PatchLift: Fast and Exact Computation of Patch Distances using Lifting, with Applications to Non-Local Means

Kunal N. Chaudhury

Abstract

In this paper, we propose a fast algorithm called PatchLift for computing distances between patches extracted from a one-dimensional signal. PatchLift is based on the observation that the patch distances can be expressed in terms of simple moving sums of an image, which is derived from the one-dimensional signal via lifting. We apply PatchLift to develop a separable extension of the classical Non-Local Means (NLM) algorithm which is at least $100\times$ faster than NLM for standard parameter settings. The PSNR obtained using the proposed extension is typically close to (and often larger than) the PSNRs obtained using the original NLM. We provide some simulations results to demonstrate the acceleration achieved using separability and PatchLift.

Index Terms

Image denoising, patch-based algorithms, non-local means, patch distance, fast algorithm, separable transform, lifting, moving sum, recursion.

I. INTRODUCTION

The Non-Local Means (NLM) algorithm was introduced by Baudes et al. [7] for denoising natural images corrupted with additive Gaussian noise. Two key innovations of NLM are the effective use of non-local correlations in images, and the use patches (blocks of neighbouring pixels) instead of single pixels to robustly compare photometric similarities. The conceptual simplicity of NLM, coupled with its excellent denoising performance, triggered a huge amount of research on the use of non-local patch-based models for image restoration [9], [13], [12], [10], [27]. Some of

K. N. Chaudhury is with the Imaging Sciences and Algorithms Lab, Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012. Correspondence: kunal@ee.iisc.ernet.in.
these methods provide state-of-the-art results in image denoising for a wide class of natural images. We refer the interested readers to [19], [27] for an exhaustive account of patch-based methods.

The non-local means of a discrete image $f = \{f(i), i \in \mathbb{Z}^2\}$ is given by the weighted average

$$f_{NL}(i) = \frac{\sum_{j \in S(i)} w_{ij} f(j)}{\sum_{j \in S(i)} w_{ij}},$$

(1)

where the weights are set to be

$$w_{ij} = \exp \left( -\frac{1}{h^2} \|P_i - P_j\|^2 \right).$$

(2)

Here patch $P_i$ is the restriction of $f$ to a square grid of length $2K + 1$ centred at $i$, $\|P_i - P_j\|$ is the Euclidean distance between $P_i$ and $P_j$ as points in $\mathbb{R}^{(2K+1)^2}$, $S(i)$ denotes the domain over which the averaging is performed, and $h$ is a smoothing parameter. To exploit non-local correlations in images, the original proposal in [7] was to set $S(i)$ to be the whole image domain. In practice, however, one restricts $S(i)$ to a sufficiently large square window of length $2S + 1$ centred at $i$. Along with the non-local averaging, it is the use of patches that makes NLM more robust than pixel-based neighborhood filters [1], [3], [4].

A drawback of NLM is its high computational complexity. In particular, if each patch is of length $2K + 1$ and the search window is of length $2S + 1$, then the direct computation of (1) requires $O(N^2 S^2 K^2)$ operations for an image of size $N \times N$. Several computational tricks and trade-offs have been proposed to accelerate NLM. These can broadly be classified into the following categories:

- **Neighborhood Selection**: The neighbourhoods are pre-selected based on intensity gradient [8], mean and variance [14], higher moments [15], tree structures [17], and probabilistic termination [24].
- **Dimensionality Reduction**: The patches are projected into a lower dimensional subspace using PCA [18] or SVD [21].
- **Fast Distance Computation**: The patch distances are computed using convolutions and FFTs [11], [25], [28], possibly at the cost of added storage.
- **Efficient Data Structures**: Here NLM is posed as a multi-dimensional filtering and efficient data-structures are used [22], [23].

