Monte Carlo Non-Local Means: Random Sampling for Large-Scale Image Filtering

Stanley H. Chan, *Member, IEEE,* Todd Zickler, *Member, IEEE,* and Yue M. Lu, *Senior Member, IEEE*

***Abstract*—We propose a randomized version of the non-local means (NLM) algorithm for large-scale image filtering. The new algorithm, called Monte Carlo non-local means (MCNLM), speeds up the classical NLM by computing a small subset of image patch distances, which are randomly selected according to a designed sampling pattern. We make two contributions. First, we analyze the performance of the MCNLM algorithm and show that, for large images or large external image databases, the random outcomes of MCNLM are tightly concentrated around the deterministic full NLM result. In particular, our error probability bounds show that, at any given sampling ratio, the probability for MCNLM to have a large deviation from the original NLM solution decays exponentially as the size of the image or database grows. Second, we derive explicit formulas for optimal sampling patterns that minimize the error probability bound by exploiting partial knowledge of the pairwise similarity weights. Numerical experiments show that MCNLM is competitive with other state-of-the-art fast NLM algorithms for single-image denoising. When applied to denoising images using an external database containing ten billion patches, MCNLM returns a randomized solution that is within 0.2 dB of the full NLM solution while reducing the runtime by three orders of magnitude.**

***Index Terms*—Non-local means, Monte Carlo, patch-based filtering, sampling, external denoising, large deviations analysis**

I. Introduction

A. Background and Motivation

In recent years, the image processing community has witnessed a wave of research aimed at developing new image denoising algorithms that exploit similarities between non-local patches in natural images. Most of these can be traced back to the non-local means (NLM) denoising algorithm of Buades *et al.* [1], [2] proposed in 2005. To date, NLM remains one of the most influential algorithms in the current denoising literature.

Given a noisy image, the NLM algorithm uses two sets of image patches for denoising. The first is a set of noisy patches $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$, where $\boldsymbol{y}_i \in \mathbb{R}^d$ is a d-dimensional (*i.e.*, d-pixel) patch centered at the ith pixel of the noisy image. The second set, $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, contains patches that are obtained from some reference images. Conceptually, NLM simply replaces each ith noisy pixel with a weighted average of pixels in the reference set. Specifically, the filtered value at the ith pixel (for $1 \le i \le m$) is given by

$$z = \frac{\sum_{j=1}^{n} w_{i,j} x_j}{\sum_{j=1}^{n} w_{i,j}}, \tag{1}$$

where x_j denotes the value of the center pixel of the jth reference patch $\boldsymbol{x}_j \in \mathcal{X}$, and the weights $\{w_{i,j}\}$ measure the similarities between the patches $\boldsymbol{y}_i$ and $\boldsymbol{x}_j$. A standard choice for the weights is

$$w_{i,j} = e^{-\|\boldsymbol{y}_i - \boldsymbol{x}_j\|_{\boldsymbol{\Lambda}}^2/(2h_r^2)}, \tag{2}$$

where h_r is a scalar parameter determined by the noise level, and $\|\cdot\|_{\boldsymbol{\Lambda}}$ is the weighted ℓ_2-norm with a diagonal weight matrix $\boldsymbol{\Lambda}$, *i.e.*, $\|\boldsymbol{y}_i - \boldsymbol{x}_j\|_{\boldsymbol{\Lambda}}^2 \overset{\text{def}}{=} (\boldsymbol{y}_i - \boldsymbol{x}_j)^T \boldsymbol{\Lambda} (\boldsymbol{y}_i - \boldsymbol{x}_j)$.

In most implementations of NLM (see, *e.g.*, [3]–[9]), the denoising process is based on a single image: the reference patches $\mathcal{X}$ are the same as the noisy patches $\mathcal{Y}$. We refer to this setting, when $\mathcal{X} = \mathcal{Y}$, as *internal denoising*. This is in contrast to the setting in which the set of reference patches $\mathcal{X}$ come from external image databases [10]–[12], which we refer to as *external denoising*. For example, $15,000$ images (corresponding to a reference set of $n \approx 10^{10}$ patches) were used in [11], [12]. One theoretical argument for using large-scale external denoising was provided in [11]: It is shown that, in the limit of large reference sets (*i.e.*, when $n \to \infty$), external NLM converges to the minimum mean squared error estimator of the underlying clean images.

Despite its strong performance, NLM has a limitation of high computational complexity. It is easy to see that computing all the weights $\{w_{i,j}\}$ requires $\mathcal{O}(mnd)$ arithmetic operations, where m, n, d are, respectively, the number of pixels in the noisy image, the number of reference patches used, and the patch dimension. Additionally, about $\mathcal{O}(mn)$ operations are needed to carry out the summations and multiplications in (1) for all pixels in the image. In the case of internal denoising, these numbers are nontrivial since current digital photographs can easily contain tens of millions of pixels (*i.e.*, $m = n \sim 10^7$ or greater). For external denoising with large reference sets (*e.g.*, $n \sim 10^{10}$), the complexity is even more of an issue, making it very challenging to fully utilize the vast number of images that are readily available online and potentially useful as external databases.

The authors are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA. E-mails: {schan,zickler,yuelu}@seas.harvard.edu.

This work was supported in part by the Croucher Foundation Post-doctoral Research Fellowship (2012-2013), and in part by the U.S. National Science Foundation under Grant CCF-1319140. Preliminary material in this paper was presented at the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, May 2013.

This paper follows the concept of reproducible research. All the results and examples presented in the paper are reproducible using the code and images available online at http://scholar.harvard.edu/stanleychan.

B. Related Work

The high complexity of NLM is a well-known challenge. Previous methods to speed up NLM can be roughly classified in the following categories:

1. Reducing the reference set $\mathcal{X}$. If searching through a large set $\mathcal{X}$ is computationally intensive, one natural solution is to pre-select a subset of $\mathcal{X}$ and perform computation only on this subset [13]–[15]. For example, for internal denoising, a spatial weight $w^s_{i,j}$ is often included so that

$$w_{i,j} = w^s_{i,j} \cdot \underbrace{e^{-\|\boldsymbol{y}_i - \boldsymbol{x}_j\|^2_{\boldsymbol{\Lambda}}/(2h_r^2)}}_{w^r_{i,j}}. \tag{3}$$

A common choice of the spatial weight is

$$w^s_{i,j} = \exp\{-d^2_{i,j}/(2h_s^2)\} \cdot \mathbb{I}\{d'_{i,j} \le \rho\}, \tag{4}$$

where $d_{i,j}$ and $d'_{i,j}$ are, respectively, the Euclidean distance and the ℓ_∞ distance between the spatial locations of the ith and jth pixels; $\mathbb{I}$ is the indicator function; and ρ is the width of the spatial search window. By tuning h_s and ρ, one can adjust the size of $\mathcal{X}$ according to the heuristic that nearby patches are more likely to be similar.

2. Reducing dimension d. The patch dimension d can be reduced by several methods. First, SVD projection [8], [16]–[18] can be used to project the d-dimensional patches onto a lower dimensional space spanned by the principal components computed from $\mathcal{X}$. Second, the integral image method [19]–[21] can be used to further speed up the computation of $\|\boldsymbol{y}_i - \boldsymbol{x}_j\|^2_{\boldsymbol{\Lambda}}$. Third, by assuming a Gaussian model on the patch data, a probabilistic early termination scheme [22] can be used to stop computing the squared patch difference before going through all the pixels in the patches.

3. Optimizing data structures. The third class of methods embed the patches in $\mathcal{X}$ and $\mathcal{Y}$ in some form of optimized data structures. Some examples include the fast bilateral grid [23], the fast Gaussian transform [24], the Gaussian KD tree [25], [26], the adaptive manifold method [27], and the edge patch dictionary [28]. The data structures used in these algorithms can significantly reduce the computational complexity of the NLM algorithm. However, building these data structures often requires a lengthy pre-processing stage, or require a large amount of memory, thereby placing limits on one's ability to use large reference patch sets $\mathcal{X}$. For example, building a Gaussian KD tree requires the storage of $\mathcal{O}(nd)$ double precision numbers (see, *e.g.*, [26], [29].)

4. Exploiting low-rank structures. Several recent approaches, *e.g.*, [30], [31], explore low-rank structures of the weight matrix $\boldsymbol{W} = [w_{i,j}]$. In [31], Talebi and Milanfar apply the Nyström approximation [32] to estimate all pairwise similarity weights $\{w_{i,j}\}$ from a subset of samples. This approach requires storing and computing the SVD of a matrix of size $\xi n \times \xi n$, where ξ is the sampling ratio. In contrast, the algorithm presented here requires no additional storage beyond the input image (and the database for the external case.)

C. Contributions

In this paper, we propose a randomized algorithm to reduce the computational complexity of NLM for both internal and external denoising. We call the method *Monte Carlo Non-Local Means* (MCNLM), and the basic idea is illustrated in Figure 1, for the case of internal denoising. For each pixel i in the noisy image, we randomly select a set of k reference pixels according to some sampling pattern and compute a k-subset of the weights $\{w_{i,j}\}_{j=1}^n$ to form an approximated solution to (1). The computational complexity of MCNLM is $\mathcal{O}(mkd)$, which can be significantly lower than the original complexity $\mathcal{O}(mnd)$ when only $k \ll n$ weights are computed. Furthermore, since there is no need to re-organize the data, the memory requirement of MCNLM is $\mathcal{O}(m + n)$. Therefore, MCNLM is scalable to large reference patch sets $\mathcal{X}$ like those used for external denoising, as we will demonstrate in Section V.

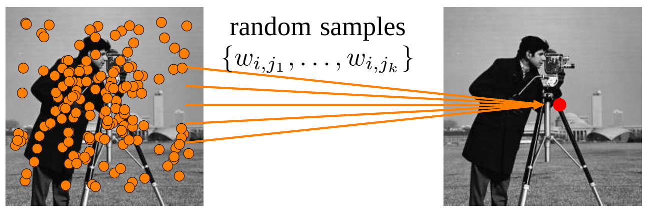


Fig. 1: Illustration of the proposed MCNLM algorithm for internal denoising: We randomly select, according to a given sampling pattern, a set of k weights $\{w_{i,j_1}, \ldots, w_{i,j_k}\}$, and use these to compute an approximation of the full NLM result in (1). The output of MCNLM is random. However, as the size of the problem (*i.e.*, n) gets larger, these random estimates become tightly concentrated around the true result.

The two main contributions of this paper are as follows.

1. Performance guarantee. MCNLM is a randomized algorithm. It would not be a useful one if its random outcomes fluctuated widely in different executions on the same input data. In Section III, we address this concern by showing that, as the size of the reference set $\mathcal{X}$ increases, the randomized MCNLM solutions become tightly concentrated around the original NLM solution. In particular, we show in Theorem 1 (and Proposition 1) that, for *any* given sampling pattern, the probability of having a large deviation from the original NLM solution drops exponentially as the size of $\mathcal{X}$ grows.

2. Optimal sampling patterns. We derive optimal sampling patterns to minimize the approximation error probabilities established in our performance analysis. We show that seeking the optimal sampling pattern is equivalent to solving a variant of the classical water-filling problem, for which a closed-form expression can be found (see Theorem 2). We also present two practical sampling pattern designs that exploit partial knowledge of the pairwise similarity weights.

The rest of the paper is organized as follows. We present the MCNLM algorithm and discuss its basic properties in Section II. The performance of the algorithm is analyzed in Section III, and optimal sampling patterns are presented in Section IV. We demonstrate in Section V the effectiveness of the proposed algorithm by showing simulation results for both internal and external denoising. Section VI concludes the paper.

II. Monte Carlo Non-local Means

Notation: Throughout the paper, we use m to denote the number of pixels in the noisy image, and n the number patches in the reference set $\mathcal{X}$. We use upper-case letters, such as X, Y, Z, to represent random variables, and lower-case letters, such as x, y, z, to represent deterministic variables. Vectors are represented by bold letters, and $\mathbf{1}$ denotes a constant vector of which all entries are one. Finally, for notational simplicity in presenting our theoretical analysis, we assume that all pixel intensity values have been normalized to the range $[0, 1]$.

A. The Sampling Process

As discussed in Section I, computing all the weights $\{w_{i,j}\}_{1\le i\le m, 1\le j\le n}$ is computationally prohibitive when m and n are large. To reduce the complexity, the basic idea of MCNLM is to randomly select a subset of k representatives of $\{w_{i,j}\}$ (referred to as samples) to approximate the sums in the numerator and denominator in (1). The sampling process in the proposed algorithm is applied to each of the m pixels in the noisy image *independently*. Since the sampling step and subsequent computations have the same form for each pixel, we shall drop the pixel index i in $\{w_{i,j}\}$, writing the weights as $\{w_j\}_{1\le j\le n}$ for notational simplicity.

The sampling process of MCNLM is determined by a sequence of *independent* random variables $\{I_j\}_{j=1}^n$ that take the value 0 or 1 with the following probabilities

$$\Pr[I_j = 1] = p_j \qquad \text{and} \qquad \Pr[I_j = 0] = 1 - p_j. \tag{5}$$

The jth weight w_j is sampled if and only if $I_j = 1$. In what follows, we assume that $0 < p_j \le 1$, and refer to the vector of all these probabilities $\boldsymbol{p} \overset{\text{def}}{=} [p_1, \ldots, p_n]^T$ as the *sampling pattern* of the algorithm.

The ratio between the number of samples taken and the number of reference patches in $\mathcal{X}$ is a random variable

$$S_n = \frac{1}{n}\sum_{j=1}^n I_j, \tag{6}$$

of which the expected value is

$$\mathbb{E}[S_n] = \frac{1}{n}\sum_{j=1}^n \mathbb{E}[I_j] = \frac{1}{n}\sum_{j=1}^n p_j \overset{\text{def}}{=} \xi. \tag{7}$$

We refer to S_n and ξ as the *empirical sampling ratio* and the *average sampling ratio*, respectively. ξ is an important parameter of the MCNLM algorithm. The original (or "full") NLM corresponds to the setting when $\xi = 1$: In this case, $\boldsymbol{p} = \mathbf{1} \overset{\text{def}}{=} [1, \ldots, 1]^T$, so that all the samples are selected with probability one.

B. The MCNLM Algorithm

Given a set of random samples from $\mathcal{X}$, we approximate the numerator and denominator in (1) by two random variables

$$A(\boldsymbol{p}) \overset{\text{def}}{=} \frac{1}{n}\sum_{j=1}^n \frac{x_j w_j}{p_j} I_j \quad \text{and} \quad B(\boldsymbol{p}) \overset{\text{def}}{=} \frac{1}{n}\sum_{j=1}^n \frac{w_j}{p_j} I_j, \tag{8}$$

Algorithm 1 Monte Carlo Non-local Means (MCNLM)

```
1: For each noisy pixel $i = 1, \ldots, m$, do the followings.
2: Input: Noisy patch $\boldsymbol{y}_i \in \mathcal{Y}$, database $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$
   and sampling pattern $\boldsymbol{p} = [p_1, \ldots, p_n]^T$ such that $0 <$
   $p_j \le 1$, and $\sum_{j=1}^n p_j = n\xi$.
3: Output: A randomized estimate $Z(\boldsymbol{p})$.
4: for $j = 1, \ldots, n$ do
5:     Generate a random variable $I_j \sim$ Bernoulli$(p_j)$.
6:     If $I_j = 1$, then compute the weight $w_j$.
7: end for
8: Compute $A(\boldsymbol{p}) = \frac{1}{n}\sum_{j=1}^n \frac{w_j x_j}{p_j} I_j$.
9: Compute $B(\boldsymbol{p}) = \frac{1}{n}\sum_{j=1}^n \frac{w_j}{p_j} I_j$.
10: Output $Z(\boldsymbol{p}) = A(\boldsymbol{p})/B(\boldsymbol{p})$.
```

where the argument $\boldsymbol{p}$ emphasizes the fact that the distributions of A and B are determined by the sampling pattern $\boldsymbol{p}$.

