Statistical Nearest Neighbors for Image Denoising

Iuri Frosio and Jan Kautz

***Abstract*— Non-local-means image denoising is based on processing a set of neighbors for a given reference patch. few nearest neighbors (NN) can be used to limit the computational burden of the algorithm. Resorting to a toy problem, we show analytically that sampling neighbors with the NN approach introduces a bias in the denoised patch. We propose a different neighbors' collection criterion to alleviate this issue, which we name statistical NN (SNN). Our approach outperforms the traditional one in case of both white and colored noise: fewer SNNs can be used to generate images of superior quality, at a lower computational cost. A detailed investigation of our toy problem explains the differences between NN and SNN from a grounded point of view. The intuition behind SNN is quite general, and it leads to image quality improvement also in the case of bilateral filtering. The MATLAB code to replicate the results presented in the paper is freely available.**

***Index Terms*— Denoising, non-local-means, nearest neighbors.**

I. Introduction

SELF-SIMILARITY driven algorithms are based on the assumption that, for any patch in a natural image, replicas of the same patch exist within the image and can be employed, among other applications, for effective denoising [1]–[4]. Since processing uses non-local (NL) information, these algorithms are commonly referred to as NL algorithms.

Non-Local-Means (NLM), one of the most well-known denoising algorithms, has been widely investigated by researchers. It is conceptually simple: denoising of a given patch is obtained as a weighted average of the surrounding patches, with weights proportional to the patch similarity. Duval *et al.* [5] analyzed the complex relation between the filtering parameters and the quality of the output images, whereas others concentrated their attention on reducing the computational burden of the filter to make it useful in practical applications [6]. A widely used practice is to reduce the number of neighbors collected for each reference patch: noticeably, the 3D Block-Matching (BM3D) denoising filter achieves state-of-the-art results in this way [2]. The neighbors' set is collected through a Nearest-Neighbors (NN) approach, which can be efficiently (although in an approximate manner) implemented [6]. Reducing the number of neighbors leads to images with sharp edges [5], but it also introduces low-frequency artifacts, clearly visible for instance in Fig. 1.

Manuscript received August 16, 2017; revised April 24, 2018 and June 19, 2018; accepted August 28, 2018. Date of publication September 13, 2018; date of current version October 11, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christos Bouganis. *(Corresponding author: Iuri Frosio.)*

The authors are with Nvidia, Santa Clara, CA 95051 USA (e-mail: ifrosio@nvidia.com; jkautz@nvidia.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. The material includes information related to the main paper. The total size of the file is 63.3 MB. Contact ifrosio@nvidia.com for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2869685

Resorting to a toy problem, we show that this artifact occurs because the estimate of the noise-free patch from the set of NNs is biased towards the noisy reference patch. To the best of our knowledge, this is the first time that the neighbor collection strategy is explicitly investigated in detail as a potential source of bias, although other authors identified and tried reducing it through different weighting schemes [7], analyzed it in relation to the size of the local search window [8], or in the context of internal external denoising [9]; other types of bias were also analyzed in the past [3], [5], [10]. Here we propose an alternative strategy to collect neighbors, named Statistical NN (SNN), which reduces the prediction error of the estimate of the noise-free patch. When filtering real images, SNN tends to blur low-contrast image details with a low signal-to-noise ratio more than NN; we explain this drawback of SNN resorting to our toy problem to analyze the differences between NN and SNN from a statistically grounded point of view, and show that a compromise between the NN and SNN strategies is easily found in practice. Our analysis and experimental results, show that, using fewer neighbors, SNN leads to an improvement in the perceived image quality, as measured by several image quality metrics on a standard image dataset, both in case of white and colored Gaussian noise. In the latter case, visual inspection reveals that NLM with SNN achieves an image quality comparable to the state-of-the-art, at a much lower computational cost. We finally show that the intuition behind SNN is indeed quite general, and it can be applied to bilateral filtering, also leading to an image quality improvement.

II. Related Work

NLM denoising [1], [4] averages similar patches and aggregates the averages in the final image. Given the vectorial representation of an $S \times S \times C$ noisy reference patch, $\boldsymbol{\mu_r} = [\mu_r^0\ \mu_r^1 .... \ \mu_r^{S^2C-1}]$, and a noisy neighbor patch $\boldsymbol{\gamma_k}$, we first define the squared distance of $\boldsymbol{\mu_r}$ and $\boldsymbol{\gamma_k}$ as:

$$d^2\left(\boldsymbol{\mu_r}, \boldsymbol{\gamma_k}\right) = \frac{1}{P} \cdot \sum_{i=0}^{P-1} \left(\mu_r^i - \gamma_k^i\right)^2, \tag{1}$$

where $P = S^2C$. Following [4], we define an exponential kernel to weight the patch $\boldsymbol{\gamma_k}$ in the average:

$$w_{\boldsymbol{\mu_r},\boldsymbol{\gamma_k}} = e^{-\frac{\max(0, d^2(\boldsymbol{\mu_r},\boldsymbol{\gamma_k}) - 2\sigma^2)}{h^2}}, \tag{2}$$

where h is referred to as the *filtering parameter* and it depends on σ^2, which is the variance of the zero-mean, white Gaussian noise affecting the image [1], [4]. Apart from the aggregation step, the estimate of the noise-free patch, $\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r})$, is:

$$\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r}) = \sum_k w_{\boldsymbol{\mu_r},\boldsymbol{\gamma}_k} \cdot \boldsymbol{\gamma}_k \quad / \quad \sum_k w_{\boldsymbol{\mu_r},\boldsymbol{\gamma}_k}. \tag{3}$$

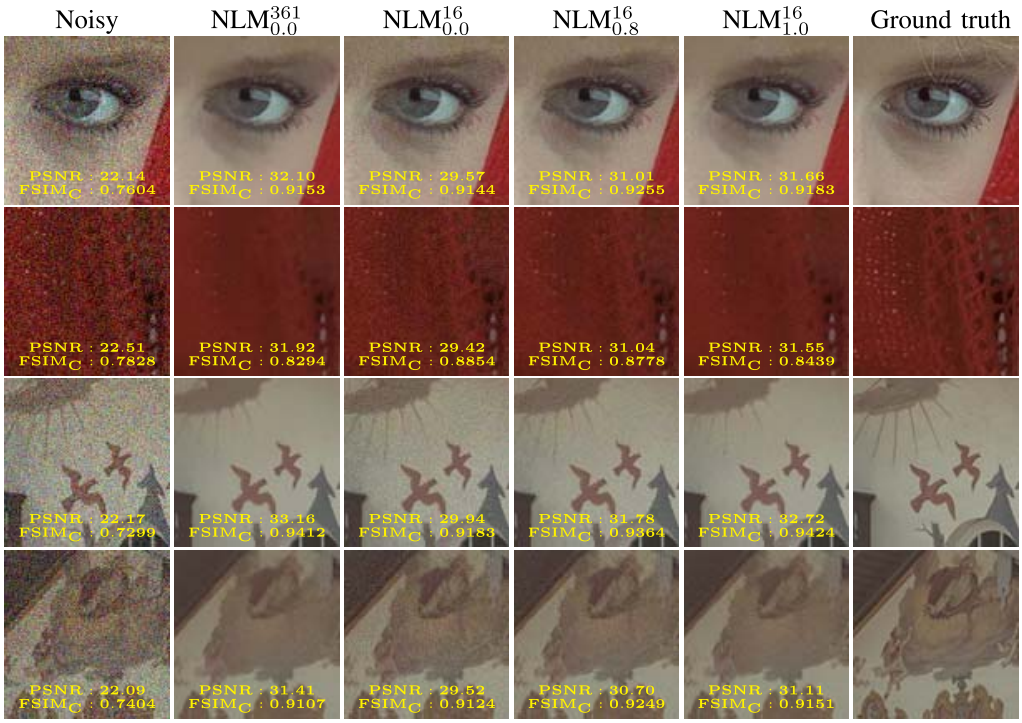


Fig. 1. Noisy, 100×100 patches from the Kodak dataset, corrupted by zero-mean Gaussian noise, $\sigma = 20$. Traditional NLM ($NLM_{0.0}^{361}$) uses 361, 3×3 patches in a 21×21 search window; it removes noise effectively in the flat areas (the skin, the wall), but it blurs the small details (see the texture of the textile, fresco details). Using 16 NNs for each patch ($NLM_{0.0}^{16}$) improves the contrast of small details, but introduces colored noise in the flat areas. The proposed SNN technique ($NLM_{1.0}^{16}$) uses 16 neighbors and mimics the results of traditional NLM in flat areas (high PSNR), while keeping the visibility of small details (high FSIM$_C$). Best results are achieved when patches are collected through SNN, with $o = 0.8$ ($NLM_{0.8}^{16}$). Better seen at 400% zoom. Full images are shown in the Supplementary Material.

Several authors proposed improvements of the original NLM idea. Lou *et al*. [11] consider scale-invariant descriptors for patch matching, extending the concept of patch similarity beyond simple translation. Enlarging the search space for similar patches has the potential of improving image quality, but it also increases the computational cost of NLM, making it impractical. Lotan and Irani [12] adopts a multi-resolution approach to better identify matching patches in case of high noise level, but do not correct for the bias introduced by the NN selection strategy. Duval *et al*. [5] interpret NLM denoising as a bias / variance dilemma. Given a noisy reference patch, the prediction error of the estimator $\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r})$ can be decomposed into the sum of a bias term, associated to the offset of $\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r})$ with respect to the ground truth $\boldsymbol{\mu}$, and a second term, associated to the variance of the estimator [5], [13]. Because of the infinite support of the weight function in Eq. (2), NLM is biased, even in absence of noise [5]: *false matchings* patches $\boldsymbol{\gamma_k}$ (dissimilar from $\boldsymbol{\mu}$) are given a non-zero weight in Eq. (3), and consequently blur small details in the estimate of the noise-free patch, $\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r})$. Decreasing the number of neighbors increases the variance term but reduces the bias, by discharging false matching patches with very small weights in Eq. (3). In practice, NLM improves by reducing the size of the search window around the reference patch and collecting few NNs through an exact or approximate search [6], [8]. This leads to a better preservation of the small details (see the texture of the textile, the sun rays and the details of the fresco in Fig. 1), but it also introduces colored noise in the flat areas of the image (see the skin and the wall in Fig. 1). We refer to this drawback as the *noise-to-noise* matching problem: since neighbors are collected around a noisy patch $\boldsymbol{\mu_r}$, their weighted average is also biased towards $\boldsymbol{\mu_r}$, which eventually leads to a partial cancellation of the noise [7]. This effect is even more evident when the size of the search window for similar patches increases [8], up to the limit case of patches taken from an external dataset [9]. Through a statistical analysis of the NLM weights, Wu *et al*. [10] identify another source of bias in the estimate $\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r})$: since the best matching neighbor for $\boldsymbol{\mu_r}$ is not associated with the highest weight, they propose a more effective weighting scheme; their approach is mainly based on the analysis of the correlation between overlapping neighbor patches and it does not explicitly consider the bias introduced by the noise in the reference patch.

The use of a small set of neighbors suggested in [5] is common in the state-of-the art algorithms like BM3D [2], BM3D-SAPCA [14], and NL-Bayes [3]. These effective NL algorithms first compute a sparse representation of the neighbors in the proper domain (*e.g.*, through a DCT or PCA transform) and filter out small coefficients associated with noise, which is assumed to be uncorrelated among different pixels and patches. Image quality is generally superior when compared to NLM, but these methods continue to suffer from visible artifacts on sharp edges (because of the limited

capability of representing any signal with the chosen basis) and in smooth image regions [15].

More recently, machine learning has also been employed to learn how to optimally combine a set of patches for denoising (alternatively to the fixed processing flow operated by BM3D) [16], but without considering the potential bias introduced by the selection of the set of NN patches.

III. Neighbor Selection for Patch Denoising

NLM denoising is a three step procedure: i) for each patch, the neighbors are identified; ii) neighbors are averaged as in Eq. (3); iii) denoised patches are aggregated (patches are partially overlapping so that multiple estimates are averaged for each pixel). None of the existing methods focus their attention on the fact that the neighbors' selection criterion affects the prediction error of $\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r})$ in step ii).

Here, we first simplify the problem of estimating the noise-free patch to a toy problem with a single-pixel patch and uniform weights, and show that the NN search strategy does introduce a bias in $\hat{\boldsymbol{\mu}}(\boldsymbol{\mu_r})$, because of the presence of noise in $\boldsymbol{\mu_r}$. Based on the expected distance between noisy patches (section III-A), we propose SNN to overcome this problem (section III-D) and demonstrate analytically the reduction of the prediction error on the toy problem (section III-E). In section IV we study our toy problem to compare the NN and SNN approaches in various cases of practical importance. Results on real data are presented in section V.

