Clustering-Based Denoising With Locally Learned Dictionaries

Priyam Chatterjee, *Student Member, IEEE*, and Peyman Milanfar, *Senior Member, IEEE*

***Abstract*—In this paper, we propose K-LLD: a patch-based, locally adaptive denoising method based on clustering the given noisy image into regions of similar geometric structure. In order to effectively perform such clustering, we employ as features the local weight functions derived from our earlier work on *steering kernel regression* [1]. These weights are exceedingly informative and robust in conveying reliable local structural information about the image even in the presence of significant amounts of noise. Next, we model each region (or cluster)—which may not be spatially contiguous—by "learning" a best basis describing the patches within that cluster using principal components analysis. This learned basis (or "dictionary") is then employed to optimally estimate the underlying pixel values using a kernel regression framework. An iterated version of the proposed algorithm is also presented which leads to further performance enhancements. We also introduce a novel mechanism for optimally choosing the local patch size for each cluster using Stein's unbiased risk estimator (SURE). We illustrate the overall algorithm's capabilities with several examples. These indicate that the proposed method appears to be competitive with some of the most recently published state of the art denoising methods.**

***Index Terms*—Clustering, dictionary learning, image denoising, kernel regression, principal component analysis, Stein's unbiased risk estimator (SURE).**

I. Introduction

In recent years, affordable hardware has made it possible for digital cameras to capture images of very high resolution. With the advent of such high resolution imaging devices, the signal sensors are becoming increasingly dense in terms of the number of pixels per unit area of the sensor. This means that relatively speaking, fewer photons are available to each sensor element (pixel) and as a result the overall sensor is increasingly more prone to the effects of noise. Hence, denoising remains an important research problem in image processing. Before we deal with the image denoising problem, we first define our observation model as

$$y_i = z_i + \eta_i \tag{1}$$

where z_i is the original pixel intensity of the ith pixel observed as y_i after being corrupted by zero mean independent identically distributed additive noise η_i. Many recently introduced denoising methods are patch based in nature [2]–[7]. Hence, it is useful to formulate the observation model in terms of image patches as well. Decomposing the image into overlapping patches, we can also write the data model as

$$\mathbf{y}_i = \mathbf{z}_i + \boldsymbol{\eta}_i \tag{2}$$

where $\mathbf{z}_i$ is the original image patch with the ith pixel at its center written in a vectorized format and $\mathbf{y}_i$ is the observed vectorized patch corrupted by a noise vector $\boldsymbol{\eta}_i$. Denoising the image is thus solving the inverse problem to estimate the pixel intensities z_i.

Many linear and nonlinear methods have been proposed to solve this problem. One of the earlier methods to achieve considerable success in this domain was the bilateral filter, proposed by Tomasi *et al.* [8]. While this method received broad attention in the image processing and computer vision communities, it fails to perform well in the presence of strong noise. A wavelet domain denoising method using a scale mixture of Gaussians (GSM) proposed by Portilla *et al.* [9] was shown to perform significantly better than the competing methods at the time of its introduction and was considered to be the state of the art until recently. Buades *et al.* [2], [3] proposed a simple patch-based algorithm that takes advantage of the presence of repeating structures in a given image and performs a weighted averaging of pixels with similar neighborhoods to suppress the noise. Kervrann *et al.* [4], [10] considerably improved a localized version of this algorithm using an iterative scheme where the variance of the intensity estimate at each pixel location is used to calculate the weights and the region of support for the averaging process. A more recent method named BM3D, proposed by Dabov *et al.* [7], [11], works on the same principle of using similar patches across the image to perform denoising. However, in contrast to the other techniques discussed here, this method performs denoising in the transform domain. This method of denoising proves to be extremely effective and can be considered to be the state of the art at this time. Another very effective method that achieves excellent denoising results is the K-SVD algorithm proposed by Aharon *et al.* [5], [12]. In this method, an optimal overcomplete dictionary of image patches adapted for the observed noisy data is first determined. Assuming that each image patch is sparse representable, denoising is carried out by coding each patch as a linear combination of only a few patches in the dictionary. In [13], Mairal *et al.* show that further performance improvements can be achieved by learning multiscale global dictionaries. Extension of the

Manuscript received July 25, 2008; revised February 26, 2009. First published May 12, 2009; current version published June 12, 2009. This work was supported in part by the U.S. Air Force under Grant FA9550-07-1-0365. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yongyi Yang.

