Image Processing Using Smooth Ordering of its Patches

Idan Ram, Michael Elad, *Fellow, IEEE*, and Israel Cohen, *Senior Member, IEEE*

***Abstract*— We propose an image processing scheme based on reordering of its patches. For a given corrupted image, we extract all patches with overlaps, refer to these as coordinates in high-dimensional space, and order them such that they are chained in the "shortest possible path," essentially solving the traveling salesman problem. The obtained ordering applied to the corrupted image implies a permutation of the image pixels to *what should be* a regular signal. This enables us to obtain good recovery of the clean image by applying relatively simple one-dimensional smoothing operations (such as filtering or interpolation) to the reordered set of pixels. We explore the use of the proposed approach to image denoising and inpainting, and show promising results in both cases.**

***Index Terms*— Patch-based processing, traveling salesman, pixel permutation, denoising, inpainting.**

I. Introduction

In RECENT years, image processing using local patches has become very popular and was shown to be highly effective – see [1]–[13] for representative work. The core idea behind these and many other contributions is the same: given the image to be processed, extract all possible patches with overlaps; these patches are typically very small compared to the original image size (a typical patch size would be 8×8 pixels). The processing itself proceeds by operating on these patches and exploiting interrelations between them. The manipulated patches (or sometimes only their center pixels) are then put back into the image canvas to form the resulting image.

There are various ways in which the relations between patches can be taken into account: 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]–[7], seeking a representative dictionary for the patches and using it to sparsely represent them, as practiced in [8]–[11], gathering groups of similar patches and applying a sparsifying transform on them [10], [12], [13]. 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. A joint treatment of these patches supports the reconstruction process by introducing a non-local force, thus enabling better recovery.

In our previous work [14] and [15] we proposed yet another patch-based image processing approach. We constructed an image-adaptive wavelet transform which is tailored to sparsely represent the given image. We used a plain 1D wavelet transform and adapted it to the image by operating on a permuted order of the image pixels[1]. The permutation we proposed is drawn from a shortest path ordering of the image patches. This way, the patches are leveraged to form a multi-scale sparsifying global transform for the image in question.

In this paper we embark from our earlier work as reported in [14] and [15], adopting the core idea of ordering the patches. However, we discard the globality of the obtained transform, the multi-scale treatment, and the sparsity-driven processing that follows. Thus, we propose a very simple image processing scheme that relies solely on patch reordering. We start by extracting all the patches of size $\sqrt{n} \times \sqrt{n}$ with maximal overlaps. Once these patches are extracted, we disregard their spatial relationships altogether, and seek a *new way* for organizing them. We propose to refer to these patches as a cloud of vectors/points in $\mathbb{R}^n$, and we order them such that they are chained in the "shortest possible path", essentially solving the traveling salesman problem [18]. This reordering is the one we have used in [14] and [15], but as opposed to our past work, our treatment from this point varies substantially. A key assumption in this work is that proximity between two image patches implies proximity between their center pixels. Therefore if the image mentioned above is of high-quality, the new ordering of the patches is expected to induce a highly regular (smooth or at least piece-wise smooth) 1D ordering of the image pixels, being the center of these patches. When the image is deteriorated (noisy, containing missing pixels, etc.), the above ordering is expected to be robust to the distortions, thereby suggesting a reordering of the corrupted pixels to "what should be" a regular signal. Thus, applying relatively simple one-dimensional (1D) smoothing operations (such as filtering or interpolation) to the reordered set of pixels should enable good recovery of the clean image.

Manuscript received October 13, 2012; revised February 6, 2013; accepted March 31, 2013. Date of publication April 12, 2013; date of current version May 16, 2013. This work was supported by the Japan Technion Society Research Fund and Robert H. Hillman Foundation for Global Security – collaboration of Technion and Northeastern University, and the European Research Council under the ERC Grant Agreement 320649. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chang-Su Kim.

I. Ram and I. Cohen are with the Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel (e-mail: idanram@tx.technion.ac.il; icohen@ee.technion.ac.il).

M. Elad is with the Department of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel (e-mail: elad@cs.technion.ac.il).

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

Digital Object Identifier 10.1109/TIP.2013.2257813

[1]Note that the idea of adapting a wavelet transform to the image by reordering its pixels appears also in [16] and [17], but the schemes proposed there do not use image patches.

This is the core process we propose in this paper – for a given corrupted image, we reorder its pixels, operate on the new 1D signal using simplified algorithms, and reposition the resulting values to their original location. We show that the proposed method, applied with several randomly constructed orderings and combined with a proposed subimage averaging scheme, is able to lead to state-of-the-art results. We explore the use of the proposed image reconstruction scheme to image denoising, and show that it achieves better results than the ones obtained with the K-SVD algorithm [8] for medium and high noise levels, and generally performs better than the BM3D algorithm [12] for high noise levels. We also explore the use of the proposed image processing scheme to image inpainting, and show that it leads to better results compared to the ones obtained with a simple interpolation scheme and the method proposed in [19] which employs sparse representation modeling via the redundant DCT dictionary. We also show that our results are mostly better than the ones of the algorithm proposed in [13], and slightly inferior to the ones of the state-of-the-art PLE [4] for two of the three test images. Finally, we draw some interesting ties between this scheme and BM3D rationale [12].

The paper is organized as follows: In Section II we introduce the basic image processing scheme. In Section III we explain how the performance of the basic scheme can be improved using a subimage averaging scheme, and describe the connection between the improved scheme and the BM3D algorithm. In Section IV we explore the use of the proposed approach to image denoising and inpainting, and present experimental results that demonstrate the advantages of the proposed scheme. We summarize the paper in Section IV with ideas for future work along the path presented here.

II. Image Processing Using Patch Ordering

A. Basic Scheme

Let $\mathbf{Y}$ be an image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{Z}$ be a corrupted version of $\mathbf{Y}$, which may be noisy or contain missing pixels. Also, let $\mathbf{z}$ and $\mathbf{y}$ be the column stacked versions of $\mathbf{Z}$ and $\mathbf{Y}$, respectively. Then we assume that the corrupted image satisfies

$$\mathbf{z} = \mathbf{M}\mathbf{y} + \mathbf{v} \tag{1}$$

where the $N \times N$ matrix $\mathbf{M}$ denotes a linear operator which corrupts the data, and $\mathbf{v}$ denotes an additive white Gaussian noise independent of $\mathbf{y}$ with zero mean and variance σ^2. In this work the matrix $\mathbf{M}$ is restricted to represent a point-wise operator, covering applications such as denoising and inpainting. The reason for this restriction is the fact that we will be permuting the pixels in the image, and thus spatial operations become far more complex to handle.