A. Statistics of Patch Distance

We consider the case of an image corrupted by white, zero-mean, Gaussian noise with variance σ^2. The search for similar patches is performed by computing the distance between $\boldsymbol{\mu_r}$ and patches of the same size, but in different positions in the image, as in Eq. (1). If the reference patch $\boldsymbol{\mu_r}$ and its neighbor $\boldsymbol{\gamma_k}$ are two noisy replica of the same patch, we get:

$$d^2(\boldsymbol{\mu_r}, \boldsymbol{\gamma_k}) = (2\sigma^2/P) \cdot \sum_{i=0}^{P-1} G(0,1)^2, \tag{4}$$

where $G(\mu, \sigma^2)$ indicates a Gaussian random variable with mean μ and variance σ^2. The sum of P squared normal variables has a χ^2_P distribution with P degrees of freedom, therefore $d^2(\boldsymbol{\mu_r}, \boldsymbol{\gamma_k}) \sim (2\sigma^2/P) \cdot \chi^2_P$, and:

$$E[d^2(\boldsymbol{\mu_r}, \boldsymbol{\gamma_k})] = 2\sigma^2. \tag{5}$$

Thus, for two noisy replicas of the same patch, the expected squared distance is not zero. This has already been noticed and effectively employed since the original NLM paper [1] to compute the weights of the patches as in Eq. (2), giving less importance to patches at a distance larger than $2\sigma^2$, or to build a better weighting scheme, as in [10]. Nonetheless, to the best of our knowledge, it has never been employed as a driver for the selection of the neighbor patches, as we do here.

B. A Toy Problem: Denoising Single-Pixel Patches

To introduce the SNN approach while dealing with a mathematically tractable problem, and support intuition with analytical evidence, we describe here a simplified toy problem where a 1×1 reference patch with noise-free value μ is corrupted by zero-mean Gaussian noise with variance σ^2, i.e. $\mu_r \sim G(\mu, \sigma^2)$. Its cumulative (cdf) and probability density function (pdf) are:

$$P(\mu_r < x) = \Phi_{\mu,\sigma^2}(x) = 1/2 \cdot [1 + \mathrm{erf}(\frac{x-\mu}{\sqrt{2}\sigma})], \tag{6}$$

$$\phi_{\mu,\sigma^2}(x) = \Phi'_{\mu,\sigma^2}(x) = \frac{1}{(\sqrt{2\pi}\sigma)} e^{-0.5\ [(x-\mu)/\sigma]^2}. \tag{7}$$

We assume that N noisy neighbors $\{\gamma_k\}_{k=1..N}$ are available; a fraction of these, $p_r \cdot N$, are noisy replicas of μ, thus they are distributed as $G(\mu, \sigma^2)$. The remaining $p_f \cdot N$ neighbors represent false matches, and follow a $G(\mu_f, \sigma^2)$ distribution. Thus the cdf and pdf of $\{\gamma_k\}_{k=1..N}$ are respectively:

$$P(\gamma_k < x) = \Phi_{mix}(x) = p_r \Phi_{\mu,\sigma^2}(x) + p_f \Phi_{\mu_f,\sigma^2}(x), \tag{8}$$

$$\phi_{mix}(x) = \Phi_{mix}(x)' = p_r \phi_{\mu,\sigma^2}(x) + p_f \phi_{\mu_f,\sigma^2}(x). \tag{9}$$

We also define a simplified estimator $\hat{\mu}(\mu_r)$ as:

$$\hat{\mu}(\mu_r) = \frac{1}{N_n} \sum_{k=1..N_n} \gamma_k, \tag{10}$$

where we neglect the weights w_{μ_r,γ_k} in Eq. (3). Beyond rendering the problem more manageable, this simplification allows us isolating the effect of the neighbors' sampling strategy from the effect of weighting, that has been analyzed by other researchers [5], [8], [10]. The validity of this assumption is further discussed in Section VI.

For any reference patch μ_r, the prediction error of $\hat{\mu}(\mu_r)$ is decomposed into its bias and variance terms as:

$$\epsilon^2(\mu_r) = \int_{\hat{\mu}} \left(\hat{\mu} - \mu\right)^2 p(\hat{\mu}) d\hat{\mu} = \epsilon^2_{bias}(\mu_r) + \epsilon^2_{var}(\mu_r)$$

$$\epsilon^2_{bias}(\mu_r) = \int_{\hat{\mu}} \left(\mathrm{E}[\hat{\mu}] - \mu\right)^2 p(\hat{\mu}) d\hat{\mu} = \{\mathrm{E}[\hat{\mu}(\mu_r) - \mu]\}^2 \tag{11}$$

$$\epsilon^2_{var}(\mu_r) = \int_{\hat{\mu}} \left(\hat{\mu} - \mathrm{E}[\hat{\mu}]\right)^2 p(\hat{\mu}) d\hat{\mu} = \mathrm{Var}[\{\hat{\mu}(\mu_r)\}], \tag{12}$$

where we omit the dependency of $\hat{\mu}(\mu_r)$ from μ_r within the integrals for notation clarity, and $p(\hat{\mu})$ is the pdf of $\hat{\mu}$. The total error of the estimator is finally computed by considering the distribution of μ_r, as:

$$\varepsilon^2 = \varepsilon^2_{bias} + \varepsilon^2_{var}, \tag{13}$$

$$\varepsilon^2_{bias} = \int_{\mu_r} \epsilon^2_{bias}(\mu_r) \phi_{\mu,\sigma^2}(\mu_r) d\mu_r, \tag{14}$$

$$\varepsilon^2_{var} = \int_{\mu_r} \epsilon^2_{var}(\mu_r) \phi_{\mu,\sigma^2}(\mu_r) d\mu_r. \tag{15}$$

C. Prediction Error of NN

When neighbors are collected with the NN approach, the set $\{\gamma_k\}_{k=1..N_n}$ contains the N_n samples closest to μ_r, and $\hat{\mu}(\mu_r)$ is its sample average. Fig. 2a shows an example of collecting $N_n = 16$ NNs, for $\mu = 0$, $\sigma = 0.2$ and $p_f = 0$, i.e. when false matches are not present.

To compute the estimation error (Eq. (13)), we need to study the statistical distribution of $\hat{\mu}(\mu_r)$ and compute $\mathrm{E}[\hat{\mu}(\mu_r)]$

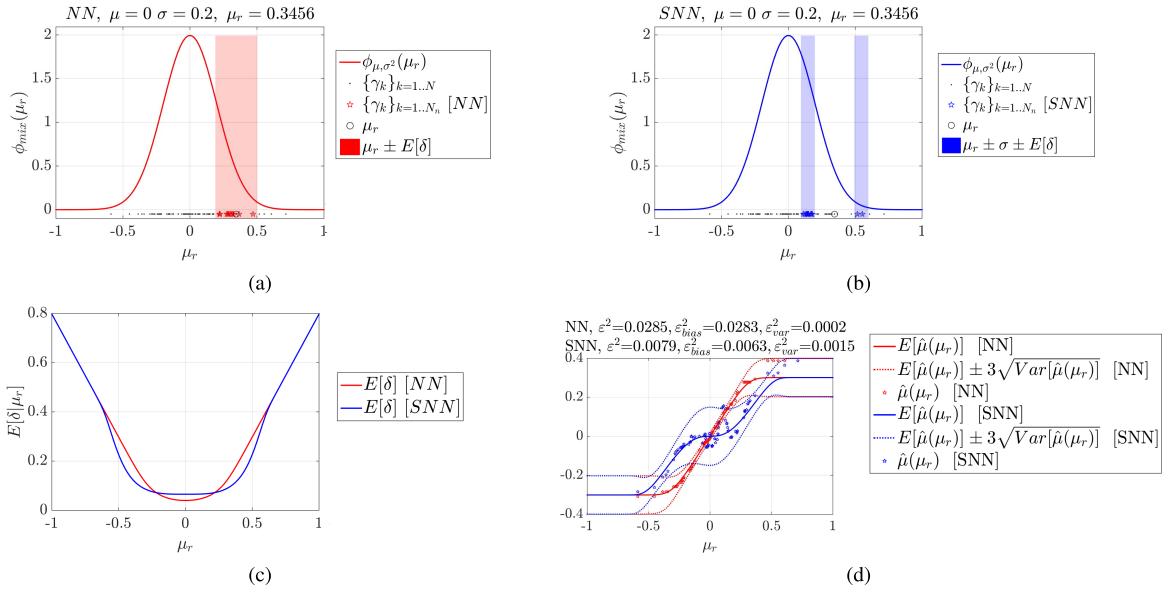


Fig. 2. Panels (a) and (b) show respectively the NN and SNN strategy to collect $N_n = 16$ neighbors $\{\gamma_k\}_{k=1..N_n}$ of $\mu_r = 0.3456$, for a 1×1 patch with noise-free value $\mu = 0$, corrupted by zero-mean Gaussian noise, $\sigma = 0.2$, from a set of $N = 100$ samples. No false matching ($p_f = 0$) are considered in this simulation. The 16 NNs of μ_r are in its immediate vicinity (red, in a); their sample mean is highly biased towards μ_r. The 16 SNNs (blue, in b) are on average closer to the actual μ, leading to a better estimate $\hat{\mu}(\mu_r)$. Panel (c) shows the expected value of the neighbor search interval, $\mathrm{E}[\delta|\mu_r]$. Panel (d) shows the expected value of the estimate, $\mathrm{E}[\hat{\mu}(\mu_r)]$, and its standard deviation, $\sqrt{\mathrm{Var}[\hat{\mu}(\mu_r)]}$, as a function of μ_r; in this panel, $\hat{\mu}(\mu_r)$ are specific estimates of μ obtained from the black samples in panels (a) and (b). SNN yields values closer to the actual $\mu = 0$, even when the noisy reference patch μ_r is far off the ground truth value μ.

and $\mathrm{Var}[\hat{\mu}(\mu_r)]$ (Eqs. (11-12)). To this aim, let's define $\delta = |\gamma_{N_n}(\mu_r) - \mu_r|$ as the distance of the N_n-th nearest neighbor from μ_r. Since each γ_k is independent from the others, and apart from a normalization factor ξ, the probability of measuring N_n neighbors in the $[\mu_r - \delta, \mu_r + \delta]$ interval is given by the product of three terms, representing respectively: the probability that $N_n - 1$ samples lie in the $[\mu_r - \delta,\ \mu_r + \delta]$ range, independently from their ordering; the probability that the N_n-th neighbor lies at distance δ from μ_r; and the probability that $N - N_n$ samples lie outside of the interval $[\mu_r - \delta, \mu_r + \delta]$, independently from their ordering. Thus, the pdf $p(\delta)$, describing the probability of finding N_n samples within distance δ from μ_r, is:

$$\begin{aligned} p_{in}(\delta) &= \Phi_{mix}(\mu_r + \delta) - \Phi_{mix}(\mu_r - \delta) \\ p_{bou}(\delta) &= \phi_{mix}(\mu_r - \delta) + \phi_{mix}(\mu_r + \delta) \\ p(\delta) &= \xi \cdot p_{in}(\delta)^{N_n - 1} p_{bou}(\delta)[1 - p_{in}(\delta)]^{N - N_n}. \end{aligned} \quad (16)$$

The normalization factor ξ is found by forcing $\int_0^{+\infty} p(\delta)d\delta = 1$, and resorting to numerical integration. From $p(\delta)$ in Eq. (16), we compute the expected value of the search interval, $\mathrm{E}[\delta|\mu_r]$ for each value of μ_r, as shown in Fig. 2c. The expected value $\mathrm{E}[\hat{\mu}(\mu_r)]$ is then computed marginalizing $\hat{\mu}$ over δ and resorting to numerical integration for the last passage. We have:

$$\begin{aligned} \mathrm{E}[\hat{\mu}(\mu_r)] &= \int_{-\infty}^{+\infty} \hat{\mu}\, p(\hat{\mu}) d\hat{\mu} \\ &= \int_{-\infty}^{+\infty} \hat{\mu} \int_0^{+\infty} \{p(\hat{\mu}|\delta) p(\delta) d\delta\} d\hat{\mu} \\ &= \int_0^{+\infty} \{\int_{-\infty}^{+\infty} \hat{\mu}\, p(\hat{\mu}|\delta) d\hat{\mu}\} p(\delta) d\delta \\ &\cong \sum_{\delta} [\int_{-\infty}^{+\infty} \hat{\mu}\, p(\hat{\mu}|\delta) d\hat{\mu} \cdot p(\delta)\Delta\delta] \\ &= \sum_{\delta} [\mathrm{E}[\hat{\mu}|\delta] \cdot p(\delta)\Delta\delta], \end{aligned} \quad (17)$$

where $\Delta\delta$ is a finite, small interval used for numerical integration. By the definition of $\hat{\mu}(\mu_r)$ in Eq. (10), we have then:

$$\mathrm{E}[\hat{\mu}(\mu_r)] = \frac{\Delta\delta}{N_n} \sum_{\delta} \sum_{k=1..N_n} \mathrm{E}[\gamma_k|\delta] \cdot p(\delta). \quad (18)$$

After marginalizing over δ, each of the N_n neighbors γ_k lies in the $[\mu_r - \delta, \mu_r + \delta]$ interval. Each of the first $N_n - 1$ neighbors is a mixture of a truncated Gaussian $G'(\mu, \sigma^2, \mu_r - \delta, \mu_r + \delta)$ with probability p_r and a second truncated Gaussian $G'(\mu_f, \sigma^2, \mu_r - \delta, \mu_r + \delta)$ with probability p_f, representing false matches. Eqs. (27–28) in the Appendix allow computing the expected value and variance of each of the two truncated Gaussians; $\mathrm{E}[\gamma_k|\delta]$ is then the expected value of their mixture, and can be computed as in Eq. (29). The expected value and variance of γ_{N_n} require a different computing procedure, as it lies exactly at distance δ from μ_r; γ_{N_n} can assume only two values, $\mu_r \pm \delta$, with probability $\phi_{mix}(\mu_r \pm \delta)/[\phi_{mix}(\mu_r - \delta) + \phi_{mix}(\mu_r + \delta)]$. Its expected value and variance are computed treating it as a Binomial random variable or, with the same result, as a mixture (see Eqs. (29-30) in the Appendix). The expected value of $\hat{\mu}(\mu_r)$ is then computed as in Eq. (18).