It is easy to compute the expected values of $A(\boldsymbol{p})$ and $B(\boldsymbol{p})$ as

$$\mu_A \overset{\text{def}}{=} \mathbb{E}[A(\boldsymbol{p})] = \frac{1}{n}\sum_{j=1}^n x_j w_j, \tag{9}$$

$$\mu_B \overset{\text{def}}{=} \mathbb{E}[B(\boldsymbol{p})] = \frac{1}{n}\sum_{j=1}^n w_j. \tag{10}$$

Thus, up to a common multiplicative constant $1/n$, the two random variables $A(\boldsymbol{p})$ and $B(\boldsymbol{p})$ are *unbiased* estimates of the true numerator and denominator, respectively.

The full NLM result z in (1) is then approximated by

$$Z(\boldsymbol{p}) \overset{\text{def}}{=} \frac{A(\boldsymbol{p})}{B(\boldsymbol{p})} = \frac{\sum_{j=1}^n \frac{x_j w_j}{p_j} I_j}{\sum_{j=1}^n \frac{w_j}{p_j} I_j}. \tag{11}$$

In general, $\mathbb{E}[Z(\boldsymbol{p})] = \mathbb{E}\left[\frac{A(\boldsymbol{p})}{B(\boldsymbol{p})}\right] \neq \frac{\mathbb{E}[A(\boldsymbol{p})]}{\mathbb{E}[B(\boldsymbol{p})]} = z$, and thus $Z(\boldsymbol{p})$ is a *biased* estimate of z. However, we will show in Section III that the probability of having a large deviation in $|Z(\boldsymbol{p}) - z|$ drops exponentially as $n \to \infty$. Thus, for a large n, the MCNLM solution (11) can still form a very accurate approximation of the original NLM solution (1).

Algorithm 1 shows the pseudo-code of MCNLM for internal denoising. We note that, except for the Bernoulli sampling process, all other steps are identical to the original NLM. Therefore, MCNLM can be thought of as adding a complementary sampling process on top of the original NLM. The marginal cost of implementation is thus minimal.

Example 1: To empirically demonstrate the usefulness of the simple sampling mechanism of MCNLM, we apply the algorithm to a 1072×712 image shown in Figure 2(a). Here, we use $\mathcal{X} = \mathcal{Y}$, with $m = n \approx 7.6 \times 10^5$. In this experiment, we let the noise be i.i.d. Gaussian with zero mean and standard deviation $\sigma = 15/255$. The patch size is 5×5. In computing the similarity weights in (3) and (4), we set the parameters as follows: $h_r = 15/255$, $h_s = \infty$, $\rho = \infty$ (*i.e.*, no spatial windowing) and $\boldsymbol{\Lambda} = \frac{1}{25}\boldsymbol{I}$. We choose a uniform sampling pattern, *i.e.*, $\boldsymbol{p} = [\xi, \ldots, \xi]^T$, for some sampling ratio $0 < \xi < 1$.

The results of this experiment are shown in Figure 2 and

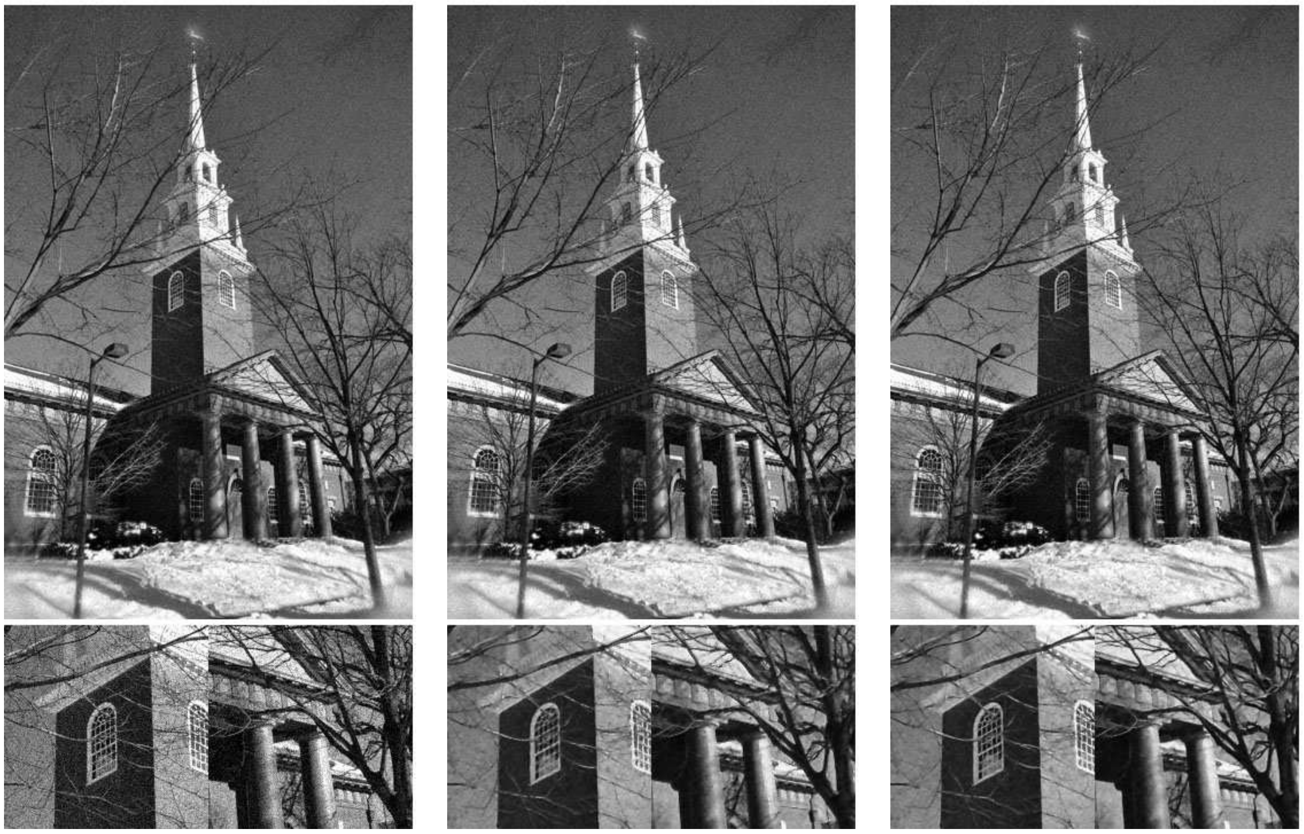


Fig. 2: Denoising an image of size 1072×712 by MCNLM with uniform sampling. (a) The original image is corrupted with i.i.d. Gaussian noise with $\sigma = 15/255$. (b) and (c) Denoised images with sampling ratio $\xi = 0.005$ and $\xi = 0.1$, respectively. Shown in parenthesis are the PSNR values (in dB) averaged over 100 trials.

Figure 3. The peak signal-to-noise ratio (PSNR) curve detailed in Figure 3 shows that MCNLM converges to its limiting value rapidly as the sampling ratio ξ approaches 1. For example, at $\xi = 0.1$ (*i.e.*, a roughly ten-fold reduction in computational complexity), MCNLM achieves a PSNR that is only 0.2dB away from the full NLM result. More numerical experiments will be presented in Section V.

III. Performance Analysis

One fundamental question about MCNLM is whether its random estimate $Z(\boldsymbol{p})$ as defined in (11) will be a good approximation of the full NLM solution z, especially when the sampling ratio ξ is small. In this section, we answer this question by providing a rigorous analysis on the approximation error $|Z(\boldsymbol{p}) - z|$.

A. Large Deviations Bounds

The mathematical tool we use to analyze the proposed MCNLM algorithm comes from the probabilistic *large deviations theory* [33]. This theory has been widely used to quantify the following phenomenon: A *smooth* function $f(X_1, \ldots, X_n)$ of a large number of *independent* random variables $X_1, \ldots, X_n$ tends to concentrate very tightly around its mean $\mathbb{E}[f(X_1, \ldots, X_n)]$. Roughly speaking, this concentration phenomenon happens because, while $X_1, \ldots, X_n$ are

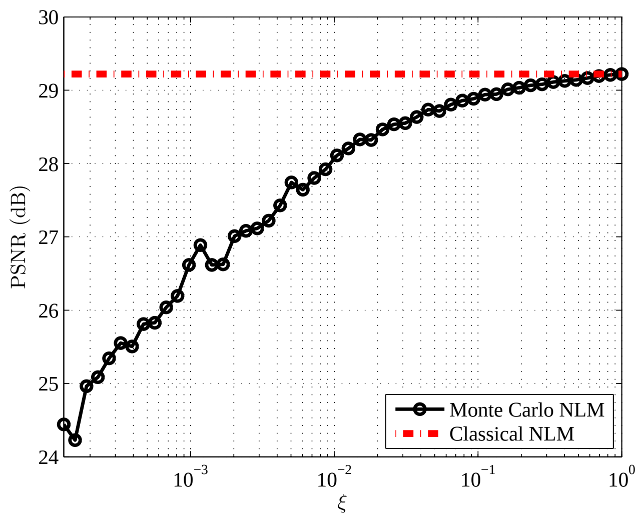


Fig. 3: PSNR as a function of the average sampling ratio ξ. The horizontal line indicates the full NLM result (*i.e.*, MCNLM at $\xi = 1$), and the "circled" line indicates the result of MCNLM. Note that at $\xi = 0.1$, MCNLM achieves a PSNR that is only 0.2 dB below the full NLM result. Results shown are average values over 100 independent trials.

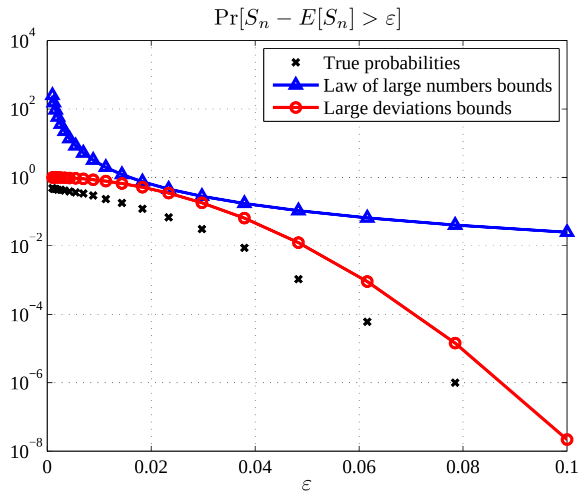


Fig. 4: Comparing the large deviations bound (13), the LLN bound (12), and the true error probability $\Pr[S_n - \mathbb{E}[S_n] > \varepsilon]$ as estimated by Monte Carlo simulations. Fixing $n = 10^6$, we plot the bounds and probabilities for different values of ε.

individually random in nature, it is unlikely for many of them to work collaboratively to significantly alter the overall system. Thus, for large n, the randomness of these variables tends to be "canceled out" and the function $f(X_1, \ldots, X_n)$ stays roughly constant.

To gain insights from a concrete example, we first apply the large deviations theory to study the empirical sampling ratio S_n as defined in (6). Here, the independent random variables are the Bernoulli random variables $\{I_j\}_{1 \le j \le n}$ introduced in (5), and the smooth function $f(\cdot)$ computes their average.

It is well known from the law of large numbers (LLN) that the empirical mean S_n of a large number of independent random variables stays very close to the true mean, which is equal to the average sampling ratio ξ in our case. In particular, by the standard Chebyshev inequality [34], we know that

$$\Pr[S_n - \mathbb{E}[S_n] > \varepsilon] \le \Pr[|S_n - \mathbb{E}[S_n]| > \varepsilon] < \frac{\mathrm{Var}[I_n]}{n\varepsilon^2}, \quad (12)$$

for every positive ε.

One drawback of the bound in (12) is that it is overly loose, providing only a linear rate of decay for the error probabilities as $n \to \infty$. In contrast, the large deviations theory provides many powerful probability inequalities which often lead to much tighter bounds with exponential decays. In this work, we will use one particular inequality, due to S. Bernstein [35]:

Lemma 1 (Bernstein Inequality [35]): Let $X_1, \ldots, X_n$ be a sequence of independent random variables. Suppose that $|X_j| \le M$ for all j, where M is a constant. Let $S_n = \frac{1}{n}\sum_{j=1}^n X_j$. Then for every positive ε,

$$\Pr\left[S_n - \mathbb{E}[S_n] > \varepsilon\right] \le \exp\left\{-\frac{n\varepsilon^2}{2\left(\frac{1}{n}\sum_{j=1}^n \mathrm{Var}[X_j] + M\varepsilon/3\right)}\right\}. \quad (13)$$

To see how Bernstein's inequality can give us a better probability bound for the empirical sampling ratio S_n, we note that $X_j = I_j$ in our case. Thus, $M = 1$ and $\mathbb{E}[S_n] = \xi$. Moreover, if the sampling pattern is uniform, ***i.e.***, $\boldsymbol{p} = [\xi, \ldots, \xi]^T$, we have $\frac{1}{n}\sum_{j=1}^n \mathrm{Var}[X_j] = \frac{1}{n}\sum_{j=1}^n p_j(1-p_j) = \xi(1-\xi)$. Substituting these numbers into (13) yields an *exponential* upper bound on the error probability, which is plotted and compared in Figure 4 against the LLN bound in (12) and against the true probabilities estimated by Monte Carlo simulations. It is clear that the exponential bound provided by Bernstein's inequality is much tighter than that provided by LLN.

B. General Error Probability Bound for MCNLM

We now derive a general bound for the error probabilities of MCNLM. Specifically, for any $\varepsilon > 0$ and any sampling pattern $\boldsymbol{p}$ satisfying the conditions that $0 < p_j \le 1$ and $\frac{1}{n}\sum_{j=1}^n p_j = \xi$, we want to study

$$\Pr\left[|Z(\boldsymbol{p}) - z| > \varepsilon\right], \quad (14)$$

where z is the full NLM result defined in (1) and $Z(\boldsymbol{p})$ is the MCNLM estimate defined in (11).

Theorem 1: Assume that $w_j > 0$ for all j. Then for every positive ε,

$$\begin{aligned}\Pr\left[|Z(\boldsymbol{p}) - z| > \varepsilon\right] &\le \exp\{-n\xi\} \\ &+ \exp\left\{\frac{-n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^n \alpha_j^2\left(\frac{1-p_j}{p_j}\right) + (\mu_B\varepsilon)M_\alpha/3\right)}\right\} \\ &+ \exp\left\{\frac{-n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^n \beta_j^2\left(\frac{1-p_j}{p_j}\right) + (\mu_B\varepsilon)M_\beta/3\right)}\right\},\end{aligned} \quad (15)$$

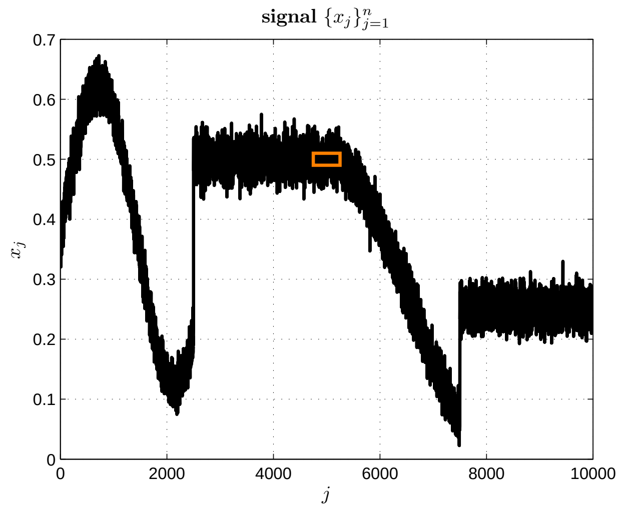
where μ_B is the average similarity weights defined in (10), $\alpha_j = w_j(x_j - z - \varepsilon)$, $\beta_j = w_j(x_j - z + \varepsilon)$, and

$$M_\alpha = \max_{1\le j\le n}\left(|\alpha_j| \max\left\{1, \frac{1-p_j}{p_j}\right\}\right),$$
$$M_\beta = \max_{1\le j\le n}\left(|\beta_j| \max\left\{1, \frac{1-p_j}{p_j}\right\}\right).$$

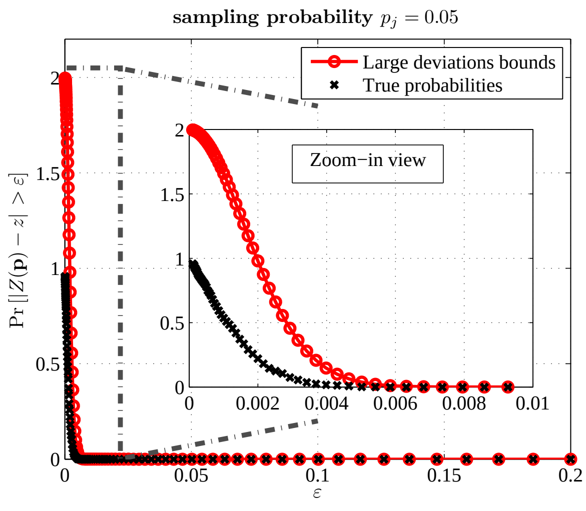
Proof: See Appendix A. ■

Remark 1: In a preliminary version of our work [36], we presented, based on the idea of martingales [37], an error probability bound for the special case when the sampling pattern is uniform. The result of Theorem 1 is more general and applies to any sampling patterns. We also note that the bound in (15) quantifies the deviation of a ratio $Z(\boldsymbol{p}) = A(\boldsymbol{p})/B(\boldsymbol{p})$, where the numerator and denominator are both weighted sums of independent random variables. It is therefore more general than the typical concentration bounds seen in the literature (see, *e.g.*, [38], [39]), where only a single weighted sum of random variables (*i.e.*, either the numerator or the denominator) is considered.