Our goal is to reconstruct $\mathbf{y}$ from $\mathbf{z}$, and for this end we employ a permutation matrix $\mathbf{P}$ of size $N \times N$. We assume that when $\mathbf{P}$ is applied to the target signal $\mathbf{y}$, it produces a smooth signal $\mathbf{y}^p = \mathbf{P}\mathbf{y}$. We will explain how such a matrix may be obtained using the image patches in Section II-B. We start by applying $\mathbf{P}$ to $\mathbf{z}$ and obtain $\mathbf{z}^p = \mathbf{P}\mathbf{z}$. Next, we take advantage of our prior knowledge that $\mathbf{y}^p$ should be smooth, and apply a "simple" 1D smoothing operator H on $\mathbf{z}^p$, such as 1D interpolation or filtering. Finally, we apply $\mathbf{P}^{-1}$ to the result, and obtain the reconstructed image

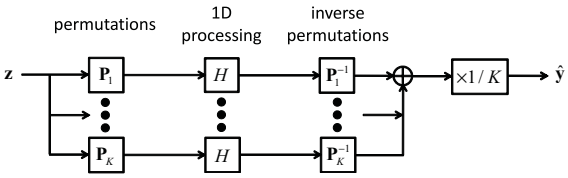

Fig. 1. The basic image processing scheme.

$$\hat{\mathbf{y}} = \mathbf{P}^{-1} H\{\mathbf{P}\mathbf{z}\}. \tag{2}$$

In order to better smooth the recovered image, we use an approach which resembles the "cycle spinning" method [20]. We randomly construct K different permutation matrices $\mathbf{P}_k$, utilize each to denoise the image $\mathbf{z}$ using the scheme described above, and average the results. This can be expressed by

$$\hat{\mathbf{y}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_k^{-1} H\{\mathbf{P}_k \mathbf{z}\}. \tag{3}$$

Figure 1 shows the proposed image processing scheme. We next describe how we construct the reordering matrix $\mathbf{P}$.

B. Building the Permutation Matrix $\mathbf{P}$

We wish to design a matrix $\mathbf{P}$ which produces a smooth signal when it is 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 corrupted image $\mathbf{Z}$ (noisy, containing missing pixels, etc.), and any permutation defined by simply reordering the corrupted pixels into a regular signal does not necessarily smooth $\mathbf{y}$. Therefore, as the pixels in the corrupted image are not helpful to us, we settle for a suboptimal ordering operation, using patches from the corrupted 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 $\sqrt{n} \times \sqrt{n}$ patch around the location of z_i in $\mathbf{Z}$. We assume that under a distance measure[2] $\omega(\mathbf{x}_i, \mathbf{x}_j)$, proximity between the two patches $\mathbf{x}_i$ and $\mathbf{x}_j$ suggests proximity between the uncorrupted 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

$$\|\mathbf{y}^p\|_{TV} = \sum_{j=2}^{N} |y^p(j) - y^p(j-1)|. \tag{4}$$

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

[2]Throughout this paper we will be using variants of the squared Euclidean distance.

Algorithm 1 The Patch Reordering Algorithm We Use

Task: Reorder the image patches $\mathbf{x}_j$.
Parameters: We are given the image patches $\{\mathbf{x}_j\}_{j=1}^N$, the distance function w, and ϵ which is used in the probabilistic part of the algorithm.
Initialization: Choose a random index j and set $\Omega(1) = \{j\}$.
Main Iteration: Perform the following steps for $k = 1, \ldots, N-1$:

- Set A_k to be the set of indices of the $B \times B$ patches around $\mathbf{x}_{\Omega(k)}$.
- If $|A_k \setminus \Omega| = 1$
  - Set $\Omega(k+1)$ to be $A_k \setminus \Omega$.
- Else
  - If $|A_k \setminus \Omega| \geq 2$
    - Find $\mathbf{x}_{j_1}$ – the nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_1 \in A_k$ and $j_1 \notin \Omega$.
    - Find $\mathbf{x}_{j_2}$ – the second nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_2 \in A_k$ and $j_2 \notin \Omega$.
  - Elseif $|A_k \setminus \Omega| = 0$
    - Find $\mathbf{x}_{j_1}$ – the nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_1 \notin \Omega$.
    - Find $\mathbf{x}_{j_2}$ – the second nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_2 \notin \Omega$.
  - Set $\Omega(k+1)$ to be:
    - $\{j_1\}$ with probability $p_1 \propto \exp\left[-\frac{w(\mathbf{x}_{\Omega(k)}, \mathbf{x}_{j_1})}{\epsilon}\right]$
    - $\{j_2\}$ with probability $p_2 = 1 - p_1 \propto \exp\left[-\frac{w(\mathbf{x}_{\Omega(k)}, \mathbf{x}_{j_2})}{\epsilon}\right]$

Output: The set Ω holds the proposed ordering.

the points $\mathbf{x}_j^p$ by the measure

$$X_{TV}^p = \sum_{j=2}^{N} w(\mathbf{x}_j^p, \mathbf{x}_{j-1}^p). \tag{5}$$

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 [18], which can become very computationally expensive for large sets of points. We choose a simple and crude 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 $p_1 = \alpha \exp\left[-w(\mathbf{x}_{j_0}, \mathbf{x}_{j_1})/\epsilon\right]$, or to its second nearest neighbor $\mathbf{x}_{j_2}$ with a probability $p_2 = \alpha \exp\left[-w(\mathbf{x}_{j_0}, \mathbf{x}_{j_2})/\epsilon\right]$, where α is determined such that $p_1 + p_2 = 1$, 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 only one unvisited patch remains in that neighborhood, we simply continue to this patch, and in the case that no unvisited patches remain, we search for the first and second nearest neighbors 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. The patch reordering scheme is summarized in Algorithm 1.

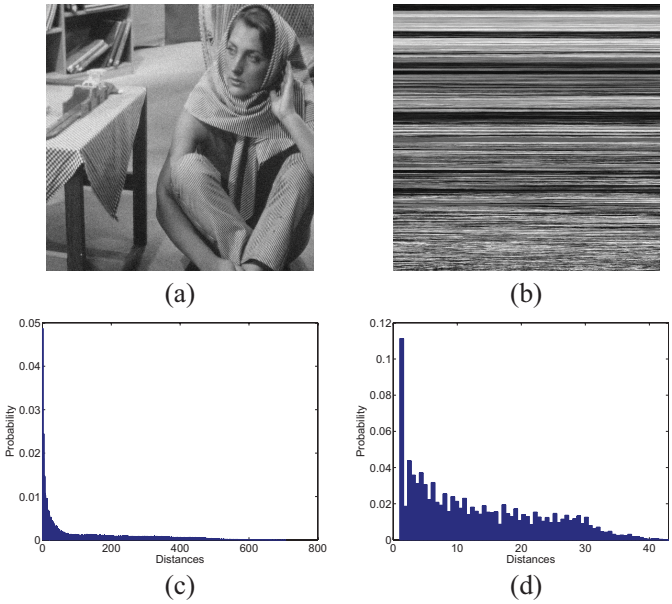

Fig. 2. (a) Noisy Barbara ($\sigma = 10$). (b) Reordered clean Barbara, folded back by s zig–zag raster scan. (c) and (d) Normalized histograms of the spatial distances between adjacent patches after the reordering, obtained with unrestricted and restricted search areas, respectively. The lengths of the pathes in the spatial domain obtained with the unrestricted and restricted searches were $3.63 \cdot 10^7$ and $3.42 \cdot 10^6$, respectively.

