IMAGE DENOISING USING NL-MEANS VIA SMOOTH PATCH ORDERING

Idan Ram[1], *Michael Elad*[2] *and Israel Cohen*[1]

[1] Department of Electrical Engineering
Technion - Israel Institute of Technology
Technion City, Haifa 32000, Israel
{idanram@tx, icohen@ee}.technion.ac.il

[2] Department of Computer Science
Technion - Israel Institute of Technology
Technion City, Haifa 32000, Israel
elad@cs.technion.ac.il

ABSTRACT

In our recent work we proposed an image denoising scheme based on reordering of the noisy image pixels to a one dimensional (1D) signal, and applying linear smoothing filters on it. This algorithm had two main limitations: 1) It did not take advantage of the distances between the noisy image patches, which were used in the reordering process; and 2) the smoothing filters required a separate training set to be learned from. In this work, we propose an image denoising algorithm, which applies similar permutations to the noisy image, but overcomes the above two shortcomings. We eliminate the need for learning filters by employing the nonlocal means (NL-means) algorithm. We estimate each pixel as a weighted average of noisy pixels in union of neighborhoods obtained from different global pixel permutations, where the weights are determined by distances between the patches. We show that the proposed scheme achieves results which are close to the state-of-the-art.

Index Terms— patch-based processing, traveling salesman, pixel permutation, denoising.

1. INTRODUCTION

In recent years, many image denoising methods achieved high quality results by operating on local patches, and exploiting interrelations between them. There are various ways in which the relations between patches are taken into account by different algorithms: weighted averaging of pixels with similar surrounding patches, as the NL-means algorithm does [1], clustering the patches into disjoint sets and treating each set differently, as performed in [2], [3], [4], [5], seeking a representative dictionary for the patches and using it to sparsely represent them, as practiced in [6], [7] and [8], gathering groups of similar patches and applying a sparsifying transform on them [8], [9], [10], [11]. A common theme to many of these methods is the expectation that every patch taken from the image may find similar ones extracted elsewhere in the image. Put more broadly, the image patches are believed to exhibit a highly-structured geometrical form in the embedding space they reside in. Thus, non-local processing schemes can achieve better recovery through joint treatment of similar patches.

In our recent work [12] we took a different approach, and proposed a denoising scheme which consists of reordering the noisy image pixels to *what should be* a 1D regular signal, and applying it linear smoothing filters. We applied several permutations to the image, each was obtained by calculating distances between the noisy image patches, and ordering them such that they were chained in the shortest possible path, essentially solving the traveling salesman problem [13]. We note that similar permutations were employed in [10] and [11] to construct image-adaptive wavelet transforms, which were used for image denoising. The algorithm proposed in [12] had two main limitations: 1) the distances between the patches were not employed in the denoising scheme, although they carry additional information regarding patch similarity; 2) the smoothing filters required a separate training set to be learned from.

In this paper we embark from our earlier work as reported in [12], and apply similar permutations to the noisy image. However, we eliminate the need for learning filters by employing the NL-means algorithm [1], which estimates each pixel as a weighted average of noisy pixels in a surrounding neighborhood. We propose to replace this neighborhood with a union of neighborhoods obtained from different global pixel permutations, and determine the weights using the distances between the patches. We demonstrate the performance of the proposed image denoising scheme, and show that combined with a patch classification and a subimage averaging schemes, it outperforms both the NL-means algorithm and the algorithm presented in [12], and achieves results which are close to those of the BM3D algorithm [9].

The paper is organized as follows: In Section 2 we introduce the basic image denoising scheme. In Section 3 we explain how the performance of the basic scheme can be improved using patch classification, subimage averaging, and

This research was supported by Japan Technion Society Research Fund and by Robert H. Hillman Foundation for Global Security - Collaboration Technion and University Northeastern, and has also received funding from the European Research Council under the ERC grant agreement no. 320649.

the application of a second iteration. In Section 4 we present experimental results that demonstrate the advantages of the proposed scheme.