Example 2: To illustrate the result of Theorem 1, we consider a one-dimensional signal as shown in Figure 5(a). The signal $\{x_j\}_{j=1}^n$ is a piecewise continuous function corrupted by i.i.d. Gaussian noise. The noise standard deviation is $\sigma = 5/255$ and the signal length is $n = 10^4$. We use MCNLM to denoise the 5001-th pixel, and the sampling pattern is uniform



(a) Noisy signal $\{x_j\}_{j=1}^n$



(b) Probability $\Pr[|Z(\boldsymbol{p}) - z| > \varepsilon]$

Fig. 5: Example to illustrate Theorem 1. (a) A one-dimensional signal with length $n = 10^4$, corrupted by i.i.d. Gaussian noise with $\sigma = 5/255$. We use the MCNLM algorithm to denoise the signal. The patch size is $d = 5$ and the parameters are $h_r = 15/255$ and $h_s = \infty$, respectively. (b) The error probability as a function of ε. In this plot, the "crosses" denote the true probabilities as estimated by 10^5 independent trials and the "circles" denote the analytical upper bound predicted by Theorem 1. For easy comparisons, we also provide a zoomed-in version of the plot in the insert.

with $p_j = \xi = 0.05$ for all j. For $\varepsilon = 0.01$, we can compute that $\frac{1}{n}\sum_{j=1}^n \alpha_j^2 = 1.194 \times 10^{-4}$, $\frac{1}{n}\sum_{j=1}^n \beta_j^2 = 1.258 \times 10^{-4}$, $\mu_B = 0.282$, $M_\alpha = 0.951$, and $M_\beta = 0.932$. It then follows from (15) that

$$\Pr[|Z(\boldsymbol{p}) - z| > 0.01] \leq 8.856 \times 10^{-6}.$$

This bound shows that the random MCNLM estimate $Z(\boldsymbol{p})$, obtained by taking only 5% of the samples, stays within one percent of the true NLM result z with overwhelming probability. A complete range of results for different values of ε are shown in Figure 5(b), where we compare the true error probability as estimated by Monte Carlo simulations with the analytical upper bound predicted by Theorem 1. We see from the "zoomed-in" portion of Figure 5(b) that the analytical bound approaches the true probabilities for $\varepsilon \geq 0.005$ (*i.e.*, 0.5% deviation.)

C. Special Case: Uniform Sampling Patterns

The error probability bound in (15) holds for all sampling patterns $\boldsymbol{p}$. In Section IV, we will use this versatile bound to design optimal nonuniform sampling patterns. In general, choosing $\boldsymbol{p}$ to be uniform leads to suboptimal performance. Nevertheless, it is still instructive to consider this case, since it provides a convenient and easily interpretable bound on the error probabilities.

Proposition 1 (Uniform Sampling): Assume that the sampling pattern is uniform, *i.e.*, $\boldsymbol{p} = \xi \mathbf{1}$. Then for every $\varepsilon > 0$ and every $0 < \xi \leq 0.5$,

$$\Pr[|Z(\boldsymbol{p}) - z| > \varepsilon] \leq \exp\{-n\xi\} + 2\exp\{-n\mu_B f(\varepsilon)\xi/(1-\xi)\}, \quad (16)$$

where $f(\varepsilon) \overset{\text{def}}{=} \varepsilon^2 / \big(2(1+\varepsilon)(1+4\varepsilon/3)\big)$.

Proof: See Appendix B. ■

We note that, for large n, the first term on the right-hand side of (16) is negligible. For example, when $n = 10^4$ and $\xi = 0.01$, we have $e^{-n\xi} = 3.7 \times 10^{-44}$. Thus, the error probability bound is dominated by the second term, whose negative exponent is determined by four factors:

1. *The size of the reference set $\mathcal{X}$.* If all other parameters are kept fixed or strictly bounded below by some positive constants, the error probability goes to zero as an exponential function of n. This shows that the random estimates obtained by MCNLM can be very accurate, when the size of the image (for internal denoising) or the size of the dictionary (for external denoising) is large.

2. *Sampling ratio ξ.* To reduce the sampling ratio ξ while still keeping the error probability small, a larger n, inversely proportional to ξ, is needed.

3. *Precision ε.* Note that the function $f(\varepsilon)$ in (16) is of order $\mathcal{O}(\varepsilon^2)$ for small ε. Thus, with all other terms fixed, a k-fold reduction in ε requires a k^2-fold increase in n or ξ.

4. *Patch redundancy μ_B.* Recall that $\mu_B = \frac{1}{n}\sum_{j=1}^n w_j$, with the weights $\{w_j\}$ measuring the similarities between a noisy patch $\boldsymbol{y}_i$ and all patches $\{\boldsymbol{x}_j\}_{j=1}^n$ in the reference set $\mathcal{X}$. Thus, μ_B serves as an indirect measure of the number of patches in $\mathcal{X}$ that are similar to $\boldsymbol{y}_i$. If $\boldsymbol{y}_i$ can find many similar (redundant) patches in $\mathcal{X}$, its corresponding μ_B will be large and so a relatively small n will be sufficient to make the probability small; and vice versa.

Using the simplified expression in (16), we derive in Appendix C the following upper bound on the mean squared error (MSE) of the MCNLM estimation:

Proposition 2 (MSE): Let the sampling pattern be uniform, with $\boldsymbol{p} = \xi\mathbf{1}$. Then for any $0 < \xi \le 0.5$,

$$\text{MSE} \overset{\text{def}}{=} \mathbb{E}\left[(Z(\boldsymbol{p}) - z)^2\right] \le e^{-n\xi} + \frac{1}{n}\left(\frac{56}{3\mu_B}\right)\left(\frac{1-\xi}{\xi}\right). \tag{17}$$

Remark 2: The above result indicates that, with a fixed average sampling ratio ξ and if the patch redundancy μ_B is bounded from below by a positive constant, then the MSE of the MCNLM estimation converges to zero as n, the size of the reference set, goes to infinity.

IV. Optimal Sampling Patterns

While the uniform sampling scheme (*i.e.*, $\boldsymbol{p} = \xi\mathbf{1}$) allows for easy analysis and provides useful insights, the performance of the proposed MCNLM algorithm can be significantly improved by using properly chosen nonuniform sampling patterns. We present the design of such patterns in this section.

A. Design Formulation

The starting point of seeking an optimal sampling pattern is the general probability bound provided by Theorem 1. A challenge in applying this probability bound in practice is that the right-hand side of (15) involves the complete set of weights $\{w_j\}$ and the full NLM result z. One can of course compute these values, but doing so will defeat the purpose of random sampling, which is to speed up NLM by *not* computing all the weights $\{w_j\}$. To address this problem, we assume that

$$0 < w_j \le b_j \le 1, \tag{18}$$

where the upper bounds $\{b_j\}$ are either known *a priori* or can be efficiently computed. We will provide concrete examples of such upper bounds in Section IV-B. For now, we assume that the bounds $\{b_j\}$ have already been obtained.

Using (18) and noting that $0 \le x_j, z \le 1$, we can see that the parameters $\{\alpha_j, \beta_j\}$ in (15) are bounded by

$$|\alpha_j| \le b_j(1+\varepsilon) \quad \text{and} \quad |\beta_j| \le b_j(1+\varepsilon),$$

respectively. Meanwhile, $0 \le \mu_B \le 1$. It then follows from (15) that

$$\begin{aligned}\Pr\left[|Z(\boldsymbol{p}) - z| > \varepsilon\right] &\le \exp\{-n\xi\} \\ &+ 2\exp\left\{\frac{-n(\mu_B\varepsilon)^2/(1+\varepsilon)}{2\left(\frac{1+\varepsilon}{n}\sum_{j=1}^n b_j^2\left(\frac{1-p_j}{p_j}\right) + \varepsilon M(\boldsymbol{p})\right)}\right\},\end{aligned} \tag{19}$$

where $M(\boldsymbol{p}) \overset{\text{def}}{=} \max\limits_{1\le j\le n}\left(b_j \max\left\{1, \frac{1-p_j}{p_j}\right\}\right)/3$.

Given a set of parameters ξ, ε and $\{b_j\}$, we seek sampling patterns $\boldsymbol{p}$ to minimize the probability bound in (19), so that the random MCNLM estimate $Z(\boldsymbol{p})$ will be tightly concentrated around the full NLM result z. Equivalently, we solve the following optimization problem.

$$(P_0): \quad \begin{array}{ll} \underset{\boldsymbol{p}}{\arg\min} & \frac{1+\varepsilon}{n}\sum_{j=1}^n b_j^2\left(\frac{1-p_j}{p_j}\right) + \varepsilon M(\boldsymbol{p}) \\ \text{subject to} & \frac{1}{n}\sum\limits_{j=1}^n p_j = \xi \text{ and } 0 < p_j \le 1. \end{array} \tag{20}$$

In general, (P_0) does not have a closed-form solution. Because of this, and since $\varepsilon \ll 1$, we omit the term $\varepsilon M(\boldsymbol{p})$ in (20) and consider a simpler problem:

$$(P_1): \quad \begin{array}{ll} \underset{\boldsymbol{p}}{\arg\min} & \sum\limits_{j=1}^n b_j^2\left(\frac{1-p_j}{p_j}\right) \\ \text{subject to} & \frac{1}{n}\sum\limits_{j=1}^n p_j = \xi \text{ and } 0 < p_j \le 1. \end{array} \tag{21}$$

Before providing a justification for using (P_1) instead of (P_0), we first note that (P_1) is a variation of the the classical water-filling optimization problem [40], for which simple closed-form solutions exist. In particular, we derive in Appendix D the following solution to (P_1).

Theorem 2 (Optimal Sampling Patterns): The solution to (P_1) is given by

$$p_j = \min\{b_j\tau, 1\}, \quad \text{for } 1 \le j \le n, \tag{22}$$

where the parameter τ is chosen so that $\sum_j p_j = n\xi$.

Remark 3: It is easy to verify that

$$g(x) = \sum_{j=1}^n \min\{b_j x, 1\} - n\xi \tag{23}$$

is a piecewise linear and monotonically increasing function. Moreover, $g(0) = -n\xi < 0$ and $g(+\infty) = n(1-\xi) > 0$. Thus, the parameter τ can be uniquely determined as the root of $g(x)$.

Given the closed-form solution in (22), we are now ready to quantify the difference between the original problem (P_0) and the simplified version (P_1).

Proposition 3: Let $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ be the solution to (P_0) and (P_1), respectively. Then

$$c(\boldsymbol{p}_0) \le c(\boldsymbol{p}_1) \le c(\boldsymbol{p}_0) + \mathcal{O}(\varepsilon), \tag{24}$$

where $c(\boldsymbol{p}) \overset{\text{def}}{=} \frac{1+\varepsilon}{n}\sum_{j=1}^n b_j^2\left(\frac{1-p_j}{p_j}\right) + \varepsilon M(\boldsymbol{p})$ is the cost function in (P_0).

The above result, shown in Appendix E, justifies the use of the simpler problem (P_1) in place of (P_0). Indeed, for $\varepsilon \ll 1$, the inequalities in (24) guarantee that the performance of the sampling pattern $\boldsymbol{p}_1$ obtained by solving (P_1) will be similar to that obtained by solving (P_0).

B. Optimal Sampling Patterns

To construct the optimal sampling pattern prescribed by Theorem 2, we need to find $\{b_j\}$, which are the upper bounds on the true similarity weights $\{w_j\}$. At one extreme, the tightest upper bounds are $b_j = w_j$, but this oracle scheme is not realistic as it requires that we know all the weights $\{w_j\}$. At the other extreme, we can use the trivial upper bound $b_j = 1$. It is easy to verify that, under this setting, the sampling pattern in (22) becomes the uniform pattern, *i.e.*, $p_j = \xi$ for all j. In what follows, we present two choices for the upper bounds that can be efficiently computed and that can utilize partial knowledge of w_j.

1) Bounds from spatial information: The first upper bound is designed for internal (*i.e.*, single image) denoising where there is often a spatial term in the similarity weight, *i.e.*,

$$w_j = w_j^s \, w_j^r. \tag{25}$$

One example of the spatial weight can be found in (4). Since $w_j^r \leq 1$, we always have $w_j \leq w_j^s$. Thus, a possible choice is to set

$$b_j^s = w_j^s. \tag{26}$$

The advantage of the above upper bound is that b_j^s is a function of the spatial distance $d_{i,j}$ between a pair of pixels, which is independent of the image data $\mathcal{X}$ and $\mathcal{Y}$. Therefore, it can be pre-computed before running the MCNLM algorithm. Moreover, since $\{b_j\}$ is *spatially invariant*, they can be reused at all pixel locations.

2) Bounds from intensity information: For external image denoising, the patches in $\mathcal{X}$ and $\mathcal{Y}$ do not have any spatial relationship, as they can come from different images. In this case, the similarity weight w_j is only due to the difference in pixel intensities (*i.e.*, $w_j = w_j^r$), and thus we cannot use the spatial bounds given in (26). To derive a new bound for this case, we first recall the Cauchy-Schwartz inequality: For any two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ and for any positive-definite weight matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{d \times d}$, it holds that

$$|\boldsymbol{u}^T \boldsymbol{\Lambda} \boldsymbol{v}| \leq \|\boldsymbol{u}\|_{\boldsymbol{\Lambda}} \, \|\boldsymbol{v}\|_{\boldsymbol{\Lambda}}.$$

Setting $\boldsymbol{u} = \boldsymbol{y} - \boldsymbol{x}_j$, we then have

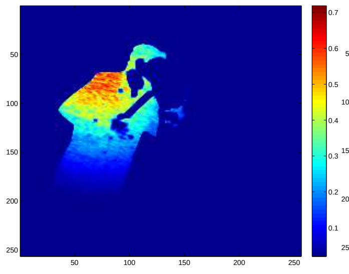
$$\begin{aligned} w_j = e^{-\|\boldsymbol{y}-\boldsymbol{x}_j\|_{\boldsymbol{\Lambda}}^2/(2h_r^2)} &\leq e^{-\left((\boldsymbol{x}_j-\boldsymbol{y})^T\boldsymbol{\Lambda}\boldsymbol{v}\right)^2/\left(2h_r^2\|\boldsymbol{v}\|_{\boldsymbol{\Lambda}}^2\right)} \\ &\leq e^{-\left(\boldsymbol{x}_j^T\boldsymbol{s}-\boldsymbol{y}^T\boldsymbol{s}\right)^2} = b_j^r, \end{aligned} \tag{27}$$

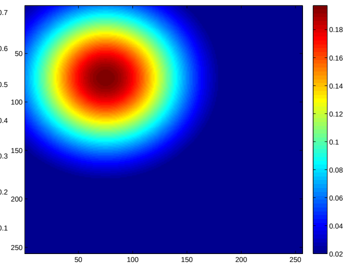
where $\boldsymbol{s} \overset{\text{def}}{=} \boldsymbol{\Lambda}\boldsymbol{v}/\left(\sqrt{2}h_r\|\boldsymbol{v}\|_{\boldsymbol{\Lambda}}\right)$. The vector $\boldsymbol{v}$ can be any nonzero vector. In practice, we choose $\boldsymbol{v} = \boldsymbol{1}$ with $\boldsymbol{\Lambda} = \operatorname{diag}\{1/d, \ldots, 1/d\}$ and we find this choice effective in our numerical experiments. In this case, $b_j^r = \exp\left\{-(\boldsymbol{x}_j^T\boldsymbol{1} - \boldsymbol{y}^T\boldsymbol{1})^2/(2d^2h_r^2)\right\}$.