We next demonstrate the smoothing effect that a permutation obtained from a corrupted image has on its clean version. We apply the patch ordering scheme described above, with the parameters $\sqrt{n} = 6$, $B = 61$ and $\epsilon = 10^6$, to the patches of the noisy Barbara image with noise standard deviation $\sigma = 10$, shown in Figure 2(a). We apply the obtained permutation to the column stacked version $\mathbf{y}$ of the clean Barbara image, and obtain the reordered signal $\mathbf{y}^p$. We then calculate the total variations of $\mathbf{y}$ and $\mathbf{y}^p$, and obtain that $\|\mathbf{y}\|_{TV} = 2.49 \cdot 10^6$ and $\|\mathbf{y}^p\|_{TV} = 1.76 \cdot 10^6$, i.e. the latter is 29 percent smaller than the former. In order to visually demonstrate the regularity of the reordered signal $\mathbf{y}^p$, we fold it back by a zig-zag raster scan into the original image size. The obtained image is shown in Figure 2(b), and it can be seen that the reordered signal is indeed piecewise regular for most of its length, but becomes less and less regular towards its end. In fact, when the last 30% of the samples of $\mathbf{y}$ and $\mathbf{y}^p$ are discarded, the total variation of the obtained reordered signal becomes 37 percent smaller than the total variation of the obtained column stacked signal. This behaviour of the reordered images might be expected due to the greedy nature of the patch ordering algorithm, which leaves for the patches near the end of the path a very small number of unvisited patches to choose from. The employed cycle spinning scheme prevents the non-regular parts of the reordered images from degrading the quality of the reconstructed images.

It is also interesting to examine the characteristics of the patch ordering in the spatial domain. To this end we also apply the patch ordering scheme to the patches of the noisy Barbara image with the same parameters, but with an unrestricted search neighborhood. We apply the permutations obtained with the two types of neighborhoods to the patches, and calculate

two normalized histograms of the spatial distances between adjacent patches, shown in Figure 2. Figure 2(c) shows that when the search neighborhood is not restricted, only about 5% of neighboring patches in the path are also immediate spatial neighbors, and that far away patches are often assigned as neighbors in the reordering process. The histogram in Figure 2(d), obtained with restricted search neighborhood, is limited to show only distances which are smaller or equal to 43, the maximal possible distance within the search window. It can be seen that despite the restriction to a smaller search neighborhood, only about 11% of neighboring patches in the path are also immediate spatial neighbors, and patches all over the search neighborhood are assigned as neighbors in the reordering process.

In order to facilitate the cycle-spinning method mentioned above, we simply run the proposed ordering solver K times, and the randomness (both in the initialization and in assigning the neighbors) leads to different permutation results. We next describe how the quality of the produced images may be further improved using a subimage averaging scheme, which can be seen as another variation of "cycle spinning".

C. Subimage Averaging

Let $N_p = (N_1 - \sqrt{n} + 1)(N_2 - \sqrt{n} + 1)$ denote the number of overlapped patches in the image $\mathbf{Z}$, and let $\mathbf{X}$ be an $n \times N_p$ matrix, containing column stacked versions of these patches. We extract these patches column by column, starting from the top left-most patch. When we calculated $\mathbf{P}$ as described in the previous section, we assumed that each patch is associated only with its middle pixel. Therefore $\mathbf{P}$ 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 matrix $\mathbf{P}$ 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 $(N_1 - \sqrt{n} + 1) \times (N_2 - \sqrt{n} + 1)$ contained in the image $\mathbf{Z}$. We denote these subimages by $\tilde{\mathbf{Z}}_j$, $j = 1, 2, \ldots, n$. An example for two of them, $\tilde{\mathbf{Z}}_1$ and $\tilde{\mathbf{Z}}_n$, contained in a noisy version of the image Barbara with noise standard deviation $\sigma = 25$, is shown Fig. 3(a).

We already observed in [14] and [15] that improved denoising results are obtained when all the n subimages of a noisy image are employed in its denoising process. Here we use a similar scheme in order to improve the quality of the recovered image. In order to avoid cumbersome notations we first describe a scheme which utilizes a single ordering matrix $\mathbf{P}$. Let the vector $\tilde{\mathbf{z}}_j = \mathbf{R}_j \mathbf{z}$ of length N_p be the column stacked version of $\tilde{\mathbf{Z}}_j$, where the $N_p \times N$ matrix $\mathbf{R}_j$ extracts the jth subimage from the image $\mathbf{z}$. We first calculate the $N_p \times N_p$ matrix $\mathbf{P}$ using the patches in $\mathbf{X}$ and apply it to each subimage $\tilde{\mathbf{z}}_j$. Then we apply the operator H to each of the reordered subimages $\tilde{\mathbf{z}}_j^p = \mathbf{P}\tilde{\mathbf{z}}_j$, apply the inverse permutation $\mathbf{P}^{-1}$ on the result, and obtain the reconstructed subimages

$$\hat{\tilde{\mathbf{y}}}_j = \mathbf{P}^{-1} H\{\mathbf{P}\tilde{\mathbf{z}}_j\} = \mathbf{P}^{-1} H\{\mathbf{P}\mathbf{R}_j \mathbf{z}\}. \tag{6}$$

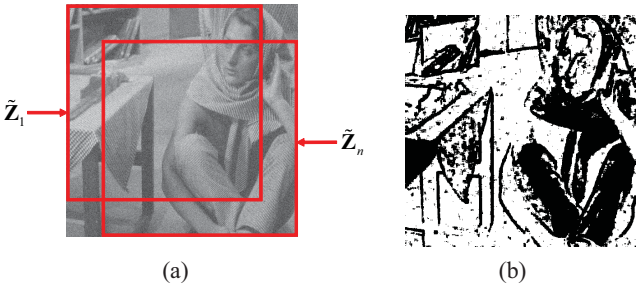

Fig. 3. (a) Two subimages $\tilde{\mathbf{Z}}_1$ and $\tilde{\mathbf{Z}}_n$ contained in a noisy version of the image Barbara ($\sigma = 25$). (b) Classification of the pixels in a noisy version of the image Barbara to centers of smooth (white) and non-smooth (black) patches.

We next reconstruct the image from all the reconstructed subimages $\hat{\tilde{\mathbf{y}}}_j$ by plugging each subimage into its original place in the image canvas and averaging the different values obtained for each pixel. More formally, we obtain the reconstructed image $\hat{\mathbf{y}}$ as follows:

$$\hat{\mathbf{y}} = \mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \hat{\tilde{\mathbf{y}}}_j = \mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{P}^{-1} H\{\mathbf{P}\mathbf{R}_j \mathbf{z}\} \tag{7}$$

where the matrix $\mathbf{R}_j^T$ plugs the estimated jth subimage into its original place in the canvas, and

$$\mathbf{D} = \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{R}_j \tag{8}$$

is an $N \times N$ diagonal weight matrix that simply averages the overlapping contributions per each pixel. When K random matrices $\mathbf{P}_k$ are employed, we obtain the final estimate by averaging the images obtained with the different permutations

$$\hat{\mathbf{y}} = \frac{1}{K} \sum_{k=1}^{K} \left(\mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{P}_k^{-1} H\{\mathbf{P}_k \mathbf{R}_j \mathbf{z}\} \right). \tag{9}$$

This formula reveals two important properties of our scheme: 1) the two summations that correspond to the two cycle-spinning versions lead to an averaging of nK candidate solutions, a fact that boosts the overall performance of the recovery algorithm; and 2) if H is chosen as linear, then the overall processing is linear as well, provided that we disregard the highly non-linear dependency of $\mathbf{P}$ on $\mathbf{z}$.