2. IMAGE DENOISING USING PATCH ORDERING

2.1. Problem Formulation

Let $\mathbf{Y}$ be an image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{Z}$ be its noisy version

$$\mathbf{Z} = \mathbf{Y} + \mathbf{V}. \tag{1}$$

$\mathbf{V}$ denotes an additive white Gaussian noise independent of $\mathbf{Y}$ with zero mean and variance σ^2. Also, let $\mathbf{z}$ and $\mathbf{y}$ be the column stacked representations of $\mathbf{Z}$ and $\mathbf{Y}$, respectively. In our previous work [12], we employed K different permutations matrices $\mathbf{P}_k$ of size $N \times N$ to reconstruct $\mathbf{y}$ from $\mathbf{z}$. When each such matrix is applied to the unknown target signal $\mathbf{y}$, it produces a smooth signal $\mathbf{y}_k^p = \mathbf{P}_k\mathbf{y}$. Therefore if we apply the same permutation to $\mathbf{z}$, we attain the prior knowledge that the clear version of the obtained signal $\mathbf{z}_k^p = \mathbf{P}_k\mathbf{z}$ is smooth, which can simplify the reconstruction process. The denoising algorithm in [12] recovers $\mathbf{y}$ by applying linear smoothing filters to $\mathbf{z}_k^p$, and applying the inverse permutation $\mathbf{P}_k^{-1}$ to the result.

The algorithm proposed in [12] has two main limitations. First, it ignores altogether the distances between the patches. These distances carry additional information, because they have a key role in the construction of the matrices $\mathbf{P}_k$, as will be explained in Section 2.2. Additionally, the smoothing filters require a separate training set to be learned from. Our goal is to develop an image denoising algorithm which employs the matrices $\mathbf{P}_k$, takes into account the distances between the patches, and do not require filter learning. To this end we modify the NL-means algorithm, which takes into account these distances in the denoising process. We next describe how we construct the reordering matrices $\mathbf{P}_k$. Afterwards we present our image denoising scheme.

2.2. Building a Permutation Matrix $\mathbf{P}$

We wish to design a matrix $\mathbf{P}$ which produces a smooth signal when applied to the target image $\mathbf{y}$. When the image $\mathbf{Y}$ is known, the optimal solution would be to reorder it as a vector, and then apply a simple *sort* operation on the obtained vector. However, we are interested in the case where we only have the noisy image $\mathbf{Z}$. Therefore, we seek a suboptimal ordering operation, using patches from this image.

Let y_i and z_i denote the ith samples in the vectors $\mathbf{y}$ and $\mathbf{z}$, respectively. We denote by $\mathbf{x}_i$ the column stacked version of the $\sqrt{g} \times \sqrt{g}$ patch around the location of z_i in $\mathbf{Z}$. We refer to the patches as points in $\mathbb{R}^g$, and assume that a small Euclidean distance between the two points $\mathbf{x}_i$ and $\mathbf{x}_j$ suggests proximity between the clear versions of their center pixels y_i and y_j. Thus, we shall try to reorder the points $\mathbf{x}_i$ so that they form a smooth path, hoping that the corresponding reordered 1D signal $\mathbf{y}^p$ will also become smooth. The "smoothness" of the reordered signal $\mathbf{y}^p$ can be measured using its total-variation measure

$$\|\mathbf{y}^p\|_{TV} = \sum_{j=2}^{N} |y^p(j) - y^p(j-1)|. \tag{2}$$

Let $\{\mathbf{x}_j^p\}_{j=1}^N$ denote the points $\{\mathbf{x}_i\}_{i=1}^N$ in their new order. Then by analogy, we measure the "smoothness" of the path through the points $\mathbf{x}_j^p$ by the measure