To compute the variance of the estimator, $\mathrm{Var}[\hat{\mu}(\mu_r)]$, we marginalize again over δ and get:

$$\begin{aligned}&\mathrm{Var}[\hat{\mu}(\mu_r)]\\&=\int_{-\infty}^{+\infty}(\hat{\mu}-\mathrm{E}[\hat{\mu}])^2 p(\hat{\mu})d\hat{\mu}\\&=\int_{-\infty}^{+\infty}(\hat{\mu}-\mathrm{E}[\hat{\mu}])^2\int_0^{+\infty}\{p(\hat{\mu}|\delta)p(\delta)d\delta\}d\hat{\mu}\\&=\int_0^{+\infty}\{\int_{-\infty}^{+\infty}(\hat{\mu}-\mathrm{E}[\hat{\mu}])^2 p(\hat{\mu}|\delta)d\hat{\mu}\}p(\delta)d\delta. \quad (19)\end{aligned}$$

By adding and subtracting $\mathrm{E}[\hat{\mu}|\delta]$, resorting again to numerical integration, and after some simplifications, we get:

$$\mathrm{Var}[\hat{\mu}(\mu_r)]\cong\Delta\delta\sum_{\delta}\sum_{k=1..N_n}\{\frac{\mathrm{Var}[\gamma_k|\delta]}{N_n^2}+(\mathrm{E}[\gamma_k|\delta]-\mathrm{E}[\hat{\mu}])^2\}\cdot p(\delta). \quad (20)$$

The expected value and variance of the first N_n-1 and of the N_n-th neighbors can be computed following the same reasoning line adopted to compute their expected value in the previous paragraph. From $\mathrm{E}[\hat{\mu}(\mu_r)]$ and $\mathrm{Var}[\hat{\mu}(\mu_r)]$, ε^2 is finally computed as in Eq. (13).

Fig. 2d shows $\mathrm{E}[\hat{\mu}(\mu_r)]\pm3\sqrt{\mathrm{Var}[\hat{\mu}(\mu_r)]}$ for the case $\mu=0$, $\sigma=0.2$, when $N_n=16$ neighbors are to be collected from a total of $N=100$ samples and no false matchings are present ($p_f=0$). The NN neighbors $\{\gamma_k\}_{k=1..N_n}$ lie close to the noisy reference patch, μ_r; consequently, each estimate $\hat{\mu}(\mu_r)$ is biased towards μ_r. The bias grows almost linearly with $\mu-\mu_r$ and it saturates at approximately 5σ of distance from μ, as the set of neighbors becomes stable (since N is finite, the same 16 samples are found in the tail of the distribution). The bias and variance error (Eqs. 11-12) are equal to $0.0283+0.0002=0.0285$ for the case in Fig. 2d. In practice, when NN neighbors are collected, samples with correlated noise are likely to be chosen. This is the *noise-to-noise* matching problem: averaging NNs patches may amplify small correlations between the noise, without canceling it [7], [8]. In the context of image denoising, the *noise-to-noise* matching problems shows up as residual, colored noise in the filtered image (see $\mathrm{NLM}_{0.0}^{16}$ in Fig. 1).

D. SNN

As an alternative to NN, inspired by Eq. (5), we propose collecting neighbors whose squared distance from the reference patch is instead close to its expectation. Thus SNNs are the patches $\{\boldsymbol{\gamma}_k\}_{k=1..N_n}$ that minimize:

$$|d^2(\boldsymbol{\mu_r},\boldsymbol{\gamma}_k)-o\cdot2\sigma^2|, \quad (21)$$

where we have introduced an additional offset parameter o, that allows to continuously move from the traditional NN approach ($o=0$) to the SNN approach ($o=1$). We assume $o=1$ for now.

E. Prediction Error of SNN

To compare the prediction error of NN and SNN in our toy problem, we need to compute $\mathrm{E}[\hat{\mu}(\mu_r)]$ and $\mathrm{Var}[\hat{\mu}(\mu_r)]$, when SNN neighbors of μ_r are collected. In our toy problem, we apply Fischer's approximation ($\sqrt{2\chi_P^2}\cong G(\sqrt{2P-1},\,1)$ to Eq. (4) for a single-pixel patch ($P=1$), and get:

$$d(\boldsymbol{\mu_r},\boldsymbol{\gamma})\cong\sigma\cdot\sqrt{2P-1}+G(0,\sigma^2)=\sigma+G(0,\sigma^2), \quad (22)$$

showing that the expected distance between two 1×1 noisy patches is approximately σ. SNN neighbors are then collected close to $\mu_r\pm o\cdot\sigma$. It can be noticed that, in our toy problem, we collect the samples based on their expected distance from the noisy reference μ_r, computed through Eq. (22), while in NLM filtering we use the squared distance (Eq. (21)). This simplification makes the toy problem mathematically manageable, without changing the intuition behind SNN.

Fig. 2b illustrates the sampling of SNN neighbors. Sampling potentially occurs on both sides of μ_r, with a different probability on each side. To compute $\mathrm{E}[\hat{\mu}(\mu_r)]$ and $\mathrm{Var}[\hat{\mu}(\mu_r)]$, we have to generalize Eq. (16) to the SNN case. To this aim, we define now $\frac{\delta}{2}$ as the distance of the N_n-th SNN from $\mu_r\pm o\cdot\sigma$ (therefore 2δ is both in the case of NN and SNN the total size of the neighbors' search interval) and its pdf, $p(\delta)$, is (23), as shown at the bottom of the next page.

Computing the expected value and the variance of $\hat{\mu}(\mu_r)$ for the SNN case turns out to be a generalization of the procedure described for the NN case. As a matter of fact, each of the first N_n-1 SNNs comes either from the left interval $[\mu_r-o\cdot\sigma-\frac{\delta}{2},\mu_r-o\cdot\sigma+\frac{\delta}{2}]$, with probability $\Phi_{mix}(\mu_r-o\cdot\sigma+\frac{\delta}{2})-\Phi_{mix}(\mu_r-o\cdot\sigma-\frac{\delta}{2})$, or from the right one, with probability $\Phi_{mix}(\mu_r+o\cdot\sigma+\frac{\delta}{2})-\Phi_{mix}(\mu_r+o\cdot\sigma-\frac{\delta}{2})$. Within each interval, the value of γ_k is distributed accordingly to a mixture of two truncated Gaussian variables, as in the NN case. The N_n-th neighbor can instead assume one among four values, $\mu_r\pm o\cdot\sigma\pm\delta$, with probabilities proportional to $\phi_{mix}(\mu_r\pm o\cdot\sigma\pm\delta)$. Similarly to the NN case, the mean and variance of $\hat{\mu}(\mu_r)$ are estimated as in Eq. (18) and (20), applying repeatedly Eqs. (27-30). From these, the bias and variance error of the estimator are obtained as in Eqs. (11-12). When $\delta>2o\cdot\sigma$, the two intervals merge into a single interval and the SNN set boils down to the NN set with the same value of δ; equations for the NN case are then applied to compute the error of the estimator. The Matlab code to replicate our toy problem is freely available at https://github.com/NVlabs/SNN.

Fig. 2d shows $\mathrm{E}[\hat{\mu}(\mu_r)]\pm3\sqrt{\mathrm{Var}[\hat{\mu}(\mu_r)]}$ for the SNN case. The bias and variance errors for SNN are respectively $\epsilon_{bias}^2=0.063$ and $\epsilon_{var}^2=0.0015$, for a total error $\varepsilon^2=0.0079$. Compared to NN, SNN slightly increases the variance of $\hat{\mu}(\mu_r)$, but it drastically decreases the bias of the estimate. In practice, the SNN criterion looks for similar patches on both sides of μ_r but not close to it; compared to NNs, SNNs are more likely to be collected close to μ (as for the case of the left interval in Fig. 2b), reducing the bias of $\hat{\mu}(\mu_r)$, at the price of a slightly higher variance. This minimizes the *noise-to-noise* matching issue and leads to a more effective noise cancellation. The Matlab code to replicate the analytical results on our toy problem is freely available at https://github.com/NVlabs/SNN.

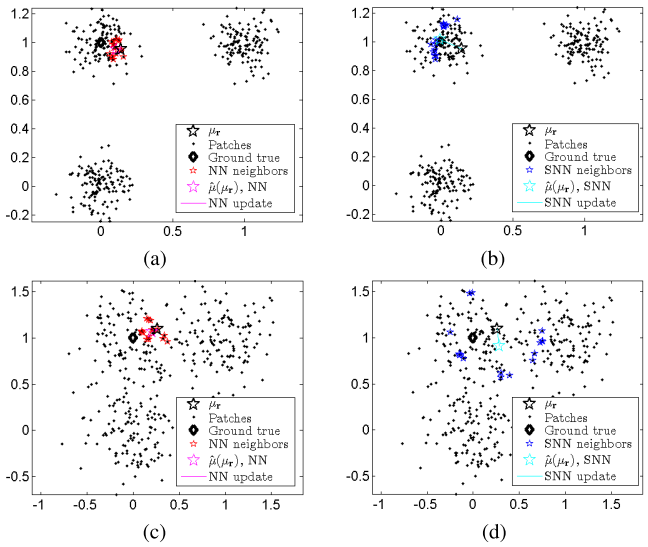


Fig. 3. Denoising with NLM and NN or SNN in 2D space. The proposed SNN strategy (b) works better than traditional NN (a) when neighbors are collected from a single cluster. When the signal-to-noise ratio is low, SNN can start collecting neighbors from different clusters (d) before traditional NN does (c). (a) NN, $\sigma = 0.1$. (b) SNN, $\sigma = 0.1$. (c) NN, $\sigma = 0.25$. (d) SNN, $\sigma = 0.25$.

IV. Neighbor Selection Analysis

In this section, we analyze and compare the prediction errors (Eq. (13)) of NN and SNN in different situations. Such analysis constitutes a proxy for the interpretation of the results on real data. We start with a graphical example showing the collection of NN and SNN neighbors in presence of noise and false matches: Fig. 3 shows the denoising of a patch with NLM in 2D space. In presence of a small amount of noise ($\sigma = 0.1$), three well-separated clusters can be identified, associated to three different kinds of patch in the search area. The average of the SNNs is generally closer to the center of the cluster (Fig. 3b), compared to NNs (Fig. 3a). This situation is representative of a flat area (all the patches belonging to the same cluster) or a very sharp edge (more clusters are present, but well separated). When the signal-to-noise ratio decreases ($\sigma = 0.25$), SNN has a higher chance to collect data from a different cluster (Fig. 3d), leading to a biased estimate of the cluster center. Thus, when the noise and signal power are comparable, SNN may fail to produce a reliable result.

For a more quantitative analysis, we perform NLM denoising of a 3×3 patch lying in an homogeneous area, close to an edge, or containing a line, texture, or an isolated detail (see Fig. 4). These can be seen as typical cases occurring during denoising of real images. Denoising is performed for the reference patch only (thus not considering the aggregation step in NLM), for different noise levels, σ, and numbers of neighbors, N_n, using NN or SNN to collect neighbors. Fig. 4 shows the ratio of the ε^2 of NN and SNN, averaged over 1000 simulations. SNN often achieves better denoising when few neighbors are employed and the noise level is low, but there are evident exceptions to this rule. In the next paragraphs, we explain these experimental results in detail resorting to our toy problem for their interpretation.

A. Homogeneous Areas

In the case of a patch in an homogeneous area (Fig. 4a), all the N neighbor patches come from the same population, thus $p_r = 1$ and the probability of collecting false matches is null ($p_f = 0$). In this situation, collecting neighbors with the SNN strategy always leads to a smaller error compared to the NN strategy, although ε^2 is comparable when very few ($N_n \to 1$) or all ($N_n \to N$) the neighbors are used. Fig. 5 explains this behavior through our toy problem. When few ($N_n = 2$) neighbors are used, the SNN approach has an higher variance (compared to NN) for $\mu_r \simeq \mu$, because SNN neighbors are collected at some distance from μ (see the peak of ϵ^2 for SNN around $\mu_r = \mu = 0$ in Fig. 5a, $N_n = 2$). Increasing N_n ($N_n = 16$, Fig. 5b) reduces the variance of $\hat{\mu}(\mu_r)$ thanks to the larger number of samples, making SNN preferable over NN. Further increasing N_n ($N_n = 64$, Fig. 5c) reduces the differences between the SNN and NN approaches, as the two sets of neighbors tend to coincide; in fact, NN and SNN generate the same result when $N_n = N$.

