp(-\varepsilon_j r^2)$. For our network we employ $M = 63$ Gaussian kernels whose centers are distributed equidistantly and they all share the same precision parameter ε. The representation of ψ_i using mixtures of RBFs is very powerful and allow us to approximate with high accuracy arbitrary non-linear functions. This is an important advantage over conventional regularization methods that mostly rely on a limited set of potential functions such as the ones reported in Section 2.1. Also note that this parameterization of the potential gradient ψ would have been computationally very expensive if we had not adopted the decoupled formulation of Eq. (6) for the potential function.

Having all the pieces of the puzzle in order, the architecture of a single "iteration" of our network, which we will refer to it as stage, is depicted in Fig. 3. We note that our network follows very closely the proximal gradient iteration in Eq. (9). The only difference is that the parameter

5

α^t has been absorbed by the potential gradient ψ, whose representation is learned. We further observe that every stage of the network consists of both convolutional and deconvolutional layers and in between there is a layer of trainable non-linear functions.

3.3. Color Image Denoising

The architecture of the proposed network as shown in Fig. 3 can only handle grayscale images. To deal with RGB color images, a simple approach would be to use the same network to process each image channel independently. However, this would result to a sub-optimal restoration performance since the network would not be able to explore the existing correlations between the different channels.

To circumvent this limitation, we follow a similar strategy as in [7] and before we feed the noisy color image to the network, we apply the same opponent color transformation which results to one luminance and two chrominance channels. Due to the nature of the color transform, the luminance channel contains most of the valuable information about primitive image structures and has a higher signal-to-noise-ratio (SNR) than the two chroma channels. We take advantage of this fact and since the block-matching operation can be sensitive to the presence of noise, we perform the grouping of the patches only from the luminance channel. Then, we use exactly the same set of group indices $I = \{i_r : r = 1 \ldots R\}$ for the other two image channels. Another important modification that we make to the original network is that for every image channel we learn a different RBF mixture. The reason for this is that due to the color transformation the three resulting channels have different SNRs that need to be correctly accounted for. Finally, it is important to note that all the image channels share the same filters of the convolutional and weighted-sum layers and their transposes. The reasoning here is that this way the network can better exploit the channel correlations. A by-product of the specific network design is that the search for similar patches needs to be performed only once compared to the naive implementation that would demand it to be computed independently for each channel. In addition, since this operation is computed only once from the noisy input and then it is re-used in all the network stages, the processing of the color channels can take place in a completely decoupled way and therefore the network admits a very efficient parallel implementation.

4. Discriminative Network Training

We train our network, which consists of S stages, for grayscale and color image denoising, where the images are corrupted by i.i.d Gaussian noise. The network parameters $\boldsymbol{\Theta} = \left[\boldsymbol{\Theta}^1, \ldots, \boldsymbol{\Theta}^S\right]$, where $\boldsymbol{\Theta}^t = \{\gamma^t, \boldsymbol{\pi}^t, \mathbf{F}^t, \mathbf{W}^t\}$ denotes the set of parameters for the t-th stage, are learned using a loss-minimization strategy given Q pairs of train-

ing data $\left\{\mathbf{y}_{(q)}, \mathbf{x}_{(q)}\right\}_{q=1}^Q$, where $\mathbf{y}_{(q)}$ is a noisy input and $\mathbf{x}_{(q)}$ is the corresponding ground-truth image. To achieve an increased capacity for the network, we learn different parameters for each stage. Therefore, the overall architecture of the network does not exactly map to the proximal gradient method but rather to an adaptive version. Nevertheless, in each stage the convolution and deconvolution layers share the same filter parameters and, thus, they correspond to proper proximal gradient iterations.

Since the objective function that we need to minimize is non-convex, in order to avoid getting stuck in a bad local-minima but also to speed-up the training, initially we learn the network parameters by following a greedy-training strategy. The same approach has been followed in [6, 34]. In this case, we minimize the cost

$$\mathcal{L}\left(\boldsymbol{\Theta}^t\right) = \sum_{q=1}^Q \ell\left(\hat{\mathbf{x}}_{(q)}^t, \mathbf{x}_{(q)}\right), \tag{15}$$

where $\hat{\mathbf{x}}_{(q)}^t$ is the output of the t-th stage and the loss function ℓ corresponds to the negative peak signal-to-noise-ratio (PSNR). This is computed as

$$\ell\left(\mathbf{y}, \mathbf{x}\right) = -20 \log_{10}\left(\frac{P_{\text{int}}\sqrt{N}}{\|\mathbf{y} - \mathbf{x}\|_2}\right), \tag{16}$$

where N is the total number of pixels of the input images and P_{int} is the maximum intensity level (*i.e.* $P_{\text{int}} = 255$ for grayscale images and $P_{\text{int}} = 1$ for color images).

To minimize the objective function in Eq. (15) w.r.t the parameters $\boldsymbol{\Theta}^t$ we employ the L-BFGS algorithm [27] (we use the available implementation of [33]). The L-BFGS is a Quasi-Newton method and therefore it requires the gradient of $\mathcal{L}$ w.r.t $\boldsymbol{\Theta}^t$. This can be computed using the chain-rule as

$$\frac{\partial \mathcal{L}\left(\boldsymbol{\Theta}^t\right)}{\partial \boldsymbol{\Theta}^t} = \sum_{q=1}^Q \frac{\partial \hat{\mathbf{x}}_{(q)}^t}{\partial \boldsymbol{\Theta}^t} \cdot \frac{\partial \ell\left(\hat{\mathbf{x}}_{(q)}^t, \mathbf{x}_{(q)}\right)}{\partial \hat{\mathbf{x}}_{(q)}^t} \tag{17}$$

where $\frac{\partial \ell(\mathbf{y}, \mathbf{x})}{\partial \mathbf{y}} = \frac{20}{\log 10} \frac{(\mathbf{y}-\mathbf{x})}{\|\mathbf{y}-\mathbf{x}\|_2^2}$, and $\frac{\partial \hat{\mathbf{x}}_{(q)}^t}{\partial \boldsymbol{\Theta}^t}$ is the Jacobian of the output of the t-th stage, which can be computed using Eq. (9). We omit the details about the computation of the derivatives w.r.t specific network parameters and we provide their derivations in the appendix. Here, it suffices to say that the gradient of the loss function can be efficiently computed using the back-propagation algorithm [32], which is a clever implementation of the chain-rule.

