FAST NON-LOCAL FILTERING BY RANDOM SAMPLING: IT WORKS, ESPECIALLY FOR LARGE IMAGES

Stanley H. Chan, Todd Zickler and Yue M. Lu

Harvard University, Cambridge, MA 02138, USA
Email:{schan, zickler, yuelu}@seas.harvard.edu

ABSTRACT

Non-local means (NLM) is a popular denoising scheme. Conceptually simple, the algorithm is computationally intensive for large images. We propose to speed up NLM by using random sampling. Our algorithm picks, uniformly at random, a small number of columns of the weight matrix, and uses these "representatives" to compute an approximate result. It also incorporates an extra column-normalization of the sampled columns, a form of symmetrization that often boosts the denoising performance on real images. Using statistical large deviation theory, we analyze the proposed algorithm and provide guarantees on its performance. We show that the probability of having a large approximation error decays exponentially as the image size increases. Thus, for large images, the random estimates generated by the algorithm are tightly concentrated around their limit values, even if the sampling ratio is small. Numerical results confirm our theoretical analysis: the proposed algorithm reduces the run time of NLM, and thanks to the symmetrization step, actually provides some improvement in peak signal-to-noise ratios.

Index Terms— Non-local means, random sampling, Sinkhorn-Knopp balancing scheme, image denoising

1. INTRODUCTION

Non-local means (NLM) [1], and its extensions and generalizations (*e.g.*, [2, 3, 4]), are widely used in image denoising. Conceptually simple, NLM filters a noisy image by replacing each pixel with a linear combinations of all other pixels. In the matrix-vector form, the algorithm can be defined as

$$\widehat{\boldsymbol{f}} = \boldsymbol{D}^{-1}\boldsymbol{W}\boldsymbol{g}, \tag{1}$$

where $\boldsymbol{g}, \widehat{\boldsymbol{f}} \in \mathbb{R}^n$ are the noisy and denoised images, respectively, $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is a symmetric matrix computed from $\boldsymbol{g}$, and $\boldsymbol{D} = \mathrm{diag}\{\boldsymbol{W}\boldsymbol{1}\}$, with $\boldsymbol{1} \overset{\text{def}}{=} [1, 1, \ldots, 1]^T$, is a diagonal matrix representing a normalization factor so that the filter weights sum to one.

The entries of $\boldsymbol{W}$, denoted by $\{w_{i,j}\}$, measure the similarity between all pairs of pixels. A standard choice is

$$w_{ij} = e^{-\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2/h^2}, \tag{2}$$

where $\boldsymbol{p}_i, \boldsymbol{p}_j \in \mathbb{R}^q$ denote pixel patches centered at the ith and jth pixel, respectively, $\|\cdot\|$ is the (weighted) ℓ^2 norm, and h is a scale parameter determined by the noise level. Several more sophisticated constructions of $\{w_{i,j}\}$ have been proposed in the literature [5, 6], further improving the denoising accuracy.

The work of S. Chan is supported by a Croucher Foundation Postdoctoral Research Fellowship (2012-2013).

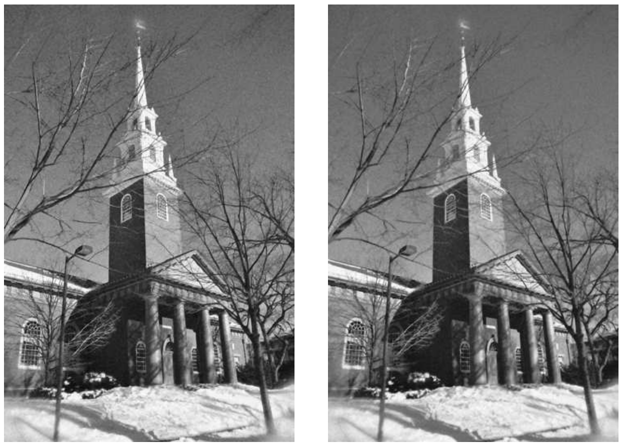
(a) Proposed: 27.58 dB
Run time = 240 seconds
(b) Classical NLM: 27.01 dB
Run time = 47170 seconds

Fig. 1: A typical comparison of the proposed algorithm and the classical NLM. The proposed algorithm computes and uses only 0.5% columns of the weight matrix, reducing the run time by a factor of 200. Also, by including a additional normalization of the sampled columns, it can actually improve performance.