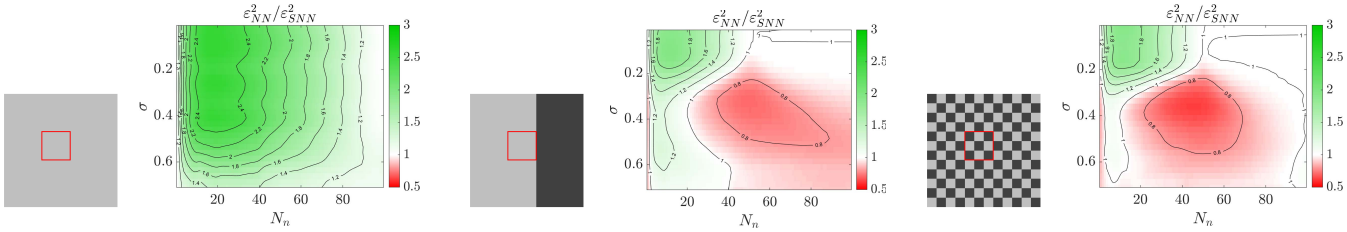
B. Bimodal and Texture Areas

For both the cases of a bimodal distribution or a texture area represented in Figs. 4b-4c, the percentage of potential false matches in the local search area is $p_f = 0.5$. This is obtained counting the patches different from the reference one in the image. In this case the SNN approach outperforms NN when the number of neighbors N_n is small, whereas NN works slightly better for large N_n and middle-high levels of noise. Fig. 6 explains this behavior through our toy problem. When the noise level is low ($\sigma = 0.07$, Fig. 6a), the reference patch and false matches are well separated, the chance of collecting false matches is small, and SNN outperforms NN as in the case of an homogeneous area. For high level of noise ($\sigma = 0.27$, Fig. 6c), SNN starts collecting false matches before NN, because the search interval is far from the reference value μ_r; this is prevented when few neighbors are used ($\sigma = 0.27$, $N_n = 8$ Fig. 6b), which reduces the size of the interval δ and therefore the chance to collect neighbors from the false matching population.

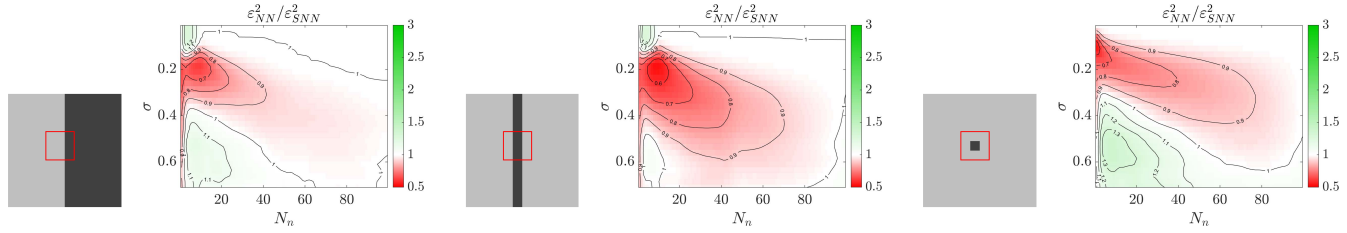
C. Edge and Line Areas

In the case of a reference patch containing an edge (Fig. 4d) or a line (Fig. 4e), the percentage of false matches

$$
\begin{aligned}
p_{in}(\delta) &= \Phi_{mix}(\mu_r - o\cdot\sigma + \frac{\delta}{2}) - \Phi_{mix}(\mu_r - o\cdot\sigma - \frac{\delta}{2}) + \Phi_{mix}(\mu_r + o\cdot\sigma + \frac{\delta}{2}) - \Phi_{mix}(\mu_r + o\cdot\sigma - \frac{\delta}{2}) \\
p_{bou}(\delta) &= \phi_{mix}(\mu_r - o\cdot\sigma - \frac{\delta}{2}) + \phi_{mix}(\mu_r - o\cdot\sigma + \frac{\delta}{2}) + \phi_{mix}(\mu_r + o\cdot\sigma - \frac{\delta}{2}) + \phi_{mix}(\mu_r + o\cdot\sigma + \frac{\delta}{2}) \\
p(\delta) &= \xi \cdot p_{in}(\delta)^{N_n - 1}\, p_{bou}(\delta)[1 - p_{in}(\delta)]^{N - N_n}.
\end{aligned}
\tag{23}
$$

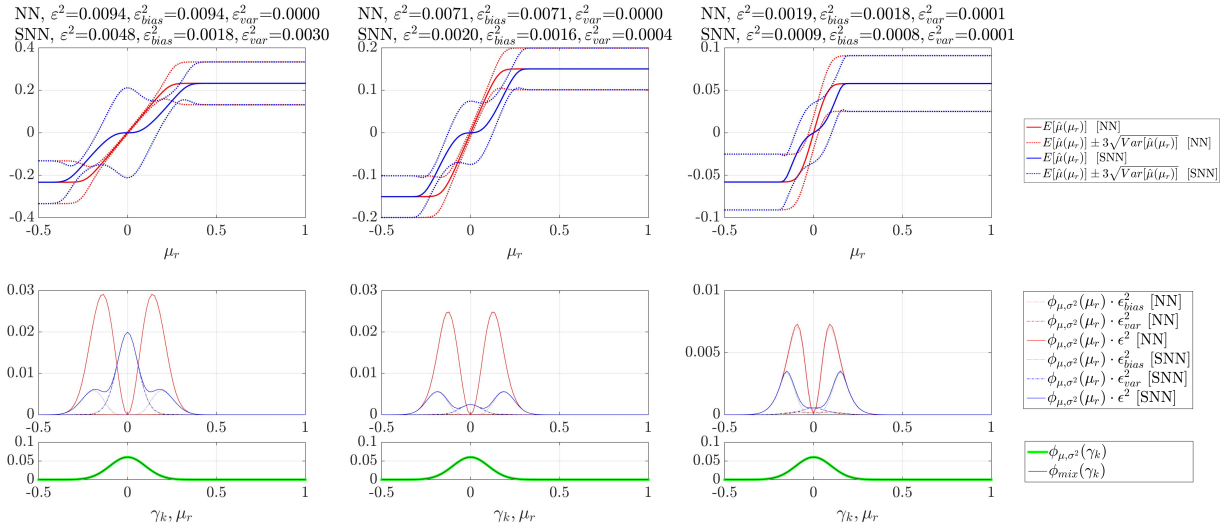


(a) Homogeneous, $p_r = 1, p_f = 0$ (b) Bimodal, $p_r = 0.5, p_f = 0.5$ (c) Texture, $p_r = 0.5, p_f = 0.5$



(d) Edge, $p_r = 0.1, p_f = 0.9$ (e) Line, $p_r = 0.1, p_f = 0.9$ (f) Isolated, $p_r = 0.01, p_f = 0.99$

Fig. 4. Each panel shows the ratio between the NN and SNN estimation errors (Eq. 13), averaged over 1000 simulations, when denoising a 3×3 reference patch μ_r (red rectangle in each panel) from a set of $N = 100$ local patches, as a function of the number of neighbors, N_n, and the noise standard deviation, σ. The bright and dark values in each image are equal to 0.75 and 0.25 respectively. The probability of finding a real (p_r) or false (p_f) matching patch within the given set is also indicated.



(a) $N_n = 2$ (b) $N_n = 16$ (c) $N_n = 64$

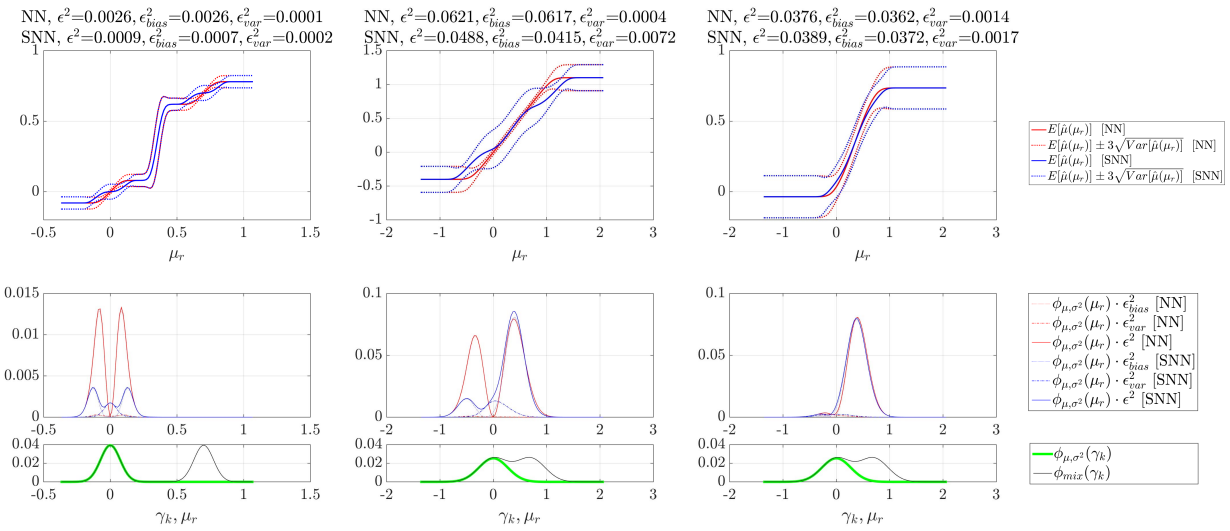
Fig. 5. For NN and SNN, the upper row shows $\mathrm{E}[\hat{\mu}(\mu_r)] \pm 3\sqrt{\mathrm{Var}[\hat{\mu}(\mu_r)]}$, the middle row show the error and bias terms as a function of μ_r, and the lower row the pdf of the neighbors γ_k, for $\mu = 0$, $\sigma = 0.1$, $p_f = 0$ and $N_n = \{2, 16, 64\}$. This case is representative of the homogeneous patch in Fig. 4. The total estimation error ε^2 with its bias and variance components (Eqs. (13-15)) is reported in the top part of each panel.

in the local search area is $p_f = 0.9$: both the SNN and NN approach generate biased results for $N_n > 10$, because of the substantial presence of false matches in the neighbor's set. For $N_n < 10$, SNN outperforms NN for low or high noise levels, whereas NN works better for middle levels of noise and large N_n.

Fig. 7 explains this behavior through our toy problem. When the noise is small and few neighbors are used ($\sigma = 0.03$, $N_n = 6$, Fig. 7a), NN and SNN capture similar sets of neighbors belonging to the reference distribution; SNN provides a slightly better estimate because of the smaller bias. As the noise level increases ($\sigma = 0.14$, $N_n = 12$, Fig. 7b) the chance to collect false matchings in the SNN set also increases, whereas false matches are less likely sampled by NN, whose search interval is closer to the reference value μ_r. As a consequence, the SNN estimate is more biased towards the false matching patches. When the noise level further increases ($\sigma = 0.44$, $N_n = 6$, Fig. 7c), also the NN strategy start collecting false matches in the neighbor set, and SNN provides again a better estimate because of the smaller bias.

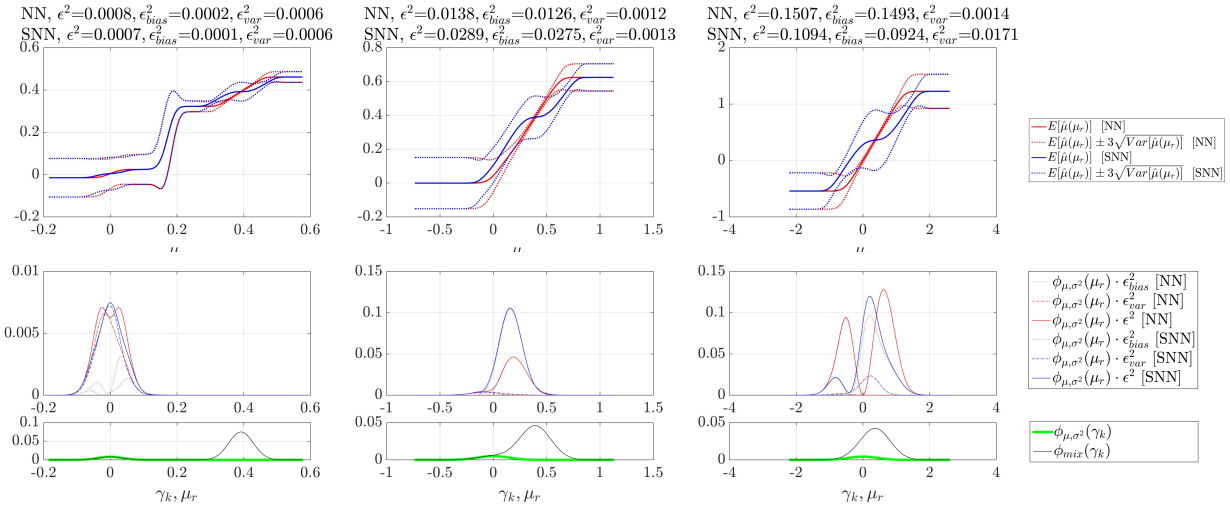
D. Isolated Detail Areas

The last case analyzed here is referred to a patch containing an isolated detail, with no matching patches but the reference patch in the local area (Fig. 4f, $p_f = 0.99$). Denoising through NLM is ill conditioned in this rare but realistic case, as the



(a) $N_n = 16, \sigma = 0.07$ (b) $N_n = 8, \sigma = 0.27$ (c) $N_n = 48, \sigma = 0.27$

Fig. 6. For NN and SNN, the upper row shows $\mathrm{E}[\hat{\mu}(\mu_r)] \pm 3\sqrt{\mathrm{Var}[\hat{\mu}(\mu_r)]}$, the middle row shows the error and bias terms as a function of μ_r, and the lower row the pdf of the neighbors γ_k, for $\mu = 0$, $\mu_f = 0.7$, $\sigma = \{0.07, 0.27\}$, $p_f = 0.5$ and $N_n = \{8, 16, 48\}$. This case is representative of the bimodal and texture patches in Fig. 4. The total estimation error ε^2 with its bias and variance components (Eqs. (13-15)) is reported in the top part of each panel.