$$X_{TV}^p = \sum_{j=2}^{N} \|\mathbf{x}_j^p - \mathbf{x}_{j-1}^p\|. \tag{3}$$

Minimizing X_{TV}^p comes down to finding the shortest path that passes through the set of points $\mathbf{x}_i$, visiting each point only once. This can be regarded as an instance of the traveling salesman problem [13], which can become very computationally expensive for large sets of points. We choose a simple approximate solution, which is to start from a random point and then continue from each point $\mathbf{x}_{j_0}$ to its nearest neighbor $\mathbf{x}_{j_1}$ with a probability $p_1 = \alpha \exp\left(-\frac{\|\mathbf{x}_{j_0} - \mathbf{x}_{j_1}\|^2}{g\epsilon}\right)$, or to its second nearest neighbor $\mathbf{x}_{j_2}$ with a probability $p_2 = 1 - p_1 = \alpha \exp\left(-\frac{\|\mathbf{x}_{j_0} - \mathbf{x}_{j_2}\|^2}{g\epsilon}\right)$, where ϵ is a design parameter, and $\mathbf{x}_{j_1}$ and $\mathbf{x}_{j_2}$ are taken from the set of unvisited points.

We restrict the nearest neighbor search performed for each patch to a surrounding square neighborhood which contains $B \times B$ patches. When no unvisited patches remain in that neighborhood, we search for the nearest neighbor among all the unvisited patches in the image. This restriction decreases the overall computational complexity, and our experiments show that with a proper choice of B it also leads to improved results. The permutation applied by the matrix $\mathbf{P}$ is defined as the order in the found path. We obtain K different matrices $\mathbf{P}_k$ by simply running the proposed ordering solver K times, and the randomness (both in the initialization and in assigning the neighbors) leads to different permutation results.

2.3. The Basic Denoising Scheme

We start by calculating K permutation matrices $\mathbf{P}_k$ from the image patches $\mathbf{x}_j$. We apply these matrices to $\mathbf{z}$ and obtain $\mathbf{z}_k^p = \mathbf{P}_k\mathbf{z}$. We wish to modify the NL-means algorithm [1] so it will make use of these signals. The NL-means algorithm estimates each pixel $y[n]$ as a weighted average of pixels in $\mathbf{Z}$, which reside in a square neighborhood S_n^{NL} surrounding $z[n]$. The weights are determined by the distances between the patch surrounding the estimated pixel and the patches surrounding the pixels in S_n^{NL}. More formally,

$$\hat{y}[n] = \frac{1}{D_n} \sum_{m \in S_n^{NL}} z[m] w_{n,m} \tag{4}$$

where the weights $w_{m,n}$ and D_n satisfy

$$w_{n,m} = \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}_m\|^2}{g\gamma}\right),\ D_n = \sum_{m\in S_n^{NL}} w_{n,m}. \quad (5)$$

Let $\mathbf{p}_k$ denote a vector containing the permutation of the pixel indices applied by the matrix $\mathbf{P}_k$. We next use the permutations to construct for each pixel $z[n]$ a neighborhood S_n. Let j_n^k be the index of the pixel $y[n]$ in $\mathbf{y}_k^p$, i.e. $y_k^p[j_n^k] = y[n]$. As $\mathbf{y}_k^p$ should be smooth, $y_k^p[j_n^k]$ should be close to its Q neighboring pixels, and therefore we choose to estimate it from their noisy versions. Thus, we first define the neighborhood $S_{k,n}$ of $z[n]$ as the set of indices

$$S_{k,n} = \{p_k[j_n^k - \frac{Q}{2}], \dots, p_k[j_n^k + \frac{Q}{2}]\}. \quad (6)$$

Then we define the total neighborhood of $z[n]$ as

$$S_n = \bigcup_{k=1}^{K} S_{k,n}. \quad (7)$$

We obtain our proposed image denoising algorithm by replacing the neighborhood S_n^{NL} with S_n in (4) and (5). As can be seen, unlike the algorithm in [12] we do make use of the distances between the patches, and we do not require a preliminary filter-learning stage. We next describe how the results obtained with our method can be further improved.

3. IMPROVING THE DENOISING RESULTS

We employ the three following methods in order to further improve the denoising results.