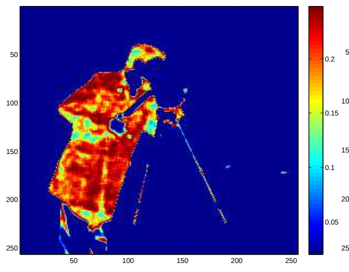
Remark 4: To obtain the upper bound b_j^r in (27), we need to compute the terms $\boldsymbol{y}^T\boldsymbol{s}$ and $\boldsymbol{x}_j^T\boldsymbol{s}$, which are the projections of the vectors $\boldsymbol{y}$ and $\boldsymbol{x}_j$ onto the one-dimensional space spanned by $\boldsymbol{s}$. These projections can be efficiently computed by convolving the noisy image and the images in the reference set with a spatially-limited kernel corresponding to $\boldsymbol{s}$. To further reduce the computational complexity, we also adopt a two-stage importance sampling procedure in our implementation, which allows us to avoid the computation of the exact values of $\{b_j\}$ at most pixels. Details of our implementation are given in a supplementary technical report [41].

Example 3: To demonstrate the performance of the various sampling patterns presented above, we consider denoising one pixel of the *Cameraman* image as shown in Figure 6(a). The similarity weights are in the form of (25), consisting of both a spatial and a radiance-related part. Applying the result of Theorem 2, we derive four optimal sampling patterns, each associated with a different choice of the upper bound, namely, $b_j = w_j, b_j = b_j^s, b_j = b_j^r$, and $b_j = b_j^s b_j^r$. Note that the first choice corresponds to an *oracle* setting, where we assume that the weights $\{w_j\}$ are known. The latter three are practically achievable sampling patterns, where b_j^s and b_j^r are defined in (26) and (27), respectively.

(a) Target pixel to be denoised


(b) Oracle sampling pattern


(c) Spatial


(d) Intensity

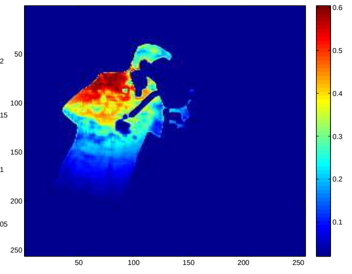

(e) Spatial + Intensity

Fig. 6: Illustration of optimal sampling probability for the case $h_r = 15/255$, $h_s = 50$. (a) Cameraman image and the target pixel. We overlay the spatial weight on top of *cameraman* for visualization. (b) Optimal sampling pattern w.r.t. w_j (oracle scheme). (c) Spatial upper bound b_j^s. (d) Intensity upper bound b_j^r. (e) Spatial and intensity upper bound $b_j^s \cdot b_j^r$.

Figure 6(b)–(e) show the resulting sampling patterns. As can be seen in the figures, various aspects of the oracle sampling pattern are reflected in the approximated patterns. For instance, the spatial approximation has more emphasis at the center than the peripherals whereas the intensity approximation has more emphasis on pixels that are similar to the target pixel.

To compare these sampling patterns quantitatively, we plot in Figure 7 the reconstruction MSE associated with different patterns as functions of the average sampling ratio ξ. Here, we set $h_r = 15/255$ and $h_s = 50$. For benchmark, we also show the performance of the uniform sampling pattern. It is clear from the figure that all the optimal sampling patterns outperform the uniform pattern. In particular, the pattern obtained by incorporating both the spatial and intensity information approaches the performance of the oracle scheme.

V. Experimental Results

In this section we present additional numerical experiments to evaluate the performance of the MCNLM algorithm and compare it with several other accelerated NLM algorithms.

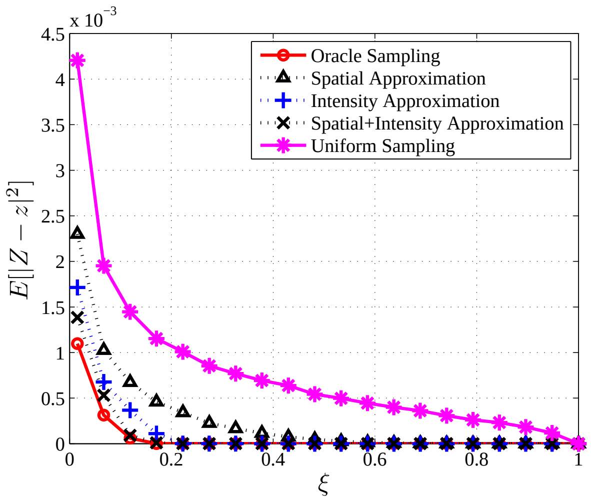


Fig. 7: Denoising results of using different sampling schemes shown in Figure 6. Setting of experiment: noise $\sigma = 15/255$, $h_s = 50$, $h_r = 15/255$, patch size 5×5.

A. Internal Denoising

A benchmark of ten standard test images are used for this experiment. For each image, we add zero-mean Gaussian noise with standard deviations equal to $\sigma = \frac{10}{255}, \frac{20}{255}, \frac{30}{255}, \frac{40}{255}, \frac{50}{255}$ to simulate noisy images at different PSNR levels. Two choices of the spatial search window size are used: 21×21 and 35×35, following the original configurations used in [1].

The parameters of MCNLM are as follows: The patch size is 5×5 (*i.e.*, $d = 25$) and $\boldsymbol{\Lambda} = \boldsymbol{I}/d$. For each choice of the spatial search window size (*i.e.*, $\rho = 21$ or $\rho = 35$), we define $h_s = (\lfloor \rho/2 \rfloor)/3$ so that three standard deviations of the spatial Gaussian will be inside the spatial search window. The intensity parameter is set to $h_r = 1.3\sigma/255$.

In this experiment, we use the spatial information bound (26) to compute the optimal sampling pattern in (22). Incorporating additional intensity information as in (27) would further improve the performance, but we choose not to do so because the PSNR gains are found to be moderate in this case due to the relatively small size of the spatial search window. Five average sampling ratios, $\xi = 0.05, 0.1, 0.2, 0.5, 1$, are evaluated. We note that when $\xi = 1$, MCNLM is identical to the full NLM.

For comparisons, we test the Gaussian KD tree (GKD) algorithm [26] with a C++ implementation (ImageStack [42]) and the adaptive manifold (AM) algorithm [27] with a MATLAB implementation provided by the authors. To create a meaningful common ground for comparison, we adapt MCNLM as follows: First, since both GKD and AM use SVD projection [18] to reduce the dimensionality of patches, we also use in MCNLM the same SVD projection method by computing the 10 leading singular values. The implementation of this SVD step is performed using an off-the-shelf MATLAB code [43]. We also tune the major parameters of GKD and AM for their best performance, *e.g.*, for GKD we set $h_r = 1.3\sigma/255$ and for AM we set $h_r = 2\sigma/255$. Other parameters are kept at their default values as reported in [26], [27].

Table I and Table II summarize the results of the experiment. Additional results, with visual comparison of denoised images, can be found in the supplementary technical report [41]. Since MCNLM is a randomized algorithm, we report the average PSNR values of MCNLM over 24 independent runs. The results show that for the 10 images, even at a very low sampling ratio, *e.g.*, $\xi = 0.1$, the averaged performance of MCNLM (over 10 testing images) is only about 0.35 dB to 0.7 dB away (depending on σ) from the full NLM solution. When the sampling ratio is further increased to $\xi = 0.2$, the PSNR values become very close (about a 0.09 dB to 0.2 dB drop depending on σ) to those of the full solution.

In Table III we report the runtime of MCNLM, GKD and AM. Since the three algorithms are implemented in different environments, namely, MCNLM in MATLAB/C++ (.mex), GKD in C++ with optimized library and data-structures, and AM in MATLAB (.m), we caution that Table III is only meant to provide some rough references on computational times. For MCNLM, its speed improvement over the full NLM can be reliably estimated by the average sampling ratio ξ.

B. External Dictionary-based Image Denoising

To test MCNLM for external dictionary-based image denoising, we consider the dataset of Levin and Nadler [11], which contains about 15,000 training images (about $n \approx 10^{10}$ image patches) from the LabelMe dataset [44]. For testing, we use a separate set of 2000 noisy patches, which are mutually exclusive from the training images. The results are shown in Figure 8.

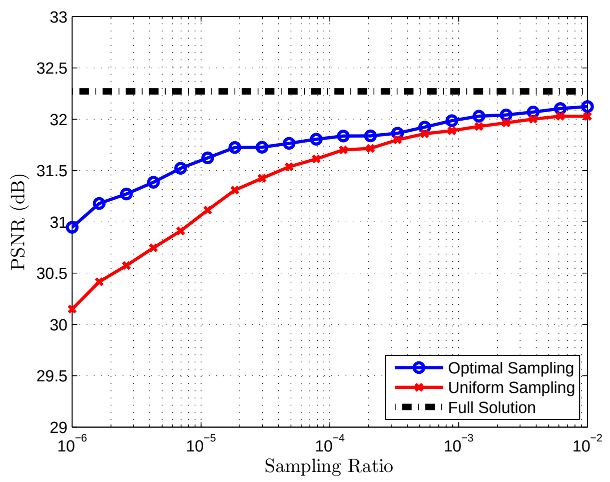


Fig. 8: External denoising using MCNLM. The external dataset contains $n = 10^{10}$ patches. 2000 testing patches are used to compute the PSNR. The "dotted" line indicates the full NLM result reported in [11]. The "crossed" line indicates the MCNLM result using uniform sampling pattern, and the "circled" line indicates the MCNLM result using the intensity approximated sampling pattern.

Due to the massive size of the reference set, full evaluation of (1) requires about one week on a 100-CPU cluster, as reported in [11]. To demonstrate how MCNLM can be used

TABLE I: Single image denoising by MCNLM, using the optimal Gaussian sampling pattern. The case when $\xi = 1$ is equivalent to the standard NLM [2]. GKD refers to [26]. AM refers to [27]. Shown in the table are PSNR values (in dB).

ξ	0.05	0.1	0.2	0.5	1	GKD	AM	0.05	0.1	0.2	0.5	1	GKD	AM
σ	*Baboon* 512×512							*Barbara* 512×512						
10	30.74	31.14	31.57	31.63	31.63	31.17	28.91	32.14	32.65	33.04	33.20	33.20	32.72	30.48
20	26.73	27.08	27.22	27.29	27.29	26.67	25.78	28.19	28.66	29.06	29.22	29.23	28.40	26.88
30	24.43	24.77	24.97	25.08	25.08	24.58	24.27	25.84	26.38	26.68	26.85	26.85	25.91	24.92
40	23.24	23.56	23.8	23.92	23.93	23.28	23.38	24.13	24.71	25.06	25.23	25.24	24.29	23.75
50	22.05	22.63	22.99	23.12	23.12	22.24	22.66	22.77	23.49	23.91	24.07	24.07	23.08	22.95
σ	*Boat* 512×512							*Bridge* 512×512						
10	32.20	32.62	32.85	32.93	32.93	32.54	30.92	29.52	29.27	29.05	29.08	29.08	29.64	28.49
20	28.56	29.26	29.57	29.70	29.70	28.69	27.99	25.46	25.36	25.36	25.41	25.41	25.72	25.40
30	26.38	27.15	27.49	27.67	27.66	26.47	26.11	23.64	23.74	23.86	23.92	23.92	23.85	23.74
40	24.88	25.69	26.02	26.21	26.21	24.90	24.83	22.28	22.66	22.83	22.92	22.92	22.67	22.65
50	23.46	24.33	24.85	25.05	25.05	23.66	23.86	21.21	21.67	21.90	22.00	22.01	21.73	21.82
σ	*Couple* 512×512							*Hill* 256×256						
10	31.98	32.40	32.62	32.72	32.72	32.40	30.84	30.54	30.51	30.44	30.49	30.49	30.89	30.12
20	28.21	28.55	28.78	28.88	28.88	28.19	27.57	26.95	27.14	27.17	27.26	27.26	27.14	26.95
30	25.80	26.42	26.69	26.81	26.81	25.97	25.76	25.13	25.46	25.64	25.75	25.75	25.28	25.34
40	24.21	25.03	25.38	25.54	25.54	24.52	24.56	23.76	24.38	24.66	24.81	24.82	24.14	24.37
50	23.14	23.93	24.42	24.59	24.59	23.39	23.72	22.83	23.46	23.87	24.02	24.02	23.10	23.57
σ	*House* 256×256							*Lena* 512×512						
10	34.07	34.76	35.36	35.50	35.50	34.51	33.07	34.69	35.48	36.02	36.15	36.15	34.90	34.04
20	30.52	31.45	32.33	32.53	32.54	30.41	29.62	30.82	31.88	32.44	32.64	32.65	30.91	30.47
30	27.89	29.06	29.81	30.04	30.04	27.81	27.27	28.49	29.64	30.23	30.50	30.50	28.52	28.44
40	26.12	27.21	27.92	28.20	28.20	26.04	25.70	26.53	27.73	28.47	28.76	28.76	26.72	26.97
50	24.70	25.86	26.66	26.98	26.99	24.76	24.79	25.16	26.29	27.03	27.34	27.35	25.29	25.81
σ	*Man* 512×512							*Pepper* 512×512						
10	32.32	32.58	32.72	32.79	32.79	32.56	31.49	32.78	33.49	33.85	33.94	33.95	33.35	31.63
20	28.56	29.13	29.37	29.49	29.49	28.78	28.37	28.99	29.87	30.32	30.46	30.46	29.38	28.43
30	26.68	27.26	27.69	27.85	27.85	26.74	26.62	26.48	27.33	27.82	27.98	27.98	26.81	25.88
40	25.12	25.94	26.45	26.65	26.66	25.27	25.39	24.68	25.61	26.09	26.29	26.29	25.01	24.25
50	23.91	24.92	25.48	25.71	25.72	24.13	24.52	23.24	23.91	24.41	24.61	24.61	23.49	23.00

TABLE II: Average PSNR over 10 testing images. Bold values are the minimum PSNRs that surpass GKD and AM.

σ	0.05	0.1	0.2	0.5	1	GKD	AM
10	32.10	**32.49**	32.75	32.84	32.84	32.47	31.00
20	28.30	**28.84**	29.16	29.29	29.29	28.43	27.75
30	26.08	**26.72**	27.09	27.25	27.24	26.19	25.84
40	24.50	**25.25**	25.67	25.85	25.86	24.68	24.59
50	23.25	**24.05**	24.55	24.75	24.75	23.49	23.67

TABLE III: Runtime (in seconds) of MCNLM, GKD and AM. Implementations: MCNLM: MATLAB/C++ (.mex) on Windows 7, GKD: C++ on Windows 7, AM: MATLAB on Windows 7.

Image Size	Search Window / Patch Size / PCA dimension	0.05	0.1	0.2	0.5	1	GKD	AM
512×512	21×21 / 5×5 / 10	0.495	0.731	1.547	3.505	7.234	3.627	0.543
(*Man*)	35×35 / 9×9 / 10	1.003	1.917	3.844	9.471	19.904	4.948	0.546
256×256	21×21 / 5×5 / 10	0.121	0.182	0.381	0.857	1.795	0.903	0.242
(*House*)	35×35 / 9×9 / 10	0.248	0.475	0.954	2.362	4.851	1.447	0.244

to speed up the computation, we repeat the same experiment on a 12-CPU cluster. The testing conditions of the experiment are identical to those in [11]. Each of the 2000 test patches is corrupted by i.i.d. Gaussian noise of standard deviation $\sigma = 18/255$. Patch size is fixed at 5×5. The weight matrix is $\boldsymbol{\Lambda} = \boldsymbol{I}$. We consider a range of sampling ratios, from $\xi = 10^{-6}$ to $\xi = 10^{-2}$. For each sampling ratio, 12 independent trials are performed and their average is recorded. Here, we show the results of the uniform sampling pattern and the optimal sampling pattern obtained using the upper bound in (27). The results in Figure 8 indicate that MCNLM achieves a PSNR within 0.2dB of the full computation at a sampling ratio of 10^{-3}, a speed-up of about 1000-fold.