D. Connection to BM3D and Clustering-Based Methods

The above processing scheme can be described a little differently. We start by calculating the permutation matrix $\mathbf{P}$ from the image patches $\mathbf{x}_i$. We then gather the patches by arranging them as the columns of a matrix $\mathbf{X}^p$ in the order defined by $\mathbf{P}$. This matrix contains in its rows the reordered subimages $\tilde{\mathbf{z}}_j^p$, therefore we next apply the operator H to its rows, and shuffle the columns of the resulting matrix according to the permutation defined by $\mathbf{P}^{-1}$. We obtain a matrix $\hat{\bar{\mathbf{X}}}$, which contains in its rows the reconstructed subimages $\hat{\tilde{\mathbf{y}}}_j$, and in its columns reconstructed versions $\bar{\mathbf{x}}_i$ of the image patches $\mathbf{x}_i$. We obtain the reconstructed image $\hat{\mathbf{y}}$ from the patches $\bar{\mathbf{x}}_i$ by plugging them into their original places in the

image canvas, and averaging the different values obtained for each pixel. When K random matrices $\mathbf{P}_k$ are employed, we apply the aforementioned scheme with each of these matrices, and average the obtained images.

Now, looking at the image processing scheme described above, we can see some similarities to the first stage of the BM3D algorithm. Both algorithms stack the image patches into groups, apply 1D processing across the patches, return the patches into their place in the image, and average the results. We note that the BM3D algorithm also applies a 2D transform to the patches before performing 1D processing across them. This feature can be easily added to our scheme if needed, and we regard this as a preprocessing part of the operator H. On the other hand, there are some key differences between the two schemes. First, while the BM3D algorithm constructs a group of neighbors for each patch, here we order *all the patches* to one chain, which defines local neighbors. Furthermore, this process is repeated K times, implying that our approach consider K different neighbors assignments. Also, while in the BM3D the patch order in each group is not restricted, ours is carefully determined as it plays a major role in our scheme. Finally, the 1D processing applied by the BM3D consists of the use of a 1D transform, followed by thresholding and the inverse transform, implying a specific denoising. Here we do not restrict ourselves to any specific 1D processing scheme, and allow the operator H to be chosen based on the application at hand.

We conclude this discussion with the following comment. When observing clustering-based restoration methods in general, such as [5], [6], [10], [12], there is a clear similarity to our approach, as near-by patches find themselves supporting each other's processing. However, as opposed to these methods, our technique also orders the near-by patches into a smooth path, and as such, our approach enables a better treatment of the patches even within the clustered groups. For example, if the corresponding center pixel forms a smooth linear slope of values, the clustering approaches would approximate this line by the average value, while our approach would regress to a smooth line. We next demonstrate the application of our proposed scheme to image denoising and inpainting.

III. Applications and Results

A. Image Denoising

The problem of image denoising consists of the recovery of an image from its noisy version. In that case $\mathbf{M} = \mathbf{I}$ and the corrupted image satisfies $\mathbf{z} = \mathbf{y} + \mathbf{v}$. The patches $\mathbf{x}_i$ contain noise, and we choose the distance measure between $\mathbf{x}_i$ and $\mathbf{x}_j$ to be the squared Euclidean distance divided by n, i.e

$$w(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{n}\|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{10}$$

In our previous works [14] and [15] we applied a complex multi-scale processing on the ordered patches. Here we wish to employ a far simpler scheme; we choose a 1D linear shift-invariant filter, and as we show next, we learn this filter from training images. Furthermore, we suggest to switch between two such filters, based on the patch content.

We desire to treat smooth areas in the image differently than areas with edges or texture, as our experiments show that this approach leads to better results. More specifically, we employ different permutation matrices and filters in the smooth and non smooth areas of the image. 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 $\text{std}(\mathbf{x}_i)$ 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 $\text{std}(\mathbf{x}_i) < C\sigma$ then $\mathbf{x}_i \in S_s$, otherwise $\mathbf{x}_i \in S_e$. Fig 3(b) demonstrates the application of this classification rule to the noisy Barbara image shown in Figure 3(a), where we use the parameters $\sqrt{n} = 8$ and $C = 1.2$ which we will later use in the denoising process of this image. White pixels are the centers of smooth patches and black pixels are the centers of patches containing texture or edges. It can be seen that the obtained image indeed contains a rough classification of the patches into smooth and non smooth sets.

We next divide each subimage $\tilde{\mathbf{z}}_j$ into two signals: $\tilde{\mathbf{z}}_{j,s}$ - a vector of length $|S_s|$ which contains the pixels corresponding to the smooth patches, and $\tilde{\mathbf{z}}_{j,e}$ - a vector of length $|S_e|$ which contains the pixels corresponding to the patches with edges and texture. We construct an $|S_s| \times N_p$ matrix $\mathbf{P}_s$, which extracts $\tilde{\mathbf{z}}_{j,s}$ from $\tilde{\mathbf{z}}_j$, and applies to it a permutation obtained from the patches in the set S_s using the nearest neighbors search method described above. We similarly construct an $|S_e| \times N_p$ matrix $\mathbf{P}_e$, which extracts $\tilde{\mathbf{z}}_{j,e}$ from $\tilde{\mathbf{z}}_j$, and applies to it a permutation obtained from the patches in the set S_e, using the same set of parameters. We apply $\mathbf{P}_s$ and $\mathbf{P}_e$ to $\tilde{\mathbf{z}}_j$ and obtain $\tilde{\mathbf{z}}^p_{j,s}$ and $\tilde{\mathbf{z}}^p_{j,e}$, respectively, which are the signals to which we apply the filters. We define the reordered subimage $\tilde{\mathbf{z}}^p_j$ and the $N_p \times N_p$ permutation matrix $\mathbf{P}$ which satisfy

$$\tilde{\mathbf{z}}^p_j = \begin{bmatrix} \tilde{\mathbf{z}}^p_{j,s} \\ \tilde{\mathbf{z}}^p_{j,e} \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_s \\ \mathbf{P}_e \end{bmatrix} \tag{11}$$

and obtain that $\tilde{\mathbf{z}}^p_j = \mathbf{P}\tilde{\mathbf{z}}_j$.

We next wish to find the filters $\mathbf{h}_s$ and $\mathbf{h}_e$, each of length N_h, applied to $\tilde{\mathbf{z}}^p_{j,s}$ and $\tilde{\mathbf{z}}^p_{j,e}$, respectively. We denote by $\mathbf{M}^p_{j,s}$ the $|S_s| \times N_h$ convolution matrix corresponding to $\tilde{\mathbf{z}}^p_{j,s}$, and by $\mathbf{M}^p_{j,e}$ the $|S_e| \times N_h$ convolution matrix corresponding to $\tilde{\mathbf{z}}^p_{j,e}$, and obtain the filtered subimages

$$\begin{aligned} \hat{\tilde{\mathbf{y}}}_j &= \mathbf{P}^{-1}\begin{bmatrix} \mathbf{M}^p_{j,s}\mathbf{h}_s \\ \mathbf{M}^p_{j,e}\mathbf{h}_e \end{bmatrix} = \mathbf{P}^{-1}\begin{bmatrix} \mathbf{M}^p_{j,s} & 0 \\ 0 & \mathbf{M}^p_{j,e} \end{bmatrix}\begin{bmatrix} \mathbf{h}_s \\ \mathbf{h}_e \end{bmatrix} \\ &= \mathbf{P}^{-1}\mathbf{M}^p_j\mathbf{h} \end{aligned} \tag{12}$$

where we defined the $N_p \times N_p$ matrix $\mathbf{M}^p_j$, and the filters vector $\mathbf{h}$ of length $2N_h$, which satisfy

$$\mathbf{M}^p_j = \begin{bmatrix} \mathbf{M}^p_{j,s} & 0 \\ 0 & \mathbf{M}^p_{j,e} \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} \mathbf{h}_s \\ \mathbf{h}_e \end{bmatrix}. \tag{13}$$