3.1. Patch classification

Similarly to our previous work [12], we improve our results by treating smooth areas in the image differently than areas with edges or texture. We first divide the patches into two sets: S_s - which contains smooth patches, and S_e - which contains patches with edges or texture. Let std$(\mathbf{x}_i)$ denote the standard deviation of the patch $\mathbf{x}_i$ and let C be a scalar design parameter. Then we use the following classification rule: if std$(\mathbf{x}_i) < C\sigma$ then $\mathbf{x}_i \in S_s$, otherwise $\mathbf{x}_i \in S_e$. We next divide the image $\mathbf{z}$ into two signals: $\mathbf{z}_s$ - which contains the pixels corresponding to the smooth patches, and $\mathbf{z}_e$ - which contains the pixels corresponding to the patches with edges and texture. We apply the denoising scheme described above to the signals $\mathbf{z}_s$ and $\mathbf{z}_e$, with the sets of patches S_s and S_e, and two sets of parameters Q_s, γ_s and Q_e, γ_e for the NL-means, respectively. We recover the two signals $\hat{\mathbf{y}}_s$ and $\hat{\mathbf{y}}_e$ from $\mathbf{z}_s$ and $\mathbf{z}_e$, respectively, and obtain the final estimate $\hat{\mathbf{y}}$ by returning the pixels in each signal to their original place in the image canvas.

3.2. Subimage Averaging

Let $\mathbf{X}$ be an $g \times (N_1 - \sqrt{g}+1)(N_2 - \sqrt{g}+1)$ matrix, containing column stacked versions of all the $\sqrt{g} \times \sqrt{g}$ patches inside the image $\mathbf{Z}$. We extract these patches column by column, starting from the top left-most patch. When we calculated each matrix $\mathbf{P}_k$ as described in Section 2.2, we assumed that each patch is associated only with its middle pixel. Therefore $\mathbf{P}_k$ was designed to reorder the signal composed of the middle points in the patches, which reside in the middle row of $\mathbf{X}$. However, we can alternatively choose to associate all the patches with a pixel located in a different position, e.g., the top left pixel in each patch. This means that the matrices $\mathbf{P}_k$ can be used to reorder any one of the signals located in the rows of $\mathbf{X}$. These signals are the column stacked versions of all the n subimages of size $(N_1 - \sqrt{g} + 1) \times (N_2 - \sqrt{g} + 1)$ contained in the image $\mathbf{Z}$. We denote these subimages by $\tilde{\mathbf{Z}}_j$, $j = 1, 2, \dots, g$.

Similarly to [12], in order to improve the quality of the recovered image we utilize all the g subimages of a noisy image in its denoising process. Let $\tilde{\mathbf{z}}_j$ be the column stacked version of $\tilde{\mathbf{Z}}_j$. Then we apply the proposed denoising scheme, with the patch classification described above, to each of the subimages $\tilde{\mathbf{z}}_j$, and obtain reconstructed subimages $\hat{\mathbf{y}}_j$. We next reconstruct the image $\mathbf{y}$ from all the subimages $\hat{\mathbf{y}}_j$ by plugging each subimage into its original place in the image canvas and averaging the different values obtained for each pixel.

3.3. Applying a Second Iteration

We can further improve the quality of the recovered image by applying the noisy image a second iteration of our proposed scheme, similarly to the algorithms proposed in [9] and [12]. In the second iteration, all the processing stages remain the same, but the applied permutations, the standard deviations of the patches, and the weights $w_{m,n}$, are all calculated using patches extracted from the first iteration clean result.