The NLM formula (1) can be extended to signals of any arbitrary dimension simply by adapting the definition of a patch. In one-dimension, a patch is simply an interval.
In this paper, we present a conceptually simple idea for reducing the complexity of one-dimensional NLM from $O(\text{NSK})$ to $O(\text{NS})$, where $N$ is the length of the signal. The idea is based on the novel observation that the patch distances involved in the non-local means of a one-dimensional signal can be computed using simple moving sums of an image, which is derived from the tensor product of the signal with itself. The complexity of the moving sum is $O(1)$ with respect to the the patch length, and as a result the proposed algorithm requires just $O(\text{NS})$ operations to compute the patch distances required in NLM (cf. section II). The present idea of using moving sums is similar in spirit to the idea of using moving sums and integral images proposed in [11], [20], [25] for fast patch distance computation. However, to the best of our knowledge, the idea of lifting is new. We next apply the proposed algorithm to develop a fast separable extension of NLM for images in section III. This was partly motivated by the work on separable bilateral filtering and NLM [6], [30]. The complexity of the fast separable NLM proposed in this paper is $O(N^2S)$ for an image with $N^2$ pixels, which is substantially smaller than the $O(N^2S^2K^2)$ complexity of the original NLM. In practice, the proposed extension turns out to be at least $100\times$ faster and yields peak-signal-to-noise ratios (PSNRs) that are typically within $0.5$ dB of the PSNRs obtained using the original NLM. In fact, what we find quite intriguing is that the PSNR obtained using separable NLM is at times significantly larger than the PSNR obtained using the original NLM (optimized with respect to the parameters in either case); see figure 3 and table II.

II. FAST PATCH DISTANCES FOR NLM

We now explain the main idea for a one-dimensional signal, which will be relevant in the context of separable NLM in section III. Later, in section IV, we will briefly discuss how the idea can be extended to higher-dimensions.

Suppose we have a one-dimensional signal $f = \{f(i), i \in \mathbb{Z}\}$. A patch of length $2K + 1$ centred at $i$ is simply the collection of samples $\{f(i + k), -K \leq k \leq K\}$. Consider two such patches centered at $i$ and $j$. The squared Euclidean distance between these patches is

$$
\rho_{ij}^2 = \sum_{k=-K}^{K} \left( f(i + k) - f(j + k) \right)^2.
$$

(3)
We require $O(K)$ operations to compute each $\rho_{ij}$. In one-dimensional NLM, the weights are given by $w_{ij} = \exp(-\rho_{ij}^2/h^2)$, and

$$f_{NL}(i) = \frac{\sum_{j=-S}^{S} w_{ij} f(j)}{\sum_{j=-S}^{S} w_{ij}}. \quad (4)$$

Note that, for any given $i$, we only required $2S+1$ distances, namely $\rho(i, i-S), \ldots, \rho(i, i+S)$. Therefore, the total complexity is $O(NSK)$ for a signal of length $N$. The question is whether it is possible to reduce this complexity somehow by exploiting the overlap between adjacent patches? We now show that this is indeed the case.

A. PatchLift

The terms on the right in (3) involve products of the signal samples. Our key observation that this non-linear dependence can be transformed into a linear one by massaging the samples. In particular, we first construct a tensor $F$ on $\mathbb{Z} \times \mathbb{Z}$ defined as

$$F(i, j) = f(i) f(j) \quad i, j \in \mathbb{Z}. \quad (5)$$

We refer to $F$ as the tensor-lift (or simply lift) of $f$. We then define

$$\bar{F}(i, j) = \sum_{k=-K}^{K} F(i+k, j+k). \quad (6)$$

Note that this is very a special form of convolution, namely the moving sums of $F$ along its sub-diagonals. Using (5) and (6), we can write (3) as

$$\rho_{ij}^2 = \bar{F}(i, i) + \bar{F}(j, j) - 2\bar{F}(i, j). \quad (7)$$

Following (7), we will refer to $\bar{F}$ as the kernel of the patch distances. Each distance computation requires just three samples of this kernel, one multiplication, and two additions. Moreover, since $F$ is symmetric, it follows from (7) that $\bar{F}$ is also symmetric: $\bar{F}(i, j) = \bar{F}(j, i)$. Therefore, it suffices to compute those $\bar{F}(i, j)$ for which, say, $i \geq j$. This reduces the computation by half.

At first sight it seems that we have not achieved anything, and that (7) is simply another way of writing (3). However, note that we have effectively transferred the non-linearity in (3) onto the lift, and kernel can now be computed efficiently using moving sums of the lift. Indeed, it is a well-known fact that moving sums in one
Fig. 1. Denoising results for a montage. We used additive Gaussian noise of standard deviation 40 to corrupt the montage. The PSNRs and run times are shown in bold. The proposed method is 200× faster than standard NLM, while being more than 2 dB better in terms of PSNR (see text for details of the algorithm and the parameter settings). Notice that the edges are visibly sharper for the proposed method, although some line artefacts are introduced by the separable filtering. The SSIM indices for (c) and (d) are 0.81 and 0.80 respectively. The computations were performed using MATLAB on a 3.40 GHz Intel quad-core machine.