The vector $\mathbf{h}$ stores the filter taps to be designed. We substitute (12) in (7), and obtain the reconstructed image

$$\hat{\mathbf{y}} = \mathbf{D}^{-1}\sum_{j=1}^{n}\mathbf{R}^T_j\mathbf{P}^{-1}\mathbf{M}^p_j\mathbf{h}. \tag{14}$$

When K random matrices $\mathbf{P}_k$ are employed, we obtain the final estimate by averaging the images obtained with the different matrices

$$\hat{\mathbf{y}} = \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{D}^{-1}\sum_{j=1}^{n}\mathbf{R}_j^T\mathbf{P}_k^{-1}\mathbf{M}_j^p\mathbf{h}\right) = \frac{1}{K}\sum_{k=1}^{K}\mathbf{D}^{-1}[\mathbf{R}_1^T,\ldots,\mathbf{R}_n^T]\begin{bmatrix}\mathbf{P}_k^{-1}\mathbf{M}_1^p\\ \vdots \\ \mathbf{P}_k^{-1}\mathbf{M}_n^p\end{bmatrix}\mathbf{h} = \mathbf{Q}\mathbf{h} \quad (15)$$

where we defined the $N \times 2N_h$ matrix

$$\mathbf{Q} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{D}^{-1}[\mathbf{R}_1^T,\ldots,\mathbf{R}_n^T]\begin{bmatrix}\mathbf{P}_k^{-1}\mathbf{M}_1^p\\ \vdots \\ \mathbf{P}_k^{-1}\mathbf{M}_n^p\end{bmatrix}. \quad (16)$$

Now let $\mathbf{y}^g$, $g = 1,\ldots,G$ be a training set which contains the column stack versions of G clean images. For each such image we create a noisy version $\mathbf{z}^g$ by adding noise with the same statistics as the noise in $\mathbf{z}$. Then we calculate for each image $\mathbf{z}^g$ a matrix $\mathbf{Q}^g$ using (16), and learn the filters vector $\mathbf{h}$ by minimizing

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}}\sum_{g=1}^{G}\|\mathbf{y}^g - \mathbf{Q}^g\mathbf{h}\|^2 = \left[\sum_{g=1}^{G}(\mathbf{Q}^g)^T\mathbf{Q}^g\right]^{-1}\sum_{k=1}^{G}(\mathbf{Q}^g)^T\mathbf{y}^g. \quad (17)$$

Once we have the filters vector $\hat{\mathbf{h}}$ we can employ it to denoise $\mathbf{z}$ by building $\mathbf{Q}$ using (16) and then calculating

$$\hat{\mathbf{y}} = \mathbf{Q}\hat{\mathbf{h}}. \quad (18)$$

We can further improve our results by applying a second iteration of our proposed scheme, in which all the processing stages remain the same, but the permutation matrices are built using patches extracted from the first iteration clean result.

In the following experiments we assess the performance of the proposed image denoising scheme by applying it to a test set containing noisy versions of 6 images, with 8 different noise standard deviations. We use a training set containing the images Man, Couple, and Hill to learn the filters vector $\mathbf{h}$. We start by demonstrating the performance gain obtained by combining the patch classification and filter learning into the proposed denoising scheme. We compare the results obtained with our scheme, for noisy images with noise standard deviation $\sigma = 25$, in three settings: 1) without patch classification and using a Gaussian smoothing filter; 2) without patch classification and using a learned filter; 3) with patch classification and using two learned filters. In all 3 cases we use the parameters shown in Table II, which correspond to $\sigma = 25$ and 1 iteration. The results are shown in Table I, and it can be seen that using a learned filter instead of a simple smoothing filter improves the results for all images, with an average increase in PSNR of 2.32 dB. Also, performing patch classification further improves the results for all images, with an average increase in PSNR of 0.54 dB.

TABLE I

DENOISING RESULTS (PSNR IN dB) OF NOISY VERSIONS OF 6 IMAGES ($\sigma = 25$), OBTAINED WITH THE PROPOSED SCHEME IN THREE SETTINGS. 1) WITHOUT PATCH CLASSIFICATION AND USING GAUSSIAN SMOOTHING FILTER (GAUSSIAN). 2) WITHOUT PATCH CLASSIFICATION AND USING A LEARNED FILTER (1 LEARNED). 3) WITH PATCH CLASSIFICATION AND USING TWO LEARNED FILTERS (2 LEARNED). FOR EACH IMAGE AND SETTING, THE BEST RESULT IS HIGHLIGHTED

Image	Gaussian	1 Learned	2 Learned
Lena	29.14	30.90	**31.54**
Barbara	27.42	30.14	**30.36**
Boats	26.66	28.91	**29.5**
Fingerprint	24.24	26.99	**27.24**
House	29.75	31.61	**32.34**
Peppers	26.38	28.99	**29.78**

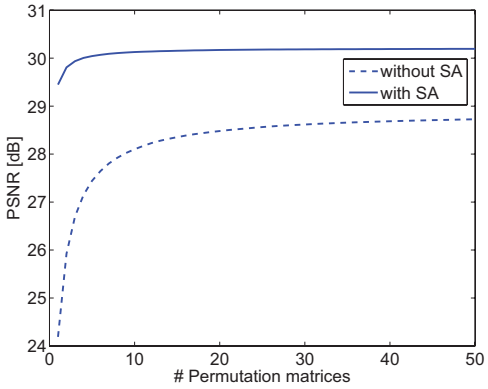

Fig. 4. Average of the PSNR values obtained for the noisy test images ($\sigma = 25$), with and without subimage averaging, as a function of the number of permutations.

We also demonstrate the improvement in the performance obtained with cycle spinning and subimage averaging. Figure 4 shows the average of the PSNR values obtained for the noisy test images with our scheme, including patch classification and filter learning, with and without subimage averaging, as a function of the number of employed permutation matrices. First it can be seen that the results in both cases improve as the number of employed permutations K increases, and that the subimage averaging improves the performance by between 1.5 dB for large values of K to 5 dB when $K = 1$. It can also be seen that when the subimage averaging scheme is used, most of the performance gain is obtained using the first 10 permutations. In fact, employing 40 more permutations increases the results by less than 0.07 dB. Thus, as a compromise between the quality of performance and computational cost, in the following experiments we utilize $K = 10$ permutation matrices when we apply our image denoising scheme.