4. EXPERIMENTAL RESULTS

In order to assess the performance of the proposed image denoising scheme we apply it to noisy versions of the images Lena, Barbara and House, with noise standard deviations $\sigma = 10, 25, 50$. In all our experiments we used $K = 10$ permutation matrices, which resulted in a PSNR gain of 1.1 to 2.2 dB, compared to the case where only one matrix is used. The rest of the parameters employed by the proposed scheme for the three noise levels are shown in Table 1. For comparison, we also apply the NL-means and BM3D [9] algorithms, and two iterations of the algorithm proposed in [12], to the noisy images. The PSNR values of the results obtained with these methods, and with two iterations of our scheme are shown in Table 2. The noisy and recovered Barbara and House images, obtained with two iterations of our scheme for $\sigma = 25$ and

Table 1: Parameters used in the denoising experiments.

σ	Iter	$\sqrt{g}$	B	C	ϵ	Q_s	Q_e	γ_s	γ_e
10	1	7	31	1.2	10	9	5	3.3	1.7
	2	4	231	1.1	10^3	33	3	0.4	1.4
25	1	12	31	1.1	10^2	11	5	4.1	1.7
	2	4	131	0.3	10^7	71	11	0.3	0.5
50	1	16	31	1.1	10^2	11	5	5	5.5
	2	6	141	0.1	10^3	91	19	0.2	0.3

Table 2: Denoising results (PSNR in dB) of noisy versions of the images Lena, Barbara and House, obtained with the NL-means and BM3D algorithms, two iterations of the algorithm proposed in [12], and two iterations of the proposed (prop.) scheme. For each image and noise level the best result is highlighted.

Image	Method	σ/PSNR		
		10/28.14	25/20.18	50/14.16
Lena	NL-means	34.73	30.43	26.84
	BM3D	**35.93**	**32.08**	29.05
	[12] (2 iter.)	35.39	31.80	28.96
	prop. (1 iter.)	35.41	31.50	28.46
	prop. (2 iter.)	35.78	32.03	**29.16**
Barbara	NL-means	33.34	28.35	24.62
	BM3D	**34.98**	30.72	27.23
	[12] (2 iter.)	34.39	30.47	27.35
	prop. (1 iter.)	34.46	30.08	26.67
	prop. (2 iter.)	34.75	**30.76**	**27.48**
House	NL-means	35.22	30.66	26.27
	BM3D	**36.71**	32.86	29.69
	[12] (2 iter.)	35.80	32.54	29.64
	prop. (1 iter.)	36.2	32.23	28.96
	prop. (2 iter.)	36.55	**33.07**	**30.21**

$\sigma = 50$, are shown in Figs. 1 and 2, respectively. First, it can be seen that the second iteration improves the results of our proposed scheme in all the cases. It can also be seen that the results obtained with two iterations of our scheme are better than the ones obtained with NL-means and the algorithm in [12]. Further, compared with BM3D results, our second iteration results are slightly better for $\sigma = 50$, comparable for $\sigma = 25$, and slightly worse for $\sigma = 10$.

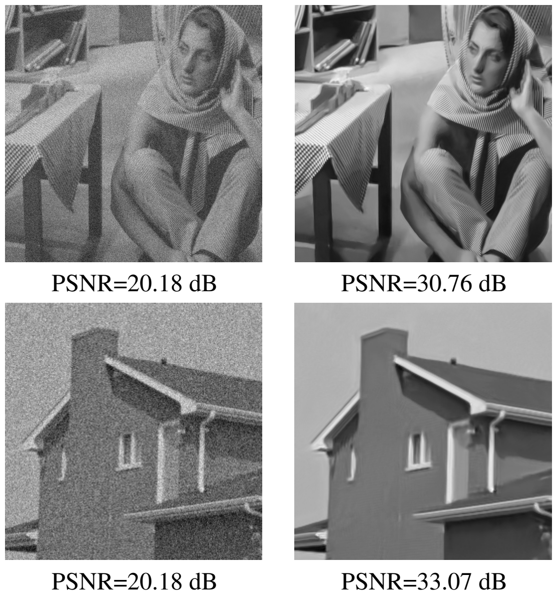

Fig. 1: Denoising results for the images Barbara and House ($\sigma = 25$): Left column - noisy images, Right column - 2 iterations results.