One limitation of NLM is computational complexity. For an image of n pixels, just computing the similarity matrix $\boldsymbol{W}$ alone requires $\mathcal{O}(qn^2)$ arithmetic operations, where q is the number of pixels in each image patch. In addition, about $\mathcal{O}(n^2)$ operations are needed to carry out the matrix-vector multiplication in (1). This complexity imposes a severe bottleneck on NLM, hindering its widespread adoption in real-world applications, where digital images easily contain tens of millions of pixels.

In this paper, we propose an algorithm to accelerate NLM filtering using random sampling [7], a technique widely used in social and political sciences. The proposed algorithm picks, uniformly at random, a small number of columns of W, and uses these "representatives" to compute a filtered image. It also incorporates an extra column-normalization of the sampled columns, a form of symmetrization that often boosts performance on real images. Figure 1[1] shows a typical comparison between the proposed algorithm and the classical NLM [as defined in (1) and (2)] on an image of size 1072×712. The proposed algorithm reduces the run time by more than two orders of magnitude, and in the meanwhile, provides some improvement in peak signal-to-noise ratios (PSNRs). Using tools from statistical large deviation theory [8, 9], we analyze the proposed algorithm and provide probabilistic guarantees on its performance.

[1]Photo courtesy of wikimedia.org, Harvard_memorial_church_winter_2009.jpg

1.1. Related Work

The high complexity of NLM is a well-known issue, and many methods have been proposed in the past decade to speed up NLM. The simplest and most widely used approach is to reduce the spatial search regions [2], essentially limiting $\boldsymbol{W}$ to be a banded matrix. While effective in reducing the complexity, this strategy reduces the ability of NLM to benefit from distant patches that are nonetheless similar. Other notable approaches include, *e.g.,* the pre-selection methods [10, 11, 12, 13], which skips the computation of the weights in (2) when the algebraic means of two patches differ by more than a threshold; SVD methods [14, 15, 16], which project the patches onto lower-dimensional subspaces; Gaussian KD trees [17, 18], which generalizes the concepts of fast Gauss transforms [19]; and variants of integral image methods [20, 21].

In this work, we approach the problem of accelerating NLM by using random sampling, a strategy that, to our knowledge, has not been considered in the image processing literature. Our method is inspired by the seminal work of Drineas, Kannan and Mahoney [22, 23, 24], who consider the general problem of approximating large-scale matrix operations (*e.g.,* matrix-vector multiplications), by random sampling. In their setting, the matrix is assumed to be known and accessible. In contrast, the weight matrix $\boldsymbol{W}$ in our problem is not available to the algorithm, so the cost of computing its entries must be taken into account. Moreover, the normalization step in (1) (in the form of multiplication by $\boldsymbol{D}^{-1}$) makes our problem different from a standard matrix-vector multiplication.

1.2. Outline and Main Contributions

After presenting the proposed random sampling algorithm in Section 2, we provide two main contributions in this paper:

1. *Concentration analysis*: The proposed algorithm picks, uniformly at random, a small number of columns of $\boldsymbol{W}$, and uses these "representatives" to compute a filtered image $\boldsymbol{f}^{(k)}$, where the superscript k indicates the number of columns chosen. When $k \ll n$, it is legitimate to worry that $\boldsymbol{f}^{(k)}$, a random vector, would vary widely at different runs of the algorithm on the same image, and in particular, $\boldsymbol{f}^{(k)}$ would be very different from $\boldsymbol{f}^* \overset{\text{def}}{=} \boldsymbol{f}^{(n)}$, *i.e.,* the limit result one can get if all the columns are used. Perhaps surprisingly, our analysis in Section 3 shows that these concerns are unnecessary. Under mild conditions on the weight matrix $\boldsymbol{W}$, we show in Theorem 1 that $\boldsymbol{f}^{(k)}$ stays within an ε-neighborhood of $\boldsymbol{f}^*$ with very high probability, as long as the product

$$n\,\varepsilon^2 \xi_k/(1-\xi_k)$$

is large. Here, $\xi_k \overset{\text{def}}{=} (k-1)/n$ is the *sampling ratio*. This result implies that, when n, the size of image, is large, the random vectors $\boldsymbol{f}^{(k)}$ will be tightly concentrated around $\boldsymbol{f}^*$, even if we use a very small sampling ratio.