For the greedy-training we run 100 L-BFGS iterations to learn the parameters of each stage independently. Then we use the learned parameters as initialization of the network and we train all the stages jointly. The joint training
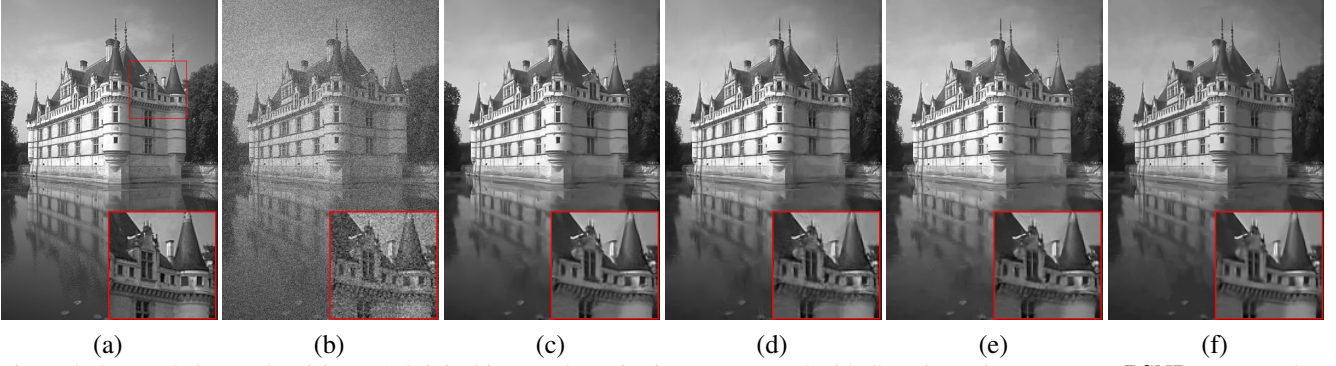
Figure 4. Grayscale image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 25$) ; PSNR = 20.16 dB. (c) Denoised image using $\mathrm{NLNet}_{7\times7}^{5}$; PSNR = 29.95 dB. (d) Denoised image using $\mathrm{TNRD}_{7\times7}^{5}$ [6] ; PSNR = 29.72 dB. (e) Denoised image using MLP [5] ; PSNR = 29.76 dB. (f) Denoised image using WNNM [15] ; PSNR = 29.76 dB.

Figure 5. Color image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 50$) ; PSNR = 14.15 dB. (c) Denoised image using $\mathrm{CNLNet}_{5\times5}^{5}$; PSNR = 26.06 dB. (d) Denoised image using CBM3D [7] ; PSNR = 25.65 dB.

corresponds to minimizing the cost function

$$\mathcal{L}\left(\mathbf{\Theta}\right) = \sum_{q=1}^{Q} \ell\left(\hat{\mathbf{x}}_{(q)}^{S}, \mathbf{x}_{(q)}\right), \qquad (18)$$

w.r.t to all the parameters of the network $\mathbf{\Theta}$. This cost function does not take into account anymore the intermediate results but only depends on the final output of the network $\hat{\mathbf{x}}_{(q)}^{S}$. In this case we run 400 L-BFGS iterations to refine the result that we have obtained from the greedy-training. Similarly to the previous case, we still employ the back-propagation algorithm to compute the required gradients.

5. Experiments

To train our grayscale and color non-local models we generated the training data using the Berkeley segmentation dataset (BSDS) [26] which consists of 500 images. We split these images in two sets, a training set which consists of 400 images and the validation/test set which consists of the remaining 100 images. All the images were randomly cropped and their resulting size was 180×180 pixel. We note that the 68 BSDS images of [30] that are used for the comparisons reported in Tables 1 and 2 are strictly excluded from the training set. The proposed models were trained on a NVIDIA Tesla K-40 GPU and the software we used for training and testing was built on top of MatConvnet [36].

Grayscale denosing Following the strategy described in Section 4, we have trained 5 stages of two different variations of our model, which we will refer to as $\mathrm{NLNet}_{5\times5}^{5}$ and $\mathrm{NLNet}_{7\times7}^{5}$. The main difference between them is the configuration of the non-local operator. For the first network we considered patches of size 5×5 while for the second one we have considered slightly larger patches of size 7×7. In both cases, the patch stride is one, that is every pixel in the image is considered as the center of a patch. Consequently, the input images at each network stage are padded accordingly, using symmetric boundaries. In addition, a non-redundant patch-transform, which was learned by training, is applied to every image-patch[2] and the group is formed using the $K = 8$ closest neighbors. The similar patches are searched on the noisy input of the network in a window of 31×31 centered around each pixel. The same group indices are then used for all the stages of the network.

In Table 1 we report comparisons of our proposed $\mathrm{NLNet}_{5\times5}^{5}$ and $\mathrm{NLNet}_{7\times7}^{5}$ models with several recent state-of-the-art denoising methods on the standard evaluation dataset of 68 images [30]. From these results we observe that both our non-local models lead to the best overall performance, with the only exception being the case of $\sigma = 50$

---

[2]Similarly to variational methods, we do not penalize the DC component of the patch-transform. Therefore, the number of the transform-domain coefficients for a patch of size P is equal to $P - 1$.

Noise	Methods									
σ (std.)	BM3D [7]	LSSC [25]	EPLL [41]	WNNM [15]	$\text{CSF}^5_{7\times7}$ [34]	$\text{TNRD}^5_{7\times7}$ [6]	DGCRF_8 [37]	MLP [5]	$\text{NLNet}^5_{5\times5}$	$\text{NLNet}^5_{7\times7}$
15	31.08	31.27	31.19	31.37	31.24	31.42	31.43	–	**31.49**	**31.52**
25	28.56	28.70	28.68	28.83	28.72	28.92	28.89	28.96	**28.98**	**29.03**
50	25.62	25.72	25.67	25.83	–	25.96	–	**26.02**	25.99	**26.07**

Table 1. Grayscale image denoising comparisons for three different noise levels over the standard set of 68 [30] Berkeley images. The restoration performance is measured in terms of average PSNR (in dB) and the best two results are highlighted in bold. The left part of the table is quoted from Chen *et al.* [6], while the results of DGCRF_8 are taken from [37].