We next apply our overall scheme, to all the noisy images in the test set. The parameters employed by the proposed denoising scheme for the different noise levels are shown in Table II. We note that the reason we chose a uniform filter length of $N_h = 25$ samples for all noise levels can be justified using Figure 5. Figure 5. shows the average of the PSNR values obtained in the first and second iterations for

TABLE II
PARAMETERS USED IN THE DENOISING EXPERIMENTS

σ	Iteration	K	$\sqrt{n}$	C	B	ϵ	N_h
5	1	10	5	2.2	61	10^6	25
	2	10	4	1.2	361	10^3	25
10	1	10	6	1.6	61	10^6	25
	2	10	4	0.8	361	10^3	25
15	1	10	7	1.4	61	10^6	25
	2	10	4	0.6	361	10^3	25
20	1	10	8	1.3	61	10^6	25
	2	10	4	0.5	361	10^3	25
25	1	10	8	1.2	61	10^6	25
	2	10	4	0.4	361	10^3	25
50	1	10	14	1.1	61	10^6	25
	2	10	5	0.3	361	10^3	25
75	1	10	16	1.1	61	10^6	25
	2	10	6	0.2	361	10^3	25
100	1	10	16	1.1	61	10^6	25
	2	10	8	0.1	361	10^3	25

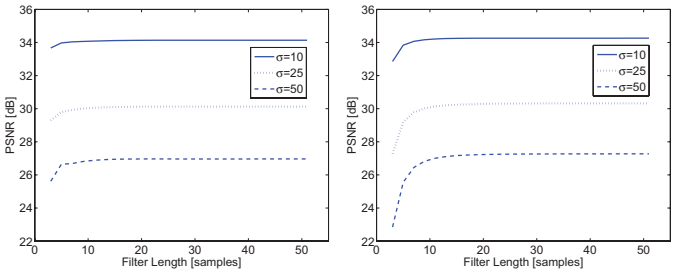

Fig. 5. Average of the PSNR values obtained for the test images, as a function of the filter length, for three different noise levels. (a) First iteration. (b) Second Iteration.

all the test images, as a function of the filter length, for three different noise levels. It can be seen that in both iterations the performance gain obtained using filters longer than 25 samples is negligible. The trained filters obtained in each iteration for the three different noise levels are shown in Figure 6. First, it can be seen that filters $\mathbf{h}_s$ and $\mathbf{h}_e$ indeed look different. It can also be seen that in the first iteration the shape of the filter $\mathbf{h}_s$ does not change much as the noise level increases, and in the second iteration the filters obtained for the higher noise levels are similar, but very different from the filter obtained for $\sigma = 10$. On the other hand, in both iterations the shape of the filter $\mathbf{h}_e$ changes greatly as the noise level increases.

For comparison, we also apply the K-SVD [8] and BM3D [12] algorithms. The PSNR values of the results obtained with these algorithms, and two iterations of our denoising scheme are shown in Table III. The noisy and recovered images obtained with our scheme for $\sigma = 25$ are shown in Figure 7. 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 inferior to the those of the K-SVD for $\sigma < 15$, but are better almost everywhere for $\sigma \geq 15$. Further, our two iterations results are generally better than those of the state-of-the-art BM3D algorithm for $\sigma \geq 50$, but are inferior for higher SNRs.

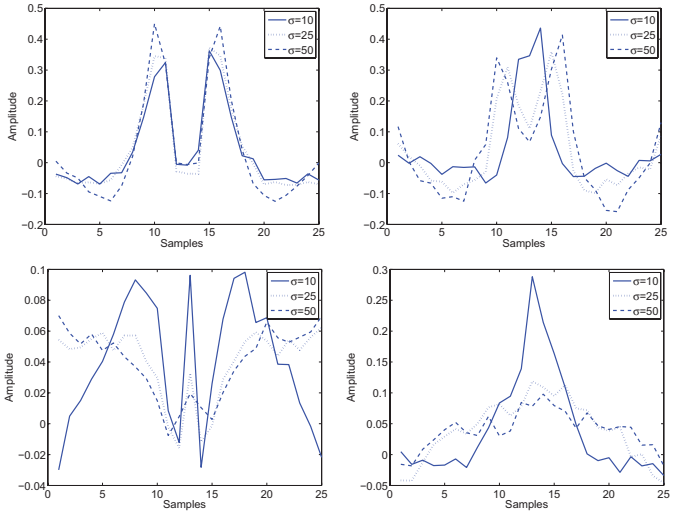

Fig. 6. Trained filters learned from the noisy images for $\sigma = 10, 25, 50$: Left column - the filters $\mathbf{h}_s$ obtained in the first (top) and second (bottom) iterations. Right column - the filters $\mathbf{h}_e$ obtained in the first (top) and second (bottom) iterations.

B. Image Inpainting

The problem of image inpainting consists of the recovery of missing pixels in the given image. Here we handle the case where there is no additive noise, therefore $\mathbf{v} = 0$, and $\mathbf{M}$ is a diagonal matrix of size $N \times N$ which contains ones and zeroes in its main diagonal corresponding to existing and missing pixels, correspondingly. Each patch may contain missing pixels, and we denote by S_i the set of indices of non-missing pixels in the patch $\mathbf{x}_i$. We choose the distance measure between patches $\mathbf{x}_i$ and $\mathbf{x}_j$ to be the average of squared differences between existing pixels that share the same location in both patches, i.e.

$$w(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k \in S_i \cap S_j} (x_i[k] - x_j[k])^2}{|S_i \cap S_j|}. \tag{19}$$

We start by calculating the matrix $\mathbf{P}$ according to the scheme described in Section II-B, with a minor difference: when a patch does not share pixels with any of the unvisited patches, the next patch in the path is chosen to be its nearest spatial neighbor. We next apply the obtained matrix to the subimages $\tilde{\mathbf{z}}_j$, and observe that the permuted vectors $\tilde{\mathbf{z}}_j^p = \mathbf{P}\tilde{\mathbf{z}}_j$ contain missing values. We bear in mind that the target signals $\tilde{\mathbf{y}}_j^p = \mathbf{P}\tilde{\mathbf{y}}^j$ should be smooth, and therefore apply on the subimages $\tilde{\mathbf{z}}_j^p$ an operator H which recovers the missing values using cubic spline interpolation. We apply the matrix $\mathbf{P}^{-1}$ on the resulting vectors and obtain the estimated subimages $\hat{\tilde{\mathbf{y}}}_j$. The final estimate is obtained from these subimages using (7). We improve our results by applying two additional iterations of a modified version of this inpainting scheme, where the only difference is that we rebuild $\mathbf{P}$ using reconstructed (and thus full) patches.