(a) $N_n = 6, \sigma = 0.03$ (b) $N_n = 12, \sigma = 0.14$ (c) $N_n = 6, \sigma = 0.44$

Fig. 7. For NN and SNN, the upper row shows $\mathrm{E}[\hat{\mu}(\mu_r)] \pm 3\sqrt{\mathrm{Var}[\hat{\mu}(\mu_r)]}$, the middle row shows the error and bias terms as a function of μ_r, and the lower row the pdf of the neighbors γ_k, for $\mu = 0$, $\mu_f = 0.39$, $\sigma = \{0.03, 0.14, 0.44\}$, $p_f = 0.9$ and $N_n = \{6, 12\}$. This case is representative of patches containing an edge or a line in Fig. 4. The total estimation error ε^2 with its bias and variance components (Eqs. (13-15)) is reported in the top part of each panel.

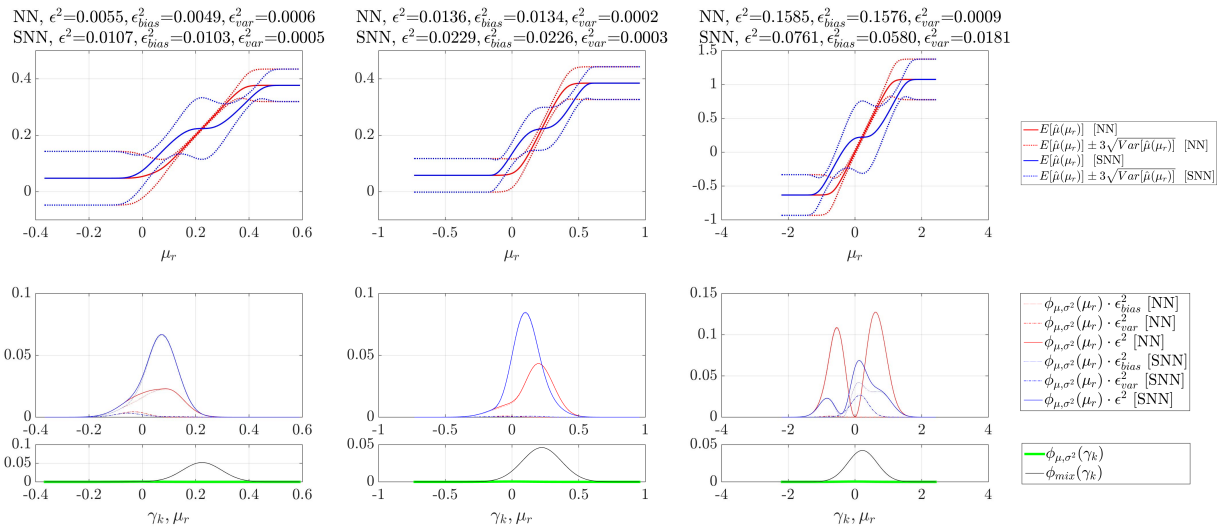
basic assumption of self-similarity is broken. Nonetheless, the output of denoising for NN and SNN is different even in this case, with NN providing a better result for low noise levels, and SNN outperforming NN in the case of high noise.

Fig. 8 explains this behavior through our toy problem. When few neighbors are considered and noise is low ($\sigma = 0.03$, $N_n = 4$, in Fig. 8a), SNN tends to collect samples from the false matching distribution, far from μ_r and providing a biased estimate. The situation does not change significantly while increasing the number of neighbors ($N_n = 32$, $\sigma = 0.14$, Fig. 8b), although SNN and NN tends to converge towards the same estimate $\hat{\mu}(\mu_r)$ in this case. On the other hand, in case of high noise, the bias term prevails for NN and SNN provides a better estimate ($N_n = 6$, $\sigma = 0.44$, Fig. 8c).

V. Result

A. White Gaussian Noise

We test the effectiveness of the SNN schema on the Kodak image dataset [17]. We first consider denoising in case of white, zero-mean, Gaussian noise with standard deviation $\sigma = \{5, 10, 20, 30, 40\}$, and evaluate the effect of the offset parameter o and number of neighbors N_n. With $\mathrm{NLM}_o^{N_n}$ we indicate NLM with N_n neighbors and an offset o. Each image is processed with NLM_o^{16} and for o ranging from $o = 0$



(a) $N_n = 4, \sigma = 0\ 03$ (b) $N_n = 32, \sigma = 0\ 14$ (c) $N_n = 6, \sigma = 0\ 44$

Fig. 8. For NN and SNN, the upper row shows $\mathrm{E}[\hat{\mu}(\mu_r)] \pm 3\sqrt{\mathrm{Var}[\hat{\mu}(\mu_r)]}$, the middle row shows the error and bias terms as a function of μ_r, and the lower row the pdf of the neighbors γ_k, for $\mu = 0$, $\mu_f = 0.22$, $\sigma = \{0.03, 0.14, 0.44\}$, $p_f = 0.99$ and $N_n = \{4, 6, 32\}$. This case is representative of patches containing an isolated detail in Fig. 4. The total estimation error ε^2 with its bias and variance components (Eqs. (13-15)) is reported in the top part of each panel.

(*i.e.*, the traditional NN strategy) to $o = 1$; $\mathrm{NLM}_{.0}^{361}$ and $\mathrm{NLM}_{.0}^{900}$ indicate the original NLM, using all the patches in the search window, respectively for $\sigma < 30$ and $\sigma \geq 30$. The patch size, the number of neighbors N_n, and the filtering parameter h in Eq. (2) are the optimal ones suggested in [1].

We resort to a large set of image quality metrics to evaluate the filtered images: PSNR, SSIM [18], MSSSIM [19], GMSD [20], FSIM and FSIM_C [21]. PSNR changes monotonically with the average squared error; SSIM and MSSSIM take inspiration from the human visual system, giving less importance to the residual noise close to edges. GMSD is a perceptually-inspired metric that compares the gradients of the filtered and reference images. FSIM is the one that correlates better with the human judgment of the image quality, and takes into consideration the structural features of the images; FSIM_C also takes into account the color component.

Table I shows the average image quality metrics measured on the Kodak dataset, whereas visual inspection of Fig. 1 reveals the correlation between these metrics and the artifacts introduced in the filtered images. The PSNR decreases when reducing the number of NN neighbors (see the first two rows for each noise level in Table I), mostly because of the residual, colored noise left by $\mathrm{NLM}_{.0}^{16}$ in the flat areas (like the skin of the girl and the wall with the fresco in Fig. 1); the presence of this colored noise is explained by the bias towards the noisy reference patch (*noise-to-noise matching*) observed for NN when few neighbors are used (see also Fig. 4a)); SSIM and MSSSIM show a similar trend, while GMSD, FSIM, and FSIM_C improve when the number of NN neighbors is reduced to 16. The improvement of these three perceptually-based image quality metrics is associated with the reduction of the NLM bias (towards false matching patches) in the case of a small set of neighbors, clearly explained in [5]. At visual inspection, edges and structures are in fact better preserved, (see the textile pattern and the sun rays in Fig. 1).

Table I shows the PSNR consistently increasing with the offset o, coherently with the decrease of the prediction error measured in our toy problem from $o = 0.0$ (NN) to $o = 1.0$ (SNN). As expected, SNN removes more noise in the flat areas (*e.g.*, the girl's skin, the wall surface in Fig. 1). For $o = 1.0$ the PSNR approaches that achieved by traditional NLM, which however requires more neighbor patches (and therefore a higher computational time), and achieves lower perceptual metric scores. The fact that SSIM, MSSIM, GMSD, FSIM and FSIM_C are maximized in the $o = [0.65, 0.9]$ range suggests that, even if none of these metrics is capable of perfectly measuring the quality perceived by a human observer, filtering with the SNN approach generates images of superior, perceptual quality compared to the traditional NLM filtering. The optimal image quality is measured for $o < 1$ because SNN can blur textures or low-contrast edges more than NN, as observed in Figs. 4b-4e. This can be visually appreciated in the texture of the textile and on the sun rays and the small details in the fresco in Fig. 1, for $o = 1$. SNN with an offset $o = 0.8$ ($\mathrm{NLM}_{0.8}^{361}$) achieves the best compromise between preserving low-contrasted details in the image and effectively smoothing the flat areas. The proposed SNN strategy always outperforms NN (when using the same number of patches) in the image quality metrics, with the exception of very low noise ($\sigma = 5$), where GMSD, FSIM and FSIM_C are constant for $.0 < o < 0.65$. Quite reasonably, the advantage of SNN over NN is larger for a large σ: in fact, for $\sigma \to 0$, NN and SNN converge to the same algorithm.

The first row of Fig. 9 shows the average PSNR and FSIM_C achieved by NLM using NN ($o = 0$) and SNN ($o = 0.8$) on the Kodak dataset, as a function of the number of neighbors N_n. Compared to NN, the SNN approach achieves an higher PSNR

TABLE I

AVERAGE IMAGE QUALITY METRICS FOR DENOISING THE KODAK IMAGE DATASET, CORRUPTED BY ZERO-MEAN GAUSSIAN NOISE, DIFFERENT STANDARD DEVIATION σ AND DIFFERENT VALUES OF THE OFFSET PARAMETER o. BOLD NUMBERS INDICATE THE BEST RESULTS

	PSNR	SSIM	MSSSIM	GMSD	FSIM	FSIM$_C$	σ
noisy	34.15	.9625	.9797	.0211	.9937	.9931	
NLM$^{361}_{.0}$	**38.34**	.9851	.9921	.0130	.9941	.9939	
NLM$^{16}_{.0}$	38.08	.9847	.9919	**.0113**	**.9949**	**.9947**	
NLM$^{16}_{.1}$	38.08	.9847	.9919	**.0113**	**.9949**	**.9947**	
NLM$^{16}_{.35}$	38.08	.9847	.9919	**.0113**	**.9949**	**.9947**	
NLM$^{16}_{.65}$	38.14	.9850	.9920	**.0113**	**.9949**	**.9947**	5
NLM$^{16}_{.8}$	38.23	**.9854**	.9922	.0115	.9948	.9946	
NLM$^{16}_{.9}$	38.28	**.9854**	**.9923**	.0118	.9946	.9945	
NLM$^{16}_{1.0}$	38.29	.9852	.9921	.0124	.9944	.9942	
noisy	28.13	.8744	.9332	.0667	.9763	.9743	
NLM$^{361}_{.0}$	**34.84**	.9634	.9804	.0338	.9814	.9811	
NLM$^{16}_{.0}$	33.96	.9580	.9773	.0269	**.9861**	.9856	
NLM$^{16}_{.1}$	33.96	.9580	.9773	.0269	**.9861**	.9856	
NLM$^{16}_{.35}$	33.96	.9580	.9773	.0269	**.9861**	.9856	
NLM$^{16}_{.65}$	34.19	.9610	.9789	**.0266**	**.9861**	**.9857**	10
NLM$^{16}_{.8}$	34.51	.9639	.9805	.0274	.9856	.9852	
NLM$^{16}_{.9}$	34.66	**.9645**	**.9809**	.0291	.9845	.9841	
NLM$^{16}_{1.0}$	34.71	.9635	.9803	.0319	.9828	.9824	
noisy	22.11	.6810	.8236	.1549	.9250	.9174	
NLM$^{361}_{.0}$	**31.18**	.9109	.9505	.0749	.9475	.9468	
NLM$^{16}_{.0}$	29.21	.8802	.9343	.0607	.9646	.9633	
NLM$^{16}_{.1}$	29.21	.8802	.9343	.0607	.9646	.9633	
NLM$^{16}_{.35}$	29.21	.8802	.9343	.0607	.9646	.9633	
NLM$^{16}_{.65}$	29.75	.8948	.9420	**.0572**	**.9653**	**.9641**	20
NLM$^{16}_{.8}$	30.45	.9086	.9493	.0583	.9631	.9621	
NLM$^{16}_{.9}$	30.81	**.9119**	**.9511**	.0635	.9585	.9576	
NLM$^{16}_{1.0}$	30.93	.9084	.9490	.0719	.9510	.9503	
noisy	18.59	.5331	.7243	.2138	.8687	.8536	
NLM$^{900}_{.0}$	**29.11**	.8631	.9197	.1071	.9154	.9144	
NLM$^{16}_{.0}$	27.40	.8273	.9035	.0827	.9438	.9419	
NLM$^{16}_{.1}$	27.40	.8273	.9035	.0827	.9438	.9419	
NLM$^{16}_{.35}$	27.40	.8273	.9035	.0827	.9438	.9419	
NLM$^{16}_{.65}$	27.48	.8311	.9055	.0817	**.9441**	**.9422**	30
NLM$^{16}_{.8}$	28.15	.8543	.9175	**.0793**	.9439	**.9422**	
NLM$^{16}_{.9}$	28.69	**.8665**	**.9233**	.0850	.9382	.9368	
NLM$^{16}_{1.0}$	28.91	.8640	.9201	.0988	.9247	.9235	
noisy	16.09	.4280	.6414	.2501	.8157	.7937	
NLM$^{900}_{0.0}$	**27.71**	.8184	.8893	.1303	.8856	.8844	
NLM$^{16}_{0.0}$	25.38	.7522	.8591	.1061	.9237	.9206	
NLM$^{16}_{0.1}$	25.38	.7522	.8591	.1061	.9237	.9206	
NLM$^{16}_{0.2}$	25.38	.7522	.8591	.1061	.9237	.9206	
NLM$^{16}_{0.35}$	25.38	.7522	.8591	.1061	.9237	.9206	40
NLM$^{16}_{0.5}$	25.38	.7522	.8591	.1061	.9237	.9206	
NLM$^{16}_{0.65}$	25.49	.7580	.8624	.1043	.9243	.9213	
NLM$^{16}_{0.8}$	26.38	.7942	.8813	**.0977**	**.9248**	**.9223**	
NLM$^{16}_{0.9}$	27.11	**.8153**	**.8910**	.1037	.9168	.9148	
NLM$^{16}_{1.0}$	27.42	.8147	.8870	.1214	.8971	.8954	