Noise	Methods			
σ (std.)	$\text{TNRD}^5_{7\times7}$ [6]	MLP [5]	CBM3D [7]	$\text{CNLNet}^5_{5\times5}$
15	31.37	–	33.50	**33.69**
25	28.88	28.92	30.69	**30.96**
50	25.94	26.00	27.37	**27.64**

Table 2. Color image denoising comparisons for three different noise levels over the standard set of 68 [30] Berkeley images. The restoration performance is measured in terms of average PSNR (in dB) and the best result is highlighted in bold.

where the MLP denoising method [5] achieves a slightly better average PSNR compared to that of $\text{NLNet}^5_{5\times5}$. It worths noting that while $\text{NLNet}^5_{5\times5}$ has a lower capacity (it uses approximately half of the parameters) than both $\text{CSF}^5_{7\times7}$ and $\text{TNRD}^5_{7\times7}$, it still produces better restoration results in all tested cases. This is attributed to the non-local information that exploits, as opposed to $\text{CSF}^5_{7\times7}$ and $\text{TNRD}^5_{7\times7}$ which are local models. Representative grayscale denoising results that demonstrate visually the restoration quality of the proposed models are shown in Fig. 4.

Color denoising Given that in the grayscale case the use of 7×7 patches did not bring any substantial improvements compared to the use of 5×5 patches, for the color case we have trained a single configuration of our model, considering only color image patches of size 5×5. Besides the standard differences, as they are described in Section 3.3, between the color and the grayscale versions of the $\text{NLNet}^5_{5\times5}$ model, the rest of the parameters about the size of the patch-group and the search window remain the same.

An important remark to make here is that most of the denoising methods that were considered previously have been explicitly designed to treat single-channel images, with the most notable exception being the BM3D, for which it indeed exists a color-version (CBM3D) [7]. In practice, this means that if we need to restore color-images then each of these methods should be applied independently on every image channel. In this case however, their denoising performance does not anymore correspond to state-of-the-art. The reason is that due to their single-channel design they fail to capture the existing correlations between the image channels, and this limitation has a direct impact in the final restoration quality. This fact is also verified by the color denoising comparisons reported in Table 2. From these results we observe that the TNRD and MLP models, which outperform BM3D for single-channel images, fall behind in restoration performance by more than 1.3 dBs. In fact, for low noise levels CBM3D, which currently produces state-of-the-art results, leads to PSNR gains that exceed 2 dBs. Comparing the proposed non-local model with CBM3D, we observe that $\text{CNLNet}^5_{5\times5}$ manages to provide better restoration results for all the reported noise levels, with the PSNR gain ranging approximately between 0.2-0.3 dBs. We are not aware of any other color-denoising method that manages to compete with CBM3D on such large set of images. For a visual inspection of the color restoration performance of $\text{CNLNet}^5_{5\times5}$ we refer to Figs. 1 and 9.

6. Conclusions and Future Work

In this work we have proposed a novel network architecture for grayscale and color image denoising. The design of the resulting models has been inspired by non-local variational methods and it exploits the non-local self-similarity property of natural images. We believe that non-local modeling coupled with discriminative learning are the key factors of the improved restoration performance that our models achieve compared to several recent state-of-the-art methods. Meanwhile, the proposed models have direct links to convolutional neural networks and therefore can take full advantage of all the latest advances on parallel GPU computing in deep learning.

We are confident that image restoration is just one of the many inverse imaging problems that our non-local networks can successfully handle. We believe that a very interesting research direction is to investigate the necessary modifications on the design of our current non-local models that would allow them to be efficiently applied to other important reconstruction problems. Another very relevant research question is if it is possible to train a single model that can handle all noise levels.

A. Derivative Calculations

In this section we provide the necessary derivations for the gradients of the loss function of the network w.r.t the

parameters $\boldsymbol{\Theta}$. We note that for all the derivative calculations we use the denominator layout notation[3]. Further, we recall that in order to learn the parameters $\boldsymbol{\Theta} = \{\gamma^t, \boldsymbol{\pi}^t, \mathbf{F}^t, \mathbf{W}^t\}_{t=1}^S$ of the network, which consists of S stages, we use two different strategies, namely greedy and joint training. During greedy training we learn the parameters $\boldsymbol{\Theta}^t$ of each stage t of the network independently from the parameters of the other stages by minimizing the loss function of Eq. (15). On the other hand, in joint training the complete set of the network parameters is learned simultaneously by minimizing the loss function given in Eq. (18).

A.1. Single-Stage Parameter Learning

First we will consider the greedy training scheme. The results computed here will also be useful in the joint estimation scheme. Since the gradient of the overall loss $\mathcal{L}$ in Eq. (15) is decomposed as:

$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta}^t)}{\partial \boldsymbol{\Theta}^t} = \sum_{q=1}^{Q} \frac{\partial \ell\left(\hat{\mathbf{x}}_{(q)}^t, \mathbf{x}_{(q)}\right)}{\partial \boldsymbol{\Theta}^t}, \qquad (19)$$

hereafter we will consider the case of a single training example $\hat{\mathbf{x}}^t$. In order to retain the notation simplicity, in the following computations we will also drop the superscript t from all the variables and use it only when it is necessary.

As we mentioned earlier, to compute the gradients w.r.t the network parameters we rely on the chain rule and we get

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \boldsymbol{\Theta}} = \frac{\partial \hat{\mathbf{x}}}{\partial \boldsymbol{\Theta}} \cdot \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \hat{\mathbf{x}}}, \qquad (20)$$

where

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \hat{\mathbf{x}}} = \frac{20}{\log 10} \frac{(\hat{\mathbf{x}} - \mathbf{x})}{\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2}, \qquad (21)$$

is a vector of size $N \times 1$. Now we focus on the computation of the Jacobian of the output of the stage, $\hat{\mathbf{x}}$, w.r.t the stage parameters. Before doing so, we recall that the output, $\hat{\mathbf{x}}$, of a stage given an input $\mathbf{z}$, is computed according to the mapping

$$\hat{\mathbf{x}} = P_{\mathcal{C}}\left(\mathbf{z}(1-\gamma) + \gamma\mathbf{y} - \sum_{r=1}^{R} \mathbf{L}_r^{\mathsf{T}} \psi(\mathbf{L}_r \mathbf{z})\right). \qquad (22)$$

Note that the Eq. (22) is just a modified version of Eq. (9), where the variable α is absorbed by the function ψ.