The authors are with the Department of Electrical Engineering, University of California, Santa Cruz, Santa Cruz, CA 95064 USA (e-mail: priyam@soe.ucsc.edu; milanfar@soe.ucsc.edu).

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.2009.2018575

1057-7149/$25.00 © 2009 IEEE

K-SVD method to denoising color images, among other image processing applications, was also proposed by Mairal *et al.* [6]. Another recent method by Takeda *et al.* [1] addresses the denoising problem by posing it in a kernel regression framework. It has been shown in [14]–[16] that several of the spatial domain denoising methods mentioned earlier, such as the *bilateral filter* [8], *nonlocal means* [2], and *optimal spatial adaptation* [4] can be cast in the kernel regression framework, directly or with locality constraints. Broadly speaking, these denoising methods inherently share the same regression framework as the *steering kernel regression* (SKR) method of [1], differing mainly in the way the kernel or the weight matrix is calculated. Such kernel-based methods have also been shown to have an equivalent variational formulation. This relationship has been studied by Brox *et al.* [17] for the *nonlocal means* (NLM) filter where an iterative version of NLM is motivated from considering an equivalent variational framework using gradient descent. Adaptation of the NLM filter in a variational framework for image denoising and segmentation has also been done by Gilboa *et al.* [18]. Barash [19] studied the relationship between the *bilateral filtering* process and anisotropic diffusion. Tschumperlé *et al.* [20] also reported the equivalence between variational denoising methods and anisotropic diffusion. In each of these works, parallel relationships can be shown for other kernel-based denoising methods which differ mainly in the way their corresponding kernel weights are calculated.

In this paper we propose a framework for denoising by learning a suitable basis function to describe image patches. Use of such basis functions to describe geometric structure has been previously explored leading to the invention of curvelets [21], contourlets [22], bandelets [23], etc. All these tools allow learning of a suitable basis to describe the image, especially the intricate edge and texture regions. We approach the problem of denoising through geometric representation by first explicitly segmenting the image based on the underlying local image structure. For this, we make use of normalized local kernels as indicators of the data geometry to segment the image into structurally similar regions. A suitable basis is learned for each of the regions to best describe the underlying image data. Then we estimate the denoised intensity of each pixel in the image by a regression analysis using the local kernels and the learned dictionary. Our major contribution in this paper is the framework which allows us to perform image denoising through efficient data representation. In this paper, we present well motivated choices of methods to achieve the objectives at each step of our denoising algorithm.

The remainder of the paper is outlined as follows: for the sake of completeness, we briefly describe the SKR method in Section II as we choose to use the *steering weights* as the indicator for local geometric structure. We go on to provide motivations for our method in Section III. Our clustering-based method which makes use of locally learned dictionaries (hence named K-LLD) is then discussed in detail in Section IV; and an iterative version of this method is presented in Section V. This is followed by Section VI, where we describe how to extend our method for better performance by varying the kernel support size in a data adaptive way. The automatic stopping rule is then described in Section VII. The work is then experimentally validated in Section VIII. We finally conclude the paper with a summary and a few words about possible future research in Section IX.

II. Steering Kernel Regression

While kernel regression is a well studied method in statistics and signal processing, it was recently used to good effect to address image processing problems like denoising, interpolation and deblurring by Takeda *et al.* [1], [24]. While many image denoising methods can be shown to have a corresponding kernel regression formulation [3], [14]–[16], the steering kernel regression (SKR) method is distinguished by the way it forms the local regression weights. The weights used in each of the methods in [1], [2], [4], [8] can be considered to be a measure of similarity of a group of pixels compared to a certain pixel or neighborhood of pixels under consideration. For localized methods, a local patch of weights around each pixel (henceforth referred to as *kernels*) needs to be estimated. In SKR, a robust estimate of the gradient is taken into consideration in analyzing the radiometric similarity of two pixels in a neighborhood (patch). This information is then used to determine the shape and size of a canonical kernel (in particular, a Gaussian). The steering kernel in this particular case can be expressed as