dimension can be implemented using just two operations per sample using recursion [2]. In particular, we see from (6) that

$$\mathbf{F}(i+1, j+1) = \mathbf{F}(i,j) + \mathbf{F}(i+1 + K, j+1 + K) - \mathbf{F}(i - K, j - K).$$

Thus given $\mathbf{F}(i, j)$, we can compute $\mathbf{F}(i+1, j+1)$ using just two additions, and so on. We will see shortly how this nicely fits with the structure of the distances required in NLM.
The above lifting trick was motivated by the idea of matrix-lifting used in semidefinite relaxation in which a quadratic form on a vector variable is transformed into a linear form on a matrix variable that has been “lifted” from the vector; see, e.g., [29], [31]. We will refer to the present idea of computing patch distances using lifting as PatchLift.

Fig. 2. This shows the two-dimensional lift $F$ arising from a one-dimensional signal. The red arrow shows the direction of the recursive moving sums (from the top left to the bottom right) for performing NLM with a window size $S$. Each recursive update requires just one addition and one subtraction along the direction of the arrow. Note that we do not need to store the entire lift. Only those $F(i,j)$ for which $|i-j| \leq S$ are required (marked as a diagonal band). Similarly, we only need to store only those $F(i,j)$ for which $|i-j| \leq S$. Thus both storage and time complexity is $O(NS)$.

B. Fast NLM using PatchLift

As noted earlier, NLM requires only the knowledge of the distances between patches that are spatially close. For a search window of length $2S + 1$, the relevant elements of the lift matrix and the kernel are shown in figure 2. In particular, for a signal of length $N$, we are required to compute the kernel at the following positions $\{(i,j): 1 \leq i \leq N, i - S \leq j \leq i + S\}$ (with boundaries handled appropriately). Using relation (8), one can compute these $O(NS)$ elements using $O(NS)$ operations. For completeness, the Matlab script for computing the kernel from the signal is provided below (note that $K$ is passed as an argument just for padding the boundary).
function barF = ComputeKernel (f, S, K)
N = length(f);
F = f*f';
padF = padarray(F, [K,K], 'symmetric');
barF = zeros(N,N);
for i = 1 : S+1
    barF(i,1) = sum( diag(padF(i+K+(-K:K), 1+K+(-K:K))) );
    barF(1,i) = barF(i,1); p = 1;
    while (i+p ¡= N)
        temp = barF(i+p-1,p) + padF(i+p+2*K,1+p+2*K)-padF(i+p-1,p);
        barF(i+p,1+p) = temp;
        barF(1+p,i+p) = temp;
        p = p+1;
    end
end

Now, given the kernel and hence the patch distances, we can compute (4) using another $O(NS)$ operations. Thus, the total complexity of computing the NLM of a one-dimensional signal using PatchLift is $O(NS)$, while the complexity of the original NLM is $O(NSK)$ . For patches of moderate-to-large length $K$, this reduction can be significant. We will refer to the proposed implementation of NLM as PatchLift-NLM.

III. SEPARABLE NON-LOCAL MEANS

We will briefly discuss in section IV how PatchLift could be extended to higher dimensions. In particular, we can use PatchLift for efficiently computing the patch distances in (2) for images. We will, however, take a different route which appears to provide an optimal trade-off between the denoising result and the run time. In particular, we consider a separable extension of the one-dimensional NLM to images in which we first apply PatchLift-NLM on the rows of the image, and then on the columns of the intermediate image. We note that result obtained using this method is generally not the same as the NLM in (1); e.g., see figures 1-3 in [30] for a comparison of the two approaches. Moreover, different results are obtained by interchanging the order of row and column operations. While we do not report it here, the denoising performances turn out to be similar for the different orderings
(within 0.3 dB), and its is often hard to visually distinguish the results. The only subtle difference that we observe is the appearance of line artefacts along horizontal and vertical directions due to the separable filtering. In this regard, we noticed that by averaging the images obtained from the row-column and the column-row operations, the final image tends to have a better visual appearance (less line artefacts) and a slightly higher PSNR. Henceforth, we will refer to the method of separately performing row-column and column-row PatchLift-NLM and then averaging the results as separable NLM (in short, S-NLM).

| Method | 32² | 64² | 128² | 256² | 512² | 1024² |
|--------|-----|-----|------|------|------|-------|
| NLM    | 1.2s| 6.5s| 25s  | 104s | 431s | 28min |
| S-NLM  | 20ms| 50ms| 150ms| 54ms | 3s   | 19s   |

Note that the total complexity of S-NLM for an $N \times N$ image is $4N \times O(NS) = O(N^2S)$. This is a substantial reduction over the $O(N^2S^2K^2)$ complexity of NLM. In table I, we compare the run times of S-NLM and NLM for the typical parameter setting $S = 10$ and $K = 3$ [7]. We see an acceleration of at least $100\times$ for the different image sizes.