The Jacobian of $\hat{\mathbf{x}}$ w.r.t the parameters of the stage, $\boldsymbol{\Theta}$, can now be expressed as

$$\frac{\partial \hat{\mathbf{x}}}{\partial \boldsymbol{\Theta}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\Theta}} \frac{\partial P_{\mathcal{C}}(\boldsymbol{u})}{\partial \boldsymbol{u}}, \qquad (23)$$

[3]For the details of this notation we refer to https://en.wikipedia.org/wiki/Matrix_calculus#Denominator-layout_notation.

where

$$\boldsymbol{u} = \mathbf{z}(1-\gamma) + \gamma\mathbf{y} - \sum_{r=1}^{R} \mathbf{L}_r^{\mathsf{T}} \psi(\mathbf{L}_r \mathbf{z}). \qquad (24)$$

Regarding the projection operator $P_{\mathcal{C}}(\boldsymbol{u})$, this is applied element-wise to the vector $\boldsymbol{u}$ and it is defined as:

$$P_{\mathcal{C}}(u) = \begin{cases} u, & \text{if } a \leq u \leq b \\ a, & \text{if } u < a \\ b, & \text{if } u > b. \end{cases} \qquad (25)$$

The derivative of $P_{\mathcal{C}}(u)$ w.r.t u is computed as:

$$\frac{dP_{\mathcal{C}}(u)}{du} = \begin{cases} 1, & \text{if } a \leq u \leq b \\ 0, & \text{elsewhere,} \end{cases} \qquad (26)$$

and therefore the Jacobian $\frac{\partial P_{\mathcal{C}}(\boldsymbol{u})}{\partial \boldsymbol{u}}$ corresponds to a binary diagonal matrix of size $N \times N$, whose diagonal elements are non-zero only if the corresponding values of $\boldsymbol{u}$ are in the range $[a\ b]$. Now, let us denote as $\boldsymbol{e}$ the $N \times 1$ vector obtained by the matrix vector product of the Jacobian $\frac{\partial P_{\mathcal{C}}(\boldsymbol{u})}{\partial \boldsymbol{u}}$ with the gradient $\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \hat{\mathbf{x}}}$, that is

$$\boldsymbol{e} = \frac{\partial P_{\mathcal{C}}(\boldsymbol{u})}{\partial \boldsymbol{u}} \cdot \frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \hat{\mathbf{x}}}. \qquad (27)$$

Weight parameter γ : Using Eq. (24) it is straightforward to show that

$$\frac{\partial \boldsymbol{u}}{\partial \gamma} = (\mathbf{y} - \mathbf{z})^{\mathsf{T}} \qquad (28)$$

and thus $\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \gamma}$ is computed as

$$\frac{\partial \ell(\hat{\mathbf{x}}, \mathbf{x})}{\partial \gamma} = (\mathbf{y} - \mathbf{z})^{\mathsf{T}} \cdot \boldsymbol{e}. \qquad (29)$$

Expansion coefficients $\boldsymbol{\pi}$: To compute the gradient of the loss function ℓ w.r.t to the expansion coefficients $\boldsymbol{\pi}$ of the mixture of Gaussian RBFs, we first express the output of the RBF mixture as a vector inner product. Specifically, it holds that

$$\psi_i(x) = \sum_{j=1}^{M} \pi_{ij} \rho_j(|x - \mu_j|) = \boldsymbol{\rho}^{\mathsf{T}}(x) \boldsymbol{\pi}_i, \qquad (30)$$

where $\boldsymbol{\rho}(x) = \begin{bmatrix} \rho(|x - \mu_1|) & \dots & \rho(|x - \mu_M|) \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^M$. We note that in the definition of $\boldsymbol{\rho}(x)$ we have dropped the subscript j from the Gaussian RBF $\rho_j(x) = \exp\left(-\varepsilon_j x^2\right)$, since we use a common precision parameter for all the mixture components, *i.e.* $\varepsilon = \varepsilon_j, \forall j$.

Based on this notation we can further express $\psi\left(\mathbf{x}\right) = \begin{bmatrix} \psi_1\left(\mathbf{x}_1\right) & \psi_2\left(\mathbf{x}_2\right) & \ldots & \psi_F\left(\mathbf{x}_F\right) \end{bmatrix}^\mathsf{T}$ as

$$\psi\left(\mathbf{x}\right) = \boldsymbol{R}^\mathsf{T}\left(\mathbf{x}\right)\boldsymbol{\pi} \tag{31}$$

where $\boldsymbol{\pi} = \begin{bmatrix} \boldsymbol{\pi}_1^\mathsf{T} & \ldots & \boldsymbol{\pi}_F^\mathsf{T} \end{bmatrix}^\mathsf{T}$, $\mathbf{x} \in R^F$ and

$$\boldsymbol{R}^\mathsf{T}\left(\mathbf{x}\right) = \begin{bmatrix} \boldsymbol{\rho}^\mathsf{T}\left(\mathbf{x}_1\right) & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\rho}^\mathsf{T}\left(\mathbf{x}_2\right) & & \mathbf{0} \\ \vdots & & \ddots & \\ \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{\rho}^\mathsf{T}\left(\mathbf{x}_F\right) \end{bmatrix} \in \mathbb{R}^{F \times (M \cdot F)}. \tag{32}$$

Now, using Eqs. (24), (30) and (31) we have

$$\boldsymbol{u} = \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R} \mathbf{L}_r^\mathsf{T} \boldsymbol{R}^\mathsf{T}\left(\mathbf{L}_r \mathbf{z}\right)\boldsymbol{\pi} \tag{33}$$

which directly leads us to compute the Jacobian $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\pi}}$ as

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\pi}} = -\sum_{r=1}^{R} \boldsymbol{R}\left(\mathbf{L}_r \mathbf{z}\right)\mathbf{L}_r. \tag{34}$$

Finally, combining Eqs. (27) and (34) we get

$$\frac{\partial \ell\left(\hat{\mathbf{x}}, \mathbf{x}\right)}{\partial \boldsymbol{\pi}} = -\sum_{r=1}^{R} \boldsymbol{R}^\mathsf{T}\left(\mathbf{L}_r \mathbf{z}\right)\mathbf{L}_r \boldsymbol{e}. \tag{35}$$

Weighted sum coefficients W : To simplify the computation of the gradient of the loss function w.r.t $\mathbf{W}$, first we obtain an equivalent expression for the non-local operator $\mathbf{L}_r$ defined in Eq. (12). Indeed, the non-local operator can be re-written as

$$\mathbf{L}_r = \sum_{k=1}^{K} w_k \mathbf{F} \mathbf{P}_{i_{r,k}} = \sum_{k=1}^{K} w_k \mathbf{T}_{i_{r,k}}. \tag{36}$$