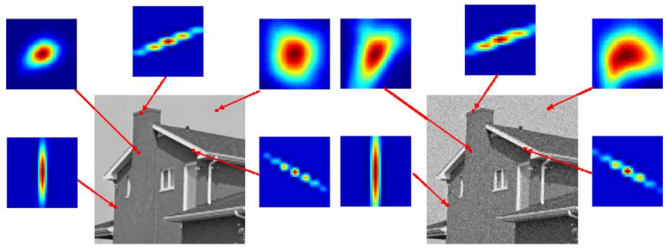
$$w_{ij} = \frac{\sqrt{\det(\mathbf{C}_j)}}{2\pi h^2} \exp\left\{-\frac{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{C}_j (\mathbf{x}_i - \mathbf{x}_j)}{2h^2}\right\} \quad (3)$$

where w_{ij} describes the similarity of the jth pixel with respect to the ith pixel, $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^2$ denote the location of the ith and the jth pixels respectively and h is a global smoothing parameter which controls the support of the steering kernel. The matrix $\mathbf{C}_j$ denotes the symmetric gradient covariance matrix formed from the estimated vertical and horizontal gradients of the jth pixel. It can be expressed as a matrix that allows the Gaussian of (3) to align with the underlying image structure by a combination of elongation and rotation operators. Mathematically, it can be expressed in the form

$$\mathbf{C}_j = \gamma_j \mathbf{U}_{\theta_j} \mathbf{\Lambda}_j \mathbf{U}_{\theta_j}^T \quad (4)$$

where $\mathbf{U}_{\theta_j}$ represents the rotation operator that aligns the Gaussian to the direction θ_j of the underlying edge and $\mathbf{\Lambda}_j$ denotes the elongation operator, and γ_j acts as a scaling parameter. The weight w_{ij} is calculated for each pixel in a neighborhood $\mathcal{N}(i)$ with the ith pixel at its center to form the weight matrix (or kernel).[1] The vectorized version of this kernel we will denote by $\mathbf{w}_i = [\ldots w_{ij} \ldots]^T$ where $j \in \mathcal{N}(i)$. We refer the reader to [1] for a more in-depth explanation of the kernel formation process. A local kernel is obtained in this way for each pixel, providing a similarity measure of a pixel to its local neighborhood. A few such local kernels are shown in Fig. 1. It can be clearly seen how the kernels are representative of the underlying image structure. Furthermore, it can be seen that different locations in the image having different intensities

[1]It is worth noting that for each pixel at $\mathbf{x}_j$ in the neighborhood of a pixel of interest at $\mathbf{x}_i$, we have a distinct covariance matrix $\mathbf{C}_j$. As can be clearly observed in Fig. 1, the steering kernel weights are not simply elliptical in shape as one might initially expect. They are indeed much more interesting and informative.



(a) (b)

Fig. 1. Steering weights formed from (a) the original house image; (b) noisy image with additive white Gaussian noise of standart deviation 15. Note how the weights roughly represent the underlying image structure in the original image.

but similar underlying structure still result in similarly shaped kernels.

For the SKR method of Takeda *et al.* [1], the data is modeled to be locally polynomial where the image is assumed to be sufficiently smooth (locally) to allow fitting of a polynomial of some low degree (usually 0, 1 or 2). We can then rewrite the data model of (2) as

$$\mathbf{y}_i = \boldsymbol{\Phi}\boldsymbol{\beta}_i + \boldsymbol{\eta}_i \tag{5}$$

where the dictionary $\boldsymbol{\Phi}$ is a matrix whose columns are formed from polynomial basis vectors and $\boldsymbol{\beta}_i$ is the vector of coefficients. For a second-order polynomial regression framework, the matrix $\boldsymbol{\Phi}$ has the form

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & (\mathbf{x}_i - \mathbf{x}_j) & \mathrm{vech}^T\{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{6}$$

where the spatial location $\mathbf{x}_j = [x_{1j}\ x_{2j}]^T$ is within a predefined neighborhood of $\mathbf{x}_i = [x_{1i}\ x_{2i}]^T$ and the $\mathrm{vech}(.)$ operator represents the raster scanned lower triangular form of a matrix and is given by (7), shown at the bottom of the page.