VI. Conclusion

We proposed Monte Carlo non-local means (MCNLM), a randomized algorithm for large-scale patch-based image filtering. MCNLM randomly chooses a fraction of the similarity weights to generate an approximated result. At any fixed sampling ratio, the probability of having large approximation errors decays exponentially with the problem size, implying that the approximated solution of MCNLM is tightly concentrated around its limiting value. Additionally, our analysis allows deriving optimized sampling patterns that exploit partial knowledge of weights of the types that are readily available in both internal and external denoising applications. Experimentally, MCNLM is competitive with other state-of-the-art accelerated NLM algorithms for single-image denoising in standard tests. When denoising with a large external database of images, MCNLM returns an approximation close to the full solution with speed-up of three orders of magnitude, suggesting its utility for large-scale image processing.

Acknowledgement

The authors thank Anat Levin for sharing the experimental settings and datasets reported in [11].

Appendix

A. Proof of Theorem 1

For notation simplicity, we shall drop the argument $\boldsymbol{p}$ in $A(\boldsymbol{p}), B(\boldsymbol{p})$ and $Z(\boldsymbol{p})$, since the sampling pattern $\boldsymbol{p}$ remains fixed in our proof. We also define $A/B = 1$ for the case when $B = 0$. We observe that

$$\begin{aligned}\Pr\left[|Z - z| > \varepsilon\right] &= \Pr\left[|A/B - z| > \varepsilon\right] \\ &= \Pr\left[|A/B - \mu| > \varepsilon \,\cap\, B = 0\right] \\ &\qquad + \Pr\left[|A/B - \mu| > \varepsilon \,\cap\, B > 0\right] \\ &\le \Pr\left[B = 0\right] \\ &\qquad + \Pr\left[|A - \mu B| > \varepsilon B \,\cap\, B > 0\right] \\ &\le \Pr\left[B = 0\right] + \Pr\left[|A - \mu B| > \varepsilon B\right]. \quad (28)\end{aligned}$$

By assumption, $w_j > 0$ for all j. It then follows from the definition in (8) that $B = 0$ if and only if $I_j = 0$ for all j. Thus,

$$\begin{aligned}\Pr[B = 0] &= \prod_{j=1}^{n}(1 - p_j) = \exp\left\{\sum_{j=1}^{n}\log(1 - p_j)\right\} \\ &\overset{(b1)}{\le} \exp\left\{-\sum_{j=1}^{n} p_j\right\} \overset{(b2)}{=} \exp\left\{-n\xi\right\}. \quad (29)\end{aligned}$$

Here, $(b1)$ holds because $\log(1-p) \le -p$ for $0 \le p \le 1$; and $(b2)$ is due to the definition that $\xi = \frac{1}{n}\sum_{j=1}^{n} p_j$.

Next, we provide an upper bound for $\Pr\left[|A - zB| > \varepsilon B\right]$ in (28) by considering the two tail probabilities $\Pr\left[A - zB > \varepsilon B\right]$ and $\Pr\left[A - zB < -\varepsilon B\right]$ separately. Our goal here is to rewrite the two inequalities so that Bernstein's inequality in Lemma 1 can be applied. To this end, we define

$$\alpha_j \overset{\text{def}}{=} w_j(x_j - z - \varepsilon) \quad \text{and} \quad Y_j \overset{\text{def}}{=} \alpha_j\left(\frac{I_j}{p_j} - 1\right).$$

We note that $z = \mu_A/\mu_B$, where μ_A and μ_B are defined in (9) and (10), respectively. It is easy to verify that

$$\begin{aligned}\Pr\left[A - zB > \varepsilon B\right] &= \Pr\left[\frac{1}{n}\sum_{j=1}^{n} Y_j > -\frac{1}{n}\sum_{j=1}^{n}\alpha_j\right] \\ &= \Pr\left[\frac{1}{n}\sum_{j=1}^{n} Y_j > \varepsilon\mu_B\right]. \quad (30)\end{aligned}$$

The random variables Y_j are of zero-mean, with variance

$$\mathrm{Var}\left[Y_j\right] = \frac{\alpha_j^2}{p_j^2}\mathrm{Var}[I_j] = \alpha_j^2\frac{1 - p_j}{p_j}.$$

Using Bernstein's inequality in Lemma 1, we can then bound the probability in (30) as

$$\begin{aligned}&\Pr\left[A - zB > \varepsilon B\right] \\ &\le \exp\left\{\frac{-n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^{n}\alpha_j^2\left(\frac{1-p_j}{p_j}\right) + M_\alpha(\mu_B\varepsilon)/3\right)}\right\}, \quad (31)\end{aligned}$$

where the constant M_α can be determined as follows:

$$\begin{aligned}|Y_j| &= \begin{cases}|\alpha_j|\left(\frac{1-p_j}{p_j}\right), & \text{if } I_j = 1\\ |\alpha_j|, & \text{if } I_j = 0.\end{cases} \\ &\le |\alpha_j|\max\left\{1, \frac{1 - p_j}{p_j}\right\}.\end{aligned}$$

Thus, $M_\alpha = \max\limits_{1\le j\le n}\left(|\alpha_j|\max\left\{1, \frac{1-p_j}{p_j}\right\}\right)$.

The other tail probability, *i.e.*, $\Pr\left[A - zB < -\varepsilon B\right]$, can be bounded similarly. In this case, we let

$$\beta_j \overset{\text{def}}{=} w_j(x_j - z + \varepsilon) \quad \text{and} \quad \widetilde{Y}_j \overset{\text{def}}{=} -\beta_j\left(\frac{I_j}{p_j} - 1\right).$$

Then, following the same derivations as above, we can show that

$$\begin{aligned}&\Pr\left[A - zB < -\varepsilon B\right] \\ &\le \exp\left\{\frac{-n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^{n}\beta_j^2\left(\frac{1-p_j}{p_j}\right) + M_\beta(\mu_B\varepsilon)/3\right)}\right\}, \quad (32)\end{aligned}$$

where $M_\beta = \max\limits_{1\le j\le n}\left(|\beta_j|\max\left\{1, \frac{1-p_j}{p_j}\right\}\right)$. Substituting (29), (31) and (32) into (28), we are done.

B. Proof of Proposition 1

The goal of the proof is to simplify (15) by utilizing the fact that $0 \le x_j \le 1$, $0 \le z \le 1$, $0 < w_j \le 1$ and $\boldsymbol{p} = \xi\mathbf{1}$. To this end, we first observe the following:

$$|\alpha_j| = w_j|x_j - z - \varepsilon| \le w_j(1 + \varepsilon).$$

Consequently, for $0 < \xi \le 1/2$, M_α is bounded as

$$M_\alpha = \max_{1\le j\le n}\left(|\alpha_j|\max\left\{1, \frac{1-\xi}{\xi}\right\}\right) \overset{(a)}{\le} (1 + \varepsilon)\left(\frac{1-\xi}{\xi}\right),$$

where in (a) we used the fact that $w_j \leq 1$ and $\xi \leq 1/2$. Similarly,

$$|\beta_j| \leq w_j(1+\varepsilon) \quad \text{and} \quad M_\beta \leq (1+\varepsilon)\left(\frac{1-\xi}{\xi}\right).$$

Therefore, the two negative exponents in (15) are lower bounded by the following common quantity:

$$\begin{aligned}
&\frac{n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^n w_j^2(1+\varepsilon)^2\left(\frac{1-\xi}{\xi}\right)+\mu_B\varepsilon(1+\varepsilon)\left(\frac{1-\xi}{\xi}\right)/3\right)}\\
&\overset{(b)}{\geq} \frac{n(\mu_B\varepsilon)^2}{2\left(\frac{1}{n}\sum_{j=1}^n w_j(1+\varepsilon)^2+\mu_B\varepsilon(1+\varepsilon)/3\right)}\left(\frac{\xi}{1-\xi}\right)\\
&= \frac{n(\mu_B\varepsilon)^2}{2\left(\mu_B(1+\varepsilon)^2+\mu_B\varepsilon(1+\varepsilon)/3\right)}\left(\frac{\xi}{1-\xi}\right)\\
&\geq \frac{n\mu_B\varepsilon^2}{2(1+\varepsilon)(1+4\varepsilon/3)}\left(\frac{\xi}{1-\xi}\right).
\end{aligned}$$

Defining $f(\varepsilon) \overset{\text{def}}{=} \varepsilon^2/(2(1+\varepsilon)(1+4\varepsilon/3))$ yields the desired result.

C. Proof of Proposition 2

The MSE can be computed as

$$\begin{aligned}
\mathbb{E}\left[(Z(\boldsymbol{p})-z)^2\right] &\overset{(a)}{=} \int_0^\infty \Pr\left[(Z(\boldsymbol{p})-z)^2 > \varepsilon\right] d\varepsilon\\
&\overset{(b)}{=} \int_0^1 \Pr\left[(Z(\boldsymbol{p})-z)^2 > \varepsilon\right] d\varepsilon,
\end{aligned}$$

where (a) is due to the "layer representation" of the expectations (See, *e.g.*, [45, Chapter 5.6]), and (b) is due to the fact that $|Z(\boldsymbol{p})-z| \leq 1$. Then, by (16), we have that

$$\begin{aligned}
&\int_0^1 \Pr\left[(Z(\boldsymbol{p})-z)^2 > \varepsilon\right] d\varepsilon\\
&\qquad \leq e^{-n\xi} + 2\int_0^1 \exp\left\{-n\mu_B f(\sqrt{\varepsilon})\left(\frac{\xi}{1-\xi}\right)\right\} d\varepsilon.
\end{aligned}$$

By the definition of $f(\varepsilon)$, it is easy to verify $f(\sqrt{\varepsilon}) \geq 3\varepsilon/28$. Thus,

$$\begin{aligned}
&\int_0^1 \Pr\left[(Z(\boldsymbol{p})-z)^2 > \varepsilon\right] d\varepsilon\\
&\qquad \leq e^{-n\xi} + 2\int_0^1 \exp\left\{-n\mu_B\,(3\varepsilon/28)\left(\frac{\xi}{1-\xi}\right)\right\} d\varepsilon\\
&\qquad \leq e^{-n\xi} + \frac{1}{n}\left(\frac{56}{3\mu_B}\right)\left(\frac{1-\xi}{\xi}\right).
\end{aligned}$$

D. Proof of Theorem 2

Removing constant terms in the objective in (21) that are independent of $\boldsymbol{p}$, we can simplify (P_1) as

$$\underset{p_1,\ldots,p_n}{\text{minimize}} \quad \sum_{j=1}^n \frac{b_j^2}{p_j} \quad \text{s.t.} \quad \frac{1}{n}\sum_{j=1}^n p_j = \xi, \quad \text{and} \quad p_j \leq 1.$$

Here, we have ignored the constraints that $p_j \geq 0$ for all j, but we will verify that these positivity constraints are indeed satisfied by the solution we get.

The Lagrangian of the above problem is

$$\mathcal{L}(\boldsymbol{p},\boldsymbol{\lambda},\nu) = \sum_{j=1}^n \frac{b_j^2}{p_j} + \sum_{j=1}^n \lambda_j(p_j-1) + \nu\left(\sum_{j=1}^n p_j - n\xi\right), \tag{33}$$

where $\boldsymbol{p} = [p_1,\ldots,p_n]^T$ are the primal variables, $\boldsymbol{\lambda} = [\lambda_1,\ldots,\lambda_n]^T$ are the Lagrange multipliers associated with the constraints $p_j \leq 1$, and ν is the Lagrange multiplier associated with the constraint $\sum_{j=1}^n p_j = n\xi$.

The first order optimality conditions imply the following:

- *Stationarity*: $\nabla_{\boldsymbol{p}}\,\mathcal{L} = 0$. That is, $-\frac{b_j^2}{p_j^2} + \lambda_j + \nu = 0$.
- *Primal feasibility*: $\sum_{j=1}^n p_j = n\xi$ and $p_j \leq 1$.
- *Dual feasibility*: $\lambda_j \geq 0$, and $\nu \geq 0$.
- *Complementary slackness*: $\lambda_j(p_j-1) = 0$.

The last condition, complementary slackness, implies that, for each j, one of the following cases always holds: $\lambda_j = 0$ or $p_j = 1$.

Case 1: $\lambda_j = 0$. Substituting this into the stationarity condition yields $p_j = b_j/\sqrt{\nu}$. Since $p_j \leq 1$, we must have $b_j \leq \sqrt{\nu}$.

Case 2: $p_j = 1$. In this case, the stationarity implies that $b_j^2 = \lambda_j + \nu = 0$. Since $\lambda_j \geq 0$, we must have $b_j > \sqrt{\nu}$.

Combining these two cases, we obtain

$$p_j = \begin{cases} \frac{b_j}{\sqrt{\nu}}, & \text{if} \quad b_j \leq \sqrt{\nu},\\ 1, & \text{if} \quad b_j > \sqrt{\nu}.\end{cases}$$

Thus, it remains to determine ν. This can be done by using the primal feasibility condition that $\frac{1}{n}\sum_{j=1}^n p_j = \xi$. In particular, consider the function $g(x)$ defined in (23), where $x = 1/\sqrt{\nu}$. The desired value of ν can thus be obtained by finding the root of the equation $g(x)$. Since $g(x)$ is a monotonically increasing piecewise-linear function, the parameter ν is uniquely determined, so is $\boldsymbol{p}$.

E. Proof of Propositon 3

It follows from the definition of $\boldsymbol{p}_0$ that $c(\boldsymbol{p}_0) \leq c(\boldsymbol{p}_1)$. So we just need to establish the second inequality in (24). We note that, since $\boldsymbol{p}_1$ is the optimal solution to (P_1), it holds that

$$c(\boldsymbol{p}_1) - \varepsilon M(\boldsymbol{p}_1) \leq c(\boldsymbol{p}_0) - \varepsilon M(\boldsymbol{p}_0),$$

and thus

$$c(\boldsymbol{p}_1) \leq c(\boldsymbol{p}_0) + \varepsilon M(\boldsymbol{p}_1) - \varepsilon M(\boldsymbol{p}_0) \leq c(\boldsymbol{p}_0) + \varepsilon M(\boldsymbol{p}_1),$$

where the second inequality is due to the fact that $M(\boldsymbol{p}_0) \geq 0$.

Next, we provide an upper bound for

$$\varepsilon M(\boldsymbol{p}_1) = \max_{1\leq j\leq n}\left(b_j \max\left\{1, \frac{1-p_j}{p_j}\right\}\right)\varepsilon/3, \tag{34}$$

where $p_j = \min(b_j\tau, 1)$ as given in (22). For each j, if $b_j\tau \geq 1/2$, then $p_j \geq 1/2$ and

$$b_j \max\{1, 1/p_j - 1\} = b_j \leq 1. \tag{35}$$

For the other case, when $p_j = b_j\tau < 1/2$, we have

$$b_j \max\{1, 1/p_j - 1\} = 1/\tau - b_j \leq 1/\tau. \tag{36}$$

Substituting (35) and (36) into (34), we get

$$\varepsilon M(\boldsymbol{p}_1) \leq \varepsilon \max\{1, 1/\tau\}/3. \tag{37}$$

Finally, to obtain an estimate of $1/\tau$, we note that τ is the unique root of the function $g(x)$ defined in (23). It is easy to verify that

$$g\left(\frac{n\xi}{\sum_j b_j}\right) \leq \frac{n\xi}{\sum_j b_j}\sum_j b_j - n\xi = 0 = g(\tau).$$

Since $g(x)$ is a monotonically increasing function, we must have $\tau \geq n\xi/\sum_j b_j \geq \xi/\mu_B$. Combining this bound with (37), we get

$$\varepsilon M(\boldsymbol{p}_1) \leq \max(1, \mu_B/\xi)\,\varepsilon/3 = \mathcal{O}(\varepsilon).$$

REFERENCES

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

[2] A. Buades, B. Coll, and J. Morel, "Denoising image sequences does not require motion estimation," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance (AVSS)*, Sep. 2005, pp. 70–74.