2. *Performance improvement*: We propose a column normalization scheme, effectively re-weighting each sampled column. Only marginally increasing the total complexity of the algorithm, this additional step turns out to be very effective in improving the denoising performance. We show in Section 4 that, with column normalization, the limit result $\boldsymbol{f}^*$ of our algorithm is equivalent to applying to $\boldsymbol{W}$ two iterations of the Sinkhorn-Knopp scheme [25], an iterative procedure that converts a matrix to a "nearby" doubly stochastic matrix. The advantage of symmetrizing $\boldsymbol{W}$ by using the Sinkhorn-Knopp scheme was recently observed and studied by Milanfar [26]. Our numerical experiments confirm this improvement, with $\boldsymbol{f}^*$ outperforming the classical NLM by more than 1 dB on average in our numerical experiments.

Algorithm 1 Fast NLM Filtering by Random Sampling

```
1:  Input: $\boldsymbol{g} \in \mathbb{R}^{n\times 1}$, and $k$, where $1 \le k \le n$
2:  Output: $\boldsymbol{f}^{(k)} \in \mathbb{R}^{n\times 1}$
3:  Initialize $\boldsymbol{f}_{\text{den}} = \boldsymbol{0}_{n\times 1}$, $\boldsymbol{f}_{\text{num}} = \boldsymbol{0}_{n\times 1}$
4:  for $t = 1, \ldots, k$ do
5:      Pick $j_t \in \{1, \ldots, n\}$ uniformly at random without replacement.
6:      Compute $\boldsymbol{W}_{:,j_t}$, the $j_t$th column of $\boldsymbol{W}$.
7:      Let $\boldsymbol{v} = \boldsymbol{W}_{:,j_t}/(\boldsymbol{W}_{:,j_t}^T \boldsymbol{1})$    (column normalization)
8:      $\boldsymbol{f}_{\text{num}} = \boldsymbol{f}_{\text{num}} + g_{j_t}\boldsymbol{v}$
9:      $\boldsymbol{f}_{\text{den}} = \boldsymbol{f}_{\text{den}} + \boldsymbol{v}$
10: end for
11: Output $\boldsymbol{f}^{(k)} = \boldsymbol{f}_{\text{num}}./\boldsymbol{f}_{\text{den}}$
```

We conclude the paper in Section 5. Due to space limitations, we only present the main results and ideas in this paper, and leave the proofs and other technical details to a follow-up paper.

2. THE PROPOSED ALGORITHM

In the section, we present the proposed random sampling algorithm and discuss its basic properties. First, we note that the classical NLM scheme defined in (1) can be rewritten as

$$\widehat{\boldsymbol{f}} = \Big(\sum_{1\le j\le n} g_j \boldsymbol{W}_{:,j}\Big)./\sum_{1\le j\le n} \boldsymbol{W}_{:,j}, \tag{3}$$

where $\boldsymbol{W}_{:,j}$ denotes the jth column of $\boldsymbol{W}$, and we use the MATLAB notation $./$ for element-wise division.

The basic idea of the proposed algorithm is simple: since it is computationally expensive to compute the entire matrix $\boldsymbol{W}$, we only use a small number of representative columns to approximately compute the sums in the numerator and denominator. Let $\{j_1, j_2, \ldots, j_k\}$ be k indices picked uniformly at random from the set $\{1, 2, \ldots, n\}$ *without* replacement. Then an approximation of (3) will be $\sum_{1\le t\le k} \boldsymbol{W}_{:,j_t} g_{j_t}./\sum_t \boldsymbol{W}_{:,j_t}$.

In our algorithm, we first carry out a normalization step by computing

$$\widetilde{\boldsymbol{W}} \overset{\text{def}}{=} \boldsymbol{W}\boldsymbol{D}^{-1},$$

where the diagonal elements of $\boldsymbol{D}$ are the column (and due to symmetry, row) sums of $\boldsymbol{W}$. Essentially, $\widetilde{\boldsymbol{W}}$ is a column-normalized version of $\boldsymbol{W}$. By introducing two vectors,

$$\boldsymbol{f}_{\text{num}} = \sum_{1\le t\le k} g_{j_t}\widetilde{\boldsymbol{W}}_{:,j_t} \quad \text{and} \quad \boldsymbol{f}_{\text{den}} = \sum_{1\le t\le k} \widetilde{\boldsymbol{W}}_{:,j_t}, \tag{4}$$

the output of the proposed algorithm is just

$$\boldsymbol{f}^{(k)} = \boldsymbol{f}_{\text{num}}./\boldsymbol{f}_{\text{den}}.$$

Algorithm 1 provides the pseudocode of the above scheme.