Once the local weight matrices are formed using (3), the denoising process is carried out by estimating the pixel intensity at each location in a local polynomial regression framework. This is done by solving the optimization problem

$$\begin{aligned} \hat{\boldsymbol{\beta}}_i &= \arg\min_{\boldsymbol{\beta}_i} (\mathbf{y}_i - \boldsymbol{\Phi}\boldsymbol{\beta}_i)^T \mathbf{W}_i (\mathbf{y}_i - \boldsymbol{\Phi}\boldsymbol{\beta}_i) \\ &= \arg\min_{\boldsymbol{\beta}_i} \|\mathbf{y}_i - \boldsymbol{\Phi}\boldsymbol{\beta}_i\|^2_{\mathbf{W}_i} \end{aligned} \tag{8}$$

where $\mathbf{W}_i = \mathrm{diag}[\mathbf{w}_i]$. The denoised estimate of the ith pixel is given by $\hat{z}_i = \mathbf{e}_1^T \hat{\boldsymbol{\beta}}_i$ where $\mathbf{e}_1$ is the first column of the identity matrix. The premultiplication by $\mathbf{e}_1$ results in picking of the first element of the $\hat{\boldsymbol{\beta}}_i$ vector as the estimated intensity. The weighted least squares problem of (8) has a solution in

$$\hat{\boldsymbol{\beta}}_i = (\boldsymbol{\Phi}^T \mathbf{W}_i \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{W}_i \mathbf{y}_i. \tag{9}$$

Note that while the optimization framework of (8) allows us to estimate all the coefficients of the vector $\widehat{\boldsymbol{\beta}}_i = [\widehat{\beta}_{(0)i} \widehat{\beta}_{(1)i} \ldots \widehat{\beta}_{(M)i}]^T$ simultaneously, only the first entry $\widehat{\beta}_{(0)i}$ is retained as the estimate of the intensity. The other coefficients correspond to estimates of higher order derivatives and are consequently used in obtaining a better estimate of the steering kernel in the iterative version of the SKR algorithm. Further details of that algorithm are explained at length in [1].

III. Motivation

In the SKR framework, the weights or kernels locally adapt to the underlying structure of the image. The weights follow edge directions and dictate the contribution of various pixels in a local neighborhood of the pixel to be denoised. However, this regression framework has two inherent limitations: the basis function remains the same (polynomial) over the entire image, and the order of regression is constant for the entire image. These drawbacks force both smooth and textured regions of any image to be reconstructed using the same basis vectors (and, hence, the same order of regression). Our method aims to alleviate these problems by the use of regression where both the type and number of basis vectors are dictated by the given image data.

In [1], the authors work primarily with regression models of orders 0, 1, and 2. They show that the second-order regression generally leads to a better denoising performance, as compared to lower orders. In their case, the second-order polynomial dictionary ($\boldsymbol{\Phi}$) is formed of six columns, consisting of vectors corresponding to the zeroth-, first-, and second-order terms of a polynomial expansion (in two spatial dimensions). The estimated $\hat{\boldsymbol{\beta}}$ is a vector of six coefficients, in keeping with the number of columns in the dictionary. In their pointwise reconstruction scheme, the authors retain only the first coefficient as the estimated intensity. This can be shown to be a special case of a patch-based version of the same framework. More specifically, once the $\hat{\boldsymbol{\beta}}$ parameters are estimated, one can proceed to reconstruct a vectorized version of each patch in the image as $\hat{\mathbf{z}}_i = \boldsymbol{\Phi}\hat{\boldsymbol{\beta}}_i$. Now, if just the center pixel of the estimated patch is retained, we get the pointwise reconstruction scheme

$$\hat{z}_i = \mathbf{c}^T \hat{\mathbf{z}}_i = \mathbf{c}^T \boldsymbol{\Phi} \hat{\boldsymbol{\beta}}_i \text{ where } \mathbf{c} = [0 \ldots 0\ 1\ 0 \ldots 0]^T. \tag{10}$$

When $\boldsymbol{\Phi}$ is the polynomial basis, it turns out that $\mathbf{c}^T\boldsymbol{\Phi} = \mathbf{e}_1^T$, leading exactly to the same pointwise estimate as in [1]. Hence, one can easily extend the SKR method to perform denoising by

$$\begin{aligned} &\mathrm{vech}\{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\} \\ &\quad = \mathrm{vech}\left\{ \begin{bmatrix} (x_{1i} - x_{1j})^2 & (x_{1i} - x_{1j})(x_{2i} - x_{2j}) \\ (x_{1i} - x_{1j})(x_{2i} - x_{2j}) & (x_{2i} - x_{2j})^2 \end{bmatrix} \right\} \\ &\quad = [\,(x_{1i} - x_{1j})^2 \quad (x_{1i} - x_{1j})(x_{2i} - x_{2j}) \quad (x_{2i} - x_{2j})^2\,]^T \end{aligned} \tag{7}$$