Plugging the new expression of $\mathbf{L}_r$ into Eq. (24) we get

$$\boldsymbol{u} = \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R}\sum_{k=1}^{K} w_k \mathbf{T}_{i_{r,k}}^\mathsf{T} \psi\left(\sum_{k=1}^{K} w_k \mathbf{z}_{i_{r,k}}\right), \tag{37}$$

where $\mathbf{z}_{i_{r,k}} = \mathbf{T}_{i_{r,k}}\mathbf{z}$. Now, it is straightforward to compute the partial derivative of $\boldsymbol{u}$ w.r.t each w_i. Based on Eq. (37), we obtain

$$\frac{\partial \boldsymbol{u}}{\partial w_i} = -\sum_{r=1}^{R} \frac{\partial}{\partial w_i}\left(\left(w_i \mathbf{T}_{i_{r,i}}^\mathsf{T} + \sum_{k \neq i} w_k \mathbf{T}_{i_{r,k}}^\mathsf{T}\right)\psi\left(\mathbf{z}_{i_r}\right)\right)$$

$$= -\sum_{r=1}^{R}\left(\psi^\mathsf{T}\left(\mathbf{z}_{i_r}\right)\mathbf{T}_{i_{r,i}} + \sum_{k=1}^{K} w_k \mathbf{z}_{i_{r,i}}^\mathsf{T} \frac{\partial \psi\left(\mathbf{z}_{i_r}\right)}{\partial \mathbf{z}_{i_r}}\mathbf{T}_{i_{r,k}}\right), \tag{38}$$

where $\mathbf{z}_{i_r} = \sum_{k=1}^{K} w_k \mathbf{z}_{i_{r,k}}$. Note that due to the decoupled formulation of ψ, the Jacobian $\frac{\partial \psi(\mathbf{z}_{i_r})}{\partial \mathbf{z}_{i_r}}$ is a diagonal matrix of the form:

$$\frac{\partial \psi\left(\mathbf{x}\right)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \psi_1(\mathbf{x}_1)}{\partial \mathbf{x}_1} & 0 & \ldots & 0 \\ 0 & \frac{\partial \psi_2(\mathbf{x}_2)}{\partial \mathbf{x}_2} & & 0 \\ \vdots & & \ddots & \\ 0 & \ldots & 0 & \frac{\partial \psi_F(\mathbf{x}_F)}{\partial \mathbf{x}_F} \end{bmatrix}, \tag{39}$$

where

$$\frac{\partial \psi_i\left(x\right)}{\partial x} = -2\varepsilon \sum_{j=1}^{M} \pi_{ij}\left(x - \mu_j\right)\exp\left(-\varepsilon\left(x - \mu_j\right)^2\right). \tag{40}$$

Combining Eqs (27) and (38) we obtain:

$$\frac{\partial \ell\left(\hat{\mathbf{x}}, \mathbf{x}\right)}{\partial w_i} = -\sum_{r=1}^{R}\left(\psi^\mathsf{T}\left(\mathbf{z}_{i_r}\right)\mathbf{T}_{i_{r,i}} + \sum_{k=1}^{K} w_k \mathbf{z}_{i_{r,i}}^\mathsf{T} \frac{\partial \psi\left(\mathbf{z}_{i_r}\right)}{\partial \mathbf{z}_{i_r}}\mathbf{T}_{i_{r,k}}\right)\boldsymbol{e}. \tag{41}$$

Patch-transform coefficients F : Let us express the matrix $\mathbf{F} \in \mathbb{R}^{F \times P}$ in terms of its column vectors, *i.e.* $\mathbf{F} = \begin{bmatrix} \mathbf{f}_1 & \ldots & \mathbf{f}_F \end{bmatrix}^\mathsf{T}$ with $\mathbf{f}_i \in \mathbb{R}^P \,\forall\, i = 1, \ldots, F$. Now, let us also re-write $\mathbf{L}_r \mathbf{z}$ as

$$\mathbf{L}_r \mathbf{z} = \left(\sum_{k=1}^{K} w_k \mathbf{F} \mathbf{P}_{i_{r,k}}\right)\mathbf{z}$$

$$= \mathbf{F}\left(\sum_{k=1}^{K} w_k \mathbf{P}_{i_{r,k}}\right)\mathbf{z} = \mathbf{F}\left(\mathbf{B}_r \mathbf{z}\right)$$

$$= \mathbf{F}\tilde{\mathbf{z}}_r = \begin{bmatrix} \mathbf{f}_1^\mathsf{T}\tilde{\mathbf{z}}_r \\ \vdots \\ \mathbf{f}_F^\mathsf{T}\tilde{\mathbf{z}}_r \end{bmatrix}. \tag{42}$$

Next, we use Eq. (42) to re-write Eq. (24) as

$$\boldsymbol{u} = \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R} \mathbf{B}_r^\mathsf{T}\begin{bmatrix} \mathbf{f}_1 & \ldots & \mathbf{f}_F \end{bmatrix}\psi\left(\begin{bmatrix} \mathbf{f}_1^\mathsf{T}\tilde{\mathbf{z}}_r \\ \vdots \\ \mathbf{f}_F^\mathsf{T}\tilde{\mathbf{z}}_r \end{bmatrix}\right)$$

$$= \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R} \mathbf{B}_r^\mathsf{T}\begin{bmatrix} \mathbf{f}_1 & \ldots & \mathbf{f}_F \end{bmatrix}\begin{bmatrix} \psi_1\left(\mathbf{f}_1^\mathsf{T}\tilde{\mathbf{z}}_r\right) \\ \vdots \\ \psi_F\left(\mathbf{f}_F^\mathsf{T}\tilde{\mathbf{z}}_r\right) \end{bmatrix}$$

$$= \mathbf{z}\left(1-\gamma\right) + \gamma\mathbf{y} - \sum_{r=1}^{R} \mathbf{B}_r^\mathsf{T}\left(\sum_{j=1}^{F} \mathbf{f}_j \psi_j\left(\mathbf{f}_j^\mathsf{T}\tilde{\mathbf{z}}_r\right)\right). \tag{43}$$