Since $\{j_t\}_{t=1}^k$ is a set of randomly chosen indices, $\boldsymbol{f}_{\text{num}}$ and $\boldsymbol{f}_{\text{den}}$, and thus the final output $\boldsymbol{f}^{(k)}$, are all random vectors. To gain insight into the proposed random sampling algorithm, we first study the expected values of $\boldsymbol{f}_{\text{num}}$ and $\boldsymbol{f}_{\text{den}}$. To that end, it is helpful to introduce the following Bernoulli random variables:

$$b_j^{(k)} = \begin{cases} 1, & \text{if } j \in \{j_1, \ldots, j_k\}, \\ 0, & \text{otherwise.} \end{cases}$$

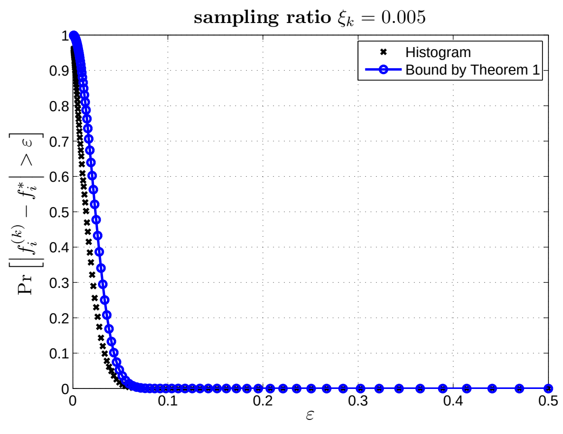

Fig. 2: $\Pr\left[|f_i^{(k)} - f_i^*| \geq \varepsilon\right]$ as a function of ε for a fixed $\boldsymbol{g}$ and $\boldsymbol{W}$, where $n = 10^4$, noise $\sigma = 15/255$, parameter $h = 15/255$, patch size $= 5 \times 1$. Crosses denote the histogram of 5000 independent trials for each ε. The circled line denotes the upper bound predicted by Theorem 1.

That is, $b_j^{(k)}$ indicates whether the jth column has been picked. The two vectors in (4) can now be rewritten as

$$\boldsymbol{f}_{\text{num}} = \sum_{1 \leq j \leq n} b_j^{(k)} g_j \widetilde{\boldsymbol{W}}_{:,j} \quad \text{and} \quad \boldsymbol{f}_{\text{den}} = \sum_{1 \leq j \leq n} b_j^{(k)} \widetilde{\boldsymbol{W}}_{:,j}.$$

Due to symmetry, the expected values of $b_j^{(k)}$ are all identical: $\mathbb{E}[b_1^{(k)}] = \mathbb{E}[b_2^{(k)}] = \ldots = \mathbb{E}[b_n^{(k)}]$. Also, since $\sum_{1 \leq j \leq n} b_j^{(k)} \equiv k$, we have $\mathbb{E}[b_j^{(k)}] = k/n$ for all $1 \leq j \leq n$. Consequently,

$$\mathbb{E}[\boldsymbol{f}_{\text{num}}] = k/n \sum_{1 \leq j \leq n} g_j \widetilde{\boldsymbol{W}}_{:,j} \quad \text{and} \quad \mathbb{E}[\boldsymbol{f}_{\text{den}}] = k/n \sum_{1 \leq j \leq n} \widetilde{\boldsymbol{W}}_{:,j}.$$

Let $\boldsymbol{f}^*$ be the element-wise ratio of the two mean vectors, we get

$$\boldsymbol{f}^* \overset{\text{def}}{=} \mathbb{E}[\boldsymbol{f}_{\text{num}}]./\mathbb{E}[\boldsymbol{f}_{\text{den}}] = \boldsymbol{f}^{(n)}.$$