for any value of N_n; SNN also achieves a better FSIM$_C$ for small values of N_n, coherently with our toy problem and simulations (Fig. 4), showing a better preservation of fine details for SNN in this case. The state-of-the-art denoising filter BM3D [2] produces images of superior quality, but at a much higher computational cost. We also compared the image quality achieved by our approach with the Sliding Discrete Cosine Transform filter (SDCT [22]), a denoising algorithm of comparable computational complexity based on signal sparsification and not using a NL approach, and achieving a comparable PSNR, but lower FSIM$_C$.

The Matlab code to filter an image with NLM, SNN, and different offset values is available at https://github.com/NVlabs/SNN.

B. Colored Noise

Although the case of white Gaussian noise is widely studied, it represents an ideal situation which rarely occurs in practice. Having in mind the practical application of denoising, we study the more general case of colored noise. Images are in fact generally acquired using a Bayer sensor and demosaicing introduces correlations between nearby pixels in the RGB domain. A direct consequence is that the NN approach is even more likely to collect false matches, amplifying low-frequency noise as well as demosaicing artifacts.

To study this case, we mosaic the images of the Kodak dataset, add noise in the Bayer domain, and subsequently demosaic them through the widely used Malvar algorithm [24]. Denoising is then performed through $NLM^{N_n}_{.0}$, $NLM^{N_n}_{.8}$ (for $N_n = \{4, 8, 16, 32\}$), BM3D [2], and SDCT [22]. All these filters require an estimate of the noise standard deviation in the RGB domain, obtained here using Eq. (32) and considering that RGB data come from a linear combination (see [24]) of independent noisy samples in the Bayer domain. To compare with the state-of-the-art, we also filter the images with BM3D-CFA [23], a modified version of BM3D specifically tailored to denoise data in the Bayer domain.

The bottom row of Fig. 9 shows the average PSNR and FSIM$_C$ on the Kodak dataset. The SNN approach is always more effective than NN in removing colored noise, both in terms of PSNR and FSIM$_C$. This is related to the higher occurrence of *noise-to-noise* matching for NN in the case of colored noise, which is also evident at visual inspection (Fig. 10). NLM with SNN also generally outperforms SDCT and BM3D in terms of PSNR and FSIM$_C$. Since BM3D adopts a NN selection strategy, it also suffers from the *noise-to-noise* matching problem in this case. Remarkably, for $\sigma \leq 20$, NLM coupled with SNN is better, in terms of PSNR, than the state-of-the-art (and computationally more intensive) BM3D-CFA, and only slightly inferior for higher noise levels. Visually, the result produced by NLM with SNN is comparable with that of BM3D-CFA: our approach generates a slightly more noisy image, but with better preserved fine details (*e.g.*, see the eyelids in Fig. 10) and without introducing grid artifacts. Further results can be observed in the Supplementary Material.

C. The Case of a "Real" Image

We finally analyze the peculiarities of NLM denoising with SNN for "real" images. We test different filters (SDCT, NLM$^{16}_{.0}$, NLM$^{16}_{.8}$, and BM3D-CFA) on a high resolution (2592×1944) image, captured at ISO 1200 with an NVIDIA Shield tablet. Notice that image resolution is much higher in this case, compared to the Kodak dataset. Futhermore, the noise distribution in the Bayer domain is far from the ideal, Gaussian distribution with constant variance. Therefore, we compute the sensor noise model and the corresponding

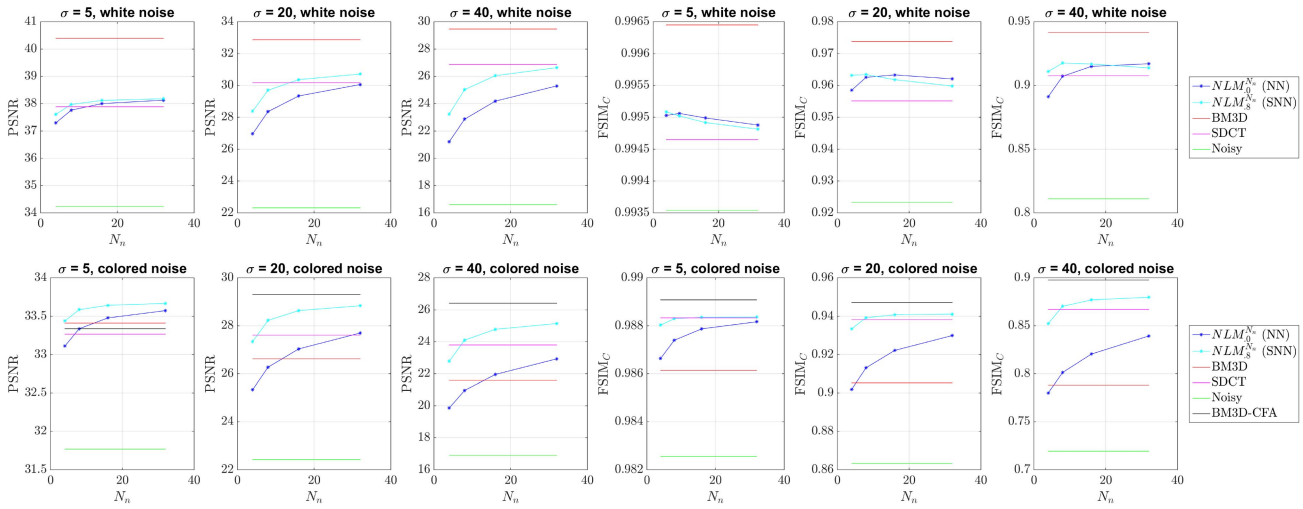


Fig. 9. Average PSNR and FSIM$_C$ measured on the Kodak dataset, as a function of the number of neighbors N_n, for NLM using the NN ($NLM_{.0}^{N_n}$) and SNN ($NLM_{.8}^{N_n}$) approaches, compared to SDCT [22], BM3D [2], and BM3D-CFA [23] in the case of additive white Gaussian noise (first row) and colored noise (bottom row). Notice that the number of patches used by BM3D and BM3D-CFA is constant and coherent with the optimal implementations described in [2] and [23].

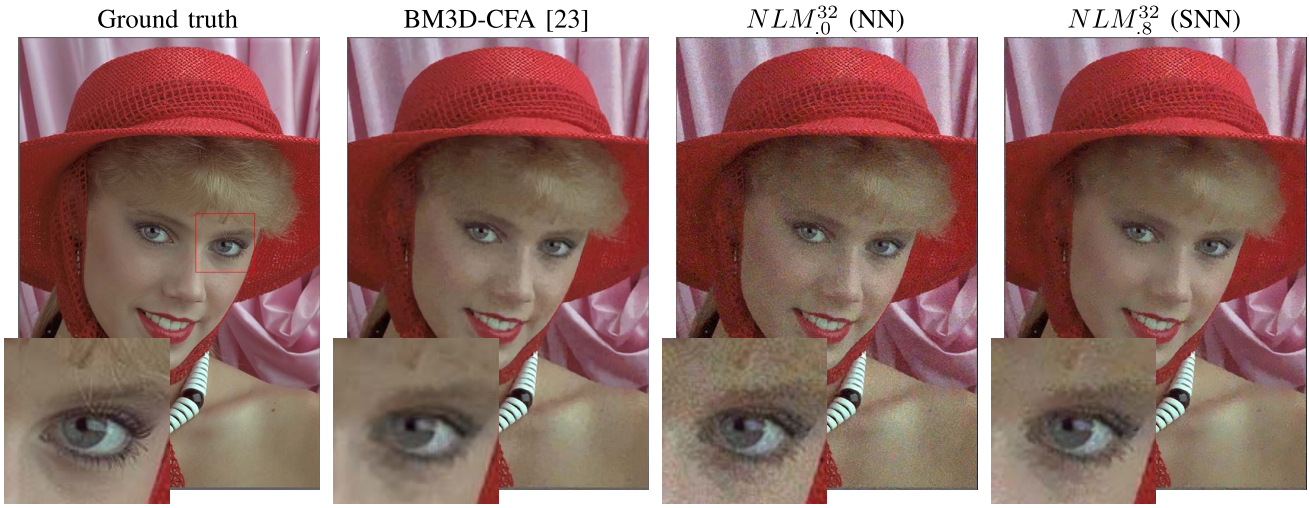


Fig. 10. Comparison of different algorithms for colored noise denoising, $\sigma = 20$. The state-of-the-art BM3D-CFA [23] achieves the highest image quality metrics (Fig. 9); a grid artifact is however visible in the white part of the eye. The traditional NLM algorithm, using $N_n = 32$ NNs ($NLM_{.0}^{32}$), does not eliminate the colored noise introduced by demosaicing. Instead, when SNNs neighbors are used ($NLM_{.8}^{32}$), the colored noise is efficiently removed and quality at visual inspection is comparable to that of BM3D-CFA. Our result appears less "splotchy" and sharper. Better seen at 400% zoom.

Variance Stabilizing Transform (VST) as in [25] and apply the VST in the Bayer domain such that the noise distribution approximately resembles the ideal one. We filter the image through the SDCT, NLM$_{.0}^{16}$, and NLM$_{.8}^{16}$ denoising algorithms in the RGB domain, after applying the VST and Malvar demosaicing, which introduces correlations among pixels. The inverse VST is then applied to the data in the RGB domain. Differently from the other algorithms, BM3D-CFA is applied directly to raw data, after the VST and before the inverse VST and demosaicing. This represents an advantage of BM3D-CFA over the other approaches, that have been designed to deal with white noise, but are applied in this case to colored noise. Finally, after the denoising step, we apply color correction, white balancing, gamma correction, unsharp masking, coherently with the typical workflow of an Image Signal Processor; these steps can easily increase the visibility of artifacts and residual noise. A stack of 650 frames is averaged to build a ground truth image of the same scene.

The results are shown in Fig. 11. Visual inspection confirms the superiority of NLM$_{.8}^{16}$ over SDCT and NLM$_{.0}^{16}$ even for the case of a real image. The NLM$_{.0}^{16}$ filter preserves high frequency details in the image, but it also generates high frequency colored noise in the flat areas, because of the *noise-to-noise* matching problem. The SDCT filter produces a more blurry image with middle frequency colored noise in the flat areas. Our approach provides again an image quality comparable to BM3D-CFA; the NLM$_{.8}^{16}$ filter is slightly more noisy than BM3D-CFA, as measured by the image quality metrics in Table II, but filtering is computationally less intensive.