We next compare the denoising performance of S-NLM and NLM for various test images with additive white Gaussian noise. The PSNRs and structural similarity indices (SSIM) [5] obtained using NLM and S-NLM at different noise levels are reported in table II. We see that the denoising performances are close in terms of these indices. Interestingly, the PSNRs obtained using S-NLM is higher most for most of the cases. However, due to the presence of line artefacts, the SSIM of NLM is often slightly higher than that of S-NLM. We have evidence to believe that a small uniform blur at the end of the processing can diffuse the lines arising from the separable processing. A full-blown discussion of this point is, however, beyond the scope of the paper and will be followed up elsewhere. We note that the results in table II are consistent with the results in [6], [30] where it was shown that the denoising performance of the separable extensions of the bilateral and NLM filter.
Comparison of NLM and fast separable NLM (S-NLM) in terms of PSNR and SSIM at noise levels $\sigma = 5, 10, 30, 50, 80, 100$ (all results are averaged over 10 noise realizations). For both methods $S = 10$ and $K = 3$; $h = 10\sigma$ for NLM and $h = 3\sigma$ for S-NLM. The sizes of Peppers and House are $256^2$, while Barbara and Lena are $512^2$.

| Image   | Method | PSNR (dB) | PSNR (dB) | PSNR (dB) | PSNR (dB) |
|---------|--------|-----------|-----------|-----------|-----------|
|         |        |           |           |           |           |
| Peppers | NLM    | 36.49     | 32.34     | 24.99     | 21.97     | 19.94     | 19.43     |
|         | S-NLM  | 36.70     | 32.62     | 26.57     | 23.47     | 20.92     | 20.26     |
| House   | NLM    | 36.98     | 34.23     | 26.29     | 24.11     | 22.36     | 21.75     |
|         | S-NLM  | 37.45     | 34.09     | 28.53     | 25.41     | 23.12     | 22.26     |
| Barbara | NLM    | 36.62     | 32.42     | 24.91     | 22.61     | 21.35     | 20.86     |
|         | S-NLM  | 36.20     | 31.88     | 25.26     | 22.77     | 21.48     | 20.95     |
| Lena    | NLM    | 36.86     | 33.22     | 27.39     | 25.21     | 23.62     | 23.08     |
|         | S-NLM  | 37.15     | 33.47     | 28.12     | 25.75     | 23.88     | 23.14     |

| Image   | Method | SSIM (%) | SSIM (%) | SSIM (%) | SSIM (%) |
|---------|--------|----------|----------|----------|----------|
|         |        |          |          |          |          |
| Peppers | NLM    | 93.64    | 88.96    | 75.57    | 67.51    | 58.50    | 54.79    |
|         | S-NLM  | 94.54    | 90.25    | 77.63    | 67.39    | 55.06    | 49.53    |
| House   | NLM    | 90.57    | 86.96    | 77.08    | 69.59    | 62.31    | 58.31    |
|         | S-NLM  | 92.62    | 87.76    | 78.30    | 68.71    | 57.39    | 50.45    |
| Barbara | NLM    | 94.75    | 90.01    | 71.30    | 60.69    | 52.66    | 49.18    |
|         | S-NLM  | 95.01    | 90.31    | 70.86    | 57.86    | 47.77    | 42.80    |
| Lena    | NLM    | 91.72    | 86.87    | 76.42    | 70.81    | 64.60    | 61.43    |
|         | S-NLM  | 93.04    | 88.01    | 76.29    | 67.83    | 57.17    | 51.58    |

are close to their original counterparts. However, we note that the separable NLM in [30] is different from the one proposed here in that a spatial filter and bilateral weights (corresponding to $K = 1$ in (2)) are used in [30]. Of course, the key difference here is that we use PatchLift to accelerate the one-dimensional NLMs.