The second equality above indicates that $\boldsymbol{f}^*$ is also the limit value of the proposed algorithm when $k = n$, *i.e.*, when we pick all the columns of $\widetilde{\boldsymbol{W}}$. To be sure, $\mathbb{E}[\boldsymbol{f}^{(k)}]$ is generally not equal to $\boldsymbol{f}^*$, since $\mathbb{E}\left[\boldsymbol{f}_{\text{num}}./\boldsymbol{f}_{\text{den}}\right] \neq \mathbb{E}[\boldsymbol{f}_{\text{num}}]./\mathbb{E}[\boldsymbol{f}_{\text{den}}]$. However, as we show in the next section, for large n, this bias is negligible and that, with high probability, $\boldsymbol{f}^{(k)}$ is tightly concentrated around $\boldsymbol{f}^*$.

3. CONCENTRATION ANALYSIS

In what follows, we assume that the pixel values have been normalized so that $0 \leq g_i \leq 1$, for all $1 \leq i \leq n$. Before stating the main result, we define the quantities

$$\mu_{Ai} = \frac{1}{n}\sum_{j=1}^{n} g_j \widetilde{W}_{ij}, \quad \mu_{Bi} = \frac{1}{n}\sum_{j=1}^{n} \widetilde{W}_{ij}, \quad \mu_{Ci} = \frac{1}{n}\sum_{j=1}^{n} g_j \widetilde{W}_{ij}^2,$$

$$\sigma_{Ai}^2 = \frac{1}{n}\sum_{j=1}^{n} (g_j \widetilde{W}_{ij} - \mu_{Ai})^2, \quad \sigma_{Bi}^2 = \frac{1}{n}\sum_{j=1}^{n} (\widetilde{W}_{ij} - \mu_{Bi})^2,$$

$$\mu_i = \frac{\mu_{Ai}}{\mu_{Bi}}, \quad \sigma_i^2 = \sigma_{Ai}^2 + \mu_{Ai}^2 + (\sigma_{Bi}^2 + \mu_{Bi}^2)\mu_i^2 - 2\mu_i\mu_{Ci}.$$

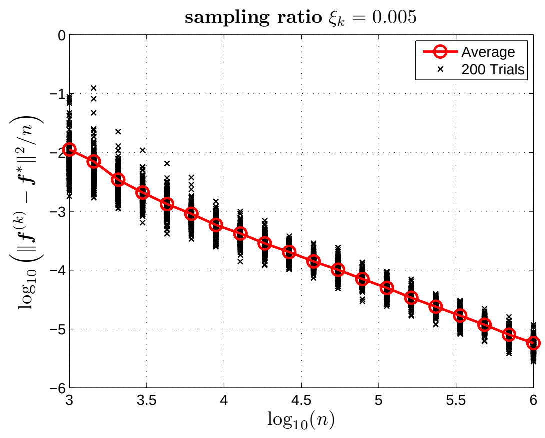

Fig. 3: The mean squared error between the random estimate $\boldsymbol{f}^{(k)}$ and the limit result $\boldsymbol{f}^*$ converges rapidly with n. The signal tested is piecewise-continuous, with noise $\sigma = 15/255$, parameter $h = 15/255$, and patch size $= 5 \times 1$. Crosses denote realizations of 200 independent trials for each n. The circled line denotes the average of these trials.

Here μ_{Ai} and μ_{Bi} can be interpreted as the averages of $\boldsymbol{f}_{\text{num}}$ and $\boldsymbol{f}_{\text{den}}$, respectively. Similarly, σ_{Ai}^2 and σ_{Bi}^2 are the corresponding variances. Thus, Theorem 1 is characterized by the statistics of the noisy data $\boldsymbol{g}$ and the weight matrix $\boldsymbol{W}$.

Theorem 1. *For any $0 < \varepsilon < 1$, and for any $1 \leq i \leq n$,*

$$\Pr\left[|f_i^{(k)} - f_i^*| \geq \varepsilon\right] \leq \exp\left\{-n\varepsilon^2 \left(\frac{\mu_{Bi}^2}{2\sigma_i^2}\right)\left(\frac{\xi_k}{1-\xi_k}\right)\right\}, \quad (5)$$

where $\xi_k = (k-1)/n$ is the sampling ratio.