[3] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the non-local-means to super-resolution reconstruction," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 36–51, Jan. 2009.

[4] 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 (ICCV)*, Oct. 2009, pp. 2272–2279.

[5] K. Chaudhury and A. Singer, "Non-local patch regression: Robust image denoising in patch space," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, 2013, available online at http://arxiv.org/abs/1211.4264.

[6] P. Milanfar, "A tour of modern image filtering," *IEEE Signal Processing Magazine*, vol. 30, pp. 106–128, Jan. 2013.

[7] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1620–1630, Apr. 2013.

[8] D. Van De Ville and M. Kocher, "SURE-based non-local means," *IEEE Signal Process. Lett.*, vol. 16, no. 11, pp. 973–976, Nov. 2009.

[9] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," *IEEE Trans. Image Process.*, vol. 15, no. 10, pp. 2866–2878, Oct. 2006.

[10] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 977–984.

[11] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 2833–2840.

[12] A. Levin, B. Nadler, F. Durand, and W. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," in *Proc. 12th European Conf. Computer Vision (ECCV)*, Oct. 2012, vol. 7576, pp. 73–86.

[13] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 839–842, Dec. 2005.

[14] P. Coupe, P. Yger, and C. Barillot, "Fast non local means denoising for 3D MR images," in *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2006, pp. 33–40.

[15] T. Brox, O. Kleinschmidt, and D. Cremers, "Efficient nonlocal means for denoising of textural patterns," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1083–1092, Jul. 2008.

[16] T. Tasdizen, "Principal components for non-local means image denoising," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2008, pp. 1728 –1731.

[17] J. Orchard, M. Ebrahimi, and A. Wong, "Efficient nonlocal-means denoising using the SVD," in *Proc. IEEE Int. Confe. Image Process. (ICIP)*, Oct. 2008, pp. 1732 –1735.

[18] D. Van De Ville and M. Kocher, "Nonlocal means with dimensionality reduction and SURE-based parameter selection," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2683 –2690, Sep. 2011.

[19] J. Darbon, A. Cunha, T. Chan, S. Osher, and G. Jensen, "Fast nonlocal filtering applied to electron cryomicroscopy," in *Proc. IEEE Int. Sym. Biomedical Imaging*, 2008, pp. 1331–1334.

[20] J. Wang, Y. Guo, Y. Ying, Y. Liu, and Q. Peng, "Fast non-local algorithm for image denoising," in *Proc. IEEE Int.Conf. Image Process. (ICIP)*, Oct. 2006, pp. 1429–1432.

[21] V. Karnati, M. Uliyar, and S. Dey, "Fast non-local algorithm for image denoising," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2009, pp. 3873–3876.

[22] R. Vignesh, B. Oh, and J. Kuo, "Fast non-local means (NLM) computation with probabilistic early termination," *IEEE Signal Process. Lett.*, vol. 17, no. 3, pp. 277–280, Mar. 2010.

[23] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Int. J. Computer Vision*, vol. 81, no. 1, pp. 24–52, Jan. 2009.

[24] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, "Improved fast Gauss transform and efficient kernel density estimation," in *Proc. Int. Conf. Computer Vision (ICCV)*, Oct. 2003, pp. 664–671.

[25] A. Adams, J. Baek, and M. A. Davis, "Fast high-dimensional filtering using the permutohedral lattice," in *Proc. EUROGRAPHICS*, 2010, vol. 29, pp. 753–762.

[26] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high-dimensional filtering," in *Proc. of ACM SIGGRAPH*, 2009, Article No. 21.

[27] E. Gastal and M. Oliveira, "Adaptive manifolds for real-time high-dimensional filtering," *ACM Trans. Graphics*, vol. 31, no. 4, pp. 33:1–33:13, 2012.

[28] H. Bhujle and S. Chaudhuri, "Novel speed-up strategies for non-local means denoising with patch and edge patch based dictionaries," *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 356–365, Jan. 2014.

[29] S. Arietta and J. Lawrence, "Building and using a database of one trillion natural-image patches," *IEEE Computer Graphics and Applications*, vol. 31, no. 1, pp. 9–19, Jan. 2011.

[30] F. Meyer and X. Shen, "Perturbation of the eigenvectors of the graph Laplacian: Application to image denoising," *Applied and Computational Harmonic Analysis*, 2013, In press. Available online at http://arxiv.org/abs/1202.6666.

[31] H. Talebi and P. Milanfar, "Global image denoising," *IEEE Trans. Image Process.*, 2014, In press.

[32] P. Drineas and M. W. Mahoney, "On the Nystrom method for approximating a Gram matrix for improved kernel-based learning," *J. Machine Learning Research*, vol. 6, pp. 2153–2175, 2005.

[33] A. Dembo and O. Zeitouni, *Large deviations techniques and applications*, Springer, Berlin, 2010.

[34] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes*, Oxford University Press, 3rd edition, 2001.

[35] S. Bernstein, *The Theory of Probabilities*, Gastehizdat Publishing House, Moscow, 1946.

[36] S. H. Chan, T. Zickler, and Y. M. Lu, "Fast non-local filtering by random sampling: It works, especially for large images," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, 2013.

[37] R. Serfling, "Probability inequalities for the sum in sampling without replacement," *The Annals of Statistics*, vol. 2, pp. 39–48, 1974.

[38] F. Chung and L. Lu, "Concentration inequalities and martingale inequalities: a survey," *Internet Mathematics*, vol. 3, no. 1, pp. 79–127, 2006.

[39] P. Drineas, R. Kannan, and M. Mahoney, "Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication," *SIAM J. Computing*, vol. 36, pp. 132–157, 2006.

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[41] S. H. Chan, T. Zickler, and Y. M. Lu, "Monte-Carlo non-local means: Random sampling for large-scale image filtering — Supplementary material," Tech. Rep., Harvard University, 2013, [Online] available at http://scholar.harvard.edu/stanleychan.

[42] A. Adams, "Imagestack," https://code.google.com/p/imagestack/.

[43] G. Peyre, "Non-local means MATLAB toolbox," http://www.mathworks.com/matlabcentral/fileexchange/13619.

[44] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: a database and web-based tool for image annotation," *Int. J. Computer Vision*, vol. 77, no. 1-3, pp. 157–173, May 2008.

[45] W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 2, John Wiley & Sons, 2nd edition, 1971.

Monte Carlo Non-Local Means: Random Sampling for Large-Scale Image Filtering (Supplementary Material)

Stanley H. Chan, *Member, IEEE*, Todd Zickler, *Member, IEEE*,
and Yue M. Lu, *Senior Member, IEEE*

Abstract

This supplementary document provides the following additional information of the main article.

- Implementation of Theorem 2 (General Sampling Case)
- Implementation of Uniform Sampling Patterns
- Implementation of Spatially Approximated Sampling Patterns (for Internal Denoising)
- Implementation of Intensity Approximated Sampling Patterns (for External Denoising)
- Additional Experimental Results

I. Implementation of Theorem 2 (General Sampling Case)

The optimal sampling pattern presented in Theorem 2 of the main article is

$$p_j = \min\{b_j\tau, 1\}, \quad \text{for } 1 \le j \le n, \tag{1}$$

where the parameter τ is the root of the function

$$g(\tau) = \sum_{j=1}^{n} \min\{b_j\tau, 1\} - n\xi. \tag{2}$$

Since $g(\cdot)$ is piecewise linear and monotonically increasing, with $g(0) = -n\xi < 0$ and $g(+\infty) = n(1-\xi) > 0$, the parameter τ can be uniquely determined. In this section, we discuss an efficient way to determine τ.

A. The Bisection Method

In determining τ, we choose the bisection method because of its efficiency and robustness. Gradient-based and Newton type of algorithms are not recommended as these algorithms require a region of convergence. Identifying such region could be challenging for the function $g(\cdot)$ defined in (2).

The bisection method is an iterative procedure that checks the signs of the two points τ_a, τ_b and their midpoint $\tau_c = (\tau_a+\tau_b)/2$. If τ_c has the same sign as τ_a, then τ_c replaces τ_a. Otherwise, τ_c replaces τ_b. The iteration continues until the residue $|\tau_a - \tau_b|$ is less than a tolerance level, or when $g(\tau_c)$ is sufficiently close to 0.

Pseudo-code of the bisection method is shown in Algorithm 1. For a small ξ, we find that by setting $\tau_a = 0$ and $\tau_b = 10$, the bisection method converges in 10 iterations with a precision $|g(\tau_c) - 0| < 10^{-1}$, which is sufficient for our problem.

B. Cost Reduction by Quantization

The cost of evaluating the function $g(\tau)$ for a fixed τ is $\mathcal{O}(n)$, because of the multiplication of $b_j \cdot \tau$ and the minimum operator in (2). To reduce the cost, we quantize the weights b_j by constructing the histogram of b_j.

The authors are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA.
Contact Emails: {`schan,zickler,yuelu`}@`seas.harvard.edu`.

Algorithm 1 The Bisection Method

```
Input: $\tau_a$, $\tau_b$.
Output: $\tau_c$.
Initialize: $F_a = g(\tau_a)$, $F_b = g(\tau_b)$, $F_c = \infty$.

while $|\tau_a - \tau_b| > \texttt{tol}$ and $|F_c - 0| > \texttt{tol}$ do
  Define $\tau_c = (\tau_a + \tau_b)/2$, and evaluate $F_c = g(\tau_c)$.
  if $F_a < 0$ and $F_c > 0$ then
    Set $\tau_b = \tau_c$, and $F_b = F_c$.
  else
    Set $\tau_a = \tau_c$, and $F_a = F_c$.
  end if
end while
```

Let q be the number of bins of the histogram, we define two sequences $\{u_t\}_{t=1}^q$ and $\{l_t\}_{t=1}^q$ such that $l_t \le b_j \le u_t$ for some t, and for $1 \le j \le n$. These two sequences denote the upper and lower bounds of the values in the t-th bin, respectively. Also, we let the center of each bin be

$$b_t^c = \frac{u_t + l_t}{2},$$

and we let the number of elements in the t-th bin be

$$n_t = \Big|\{b_j \mid l_t \le w_j \le u_t, \quad \text{for } j = 1, \ldots, n\}\Big|.$$

Then, the value $g(\tau)$ can be approximated by

$$g(\tau) \approx \sum_{t=1}^{q} n_t \min\{1, b_t^c \tau\} - n\xi. \tag{3}$$

The advantage of using (3) instead of (2) is that the cost of evaluating (3) is $\mathcal{O}(q)$, which is smaller than $\mathcal{O}(n)$ as in (2).

II. Implementation of Uniform Sampling Patterns

Uniform sampling is the fundamental building block of MCNLM's optimal sampling patterns. In this section, we discuss the implementation of uniform sampling for MCNLM. The techniques presented here will be used in other sections of this report.

For clarity we present the pseudo-codes using MATLAB language, although in practice the codes are implemented in C++.

A. Naive Implementation

To begin with, we consider the following naive implementation of uniform sampling:

```
if (rand(1)<xi)
  I(j) = 1;
else
  I(j) = 0;
end
```

where `rand(1)` is the MATLAB command for generating a random number from Uniform$[0, 1]$. The output of the above procedure is a sequence of i.i.d. Bernoulli random variables $\{I_j\}_{j=1}^n$ with probability ξ.

The problem of this naive implementation is that the random number `rand(1)` has to be generated on-the-fly for n times. These random numbers have to be compared against a double precision number ξ for n times, and to be multiplied by the number of pixels of the image. Thus, the overall cost is not trivial.

Remark: In practice, the line `I(j)=1` can be replaced by the actual denoising steps, *e.g.* `A = A + w(j)*x(j)/xi` and `B = B + w(j)/xi`, to improve efficiency.

B. Fast Implementation

Our proposed implementation is to replace the on-the-fly Bernoulli sampling by a pre-defined (fixed) sequence of sampling indices. To this end, we define

```
k   = round(xi*n);
idx = randi(n,k,1);
```

The command `k = round(xi*n)` returns the average number of samples to be picked, and the command `idx = randi(n,k,1)` returns a list of k random indices drawn uniformly from $\{1,\ldots,n\}$. Different from the naive implementation, the indices `idx` are *reused* for denoising all m pixels. Therefore, the overall cost of the algorithm is $\mathcal{O}(k)$ (for generating the random indices), as compared with $\mathcal{O}(nm)$ operations in the naive implementation. The pseudo-code of the alternative implementation is shown in Algorithm 2.

Algorithm 2 Uniform Sampling

```
Input: xi,n.
Determine k = round(xi*n).
Construct idx = randi(n,k,1).
for i=1:m do
  for t=1:k do
    Set j = idx(t).
    Compute w(j) and perform other steps of NLM.
  end for
end for
```

We emphasize that the alternative implementation is only an *approximation* of the method described in the main article, because the same sampling pattern is used for all m pixels and hence introduces correlation. However, in practice, we find that the impact of the correlation is small to the denoising quality. One way to minimize the correlation is to define multiple sampling patterns and use different patterns within certain spatial neighborhood.

III. Implementation of Spatially Approximated Sampling Patterns (for Internal Denoising)

In this section we discuss the implementation of the spatially approximated sampling patterns presented in Section IV.B.1 of the main article. To begin with, we recall that the spatially approximated sampling pattern is derived from the spatial weight

$$b_j^s = e^{-d_{i,j}^2/(2h_s^2)}, \tag{4}$$

where $d_{i,j}$ is the Euclidean distance between the spatial locations of the i-th and j-th pixels. Since $d_{i,j}$ is the distance, without loss of generality we can set $i = 0$.

Since $b_j^s \approx 0$ when $d_{i,j} > 3h_s$, we set a cutoff $\rho = 3h_s$ so that any pixel j located at a position farther than ρ from the i-th pixel will be discarded. (See Section I.B.1 for the definition of ρ.) We let the number of nonzero elements of $\{b_j^s\}$ be n_s.

A. Pre-Defined Sampling Indices

The goal of the fast implementation is to generate a sequence of sampling indices $j_1,\ldots,j_k$ by exploiting $\{b_j^s\}$. To this end, we first compute the parameter τ using the bisection method:

```
tau = bisection_method(bs, xi, n_s);
```

where the limit n_s is the number of non-zero $\{b_j^s\}$. The computed parameter τ determines the sampling pattern $\{p_j\}$:

$$p_j = \begin{cases} b_j^s\,\tau, & b_j^s \geq 1/\tau, \\ 1, & b_j^s < 1/\tau. \end{cases}$$

The sampling pattern thus returns a sequence of indices for denoising:

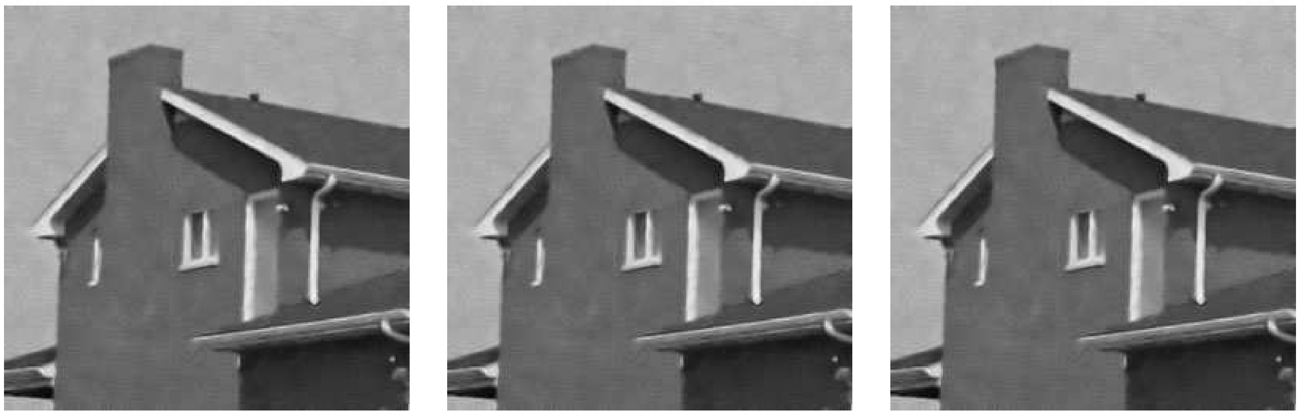
```
for j=1:n_s
```

```
    if (rand(1)<p(j))
        I(j) = 1;
    end
end
```

Similar to the uniform sampling case, the random indices generated by the above procedure are reused.