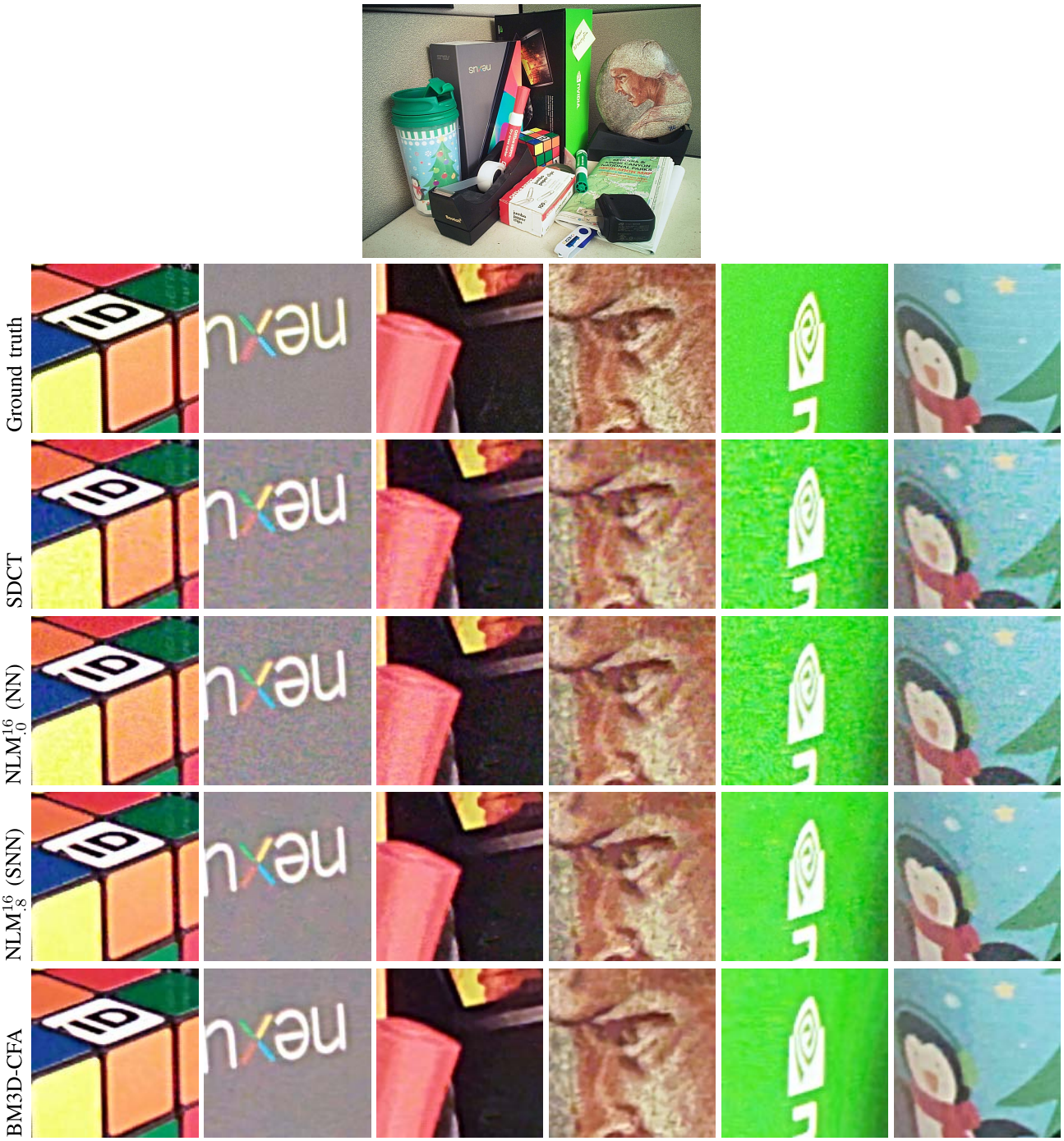


Fig. 11. Comparison of several patches from the image on the top, acquired with an NVIDIA Shield Tablet at ISO 1200, demosaiced and then denoised with different algorithms. Color correction, white balancing, gamma correction and unsharp masking are applied after denoising. The ground truth image is obtained by averaging 650 frames. Better seen at 400% zoom.

D. Other Applications of SNN

The proposed neighbors' selection principle is indeed quite general and it can be easily extended to other applications beyond NL algorithms. For instance, once properly reformulated, it can be beneficial also for the case of the bilateral filter [26], which is described by:

$$\hat{\mu}^{i,j} = \sum_{\delta i,\delta j=-n}^{n} s^{\delta i,\delta j} \cdot r[\mu_r^{i,j}, \gamma^{\,i+\delta i,j+\delta j}] \cdot \gamma^{\,i+\delta i,j+\delta j} \tag{24}$$

where the filter size is $(2n+1)\times(2n+1)$, $\hat{\mu}^{i,j}$ is the (i,j) pixel of the filtered image, $\gamma^{\,i,j}$ is a pixel in the noisy image, $\mu_r^{i,j} = \gamma^{\,i,j}$, $s^{\delta i,\delta j}$ is a spatial weight, and the range weight is (apart from a normalization constant):

$$r[\mu_r^{i,j}, \gamma^{\,h,k}] = \exp\{-0.5 \cdot [(\mu_r^{i,j} - \gamma^{\,h,k})/\sigma_r]^2\}. \tag{25}$$

Thus, the range weight for the pixel $\gamma^{\,h,k}$ is proportional to its distance from $\mu_r^{i,j}$. Assuming the image is corrupted by zero-mean Gaussian noise with variance σ^2, our approach

TABLE II
IMAGE QUALITY METRICS FOR THE "REAL" IMAGE IN FIG. 11, ACQUIRED WITH AN NVIDIA SHIELD TABLET AT ISO 1200 AND DENOISED WITH DIFFERENT ALGORITHMS

	Noisy	SDCT	$\text{NLM}_{.0}^{16}$ (NN)	$\text{NLM}_{.8}^{16}$ (SNN)	BM3D-CFA
PSNR	21.2689	23.3715	24.0972	24.5500	25.2580
SSIM	0.8741	0.8889	0.9135	0.9261	0.9607
MSSSIM	0.7133	0.7773	0.8124	0.8450	0.8943
GMSD	0.1921	0.1499	0.1356	0.1145	0.0921
FSIM	0.9920	0.9927	0.9933	0.9940	0.9952
FSIM_C	0.9890	0.9899	0.9911	0.9921	0.9941

instead is to give a larger weight to those pixels that differ in value $\pm o \cdot \sigma$ from $\mu_r^{i,j}$, as:

$$r[\mu_r^{i,j}, \gamma^{h,k}] = \exp[(-0.5 \cdot \frac{|\mu_r^{i,j} - \gamma^{h,k}| - o \cdot \sigma}{\sigma_r})^2]. \quad (26)$$

The Matlab code of the modified bilateral filter is available at https://github.com/NVlabs/SNN. We applied this filter to the Kodak dataset (gray scale), corrupted by Gaussian noise ($\sigma = \{5, 10\}$)[1] and setting $\sigma_r = \sigma$. The average image quality metrics (Table III) show also in this case a significant improvement, compared to the traditional formulation of the bilateral filter. Fig. 12 shows that, also for the case of bilateral filtering and similarly to the case of NLM, our approach leads to a better smoothing of the flat areas, while edges are similarly well preserved.

Potentially, the SNN approach can be applied to any other NL denoising algorithm, like BM3D [2]. We did an attempt in this direction, but measured only a slight, statistically non-significant improvement in image quality. Our interpretation for this is the following. BM3D is a two steps procedure: filtering is performed a first time on the raw, noisy image; the quality of the oracle image generated in this way actually improves with the application of SNN, but in the second phase, the search for neighbors is repeated on the oracle image, whose noise level is low. In these conditions, the advantage of SNN over NN is questionable: the reference patch $\boldsymbol{\mu_r}$ is often close to the ground truth $\boldsymbol{\mu}$ and in this case the SNN reconstruction error can be higher (compared to NN) because of the higher variance. This is coherent with the prediction of our toy problem, for instance in Fig. 4a where the error curve of SNN is higher than the error curve of NN for $\mu_r \simeq \mu$.

TABLE III
IMAGE QUALITY METRICS FOR DENOISING THE GRAYLEVEL KODAK DATASET, CORRUPTED BY ZERO-MEAN GAUSSIAN NOISE, $\sigma = \{5; 10\}$, USING A BILATERAL FILTER AND DIFFERENT OFFSETS o FOR THE RANGE WEIGHTS. THE CASE $o = 0$ CORRESPONDS TO THE TRADITIONAL BILATERAL FILTER

	$\sigma = 5$				
	PSNR	SSIM	MSSSIM	GMSD	FSIM
Noisy	34.15	0.8487	0.9775	0.0222	0.9447
o = 0	36.22	0.9142	0.9866	0.0142	0.9660
o = 0.1	36.34	0.9174	0.9870	0.0141	0.9667
o = 0.35	36.63	0.9252	0.9878	**0.0139**	0.9681
o = 0.65	36.93	0.9334	0.9885	0.0143	**0.9683**
o = 0.8	37.03	0.9366	**0.9886**	0.0149	0.9676
o = 0.9	37.08	0.9381	**0.9886**	0.0154	0.9666
o = 1.0	**37.09**	**0.9392**	0.9885	0.0161	0.9652
	$\sigma = 10$				
	PSNR	SSIM	MSSSIM	GMSD	FSIM
Noisy	28.13	0.6267	0.9263	0.0702	0.8497
o = 0	31.13	0.7735	0.9581	0.0418	0.9114
o = 0.1	31.32	0.7827	0.9596	0.0407	0.9142
o = 0.35	31.80	0.8067	0.9631	0.0385	0.9207
o = 0.65	32.37	0.8351	0.9666	**0.0372**	0.9259
o = 0.8	32.62	0.8477	0.9678	0.0373	**0.9267**
o = 0.9	32.75	0.8550	0.9684	0.0378	0.9262
o = 1.0	**32.86**	**0.8610**	**0.9687**	0.0387	0.9248

VI. DISCUSSION

The NLM algorithm can be analyzed as a bias / variance dilemma. Duval *et al.* [5] found that decreasing the number of neighbors reduces the bias towards false matches in $\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}$, which affects NLM even in absence of noise. Their analysis, as well as that of other researchers ([8], [10]), is mostly centered around the role of the weights in Eq. (2). Instead, we concentrate our attention on the selection of the neighbors: since the NNs of any noisy reference patch $\boldsymbol{\mu_r}$ lie close to it, their average is biased towards $\boldsymbol{\mu_r}$, independently from the weights (*noise-to-noise* matching). The final effect is the creation of splotchy, colored artifacts in the filtered images, that can be mitigated resorting to the SNN sampling strategy.

We propose using a toy problem to interpret and provide insights on the differences between NN and SNN. Compared to the original implementation of NLM, the math in our toy is slightly simplified, as we neglect the weights assigned to the neighbors (Eq. (10)). This simplification allows studying the effect of the neighbors' sampling procedure independently from the weights, but it also has potential limitations. It is in fact well representative of an homogeneous area corrupted by a limited amount of noise, where all neighbors belong to the same cluster and their distance from the reference patch is close to the expected one (Eq. (5)). When more complex structures are present in the search window, or for high noise levels, neighbors at a squared distance significantly larger than $2\sigma^2$ can be collected. In this case the weights in Eq. (2) cannot be neglected, and the interpretation of the results obtained with our toy problem has to be made with a grain of salt. Even with this in mind, our results show that the empirical advantage of NN over SNN can be explained in a principled way through our toy problem. In fact, analytical results in our toy problem (Fig. 2d) and experimental results on the Kodak dataset (Table I) are in agreement, suggesting that the proposed SNN approach mitigates the *noise-to-noise* matching problem, reduces the bias towards $\boldsymbol{\mu_r}$, and produces images

[1]Although we registered an improvement in the image quality metrics for $\sigma > 10$, we did not report these results here because of the inadequacy of the bilateral filter for such large noise levels.

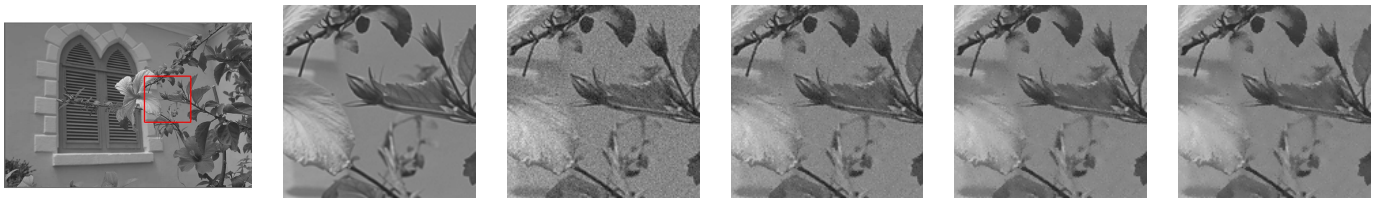


Fig. 12. Filtering of an image of the Kodak dataset, corrupted by zero-mean, Gaussian noise, $\sigma = 10$, with the traditional bilateral filter ($o = 0$) [26] and with the proposed strategy ($o = 0.8$ and $o = 1.0$). Flat areas are denoised better with the proposed approach, while edges are preserved well.

of quality similar to the original NLM in the flat areas (high PSNR in Table I), while keeping at the same time the visibility of the small details (low GMSD, high FSIM in Table I).

The advantage of SNN is more evident in the case of colored noise, when *noise-to-noise* matching is more likely to occur. Visual inspection reveals that in this case NLM with SNN achieves an image quality comparable to the state-of-the-art BM3D-CFA [23], but at a lower computational cost (see Fig. 10 and other examples in the Supplementary Material). Our experiments (Fig. 11 and Table II) suggest that NLM with SNN is close to BM3D-CFA also in the case of "real" images, where white balance, color correction, gamma correction and edge enhancement are applied after denoising, and can significantly enhance the visibility of artifacts and residual noise.