This last reformulation of $\boldsymbol{u}$ greatly facilitates the computation of its Jacobian w.r.t $\mathbf{f}_i$, $\forall i = 1, \ldots, F$. Now, we can show that

$$\frac{\partial \boldsymbol{u}}{\partial \mathbf{f}_i} = -\sum_{r=1}^{R} \left(\mathbf{I}_P \cdot \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right) + \mathbf{f}_i \tilde{\mathbf{z}}_r^{\mathsf{T}} \cdot \frac{\partial \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)}{\left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)} \right) \mathbf{B}_r, \tag{44}$$

where $\mathbf{I}_P \in \mathbb{R}^{P \times P}$ is the identity matrix. Consequently, it holds

$$\frac{\partial \ell \left(\hat{\mathbf{x}}, \mathbf{x} \right)}{\partial \mathbf{f}_i} = -\sum_{r=1}^{R} \left(\mathbf{I}_P \cdot \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right) + \mathbf{f}_i \tilde{\mathbf{z}}_r^{\mathsf{T}} \cdot \frac{\partial \psi_i \left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)}{\left(\mathbf{f}_i \tilde{\mathbf{z}}_r \right)} \right) \mathbf{B}_r \boldsymbol{e}. \tag{45}$$

A.2. Joint Parameter Learning

In the joint-training scheme the parameters of all the stages of the network are learned simultaneously by minimizing the loss function of Eq. (18) which depends only on the final output of the network $\hat{\mathbf{x}}^S$. In this case we need to compute the gradient of the loss function $\ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)$ w.r.t the parameters $\boldsymbol{\Theta}^t$ of each stage t. Using the chain-rule this can be computed as

$$\frac{\partial \ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)}{\partial \boldsymbol{\Theta}^t} = \frac{\partial \hat{\mathbf{x}}^t}{\partial \boldsymbol{\Theta}^t} \cdot \frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^t} \cdot \frac{\partial \ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)}{\partial \hat{\mathbf{x}}^S}, \tag{46}$$

where $\frac{\partial \hat{\mathbf{x}}^t}{\partial \boldsymbol{\Theta}^t}$ is calculated by combining Eq. (23) and the results of Section A.1, while $\frac{\partial \ell \left(\hat{\mathbf{x}}^S, \mathbf{x} \right)}{\partial \hat{\mathbf{x}}^S}$ is given by Eq. (21). Therefore, the only remaining Jacobian that we need to compute is $\frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^t}$. This quantity can be computed recursively as

$$\frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^t} = \frac{\partial \hat{\mathbf{x}}^{t+1}}{\partial \hat{\mathbf{x}}^t} \cdot \frac{\partial \hat{\mathbf{x}}^{t+2}}{\partial \hat{\mathbf{x}}^{t+1}} \cdots \frac{\partial \hat{\mathbf{x}}^S}{\partial \hat{\mathbf{x}}^{S-1}}. \tag{47}$$

Consequently, it suffices to derive the expression for the Jacobian $\frac{\partial \hat{\mathbf{x}}^{t+1}}{\partial \hat{\mathbf{x}}^t}$ where $\hat{\mathbf{x}}^{t+1}$ is obtained from $\hat{\mathbf{x}}^t$ according to

$$\hat{\mathbf{x}}^{t+1} = P_{\mathcal{C}} \Bigg(\hat{\mathbf{x}}^t \left(1 - \gamma^{t+1} \right) + \gamma^{t+1} \mathbf{y}$$
$$- \sum_{r=1}^{R} \left(\mathbf{L}_r^{t+1} \right)^{\mathsf{T}} \psi^{t+1} \left(\mathbf{L}_r^{t+1} \hat{\mathbf{x}}^t \right) \Bigg)$$
$$= P_{\mathcal{C}} \left(\boldsymbol{u}^{t+1} \right). \tag{48}$$

Using Eq. (48), we finally get

$$\frac{\partial \hat{\mathbf{x}}^{t+1}}{\partial \hat{\mathbf{x}}^t} = \frac{\partial \boldsymbol{u}^{t+1}}{\partial \hat{\mathbf{x}}^t} \cdot \frac{\partial P_{\mathcal{C}} \left(\boldsymbol{u}^{t+1} \right)}{\partial \boldsymbol{u}^{t+1}}$$
$$= \Bigg(\mathbf{I}_N \left(1 - \gamma^{t+1} \right)$$
$$- \sum_{r=1}^{R} \left(\mathbf{L}_r^{t+1} \right)^{\mathsf{T}} \frac{\partial \psi^{t+1} \left(\hat{\mathbf{x}}_r^t \right)}{\partial \hat{\mathbf{x}}_r^t} \mathbf{L}_r^{t+1} \Bigg) \mathbf{P}^{t+1}, \tag{49}$$

where $\mathbf{I}_N \in R^{N \times N}$ is the identity matrix, $\hat{\mathbf{x}}_r^t = \mathbf{L}_r^{t+1} \hat{\mathbf{x}}^t$ and $\mathbf{P}^{t+1} = \frac{P_c \left(\boldsymbol{u}^{t+1} \right)}{\partial \boldsymbol{u}^{t+1}}$.

B. Grayscale and Color Image Denoising Comparisons

In this section we provide additional grayscale and color image denoising results for different noise levels. For grayscale image denoising we compare the performance of our non-local models with TNRD [6], MLP [5], EPLL [41] and BM3D [7], while for color image denoising we compare our non-local CNN with the state-of-the-art CBM3D method [7]. Besides the visual comparisons, in the captions of the figures we provide the PSNR score (in dB) of each method to also allow a quantitative comparison.