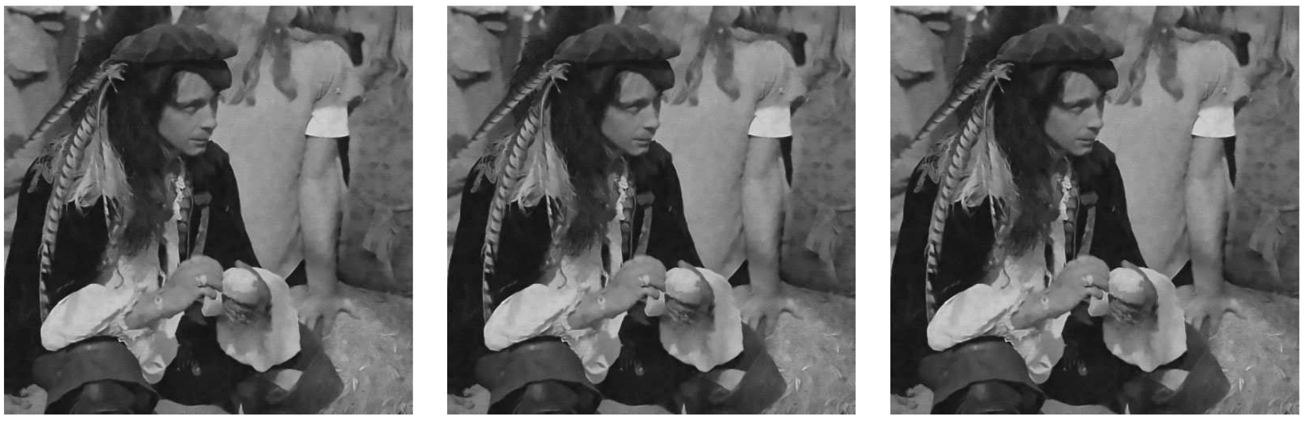
B. *Comparisons*

The performance of the spatially approximated sampling pattern is useful for small h_s. In Figure 1 and Figure 2, we show two denoising examples of using the oracle sampling pattern, the uniform sampling pattern, and the spatially approximated sampling pattern. The algorithm is implemented on MATLAB/C++ (`.mex`), and supports multi-core processing. The run time shown in the figures are recorded based on a 4-CPU 3.5GHz PC.



(a) Oracle sampling 1.07 sec, 31.78 dB (b) Uniform sampling 0.37 sec, 28.58 dB (c) Spatially approx. sampling 0.40 sec, 31.47 dB

Fig. 1: *House* (256×256). Noise level is $\sigma = 20/255$. Search radius $= 21 \times 21$. Parameters are $h_r = 20/255$, $h_s = 10/3$, $\rho = 10$. Patch size $= 5 \times 5$. Sampling Ratio $\xi = 0.1$.



(a) Oracle sampling 3.58 sec, 29.14 dB (b) Uniform sampling 2.02 sec, 27.68 dB (c) Spatially approx. sampling 2.53 sec, 29.10 dB

Fig. 2: *Man* (512×512). Noise level is $\sigma = 20/255$. Search radius $= 21 \times 21$. Parameters are $h_r = 20/255$, $h_s = 10/3$, $\rho = 10$. Patch size $= 5 \times 5$. Sampling Ratio $\xi = 0.1$.

IV. Implementation of Intensity Approximated Sampling Patterns (for External Denoising)

The problem of the spatially approximated sampling pattern is that it is not applicable to external denoising, because patches in external databases do not necessarily have spatial correlations. In this case, the intensity approximated sampling pattern described in Section IV.B.2 can be used.

The idea of intensity approximated sampling is to realize that

$$w_j \leq e^{-(\boldsymbol{x}_j^T \boldsymbol{s} - \boldsymbol{y}^T \boldsymbol{s})^2} = b_j^r,$$

where $\boldsymbol{s} = \boldsymbol{\Lambda 1}/(\sqrt{2}h_r \|\mathbf{1}\|_{\boldsymbol{\Lambda}})$. (See Section IV.B.2 for details.) The quantities $\boldsymbol{x}_j^T \boldsymbol{s}$ and $\boldsymbol{y}^T \boldsymbol{s}$ can be effectively computed by projecting $\boldsymbol{x}_j$ (and $\boldsymbol{y}$) onto the one-dimensional space spanned by $\boldsymbol{s}$. If $\{\boldsymbol{x}_j\}$ are patches collected from an image, then $\boldsymbol{x}_j^T \boldsymbol{s}$ can be computed through convolution [1].

An implementation concern about the projection is that since the number of w_j (*i.e.,* n) is large for external denoising, it will be inefficient to compute projections $\boldsymbol{x}_j^T \boldsymbol{s}$ and $\boldsymbol{y}^T \boldsymbol{s}$ for all $j = 1, \ldots, n$. In this section, we present a fast method that implements the intensity approximated sampling pattern without computing all projections.

A. Overview

The overall idea of the method is to use a two-stage importance sampling procedure [2]. The motivation is that if sampling a probability distribution p_j (which is b_j^r in our problem) is difficult, we can first sample an easy-to-compute distribution r_j such that

$$p_j \leq r_j, \tag{5}$$

and re-sample the already picked samples according to the probability $\frac{p_j}{r_j}$. It can be justified by seeing that for a Bernoulli random variable I_j, the probability of getting $I_j = 1$ is

$$p_j = r_j \cdot \frac{p_j}{r_j}. \tag{6}$$

Clearly, the critical point is that r_j must be an upper bound of p_j for all j. Therefore, in the following we discuss a procedure to find a valid and efficient upper bound r_j.

B. Quantization of $\{\overline{x}_j\}$

For notational simplicity we define

$$\overline{x}_j = \boldsymbol{x}_j^T \boldsymbol{s}, \quad \text{and} \quad \overline{y} = \boldsymbol{y}^T \boldsymbol{s}$$

as the projected signals. Then, we quantize the sequence $\{\overline{x}_j\}_{j=1}^n$ into a q-bin histogram with bins $\mathcal{B}_1, \ldots, \mathcal{B}_q$. Each bin $\mathcal{B}_t$ ($t = 1, \ldots, q$) contains a lower boundary and an upper boundary, l_t and u_t, respectively. In other words,

$$\mathcal{B}_t \overset{\text{def}}{=} \{j \mid l_t \leq \overline{x}_j \leq u_t\}, \tag{7}$$

for $t = 1, \ldots, q$. To illustrate this idea pictorially, in Figure 3 we show a sorted sequence $\{\overline{x}_j\}$. The dotted horizontal lines are the bin boundaries. In this plot, there are $q = 16$ bins.

Note that the quantization is independent of the denoising process. Therefore, it can be executed off-line when preparing the dataset.

C. Quantization of $\{b_j^r\}$

Our next step is to determine an upper bound r_j of p_j using $\{\mathcal{B}_t\}_{t=1}^q$. First, We determine the bin $\mathcal{B}_{t_0}$ that covers the point $\overline{y}$:

$$l_{t_0} \leq \overline{y} \leq u_{t_0},$$

by sweeping through $t_0 = 1, \ldots, q$. The motivation is that we want $\mathcal{B}_{t_0}$ to contain indices that are potentially the closest to $\overline{y}$.

Then, for all $j \in \{1, \ldots, n\}$, we define the upper bound

$$r_j = \begin{cases} e^{-(\overline{y} - u_t)^2}, & t \in \{1, \ldots, t_0 - 1\}, \text{ and } j \in \mathcal{B}_t, \\ 1, & j \in \mathcal{B}_{t_0}, \\ e^{-(\overline{y} - l_t)^2}, & t \in \{t_0 + 1, \ldots, q\}, \text{ and } j \in \mathcal{B}_t. \end{cases} \tag{8}$$

It follows that

$$b_j^r \leq r_j, \tag{9}$$

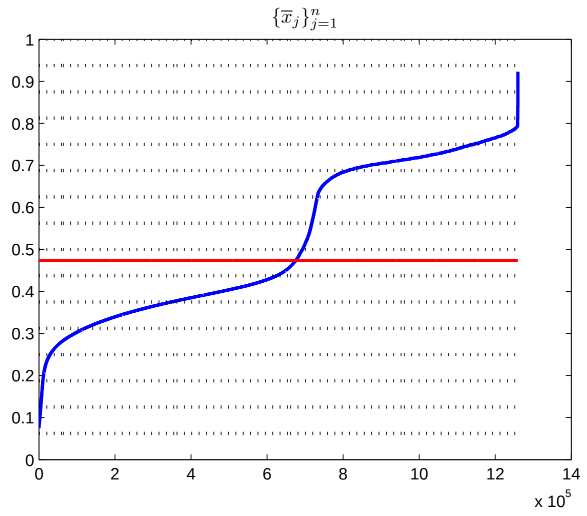


Fig. 3: Illustration of the quantization process. The dataset $\mathcal{X}$ used in this example contains $n = 1.26 \times 10^6$ samples. The blue line is the sequence $\{\overline{x}_j\}_{j=1}^n$ (sorted). The black dotted lines are the quantization boundaries. The red solid line is $\overline{y}$.

as illustrated in Figure 4.

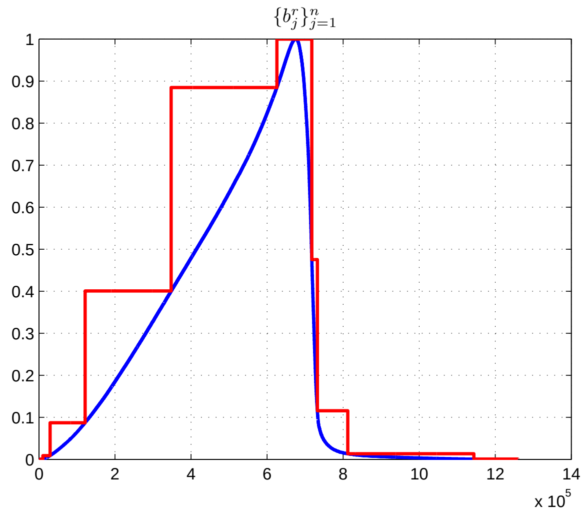


Fig. 4: Illustration of the quantization of b_j^r. The blue solid line is $\{b_j^r\}_{j=1}^n$. The red solid line is the upper bound r_j defined by (8).

D. Drawing Samples

Once r_j is defined, the two-stage sampling procedure can be described as follows. We compute τ for the sequence $\{r_j\}$ to determine a probability distribution

$$\overline{r}_j = r_j \tau. \tag{10}$$

Because r_j is piecewise constant, $\overline{r}_j$ is also piecewise constant. Therefore, drawing samples according to $\overline{r}_j$ is equivalent to drawing *uniformly* random samples at a probability $\overline{r}_j$. Thus, the fast implementation presented in Section II.B above can be used.

Since $\overline{r}_j$ is an upper bound of b_j^r, the number of samples collected at the Stage-1 sampling is guaranteed to be more than $n\xi$. However, an excessively large number of samples is undesirable as it requires more computation for Stage-2. In order to control the number of samples, we can choose an appropriate number of quantization levels q. In our experiment, we find that $q = 8$ to $q = 64$ is sufficient for most cases.

In the Stage-2 sampling, we compute the weight b_j^r

$$b_j^r = e^{-(\overline{x}_j - \overline{y})^2}, \tag{11}$$

for all j's that are picked in Stage-1. Then we define the probability

$$p_j = b_j^r \tau', \tag{12}$$

by computing an appropriate τ'. Finally, we pick the weights at a probability

$$p_j / \overline{r}_j = \frac{b_j^r \tau'}{r_j \tau}.$$

V. Additional Experimental Results

In this section we provide additional numerical results for Section V of the main article. The 10 testing images are shown in Figure 5.

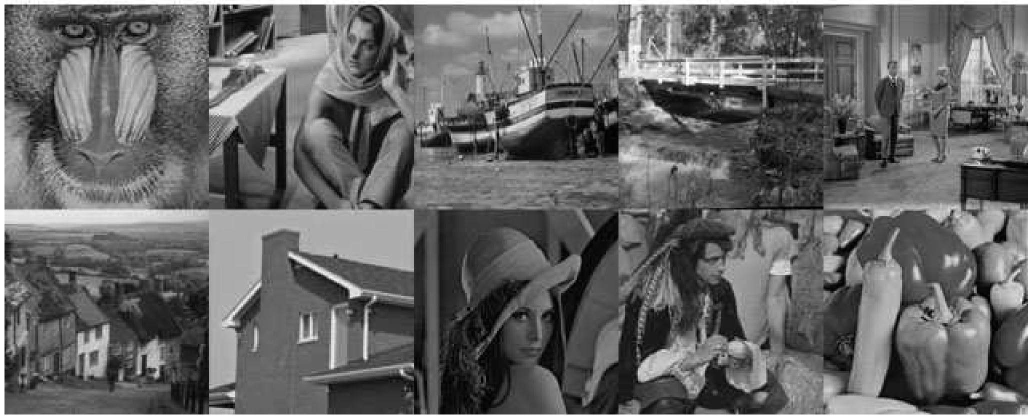


Fig. 5: Ten "standard" testing images for experiments.

Tables I and II are the extended version of the table in the main article. Table I shows the results when the window size is 21×21 and the patch size is 5×5, whereas II shows the results when the window size is 35×35 and the patch size is 9×9. In the extended tables, we show the PSNR values of using both the uniform sampling pattern and the spatially approximated sampling pattern. It is evident that the spatially approximated sampling pattern produces significantly higher PSNR than the uniform sampling pattern, *e.g.*, 32.14dB (approx.) versus 30.47dB (uniform) for *Barbara* at $\xi = 0.05$ (Table I). It is also indicative to see that when the spatial search window increases, *i.e.*, h_s increases, the spatially approximated sampling pattern has a small advantage over the uniform sampling pattern. For example, the gain of *Barbara* at $\xi = 0.05$ for the case of 35×35 search window is 0.36dB, whereas the gain is 1.67dB for the case of 21×21 search window.

In Figure 6, we show a visual comparison between MCNLM, NLM [3], GKD [4] and AM [5]. In this experiment, we considered the image *Man* (512×512) corrupted with noise of standard deviation $\sigma = 30/255$. To denoise the image, we set search window size as 21×21, and patch size as 5×5. The sampling pattern used is the spatially approximated sampling pattern.

References

[1] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 839–842, Dec. 2005.

[2] K. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.

[3] A. Buades, B. Coll, and J. Morel, "Denoising image sequences does not require motion estimation," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance (AVSS)*, Sep. 2005, pp. 70–74.

[4] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high-dimensional filtering," in *Proc. of ACM SIGGRAPH*, 2009, Article No. 21.

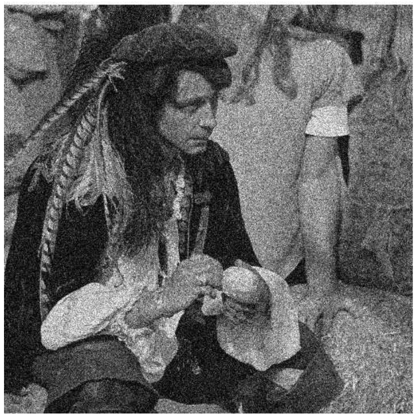
[5] E. Gastal and M. Oliveira, "Adaptive manifolds for real-time high-dimensional filtering," *ACM Trans. Graphics*, vol. 31, no. 4, pp. 33:1–33:13, 2012.

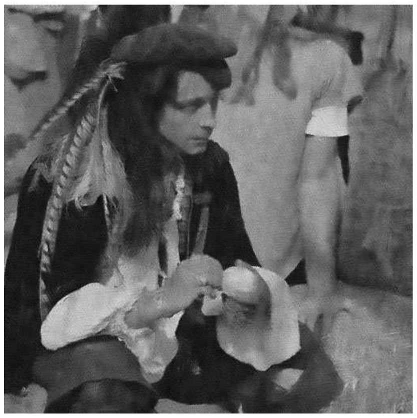
TABLE I: Single image denoising by MCNLM, using the spatially approximated sampling pattern. The case when $\xi = 1$ is equivalent to the standard NLM [3]. GKD refers to [4]. AM refers to [5]. The figures are PSNR values (dB). Window size is 21×21. Patch size is 5×5.