The role played by the reference patch $\boldsymbol{\mu_r}$ deserves a mention here. If neighbors are collected through the NN strategy, $\boldsymbol{\mu_r}$ always belongs to set of neighbors, but this is not the case for SNN. Our experiments, as well as the theoretical analysis derived for our toy problem, have been performed without assigning any special role to $\boldsymbol{\mu_r}$. An alternative to this is to add $\boldsymbol{\mu_r}$ by default to the set of neighbors, increasing the bias ε^2_{bias} of the estimator (due to the *noise-to-noise* matching problem), while decreasing at the same time the variance ε^2_{var}. This may be beneficial or not, depending on the noise level, the patch size, and other parameters of NLM. The problem in this case becomes that of assigning the proper weight (see Eq.(2)) to $\boldsymbol{\mu_r}$; this remains an open research question, that can be investigated on the basis of the NLM model introduced here.

It is worth mentioning that both the NN and SNN approaches to NLM denoising require the knowledge of the noise power, σ^2: for NN-based NLM denoising, σ^2 is used to properly set the filtering parameter h in Eq. (2) only, whereas it also guides the selection of the neighbors in the case of SNN. In the case of "real" images, several effective noise estimation methods have been proposed [3], but the noise distribution is far more complicated than zero-mean Gaussian [25]. Even if a comprehensive analysis of the application of SNN to NLM denoising for real images goes beyond the scope of this paper, we show that our approach can be effectively applied to "real" images through the VST approach [25].

A final observation is that our toy problem can also be used to confirm and better analyze results already presented in the literature, and to predict differences between NN and SNN in those cases. An interesting example is represented by the

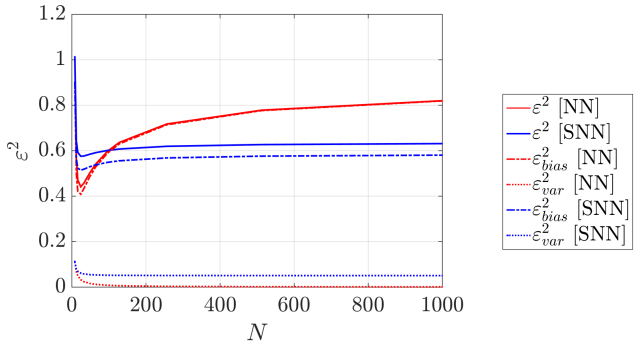


Fig. 13. The prediction error ε^2 (Eq. (13)) for NN and SNN, with its bias and variance components, as predicted for our toy problem and $\mu = 0$, $\sigma = 0.1$, $p_r = 1$, using a set of $N_n = 8$ neighbors extracted from a population of N patches. The result for NN is coherent with those reported in Postec *et al.* [8], showing the capability of our toy problem to replicate theoretical results from literature.

work by Postec *et al.* [8], that suggests that the bias towards the noisy reference patch decreases with the size of the search window up to some point, then it increases. This is equivalent to increasing N in our toy problem. Fig. 13 demonstrates that our toy problem actually reproduces this behavior (even without considering the effect of the weights in Eq. (2)), which is explained considering that the variance and bias of the estimator first decrease collecting a set of different neighbors; on the other hand, when N is too large, NN is more likely to collect samples clustered around μ_r and therefore biased on the average. The SNN approach suffers from a higher variance when N is only slightly larger than N_n, but does not suffer from the same drawback when increasing N: in fact, as samples are collected at distance $o \cdot \sigma$ from μ_r, the bias towards μ_r is largely reduced. This result finally establishes a connection between the role of sampling and that of the weights in NLM, as sampling can be interpreted as a weighting scheme assigning a unitary weight to the selected neighbors, and a null weight to the other ones.

VII. Conclusion

Collecting neighbors through a traditional NN approach can bias the estimate of the noise-free patches in NLM denoising, leading to the introduction of colored noise in the filtered images. The proposed SNN approach for collecting neighbors mitigates this drawback, leading to image quality that is visually comparable to state-of-the-art results in the case of colored noise, but at a lower computational cost. Compared

to the traditional NN approach, SNN is particularly effective in the homogeneous areas of an image, whereas in presence of texture and small details the superiority of NN or SNN is determined by various factors including, among others, the noise level and the number of available patches. To further improve the image quality, an optimal strategy to adpatively modify the offset parameter and consequently switch between NN and SNN could be learned from examples [27], but this goes beyond the scope of the paper. The intuition behind SNN is general and can be effectively applied in other domains like bilateral filtering. The application of SNN to other algorithms and problems, like studying its effect on enlarged search windows in NLM [8], on internal and external denoising [9], or even beyond image processing, remains nonetheless an open research question to be investigated in future.

APPENDIX

A. The Truncated Gaussian Distribution

Let's consider a set of random values generated by a Gaussian distribution with mean μ and variance σ^2, and let γ be a random variable obtained collecting values in the $[a, b]$ interval only; γ follows a truncated Gaussian distribution, $\gamma \sim G'(\mu, \sigma^2, a, b)$, whose expected value and variance are:

$$\alpha = \frac{a-\mu}{\sigma},\ \beta = \frac{b-\mu}{\sigma}$$

$$\mathrm{E}[\gamma] = \mu - \sigma \cdot \frac{\phi(\beta) - \phi(\alpha)}{\Phi(\alpha) - \Phi(\beta)} \tag{27}$$

$$\mathrm{Var}[\gamma] = \sigma^2 \cdot \{1 - \frac{\beta \cdot \phi(\beta) - \alpha \cdot \phi(\alpha)}{\Phi(\beta) - \Phi(\alpha)} + -[\frac{\phi(\beta) - \phi(\alpha)}{\Phi(\beta) - \Phi(\alpha)}]^2\}, \tag{28}$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-0.5\ x^2}$ and $\Phi(x) = 1/2 \cdot [1 + \mathrm{erf}(\frac{x}{\sqrt{2}})]$.

B. Mixture of Independent Random Variables

Given a set of independent random variables, $\{\gamma_i\}_{i=1..N}$, each with pdf $p_i(\gamma_i)$, let's consider the linear mixture γ with pdf $p(\gamma) = \sum_i P_i \cdot p_i(\gamma)$, where P_i represents the probability to select the component i in the mixture. The expected value and variance of γ are then given by:

$$\mathrm{E}[\gamma] = \sum_i P_i E[\gamma_i], \tag{29}$$

$$\mathrm{Var}[\gamma] = \sum_i P_i \{\mathrm{Var}[\gamma_i] + (\mathrm{E}[\gamma_i])^2\} - (\sum_i P_i \mathrm{E}[\gamma_i])^2. \tag{30}$$

C. Linear Combination of Independent Random Variables

Given a set of independent random variables, $\{\gamma_i\}_{i=1..N}$, let's consider the linear combination $\gamma = \sum_i m_i \gamma_i$. The expected value and variance of γ are given by:

$$\mathrm{E}[\gamma] = \sum_i m_i \cdot E[\gamma_i] \tag{31}$$

$$\mathrm{Var}[\gamma] = \sum_i m_i^2 \cdot \mathrm{Var}[\gamma_i]. \tag{32}$$

REFERENCES

[1] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, 2005.

[2] 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, Aug. 2007.

[3] M. Lebrun, M. Colom, A. Buades, and J. M. Morel, "Secrets of image denoising cuisine," *Acta Numer.*, vol. 21, pp. 475–576, May 2012.

[4] A. Buades, B. Coll, and J.-M. Morel, "Non-local means denoising," *Image Process. Line*, vol. 1, pp. 208–212, Sep. 2011.

[5] V. Duval, J.-F. Aujol, and Y. Gousseau, "On the parameter choice for the non-local means," *SIAM J. Imag. Sci.*, p. 37, Mar. 2010. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00468856

[6] Y.-T. Tsai, M. Steinberger, D. Pajak, and K. Pulli, "Fast ANN for high-quality collaborative filtering," *Comput. Graph. Forum*, vol. 35, no. 1, pp. 138–151, 2016.

[7] H. Xu, J. Xu, and F. Wu, "On the biased estimation of nonlocal means filter," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jun./Apr. 2008, pp. 1149–1152.

[8] S. Postec, J. Froment, and B. Vedel. (Nov. 2013). "Non-local means est un algorithme de dÃ©bruitage local (non-local means is a local image denoising algorithm)." [Online]. Available: https://arxiv.org/abs/1311.3768

[9] I. Mosseri, M. Zontak, and M. Irani, "Combining the power of internal and external denoising," in *Proc. IEEE Int. Conf. Comput. Photogr. (ICCP)*, Apr. 2013, pp. 1–9.

[10] Y. Wu, B. Tracey, P. Natarajan, and J. P. Noonan, "Probabilistic non-local means," *IEEE Signal Process. Lett.*, vol. 20, no. 8, pp. 763–766, Aug. 2013.

[11] Y. Lou, P. Favaro, S. Soatto, and A. Bertozzi, "Nonlocal similarity image filtering," in *Image Analysis and Processing*, P. Foggia, C. Sansone, and M. Vento, Eds. Berlin, Germany: Springer, 2009, pp. 62–71.

[12] O. Lotan and M. Irani, "Needle-match: Reliable patch matching under high uncertainty," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 439–448.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer, 2001. [Online]. Available: https://www.bibsonomy.org/bibtex/2f58afc5c9793fcc8ad8389824e57984c/sb3000

[14] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D image denoising with shape-adaptive principal component analysis," in *Proc. Workshop Signal Process. Adapt. Sparse Structured Represent. (SPARS)*, 2009.

[15] C. Knaus and M. Zwicker, "Progressive image denoising," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 3114–3125, Jul. 2014.

[16] B. Ahn and N. I. Cho, "Block-matching convolutional neural network for image denoising," *CoRR*, vol. abs/1704.00524, 2017. [Online]. Available: http://arxiv.org/abs/1704.00524

[17] N. Ponomarenko *et al.*, "Image database TID2013: Peculiarities, results and perspectives," *Signal Process., Image Commun.*, vol. 30, pp. 57–77, Jan. 2015.

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[19] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. ACSSC*, Nov. 2003, pp. 1398–1402.

[20] W. Xue, L. Zhang, X. Mou, and A. C. Bovik, "Gradient magnitude similarity deviation: A highly efficient perceptual image quality index," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 684–695, Feb. 2014.

[21] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011.

[22] G. Yu and G. Sapiro, "DCT image denoising: A simple and effective image denoising algorithm," *J. Image Process. On Line*, vol. 1, pp. 292–296, Oct. 2011.

[23] A. Danielyan, M. Vehvilainen, A. Foi, V. Katkovnik, and K. Egiazarian, "Cross-color BM3D filtering of noisy raw data," in *Proc. Int. Workshop Local Non-Local Approx. Image Process.*, Aug. 2009, pp. 125–129.

[24] H. S. Malvar, L.-W. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, May 2004, p. III-485.

[25] A. Foi, "Clipped noisy images: Heteroskedastic modeling and practical denoising," *Signal Process.*, vol. 89, no. 12, pp. 2609–2629, 2009.
[26] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. ICCV*, Jan. 1998, pp. 839–846.
[27] J. Dong, I. Frosio, and J. Kautz, "Learning adaptive parameter tuning for image processing," *Electron. Imag.*, vol. 2018, no. 13, pp. 196-1–196-8, 2018. [Online]. Available: https://www.ingentaconnect.com/content/ist/ei/2018/00002018/00000013/art00004, doi: doi:10.2352/ISSN.2470-1173.2018.13.IPAS-196.

Iuri Frosio received the M.S. and Ph.D. degrees in biomedical engineering from the Politecnico di Milano in 2003 and 2006, respectively. He was a Research Fellow with the Computer Science Department, Universitá degli Studi di Milano, from 2003 to 2006, where he was also an Assistant Professor from 2006 to 2014. In the same period, he was a consultant for various companies in Italy and abroad. He joined Nvidia as a Senior Research Scientist in 2014. His research interests include image processing, computer vision, inertial sensors, reinforcement learning, machine learning, and parallel computing. He is currently an Associate Editor of the *Journal of Electronic Imaging*.

Jan Kautz received the B.Sc. degree in computer science from the University of Erlangen-Nurnberg in 1999, the M.Math. degree from the University of Waterloo in 1999, and the Ph.D. degree from the Max-Planck Institut fur Informatik in 2003. He held a post-doctoral position at the Massachusetts Institute of Technology from 2003 to 2006. He leads the Visual Computing Research Team at Nvidia, where he is currently involved predominantly in computer vision problems from low-level vision (denoising, super-resolution, computational photography) and geometric vision (structure from motion, SLAM, optical flow) to high-level vision (detection, recognition, classification) and machine learning problems, such as deep reinforcement learning and efficient deep learning. Before joining Nvidia in 2013, he was a tenured Faculty Member at the University College London. He was the Program Co-Chair of the Eurographics Symposium on Rendering 2007 and Pacific Graphics 2011, and the Program Chair of the IEEE Symposium on Interactive Ray-Tracing 2008 and CVMP 2012 . He has Co-Chaired Eurographics 2014. He was on the Editorial Board of the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS and *The Visual Computer*.