C. Acknowledgments

References

[1] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 4015–4023, 2015. 1

[2] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. IOP Publishing, 1998. 2

[3] K. Bredies, K. Kunisch, and T. Pock. Total generalized variation. *SIAM J. Imaging Sci.*, 3:492–526, 2010. 2

[4] A. Buades, B. Coll, and J.-M. Morel. Image denoising methods. A new nonlocal principle. *SIAM review*, 52:113–147, 2010. 3

[5] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2392–2399, 2012. 1, 7, 8, 11, 13

[6] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Trans. Pattern Anal. Mach. Intell*, 2016. to appear. 1, 6, 7, 8, 11, 12, 13

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. Image Process.*, 16(8):2080–2095, 2007. 1, 3, 6, 7, 8, 11, 12, 14, 15

[8] A. Danielyan, V. Katkovnik, and K. Egiazarian. Bm3d frames and variational image deblurring. *IEEE Trans. Image Process.*, 21(4):1715–1728, 2012. 4, 5

[9] M. Elad, P. Milanfar, and R. Rubinstein. Analysis versus synthesis in signal priors. *Inverse problems*, 23(3):947, 2007. 2

[10] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE Trans. Image Proces.*, 17:1047–1060, 2008. 3
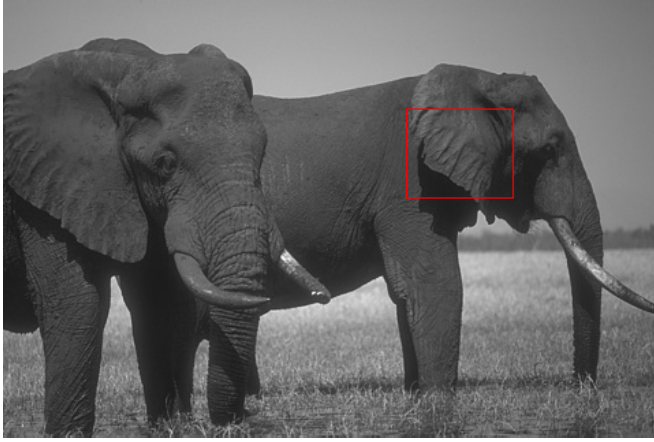
Figure 6. Grayscale image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 15$) ; PSNR = 24.57 dB. (c) Denoised image using $\text{NLNet}_{5\times5}^{5}$; PSNR = **31.58 dB**. (d) Denoised image using $\text{TNRD}_{7\times7}^{5}$ [6] ; PSNR = 31.34 dB. (e) Denoised image using EPLL [41] ; PSNR = 31.02 dB. (f) Denoised image using BM3D [7] ; PSNR = 31.29 dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region.**

[11] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2147–2154, 2014. 1

[12] S. Esedoglu and S. Osher. Decomposition of images by the anisotropic Rudin-Osher-Fatemi model. *Communications on*

(a)

(b)

(c)

(d)

(e)

(f)

Figure 7. Grayscale image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 25$) ; PSNR = 20.18 dB. (c) Denoised image using $\text{NLNet}^5_{7\times7}$; PSNR = **30.25 dB**. (d) Denoised image using $\text{TNRD}^5_{7\times7}$ [6] ; PSNR = 30.15 dB. (e) Denoised image using EPLL [41] ; PSNR = 29.92 dB. (f) Denoised image using MLP [5] ; PSNR = 30.16dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region**.

pure and applied mathematics, 57(12):1609–1626, 2004. 3

[13] M. Figueiredo, J. Bioucas-Dias, and R. Nowak. Majorization–minimization algorithms for wavelet-

based image restoration. *IEEE Trans. Image Process.*, 16:2980–2991, 2007. 2, 3

[14] G. Gilboa and S. Osher. Nonlocal operators with applications
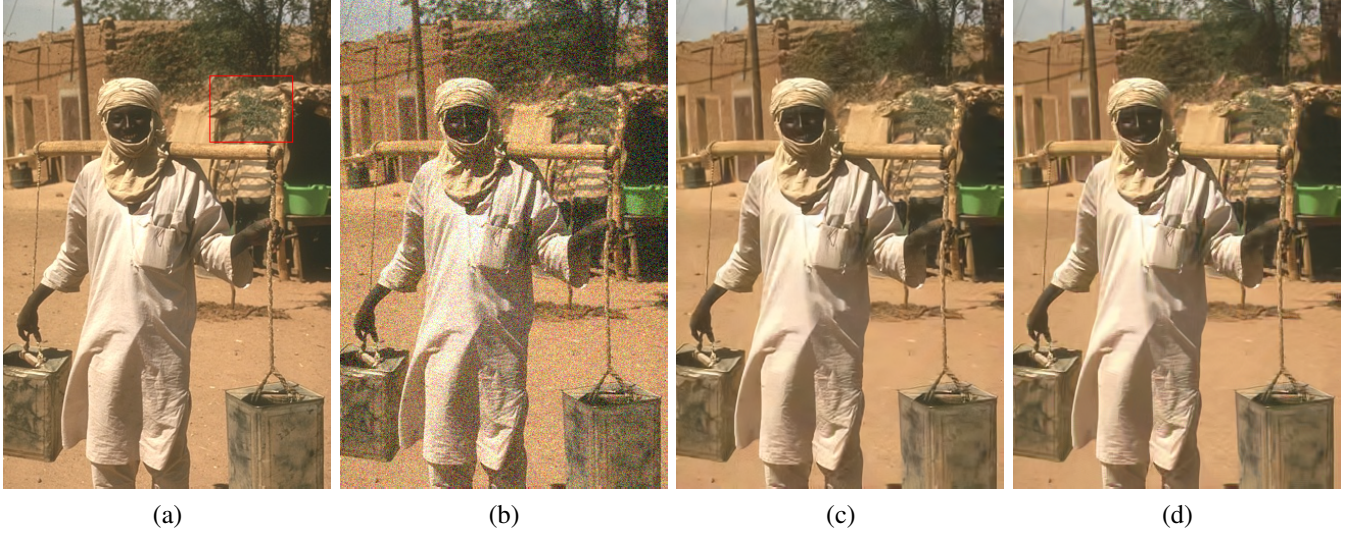
| (a) | (b) | (c) | (d) |

Figure 8. Color image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 25$) ; PSNR = 20.34 dB. (c) Denoised image using $\text{CNLNet}_{5\times5}^{5}$; PSNR = **31.14 dB**. (d) Denoised image using CBM3D [7] ; PSNR = 30.75 dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region**.

to image processing. *Multiscale Model. Simul.*, 7:1005–1028, 2008. 3, 4

[15] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2862–2869, 2014. 3, 7, 8

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2016. 1

[17] Y. H. Hu and J.-N. Hwang. *Handbook of neural network signal processing*. CRC press, 2001. 5

[18] J. Kim, K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 1646–1654, 2016. 1

[19] S. Kindermann, S. Osher, and P. W. Jones. Deblurring and denoising of images by nonlocal functionals. *Multiscale Model. Simul.*, 4:1091–1115, 2005. 3

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[21] S. Lefkimmiatis, A. Bourquard, and M. Unser. Hessian-based norm regularization for image restoration with biomedical applications. *IEEE Trans. Image Process.*, 21(3):983–995, 2012. 2