IV. DISCUSSION

We presented a separable extension of the original non-local means filter of Buades et al. [7] and showed that its denoising performance is close to that of [7] while being $100\times$ faster. The proposed algorithm has a straightforward extension to video and volume data processing.

We note that the bilateral filter is a special case of the non-local means corresponding to $K = 1$. While the complexity of our algorithm for $K = 1$ continues to
Fig. 3. Denoising results for Peppers (256$^2$) and House (256$^2$) using NLM and S-NLM at $\sigma = 30$. The corresponding PSNR and SSIM are given in table II. The run times are reported in table I.

be $O(N^2S)$, one can further reduce the complexity to $O(N^2)$ using approximations [16], [26]. This leads to the question as to whether it is possible to design similar approximation-based algorithms for NLM of complexity $O(N^2K)$ or even $O(N^2)$? One could then use large search windows, possibly the while image [7].

Before concluding, we briefly discuss how PatchLift can be extended to $d$-dimensional signals and, in particular, images and volumes. Suppose the signal $f$ is defined on the grid $\mathbb{Z}^d$, where the coordinates of a point $i \in \mathbb{Z}^d$ is denoted by $i = (i_1, \ldots, i_d)$. We use $\Box_K$ to denote the hypercube centered at the origin whose edges are of length $2K + 1$. In particular, $\Box_K$ consists of those grid points $i$ for which $-K \leq i_\ell \leq K$
for $1 \leq \ell \leq d$. A patch of size $K$ centred at $i$ is defined to the collection of signal samples $\{f(i+k), k \in \square_K\}$. Based on this notation, the squared distance between two patches of size $K$ centered at $i$ and $j$ is

$$\rho(i, j)^2 = \sum_{k \in \square_K} (f(i+k) - f(j+k))^2.$$ 

The complexity of computing $\rho(i, j)$ is $O(K^d)$, which is proportional to the number of points in $\square_K$. As in the one-dimensional case, we construct a tensor $F$ on $\mathbb{Z}^d \times \mathbb{Z}^d$ given by $F(i, j) = f(i)f(j)$, and let

$$F(i, j) = \sum_{k \in \square_K} F(i+k, j+k).$$

This again just are moving sums over the sub-diagonals of the tensor. Let $e$ denote the vector whose components are 1 or 0. Then it can be verified that

$$F(i+e, j+e) = F(i, j) + \text{sum of } O(K^{d-1}) \text{ samples of } F.$$ 

This is essentially because the non-overlapping regions between the shifted hyper-cubes have $K^d - (K-1)^d = O(K^{d-1})$ points. Thus, given $F(i, j)$, we can compute $F(i+e, j+e)$ using $O(K^{d-1})$ additions. We skip the details, but it is not hard to see that PatchLift allows us to reduce the complexity of NLM in $d$-dimensions from $O(N^dS^dK^d)$ to $O(N^dS^dK^{d-1})$. It is also clear that we only need to store $O(N^dS^d)$ elements of $F$ and $F$ out of the total of $O(N^{2d})$ elements to compute the patch distances. To fully optimize the implementation, one needs to efficiently store and manipulate the sparse tensors $F$ and $F$. The details of such a scheme would be presented in a future work.