Remark 1. *We obtain this result by adapting and generalizing the bounding techniques developed in [8], and the proof is left to a follow-up paper. In essence, the theorem says that the probability of having a large approximation error decays exponentially as n increases. Thus, for large images, the random estimates $\boldsymbol{f}^{(k)}$ will be highly concentrated around $\boldsymbol{f}^*$, even if the sampling ratio ξ_k and error threshold ε are kept small.*

The concentration of the random estimates can also be measured in terms of the expected mean squared errors (MSEs), as follows:

Corollary 1. $\mathbb{E}\left[\frac{1}{n}\|\boldsymbol{f}^{(k)} - \boldsymbol{f}^*\|^2\right] \leq \frac{2}{n}\left(\sum_{i=1}^{n} \frac{1}{n}\frac{\sigma_i^2}{\mu_{Bi}^2}\right)\left(\frac{1-\xi_k}{\xi_k}\right).$

Example 1. *To illustrate the results of Theorem 1 and Corollary 1, we apply the denoising algorithm to a 1D piecewise-continuous signal with a fixed sampling ratio $\xi_k = 0.005$, and $n = 10^4$. For this example, $\mu_{Bi} = 3.2314 \times 10^{-1}$, $\sigma_i^2 = 2.3362 \times 10^{-3}$ and $\frac{\mu_{Bi}^2}{2\sigma_i^2} = 2..2349 \times 10^1$. If $\varepsilon = 0.1$, then $\Pr\left[|f_i^{(k)} - f_i^*| \geq \varepsilon\right] \leq 1.3261 \times 10^{-5}$. Figure 2 illustrates the tightness of the bound predicted by Theorem 1. Figure 3 shows that the MSE converges at a rate of $1/n$ and becomes tightly concentrated around 0.*

Figure 4 compares the performance of the random sampling algorithm and the classical NLM, when both are applied to the image shown in Figure 1. We use PSNR($\boldsymbol{x}$) to denote the PSNR achieved by an estimate $\boldsymbol{x}$, where $\boldsymbol{x}$ can be $\boldsymbol{f}^{(k)}$, the random estimate with

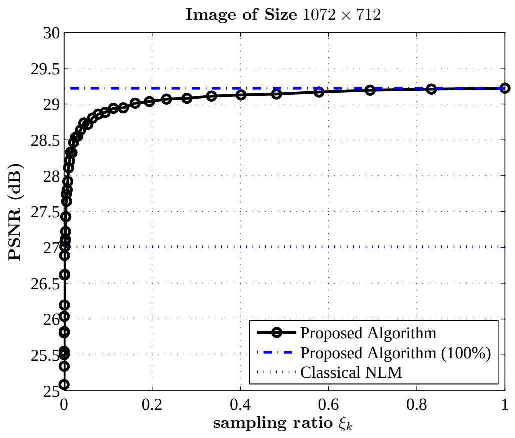

Fig. 4: The convergence plot of denoising the image shown in Figure 1 with 1072×712 pixels. Noise $\sigma = 15/255$, parameter $h = 15/255$, and patch size $= 5 \times 5$. Speed up factor $= 1/\xi_k$.

k columns; $\boldsymbol{f}^*$, the limit result of the proposed algorithm; or $\widehat{\boldsymbol{f}}$, the reconstruction obtained by the classical NLM as in (1).

We observe from Figure 4 that, with $\xi_k = 0.3$, the deviation of PSNR($\boldsymbol{f}^{(k)}$) from PSNR($\boldsymbol{f}^*$) is less than 0.1 dB. This suggests that $\boldsymbol{f}^{(k)}$ is already a good approximation of $\boldsymbol{f}^*$. It is also interesting to note that PSNR($\boldsymbol{f}^*$) is higher than PSNR($\widehat{\boldsymbol{f}}$) by more than 2 dB. Consequently, with a very small sampling ratio $\xi_k = 0.002$, PSNR($\boldsymbol{f}^{(k)}$) is already as good as PSNR($\widehat{\boldsymbol{f}}$).

4. DISCUSSIONS

4.1. Performance Improvement

It is perhaps a surprising observation from Figure 4 that $\boldsymbol{f}^*$, the limit result of the proposed scheme, yields much higher PSNR than $\widehat{\boldsymbol{f}}$, the classical NLM. In this section, we provide some initial explanations, and leave more detailed analysis to a follow-up work.