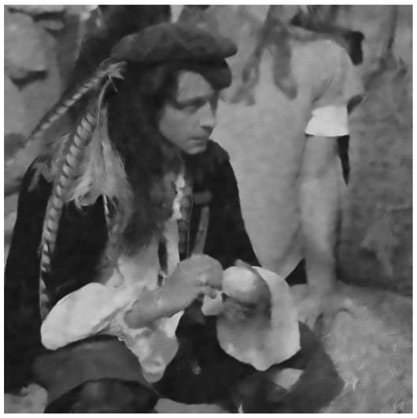
	ξ	0.05	0.1	0.2	0.5	1	GKD	AM	0.05	0.1	0.2	0.5	1	GKD	AM
σ	Sampling	*Baboon* 512×512							*Barbara* 512×512						
10	approx.	30.74	31.14	31.57	31.63	31.63	31.17	28.91	32.14	32.65	33.04	33.20	33.20	32.72	30.48
	uniform	29.51	29.95	30.45	31.12				30.47	31.26	32.10	32.87			
20	approx.	26.73	27.08	27.22	27.29	27.29	26.67	25.78	28.19	28.66	29.06	29.22	29.23	28.40	26.88
	uniform	25.03	26.27	26.95	27.29				26.01	27.27	28.32	29.11			
30	approx.	24.43	24.77	24.97	25.08	25.08	24.58	24.27	25.84	26.38	26.68	26.85	26.85	25.91	24.92
	uniform	22.40	23.85	24.89	25.08				23.06	24.95	25.98	26.70			
40	approx.	23.24	23.56	23.80	23.92	23.93	23.28	23.38	24.13	24.71	25.06	25.23	25.24	24.29	23.75
	uniform	21.09	22.61	23.44	23.92				21.94	23.24	24.45	25.11			
50	approx.	22.05	22.63	22.99	23.12	23.12	22.24	22.66	22.77	23.49	23.91	24.07	24.07	23.08	22.95
	uniform	19.42	21.43	22.44	22.91				20.38	22.06	23.18	23.88			
σ	Sampling	*Boat* 512×512							*Bridge* 512×512						
10	approx.	32.20	32.62	32.85	32.93	32.93	32.54	30.92	29.52	29.27	29.05	29.08	29.08	29.64	28.49
	uniform	30.47	31.42	32.24	32.70				29.05	29.29	29.63	29.51			
20	approx.	28.56	29.26	29.57	29.70	29.70	28.69	27.99	25.46	25.36	25.36	25.41	25.41	25.72	25.40
	uniform	26.16	27.67	28.74	29.40				24.59	25.33	25.73	25.65			
30	approx.	26.38	27.15	27.49	27.67	27.66	26.47	26.11	23.64	23.74	23.86	23.92	23.92	23.85	23.74
	uniform	23.66	25.53	26.60	27.38				22.13	23.41	23.93	23.95			
40	approx.	24.88	25.69	26.02	26.21	26.21	24.90	24.83	22.28	22.66	22.83	22.92	22.92	22.67	22.65
	uniform	21.84	23.73	25.03	25.97				20.78	21.88	22.62	22.93			
50	approx.	23.46	24.33	24.85	25.05	25.05	23.66	23.86	21.21	21.67	21.90	22.00	22.01	21.73	21.82
	uniform	20.28	22.50	23.76	24.74				19.57	20.75	21.70	21.94			
σ	Sampling	*Couple* 512×512							*Hill* 256×256						
10	approx.	31.98	32.4	32.62	32.72	32.72	32.40	30.84	30.54	30.51	30.44	30.49	30.49	30.89	30.12
	uniform	30.32	31.35	31.93	32.43				29.82	30.40	30.73	30.75			
20	approx.	28.21	28.55	28.78	28.88	28.88	28.19	27.57	26.95	27.14	27.17	27.26	27.26	27.14	26.95
	uniform	26.31	27.35	28.41	28.83				25.68	26.59	27.18	27.36			
30	approx.	25.80	26.42	26.69	26.81	26.81	25.97	25.76	25.13	25.46	25.64	25.75	25.75	25.28	25.34
	uniform	23.61	25.00	26.14	26.69				23.28	24.59	25.39	25.77			
40	approx.	24.21	25.03	25.38	25.54	25.54	24.52	24.56	23.76	24.38	24.66	24.81	24.82	24.14	24.37
	uniform	21.58	23.54	24.62	25.35				21.85	23.15	24.10	24.57			
50	approx.	23.14	23.93	24.42	24.59	24.59	23.39	23.72	22.83	23.46	23.87	24.02	24.02	23.10	23.57
	uniform	20.37	22.12	23.47	24.33				20.41	21.97	23.11	23.75			
σ	Sampling	*House* 256×256							*Lena* 512×512						
10	approx.	34.07	34.76	35.36	35.50	35.50	34.51	33.07	34.69	35.48	36.02	36.15	36.15	34.90	34.04
	uniform	31.65	32.89	33.90	34.95				32.07	33.21	34.71	35.72			
20	approx.	30.52	31.45	32.33	32.53	32.54	30.41	29.62	30.82	31.88	32.44	32.64	32.65	30.91	30.47
	uniform	26.98	28.67	30.42	31.87				27.70	29.44	30.97	32.20			
30	approx.	27.89	29.06	29.81	30.04	30.04	27.81	27.27	28.49	29.64	30.23	30.50	30.50	28.52	28.44
	uniform	23.91	25.82	27.96	29.51				24.63	26.96	28.71	29.96			
40	approx.	26.12	27.21	27.92	28.20	28.20	26.04	25.70	26.53	27.73	28.47	28.76	28.76	26.72	26.97
	uniform	21.57	24.20	26.23	27.67				22.80	24.89	26.78	28.19			
50	approx.	24.70	25.86	26.66	26.98	26.99	24.76	24.79	25.16	26.29	27.03	27.34	27.35	25.29	25.81
	uniform	20.87	22.89	24.92	26.37				21.29	23.39	25.33	26.75			
σ	Sampling	*Man* 512×512							*Pepper* 512×512						
10	approx.	32.32	32.58	32.72	32.79	32.79	32.56	31.49	32.78	33.49	33.85	33.94	33.95	33.35	31.63
	uniform	30.85	31.57	32.39	32.71				30.76	31.73	32.52	33.53			
20	approx.	28.56	29.13	29.37	29.49	29.49	28.78	28.37	28.99	29.87	30.32	30.46	30.46	29.38	28.43
	uniform	26.60	27.61	28.74	29.37				25.75	27.55	28.72	30.11			
30	approx.	26.68	27.26	27.69	27.85	27.85	26.74	26.62	26.48	27.33	27.82	27.98	27.98	26.81	25.88
	uniform	23.89	25.58	26.95	27.67				23.32	25.18	26.76	27.68			
40	approx.	25.12	25.94	26.45	26.65	26.66	25.27	25.39	24.68	25.61	26.09	26.29	26.29	25.01	24.25
	uniform	21.87	23.89	25.38	26.37				21.58	23.56	24.94	25.95			
50	approx.	23.91	24.92	25.48	25.71	25.72	24.13	24.52	23.24	23.91	24.41	24.61	24.61	23.49	23.00
	uniform	20.75	22.66	24.24	25.36				19.99	22.04	23.42	24.29			

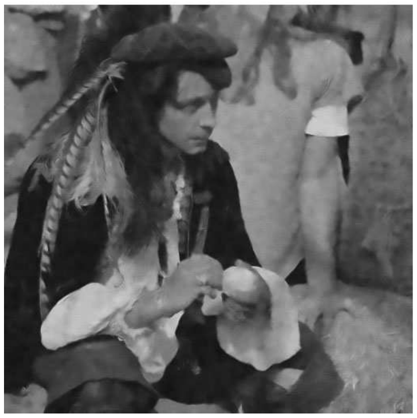
TABLE II: Single image denoising by MCNLM, using the spatially approximated sampling pattern. The case when $\xi = 1$ is equivalent to the standard NLM [3]. GKD refers to [4]. AM refers to [5]. The figures are PSNR values (dB). Window size is 35×35. Patch size is 9×9.

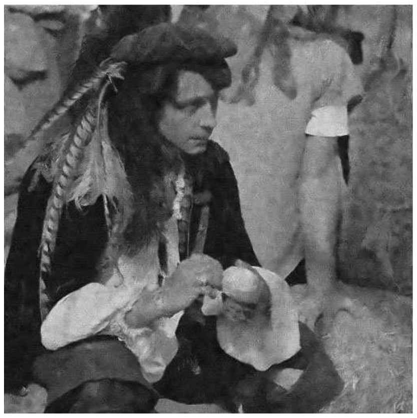
	ξ	0.05	0.1	0.2	0.5	1	GKD	AM	0.05	0.1	0.2	0.5	1	GKD	AM
σ	Sampling	*Baboon* 512×512							*Barbara* 512×512						
10	approx.	30.02	30.45	30.90	30.91	30.91	30.79	27.86	30.38	30.32	30.30	30.34	30.35	32.67	29.41
	uniform	29.11	29.55	29.93	30.44				30.02	30.30	30.45	30.49			
20	approx.	26.51	26.8	27.01	27.03	27.03	26.23	24.91	27.01	27.23	27.29	27.33	27.33	28.06	26.04
	uniform	24.72	25.72	26.43	26.75				26.01	26.55	27.15	27.26			
30	approx.	24.52	24.68	24.68	24.73	24.73	23.90	23.45	25.55	25.86	25.94	25.99	25.99	25.60	24.13
	uniform	22.88	24.04	24.64	24.79				24.13	25.02	25.59	25.94			
40	approx.	23.40	23.41	23.46	23.51	23.52	22.57	22.66	24.53	24.75	24.94	24.99	25.00	24.02	23.03
	uniform	22.07	23.02	23.47	23.47				22.94	24.01	24.64	24.89			
50	approx.	22.39	22.66	22.69	22.74	22.74	21.71	22.14	23.55	23.85	24.01	24.08	24.08	22.86	22.28
	uniform	21.33	22.30	22.63	22.69				22.01	23.09	23.65	24.00			
σ	Sampling	*Boat* 512×512							*Bridge* 512×512						
10	approx.	31.69	32.14	32.41	32.43	32.43	32.16	29.51	29.16	29.11	28.98	28.98	28.98	29.44	27.38
	uniform	30.37	31.08	31.42	32.00				28.69	29.01	29.10	29.21			
20	approx.	28.57	29.16	29.47	29.52	29.52	28.21	26.88	25.30	25.23	25.27	25.29	25.29	25.13	24.43
	uniform	26.38	27.64	28.43	29.17				24.16	24.84	25.14	25.30			
30	approx.	26.89	27.30	27.54	27.59	27.59	25.83	25.05	23.64	23.71	23.75	23.77	23.77	23.26	22.69
	uniform	24.70	25.82	26.84	27.34				22.44	23.24	23.64	23.73			
40	approx.	25.55	25.98	26.13	26.21	26.21	24.26	23.90	22.45	22.58	22.64	22.67	22.68	21.97	21.65
	uniform	23.30	24.77	25.46	26.12				21.48	22.17	22.50	22.69			
50	approx.	24.32	24.75	24.92	25.01	25.01	23.11	23.08	21.68	21.77	21.86	21.90	21.90	21.16	20.97
	uniform	22.16	23.64	24.54	24.93				20.64	21.39	21.79	21.82			
σ	Sampling	*Couple* 512×512							*Hill* 256×256						
10	approx.	31.72	32.22	32.51	32.54	32.54	32.17	29.45	30.08	30.15	30.08	30.10	30.10	30.59	29.07
	uniform	30.30	30.90	31.57	32.19				29.41	29.91	30.16	30.26			
20	approx.	28.41	28.80	29.08	29.12	29.12	27.70	26.44	26.88	27.06	27.03	27.05	27.05	26.51	25.90
	uniform	26.20	27.30	28.19	28.90				25.26	26.14	26.79	27.12			
30	approx.	26.39	26.69	26.87	26.92	26.92	25.34	24.72	25.34	25.50	25.59	25.62	25.62	24.72	24.45
	uniform	24.26	25.38	26.36	26.90				23.79	24.82	25.33	25.66			
40	approx.	24.95	25.31	25.45	25.51	25.51	23.88	23.69	24.26	24.44	24.58	24.62	24.62	23.48	23.52
	uniform	23.31	24.40	25.08	25.44				22.74	23.68	24.26	24.55			
50	approx.	23.82	24.17	24.34	24.41	24.41	22.88	22.98	23.20	23.51	23.62	23.68	23.68	22.58	22.88
	uniform	22.20	23.32	24.03	24.37				21.81	22.85	23.35	23.67			
σ	Sampling	*House* 256×256							*Lena* 512×512						
10	approx.	33.57	34.30	34.69	34.75	34.75	34.27	31.45	34.70	35.54	35.92	35.97	35.97	34.57	32.86
	uniform	32.07	32.79	33.51	34.22				32.48	33.37	34.34	35.32			
20	approx.	30.66	31.80	32.59	32.69	32.69	30.10	28.19	31.73	32.41	32.85	32.94	32.94	30.46	29.40
	uniform	27.90	29.16	30.28	31.87				28.81	30.19	31.52	32.55			
30	approx.	28.88	30.04	30.66	30.79	30.79	27.32	26.01	29.65	30.22	30.59	30.70	30.70	28.07	27.43
	uniform	25.26	27.26	28.75	30.11				26.71	28.53	29.62	30.42			
40	approx.	27.28	28.19	28.63	28.74	28.74	25.49	24.52	27.70	28.34	28.70	28.81	28.82	26.41	26.13
	uniform	23.92	25.80	27.30	28.41				25.16	26.84	27.90	28.58			
50	approx.	26.08	26.80	27.16	27.32	27.33	24.22	23.72	26.31	26.89	27.21	27.34	27.34	25.17	25.25
	uniform	23.06	25.03	26.14	27.05				23.84	25.43	26.49	27.08			
σ	Sampling	*Man* 512×512							*Pepper* 512×512						
10	approx.	31.73	32.04	32.32	32.34	32.34	32.14	30.25	32.11	32.80	33.28	33.30	33.30	32.99	29.66
	uniform	30.30	30.92	31.42	32.08				30.27	30.69	31.59	32.60			
20	approx.	28.50	28.89	29.17	29.21	29.21	28.16	27.23	28.78	29.64	30.16	30.21	30.21	28.70	26.79
	uniform	26.63	27.67	28.38	29.04				25.84	27.03	28.12	29.44			
30	approx.	26.85	27.36	27.51	27.56	27.56	26.06	25.52	26.84	27.52	27.88	27.94	27.94	25.99	24.29
	uniform	24.71	26.09	26.89	27.45				23.80	25.21	26.51	27.52			
40	approx.	25.81	26.07	26.30	26.37	26.37	24.66	24.43	25.26	25.81	26.15	26.21	26.21	24.35	22.74
	uniform	23.67	24.97	25.78	26.26				22.56	23.95	25.18	25.95			
50	approx.	24.74	25.12	25.37	25.45	25.45	23.59	23.73	23.97	24.43	24.59	24.65	24.66	22.93	21.56
	uniform	22.69	23.98	24.89	25.27				21.36	23.03	23.85	24.50			


(a) noisy, 18.75dB


(b) $\xi = 0.05$, 26.68dB


(c) $\xi = 0.1$, 27.69dB


(d) $\xi = 1$, NLM [3], 27.85dB


(e) GKD [4], 26.74dB

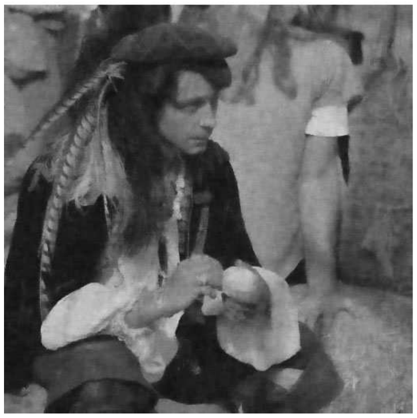

(f) AM [5], 26.62dB

Fig. 6: MCNLM on *Man* (512×512). Noise level is $\sigma = 30/255$. The search window has a finite size of 21×21. Patch size is 5×5. Sampling pattern: spatially approximated sampling pattern.