[22] S. Lefkimmiatis and S. Osher. Non-local Structure Tensor functionals for image regularization. *IEEE Trans. Comput. Imaging*, 1:16–29, 2015. 3, 4

[23] S. Lefkimmiatis, A. Roussos, P. Maragos, and M. Unser. Structure tensor total variation. *SIAM J. Imaging Sci.*, 8:1090–1122, 2015. 2

[24] S. Lefkimmiatis, J. Ward, and M. Unser. Hessian Schatten-norm regularization for linear inverse problems. *IEEE Trans. Image Process.*, 22(5):1873–1888, 2013. 2

[25] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Proc. IEEE Int. Conf. Computer Vision*, pages 2272–2279, 2009. 3, 8

[26] 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. IEEE Int. Conf. Computer Vision*, pages 416–423, 2001. 7

[27] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006. 6

[28] N. Parikh and S. Boyd. *Proximal Algorithms*. Now Publishers, 2013. 3

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[30] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205–229, 2009. 3, 7, 8

[31] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992. 2, 3

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 6

[33] M. Schmidt. minFunc: unconstrained differentiable multivariate optimization in Matlab. http://www.cs.ubc.ca/~schmidtm/Software/, 2005. 6

[34] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 2774–2781, 2014. 1, 6, 8

[35] I. Selesnick and M. Figueiredo. Signal restoration with overcomplete wavelet transforms: Comparison of analysis and synthesis priors. In *SPIE (Wavelets XIII)*, 2009. 2
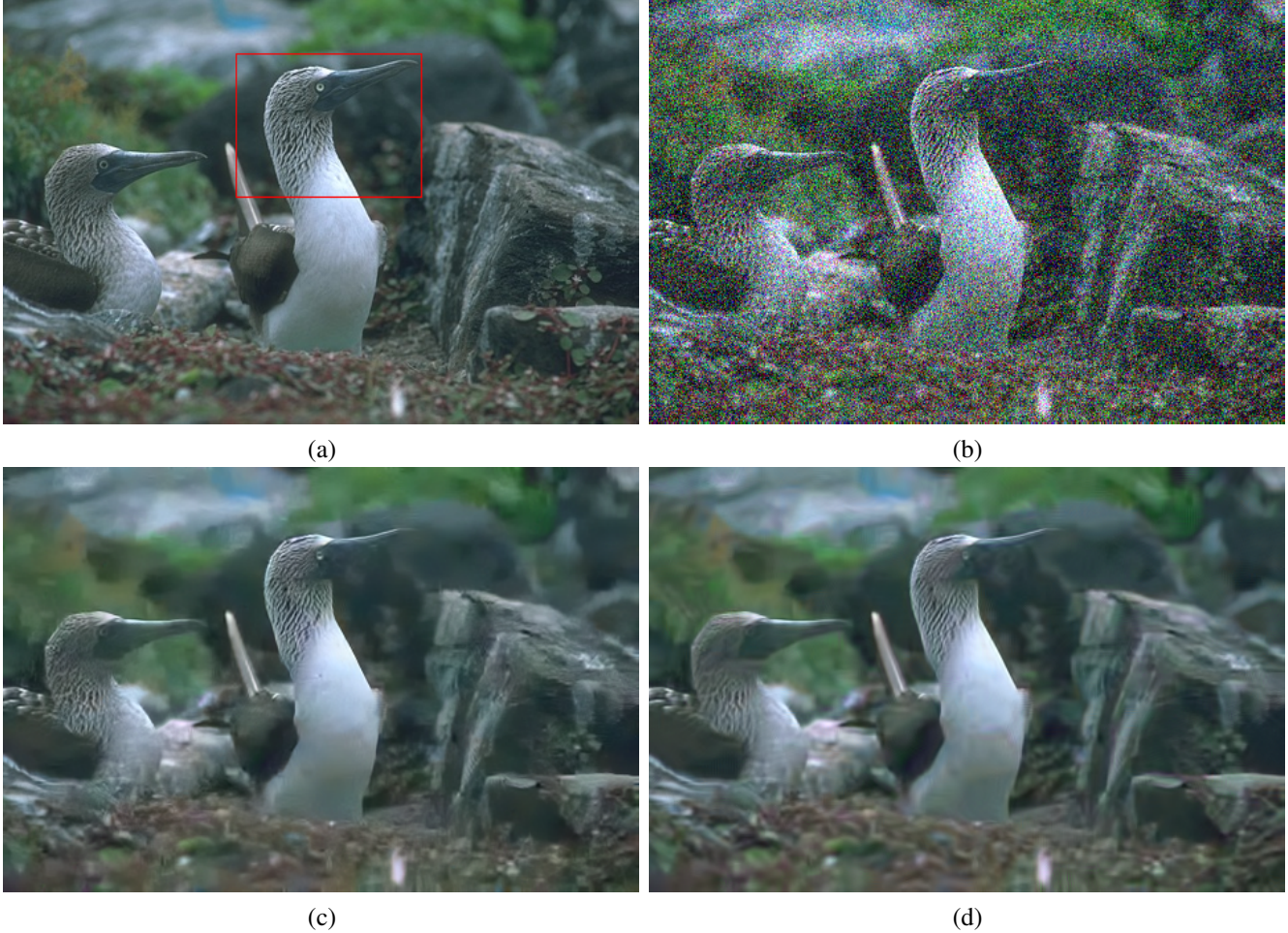
14

(a)

(b)

(c)

(d)

Figure 9. Color image denoising. (a) Original image, (b) Noisy image corrupted with Gaussian noise ($\sigma = 50$) ; PSNR = 15.10 dB. (c) Denoised image using $\text{CNLNet}_{5\times5}^{5}$; PSNR = **24.74 dB**. (d) Denoised image using CBM3D [7] ; PSNR = 24.39 dB. **Images are best viewed magnified on screen. Note the differences of the denoised results in the highlighted region**.

[36] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015. 7

[37] R. Vemulapalli, O. Tuzel, and M.-Y. Liu. Deep Gaussian conditional random field network: A model-based deep network for discriminative denoising. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 4801–4809, 2016. 8

[38] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, 2002. 2

[39] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*, pages 341–349, 2012. 1

[40] D. Zhou and B. Schölkopf. Regularization on discrete spaces. In *Pattern Recognition*, pages 361–368. Springer, 2005. 3

[41] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proc. IEEE Int. Conf. Computer Vision*, pages 479–486. IEEE, 2011. 8, 11, 12, 13