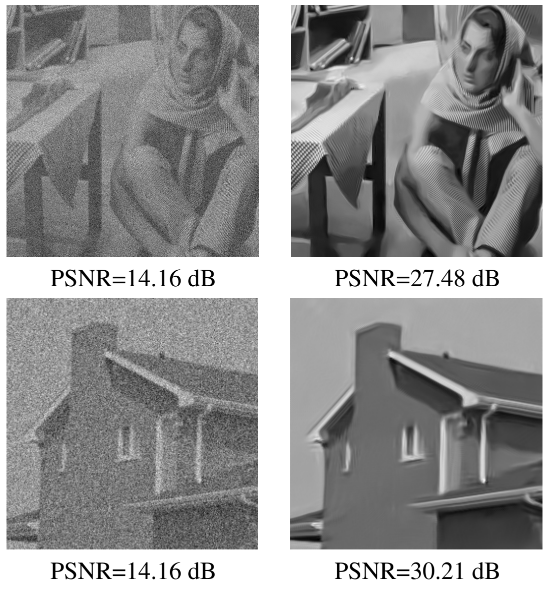

Fig. 2: Denoising results for the images Barbara and House ($\sigma = 50$): Left column - noisy images, Right column - 2 iterations results.

5. CONCLUSIONS

We have proposed a new image denoising scheme which is based on the NL-means algorithm and smooth 1D ordering of the pixels in the noisy image. We replaced the square neighborhoods employed by each pixel in the NL-means algorithm with a union of neighborhoods obtained from different global pixel permutations. We have shown that combined with patch classification and subimage averaging schemes, applying two iterations of our proposed scheme produces results which are close to the state-of-the-art.

In our future work, we wish to improve state-of-the-art algorithms like the BM3D and the algorithm proposed in [8] by employing our proposed patch neighborhoods in their block matching procedure. Additionally, the proposed image denoising scheme may be further improved by dividing the patches to more than two types, and treating each type differently.

6. REFERENCES

[1] A. Buades, B. Coll, and J. M. Morel, “A review of image denoising algorithms, with a new one,” *Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 490–530, 2006.

[2] P. Chatterjee and P. Milanfar, “Clustering-based denoising with locally learned dictionaries,” *IEEE Trans. Image Processing*, vol. 18, no. 7, pp. 1438–1451, 2009.

[3] G. Yu, G. Sapiro, and S. Mallat, “Image modeling and enhancement via structured sparse model selection,” in *Proc. 17th IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 1641–1644.

[4] W. Dong, X. Li, L. Zhang, and G. Shi, “Sparsity-based image denoising via dictionary learning and structural clustering,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 457–464.

[5] D. Zoran and Y. Weiss, “From learning models of natural image patches to whole image restoration,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 479–486.

[6] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” *IEEE Trans. Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[7] J. Mairal, M. Elad, G. Sapiro *et al.*, “Sparse representation for color image restoration,” *IEEE Trans. Image Processing*, vol. 17, no. 1, p. 53, 2008.

[8] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, “Non-local sparse models for image restoration,” in *Proc. IEEE 12th International Conference on Computer Vision (ICCV)*, 2009, pp. 2272–2279.

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-d transform-domain collaborative filtering,” *IEEE Trans. Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[10] I. Ram, M. Elad, and I. Cohen, “Generalized Tree-Based Wavelet Transform ,” *IEEE Trans. Signal Processing*, vol. 59, no. 9, pp. 4199–4209, 2011.

[11] ——, “Redundant Wavelets on Graphs and High Dimensional Data Clouds,” *IEEE Signal Processing Letters*, vol. 19, no. 5, pp. 291–294, 2012.

[12] ——, “Image processing using smooth ordering of its patches,” *Submitted to IEEE Trans. Image Processing.* [Online]. Available: http://www.cs.technion.ac.il/~elad/publications/journals/2012/Patch-Ordering-IEEE-TIP.pdf

[13] T. H. Cormen, *Introduction to algorithms*. The MIT press, 2001.