We demonstrate the performance of our proposed scheme on corrupted versions of the images Lena, Barbara and House, obtained by zeroing 80% of their pixels, which are selected at random. The parameters employed in each of the three

TABLE III

DENOISING RESULTS (PSNR IN dB) OF NOISY VERSIONS OF 6 IMAGES, OBTAINED WITH THE K-SVD AND BM3D ALGORITHMS AND TWO ITERATIONS OF THE PROPOSED SCHEME. FOR EACH IMAGE AND NOISE LEVEL, THE BEST RESULT IS HIGHLIGHTED

Image	Method	σ/PSNR							
		5/34.16	10/28.14	15/24.61	20/22.11	25/20.18	50/14.16	75/10.63	100/8.14
Lena	K-SVD	38.63	35.52	33.76	32.40	31.35	27.85	25.81	24.47
	BM3D	**38.72**	**35.93**	**34.28**	**33.04**	**32.05**	28.96	27.16	25.80
	proposed (1 iter.)	38.22	35.26	33.64	32.48	31.54	28.66	26.8	25.45
	proposed (2 iter.)	38.31	35.39	33.84	32.72	31.80	**28.96**	**27.22**	**26.01**
Barbara	K-SVD	38.08	34.40	32.33	30.79	29.54	25.43	23.02	21.89
	BM3D	**38.29**	**34.93**	**33.05**	**31.69**	**30.61**	27.16	25.11	23.61
	proposed (1 iter.)	37.63	34.29	32.54	31.32	30.36	27.19	25.14	23.56
	proposed (2 iter.)	37.74	34.39	32.65	31.43	30.47	**27.35**	**25.42**	**24.07**
Boats	K-SVD	37.25	33.65	31.74	30.35	29.30	25.94	24.04	22.85
	BM3D	**37.29**	**33.94**	**32.15**	**30.87**	**29.89**	**26.71**	**25.01**	23.88
	proposed (1 iter.)	37.09	33.64	31.79	30.49	29.50	26.35	24.58	23.34
	proposed (2 iter.)	37.10	33.70	31.91	30.67	29.70	26.69	24.99	**23.90**
Fingerprint	K-SVD	**36.66**	32.42	30.09	28.46	27.26	23.23	19.97	18.29
	BM3D	36.52	**32.47**	**30.30**	**28.83**	**27.72**	**24.54**	**22.82**	**21.57**
	proposed (1 iter.)	36.00	31.88	29.72	28.27	27.24	24.02	22.25	21.11
	proposed (2 iter.)	36.19	32.01	29.84	28.38	27.34	24.13	22.47	21.44
House	K-SVD	39.33	35.90	34.19	32.97	31.97	28.01	25.27	23.59
	BM3D	**39.84**	**36.63**	**34.87**	**33.72**	**32.79**	29.54	27.42	25.78
	proposed (1 iter.)	38.54	35.61	34.08	33.06	32.34	29.28	27.26	25.51
	proposed (2 iter.)	38.76	35.80	34.35	33.32	32.54	**29.64**	**27.79**	**26.30**
Peppers	K-SVD	37.80	34.27	32.23	30.88	29.81	26.24	23.54	21.68
	BM3D	**38.09**	**34.70**	**32.77**	**31.37**	**30.26**	26.69	24.71	23.20
	proposed (1 iter.)	37.59	34.13	32.21	30.86	29.78	26.28	24.21	22.53
	proposed (2 iter.)	37.63	34.26	32.40	31.09	30.01	**26.75**	**24.72**	**23.21**

TABLE IV

PARAMETERS USED IN THE INPAINTING EXPERIMENTS

Iteration	K	$\sqrt{n}$	B	ϵ
1	10	16	9	10^2
2	10	8	43	10^4
3	10	5	55	10^8

iterations are shown in Table IV. In order to demonstrate the advantages of our method over simpler interpolation schemes we compare our results to the ones obtained by the matlab function "griddata" which performs cubic interpolation of the missing pixels based on Delaunay triangulation [21], [22]. We also compare the performance of our algorithm to those of three other patch-based algorithms. The first is the algorithm described in chapter 15 of [19], which employs a patch-based sparse representation reconstruction algorithm with a DCT overcomplete dictionary to recover the image patches. We use a patch size of 16×16 pixels in order to improve the results this method produces. We note that we do not employ the K-SVD based algorithm which was also described in this chapter, as our experiments showed that it produces comparable or only slightly better results than the redundant DCT dictionary, at a higher computational cost. The other two algorithms to which we compare our results are the algorithm proposed in [13], which is based on transforming groups of patches, and the state-of-the-art PLE algorithm proposed in [4], which is based on patch clustering. We present here the results reported for these algorithms in [4], for the case of 80% missing pixels. The PSNR values of the results obtained with the different algorithms are shown in Table V. Figure 8 shows the corrupted and the reconstructed images, with the corresponding PSNR values, obtained using Delaunay triangulation, overcomplete DCT dictionary, and 1 and 3 iterations of the proposed scheme. First, it can be seen that the second and third iterations greatly improve the results of our proposed algorithm. It can also be seen that the results obtained with three iterations of our proposed scheme are much better than those obtained with Delaunay triangulation and the overcomplete DCT dictionary. Further, it can be seen that our three iterations results are better than the results obtained with the patch grouping algorithm [13] for the images Lena and Barbara, but slightly inferior to its results for the image House. Finally, it can be seen that the stat-of-the-art PLE algorithm [4] outperforms our three iterations results by only 0.3 dB for the images Lena and House, but by more than 1.2 dB for the image Barbara.

C. Computational Complexity

We next evaluate the computational complexity of a single iteration of the two image processing algorithms described above. We note that for the image denoising scheme, we assume that the filters training has been done beforehand, and exclude it from our calculations. First, building the matrix $\mathbf{X}$

Fig. 7. Denoising results (PSNR) for the images Lena, Barbara, Boat, and House ($\sigma = 25$, input PSNR $= 20.18$ dB): top row - noisy images, center row - 1 iteration results, and bottom row - 2 iterations results.

Fig. 8. Inpainting results (PSNR) of corrupted versions of the images Lena, Barbara, and House with 80% of their pixels missing, obtained with different reconstruction methods: First column - corrupted images, Second column - Delaunay triangulation, Third column - overcomplete DCT dictionary, Fourth column - 1 iteration of the proposed scheme, Fifth column - 3 iterations of the proposed scheme.

which contains the image patches requires $O(nN)$ operations. We assume that when the nearest-neighbor search described above is used with a search window of size $B \times B$, most of the patches do not require to calculate distances outside this neighborhood. Therefore, as calculating each of the distance measures (10) and (19) requires $O(n)$ operations, the number of operations required to calculate a single reordering matrix $\mathbf{P}_k$ can be bounded by $O(NB^2n)$. Next, applying the matrices $\mathbf{P}_k$ and $\mathbf{P}_k^{-1}$ to the n subimages $\tilde{\mathbf{z}}_j$ require $O(nN)$ operations, and so does applying either one of the operators H described

TABLE V
INPAINTING RESULTS (PSNR IN dB) OF CORRUPTED VERSIONS OF THE IMAGES LENA, BARBARA, AND HOUSE WITH 80% OF THEIR PIXELS MISSING, OBTAINED USING DELAUNAY TRIANGULATION (DT), OVERCOMPLETE DCT DICTIONARY, THE ALGORITHM IN [13], PLE [4], AND 1 (P1), 2 (P2), AND 3 (P3) ITERATIONS OF THE PROPOSED SCHEME. FOR EACH IMAGE, THE BEST RESULT IS HIGHLIGHTED

Image	DT	DCT	[13]	[4]	P1	P2	P3
Lena	30.25	29.97	31.62	**32.22**	30.25	31.80	31.96
Barb	22.88	27.15	25.40	**30.94**	27.56	29.34	29.71
House	29.21	29.69	32.87	**33.05**	29.03	32.10	32.71

above to the n subimages $\tilde{\mathbf{z}}_j^p$. Finally , constructing an estimate image by averaging the pixel values obtained with the different subimages also requires $O(nN)$ operations, and averaging the estimates obtained with the different matrices $\mathbf{P}_k$ requires $O(KN)$ operations. Therefore when K permutation matrices are employed, the total complexity is

$$O((n+K)N) + K\left[O(NB^2n) + O(nN)\right] = O(NKB^2n) \tag{20}$$

operations, which means that, as might be expected, the overall complexity is dominated by the creation of the permutation matrices. For a typical case in our experiments, $N = 512^2$, $K = 10$, $n = 64$ and $B = 61$, the above amounts to $6.24 \cdot 10^{11}$ operations. In order to better illustrate these numbers, we also provide run-times: applying two iterations of our denoising scheme to a 512×512 image with noise level $\sigma = 25$ using a non optimized and non parallel matlab implementation, on an Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz, takes about 45 min. Applying three iterations of our inpainting scheme to a 512×512 image with 80% missing pixels take only about 14 minutes. We should note that while in our experiments we employed an exact exhaustive search, approximate nearest neighbor algorithms may be used to alleviate the computational burden.