REFERENCES

[1] L. P. Yaroslavsky, Digital Picture Processing. Secaucus, NJ: Springer-Verlag, 1985.
[2] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, Discrete-Time Signal Processing, Englewood Cliffs: Prentice-Hall, 1989.
[3] S. M. Smith, J. M. Brady, “SUSAN - A new approach to low level image processing,” International Journal of Computer Vision, vol. 23, no. 1, pp. 45-78, 1997.
[4] C. Tomasi, R. Manduchi, “Bilateral filtering for gray and color images,” IEEE International Conference on Computer Vision, pp. 839-846, 1998.
[5] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, 2004.
[6] T. Q. Pham and L. J. Van Vliet, “Separable bilateral filtering for fast video preprocessing,” IEEE International Conference on Multimedia and Expo (ICME), 2005.
[7] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 60-65, 2005.
[8] M. Mahmoudi and G. Sapiro, “Fast image and video denoising via nonlocal means of similar neighborhoods,” IEEE Signal Processing Letters, vol. 12, no. 12, pp. 839-842, 2005.
[9] C. Kervrann, J. Boulanger, “Optimal spatial adaptation for patch-based image denoising,” IEEE Transactions on Image Processing, vol. 15(10), 2866-2878, 2006.
[10] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” IEEE Transactions on Image Processing, vol. 15, no. 12, pp. 3736-3745, 2006.
[11] J. Wang, Y. Guo, Y. Ying, Y. Liu, and Q. Peng, “Fast non-local algorithm for image denoising,” Proc. IEEE International Conference on Image Processing, pp. 1429-1432, 2006.
[12] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, “Image denoising by sparse 3-D transform-domain collaborative filtering,” IEEE Transactions on Image Processing, vol. 16, pp. 2080-2095, 2007.
[13] G. Gilboa, S. Osher, “Nonlocal operators with applications to image processing,” Multiscale Modeling and Simulation, vol. 7, no. 3, 1005-1028, 2008.
[14] P. Coupe, S. Prima, P. Hellier, C. Kervrann, and C. Barillot, “An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images,” IEEE Transactions on Medical Imaging, vol. 27, no. 4, pp. 425-441, 2008.
[15] A. Dauwe, B. Goossens, H. Luong, and W. Philips, “A fast non-local image denoising algorithm,” Proc. SPIE Electronic Imaging, vol. 6812, pp. 1331-1334, 2008.
[16] F. Porikli, “Constant time $O(1)$ bilateral filtering,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-8, 2008.
[17] T. Brox, O. Kleinschmidt, and D. Cremers, “Efficient non-local means for denoising of textural patterns,” IEEE Transactions on Image Processing, vol. 17, no. 7, pp. 1083-1092, 2008.
[18] T. Tasdizen, “Principal components for non-local means image denoising,” Proc. IEEE International Conference on Image Processing (ICIP), pp. 1728-1731, 2008.
[19] A. Buades, B. Coll, J. M. Morel, “Image denoising methods. A new nonlocal principle,” SIAM Review, vol. 52, pp. 113-147, 2010.
[20] J. Darbon, A. Cunha, T. F. Chan, S. Osher, and G. J. Jensen, “Fast nonlocal filtering applied to electron cryomicroscopy,” Proc. IEEE International Symposium on Biomedical Imaging (ISBI), pp. 1331-1334, 2008.
[21] J. Orchard, M. Ebrahimi, and A. Wong, “Efficient nonlocal-means denoising using the SVD,” Proc. IEEE International Conference on Image Processing (ICIP), pp. 1732-1735, 2008.
[22] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, “Gaussian KD-trees for fast high-dimensional filtering,” Proc. ACM SIGGRAPH, 2009.
[23] A. Adams, J. Baek, and A. Davis, “Fast high-dimensional filtering using the permutohedral lattice,” Proc. EuroGraphics, 2010.
[24] R. Vignesh, O. T. Byung, and C. C. Kuo, “Fast non-local means (NLM) computation with probabilistic early termination,” IEEE Signal Processing Letters, vol. 17, no. 3, pp. 277-280, 2010.
[25] L. Condat, “A simple trick to speed up and improve the non-local means,” Research Report hal-00512801, 2010.
[26] K. N. Chaudhury, D. Sage, and M. Unser, “Fast bilateral filtering using trigonometric range kernels,” IEEE Transactions on Image Processing, vol. 20, no. 12, pp. 3376-3382, 2011.
[27] P. Chatterjee, P. Milanfar, “Patch-based near-optimal image denoising,” IEEE Transactions on Image Processing, vol. 21, no. 4, pp. 1635-1649, 2012.
[28] C. Deledalle, V. Duval, J. Salmon, “Non-local methods with shape-adaptive patches (NLM-SAP),” Journal of Mathematical Imaging and Vision, vol. 43, no. 2, pp. 103-120, 2012.

[29] E. J. Candes, T. Strohmer, and V. Voroninski, “PhaseLift: Exact and stable signal recovery from magnitude measurements via convex programming,” Communications on Pure and Applied Mathematics, vol. 66, no. 8, pp. 1241-1274, 2013.

[30] Y. S. Kim, H. Lim, O. Choi, K. Lee, J. D. K. Kim and J. D. Kim, “Separable bilateral nonlocal means,” IEEE International Conference on Image Processing (ICIP), pp. 1513-1516, 2011.

[31] K. N. Chaudhury, Y. Khoo, and A. Singer, “Global registration of multiple point clouds using semidefinite programming,” arXiv:1306.5226 [cs.CV], 2013.