Let $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ be a matrix with positive entries. Define two matrix operations $\varphi_r : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ and $\varphi_c : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$:

$$\varphi_r(\boldsymbol{W}) = \text{diag}^{-1}\{\boldsymbol{W}\,\boldsymbol{1}\}\,\boldsymbol{W}, \quad \text{and} \quad \varphi_c(\boldsymbol{W}) = \boldsymbol{W}\,\text{diag}^{-1}\left\{\boldsymbol{W}^T\,\boldsymbol{1}\right\}.$$

In words, $\varphi_r(\boldsymbol{W})$ and $\varphi_c(\boldsymbol{W})$ normalize the rows, and respectively the columns, of $\boldsymbol{W}$. Using these two operators, we have $\widehat{\boldsymbol{f}} = \varphi_r(\boldsymbol{W})\,\boldsymbol{g}$, whereas

$$\boldsymbol{f}^* = \varphi_r(\widetilde{\boldsymbol{W}})\,\boldsymbol{g} = \varphi_r(\varphi_c(\boldsymbol{W}))\,\boldsymbol{g}. \tag{6}$$

So, $\boldsymbol{f}^*$ differs from $\widehat{\boldsymbol{f}}$ by an extra column normalization step.

It is interesting to note that the "double-normalization" in (6) are exactly the first two steps of an iterative matrix balancing procedure in the literature called the Sinkhorn-Knopp algorithm [25]. Iteratively applying $\varphi_r(\varphi_c(\cdot))$ to a matrix, the Sinkhorn-Knopp scheme produces, upon convergence, a doubly stochastic matrix that is close to the input matrix. The advantage of symmetrizing $\boldsymbol{W}$ by using the full Sinkhorn-Knopp scheme was recently studied by Milanfar [26]. We observe from our numerical experiments that most of the PSNR improvements can be obtained by only carrying out the scheme for two steps, as in (6). Furthermore, in our sampling scheme, once a column is picked, scaling it to have unit sum only adds marginal

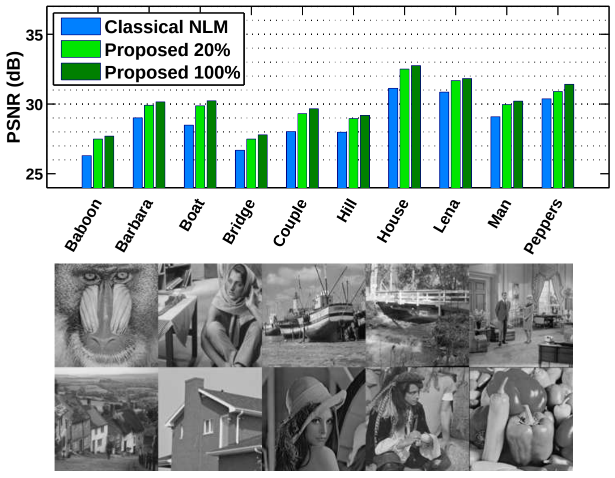

Fig. 5: Denoising PSNRs on 10 images (of size 256×256). Average run time: Classical NLM: 330 sec; Proposed 100%: 330 sec; Proposed 20%: 75 sec.

cost to the total complexity. Thus, the proposed algorithm can benefit from the Sinkhorn-Knopp scheme almost "for free".

We apply the classical NLM and the proposed algorithm to the images shown in Figure 5 and show the PSNR values in the bar chart. Evidently, the proposed method at full sampling performs better than the classical NLM, with an improvement of more than 1.3 dB on average. Even for the case of 20% columns, the proposed method outperforms the classical NLM by 1 dB on average.

4.2. Finite Search Windows

Due to the prohibited computational cost of the full NLM, most practical implementations of NLM either use a limited spatial search window (which in some sense makes non-local means local), or a pre-selection scheme [10]. Our random sampling algorithm can be extended to accommodate either case if we skip the column normalization step considered in Section 4.1: Without column normalization, each row of $\boldsymbol{W}$ can be processed independently and thus the sampling can be performed within the search window. If the image size n is large so that the window size m also has to be large, Theorem 1 still guarantees that for a fixed sampling ratio k/m, the probability of deviation drops exponentially as $m \to \infty$. A trade-off of using such approaches is that we can no longer benefit from the Sinkhorn normalization scheme.