IV. CONCLUSION

We have proposed a new image processing scheme which is based on smooth 1D ordering of the pixels in the given image. We have shown that using a carefully designed permutation matrices and simple and intuitive 1D operations such as linear filtering and interpolation, the proposed scheme can be used for image denoising and inpainting, where it achieves high quality results.

There are several research directions to extend this work that we are currently considering. The first is to make use of the distances between the patches not only to find the ordering matrices, but also in the reconstruction process of the subimages. These distances carry additional information which might improve the obtained results. Improvements can also be made to the patch ordering scheme itself. We have seen in Section II.B that this scheme performs poorly near the end of the found path, when only a small number of unvisited patches remain. A possible solution could be to develop a scheme which allows patches to be revisited more than once. A different direction is to develop new image processing algorithms which involve optimization problems in which the 1D image reorderings act as regularizers. These may both improve the image denoising and inpainting results, and allow to tackle other applications such as image deblurring, where the operator $\mathbf{M}$ is no longer restricted to be point-wise local. Additionally, the proposed image denoising scheme may be improved by dividing the patches to more than two types, and treating each type differently. Finally, we note that in our work we have not exhausted the potential of the proposed algorithms, and the choice of different parameters (e.g., B, ϵ) for each set of patches may also improve the produced results.

ACKNOWLEDGMENT

The authors thank the authors of [4] for the fruitful discussions and advices, which helped in developing the presented work. The authors also thank the anonymous reviewers for their helpful comments.

REFERENCES

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

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

[3] G. Yu, G. Sapiro, and S. Mallat, "Image modeling and enhancement via structured sparse model selection," in *Proc. 17th IEEE Int. Conf. Image Process.*, Sep. 2010, pp. 1641–1644.

[4] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity," *IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2481–2499, May 2012.

[5] W. Dong, X. Li, L. Zhang, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 457–464.

[6] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 1838–1857, Jul. 2011.

[7] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 479–486.

[8] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[9] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, Jan. 2008.

[10] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep.–Oct. 2009, pp. 2272–2279.

[11] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. 7th Int. Conf. Curves Surf.*, 2012, pp. 711–730.

[12] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[13] X. Li, "Patch-based image interpolation: Algorithms and applications," in *Proc. Int. Workshop Local Non-Local Approx. Image Process.*, 2008, pp. 1–6.

[14] I. Ram, M. Elad, and I. Cohen, "Generalized tree-based wavelet transform," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4199–4209, Sep. 2011.

[15] I. Ram, M. Elad, and I. Cohen, "Redundant wavelets on graphs and high dimensional data clouds," *IEEE Signal Processing Letters*, vol. 19, no. 5, pp. 291–294, May 2012.

[16] G. Plonka, "The easy path wavelet transform: A new adaptive wavelet transform for sparse representation of two-dimensional data," *Multiscale Model. Simul.*, vol. 7, no. 3, pp. 1474–1496, 2009.

[17] D. Heinen and G. Plonka, "Wavelet shrinkage on paths for denoising of scattered data," *Results Math.*, vol. 62, nos. 3–4, pp. 337–354, 2012.

[18] T. H. Cormen, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.

[19] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. New York, NY, USA: Springer-Verlag, 2010.

[20] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," *Wavelets and Statistics*. New York, NY, USA: Springer-Verlag, 1995, pp. 125–150.

[21] T. Yang, *Finite Element Structural Analysis*, vol. 2. Englewood Cliffs, NJ, USA: Prentice-Hall, 1986.

[22] D. Watson, *Contouring: A Guide to the Analysis and Display of Spatial Data (With Programs on Diskette)*. New York, NY, USA: Pergamon, 1992.

Idan Ram received the B.Sc. (*cum laude*) and M.Sc. degrees in electrical engineering from the Technion – Israel Institute of Technology, Haifa, Israel, in 2004 and 2009, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the Technion.

He recieved the Ollendorff Award for research in image processing and analysis in 2013. His current research interests include image processing using sparse signal representations and smooth patch ordering.

Michael Elad (F'12) received the B.Sc., M.Sc., and D.Sc. degrees from the Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa, Israel, in 1986, 1988, and 1997, respectively. Since 2003, he has been a Faculty member with the Computer Science Department, Technion, and since 2010, he has held a Full Professor position.

He works in the field of signal and image processing, specializing in particular on inverse problems, sparse representations and superresolution. He received the Technion's Best Lecturer Award six times, and he was the recipient of the 2007 Solomon Simon Mani Award for Excellence in Teaching, the 2008 Henri Taub Prize for Academic Excellence, and the 2010 Hershel-Rich Prize for Innovation. He is serving as an Associate Editor for SIAM SIIMS, IEEE-TIT, and ACHA. He is serving as a Senior Editor for IEEE SPL.

Israel Cohen (M'01–SM'03) is a Professor of electrical engineering with the Technion – Israel Institute of Technology, Haifa, Israel. He received the B.Sc. (*summa cum laude*), M.Sc., and Ph.D. degrees in electrical engineering from the Technion, in 1990, 1993 and 1998, respectively.

From 1990 to 1998, he was a Research Scientist with RAFAEL Research Laboratories, Haifa, Ministry of Defense. From 1998 to 2001, he was a Post-Doctoral Research Associate with the Computer Science Department, Yale University, New Haven, CT, USA. In 2001, he joined the Electrical Engineering Department, Technion. His current research interests include statistical signal processing, analysis and modeling of acoustic signals, speech enhancement, noise estimation, microphone arrays, source localization, blind source separation, system identification, and adaptive filtering. He is a co-editor of the Multichannel Speech Processing section of the *Springer Handbook of Speech Processing* (Springer, 2008), a co-author of *Noise Reduction in Speech Processing* (Springer, 2009), a co-editor of *Speech Processing in Modern Communication: Challenges and Perspectives* (Springer, 2010), and a general co-chair of the 2010 International Workshop on Acoustic Echo and Noise Control.

He is a recipient of the Alexander Goldberg Prize for Excellence in Research, and the Muriel and David Jacknow Award for Excellence in Teaching. He serves as a member of the IEEE Audio and Acoustic Signal Processing Technical Committee and the IEEE Speech and Language Processing Technical Committee. He served as an Associate Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING and the IEEE SIGNAL PROCESSING LETTERS, and as a Guest Editor of a special issue of the *EURASIP Journal on Advances in Signal Processing* on Advances in Multimicrophone Speech Processing and a special issue of the *Elsevier Speech Communication Journal* on Speech Enhancement.