5. CONCLUSION

We presented a randomized algorithm for computing non-local means. The proposed scheme randomly chooses a fraction of the columns of the weight matrix to generate an approximate result. For a fixed sampling ratio, the probability of having large approximation errors decays exponentially with the size of the image, implying that the approximate solution is tightly concentrated around its limit point when the image is large. Additionally, with a column normalizing step, the proposed algorithm is equivalent to the first two steps of the iterative Sinkhorn-Knopp scheme, which has been shown to yield better estimates in many real images. Therefore, in addition to savings in computing time, we find through our numerical experiments that the proposed algorithm also improves the performance of the classical NLM.

6. REFERENCES

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

[2] A. Buades, B. Coll, and J.M. Morel, “Denoising image sequences does not require motion estimation,” in *IEEE Conference on Advanced Video and Signal Based Surveillance*, September 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, pp. 36–51, 2009.

[4] P. Milanfar, “A tour of modern image filtering,” *IEEE Signal Processing Magazine*, vol. 2, 2011.

[5] D. Van De Ville and M. Kocher, “Sure-based non-local means,” *IEEE Signal Process. Lett.*, vol. 16, pp. 973–976, 2009.

[6] C. Kervrann and J. Boulanger, “Optimal spatial adaptation for patch-based image denoising,” *IEEE Trans. Image Process.*, vol. 15, pp. 2866–2878, 2006.

[7] William G. Cochran, *Sampling Techniques, 3rd Edition*, John Wiley & Sons, 3rd edition, Jan. 1977.

[8] R. J. Serfling, “Probability inequalities for the sum in sampling without replacement,” *The Annals of Statistics*, vol. 2, no. 1, pp. 39–48, Jan. 1974.

[9] T. S. Ferguson, *A Course in Large Sample Theory*, vol. 38, Chapman & Hall/CRC, 1996.

[10] M. Mahmoudi and G. Sapiro, “Fast image and video denoising via nonlocal means of similar neighborhoods,” *IEEE Signal Process. Lett.*, vol. 12, pp. 839–842, 12 2005.

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

[12] T. Brox, O. Kleinschmidt, and D. Cremers, “Efficient nonlocal means for denoising of textural patterns,” *IEEE Trans. Image Process.*, vol. 17, pp. 1083–1092, 2008.

[13] 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, 2010.

[14] J. Orchard, M. Ebrahimi, and A. Wong, “Efficient nonlocal-means denoising using the SVD,” in *IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 1732 –1735.

[15] T. Tasdizen, “Principal components for non-local means image denoising,” in *IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 1728 –1731.

[16] 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, September 2011.

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

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

[19] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, “Improved fast Gauss transform and efficient kernel density estimation,” in *International Conference on Computer Vision (ICCV)*, 2003.

[20] J. Darbon, A. Cunha, T. Chan, S. Osher, and G. Jensen, “Fast nonlocal filtering applied to electron cryomicroscopy,” in *IEEE International Symposium on Biomedical Imaging*, 2008, pp. 1331–1334.

[21] J. Wang, Y. Guo, Y.Ying, Y. Liu, and Q. Peng, “Fast non-local algorithm for image denoising,” in *IEEE International Conference on Image Processing (ICIP)*, Oct 2006, pp. 1429–1432.

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

[23] P. Drineas, R. Kannan, and M. Mahoney, “Fast Monte Carlo algorithms for matrices II: Computing low-rank approximations to a matrix,,” *SIAM Journal on Computing*, vol. 36, pp. 158 – 183, 2006.

[24] P. Drineas, R. Kannan, and M. Mahoney, “Fast Monte Carlo algorithms for matrices III: Computing an efficient approximate decomposition of a matrix,” *SIAM Journal on Computing*, vol. 36, pp. 184 – 206, 2006.

[25] R. Sinkhorn and P. Knopp, “Concerning non-negative matrices and doubly-stochastic matrices,” *Pacific Journal of Mathematics*, vol. 21, pp. 343 – 348, 1967.

[26] P. Milanfar, “Symmetrizing smoothing filters,” 2012, Submitted to SIAM Journal on Imaging Science. Available at `http://users.soe.ucsc.edu/~milanfar/publications/journal/SIIMSMay-3-12.pdf`.