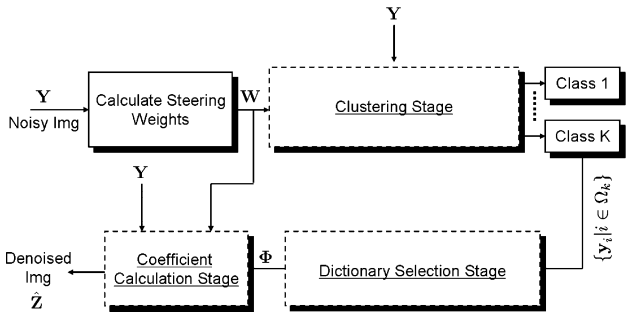


Fig. 2. Block diagram of our algorithm.

estimating patches in the image. Each of these estimated (overlapping) patches, written in a vectorized form as $\hat{\mathbf{z}}_i$, is then a linear combination of the columns in the dictionary $\mathbf{\Phi}$. One can then imagine each of the columns of the dictionary to be a vectorized version of some patch of pixel intensities. This interpretation then becomes somewhat similar to the recently introduced K-SVD algorithm developed by Aharon *et al.* [5]. However, in their work the authors develop an overcomplete dictionary, which is learned to best represent the patches of pixels in a given image. This method tends to find the best *global* overcomplete dictionary and represent each image patch as a linear combination of only a few dictionary vectors (atoms). The coefficients of the linear combination are found through a sparse coding process. Methods of learning such a globally overcomplete dictionary that best describes patches in any given image have been actively researched for various image processing tasks [25]–[27]. One can easily think of such a learned global overcomplete dictionary to be local in nature where, for each patch, the dictionary consists of only the basis vectors (or atoms) used for sparse coding. Hence, each image patch has a local dictionary which is a much smaller subset of the global dictionary.

This brings out the inherent similarities and dissimilarities of the two successful denoising techniques, namely the local kernel regression-based methods and those that learn a global data-adaptive dictionary. Our work introduces a general denoising framework that bridges the gap between these two techniques. It also addresses the question of efficient data representation through a simple preprocessing step of data clustering. Data representation has been the objective of transforms like curvelets [21], contourlets [22], bandelets [23], etc., where the main aim is to define geometrically informative bases. Such bases form dictionaries which are not data dependent and hence can be termed as *fixed* dictionaries. Contrary to such approaches, our method learns local dictionaries that will best represent each image patch. Moreover, we note that patches of similar geometric structure should ideally have similar dictionaries. In our framework, we take advantage of such similarities in learning the local *data adaptive* dictionaries by grouping together regions of similar geometric structure. Finally, we perform denoising through data adaptive kernel regression using the local steering weights.[2] While our method is directed specifically at denoising, a similar methodology has been employed by Hong *et al.* [28] to build a multiscale hybrid linear model for image encoding.

[2]Kernels from other methods such as those presented in [2], [4], [8] can be used as well. However, our choice of the steering kernel is largely motivated from its robustness to noise. Invariance to intensity difference is another property of the normalized steering weights which proves advantageous to our method. This will become clearer later in Section IV-A.

IV. Clustering-Based Denoising (K-LLD)

Our algorithm aims to improve on the successful kernel regression framework by eliminating the limitations mentioned before; namely, the static nature of the dictionary, and the constancy of the approximation order across the image. We go about this task with a clustering-based algorithm which consists of three stages (Fig. 2): the **clustering** step where the image is clustered using features that capture the local structure of the underlying image data (patches of pixels from the image), the **dictionary selection** stage where we form an optimized dictionary that adapts to the geometric structure of the image patches in each cluster; and, finally, the **coefficient calculation** stage where the coefficients for the linear combination of dictionary atoms are estimated, subject to the (steering) kernel weights. In the following sections, we describe and motivate each of the above stages in detail.

A. Clustering

Feature Selection: In the initial stage, our algorithm attempts to perform clustering to identify regions of similar structure in the image. In the existing literature, a wide variety of clustering algorithms are available [29]. To perform clustering we need to first identify informative features from the image. While the choice of features remains an open research problem, in many cases the features are directly computed from the input image. Commonly used low level features to identify similar pixels (or patches) are pixel intensities [2], [8], [28], gradient information [30], etc., or a combination of these. The use of such features directly from the input image is not advisable for our denoising problem due to their instability in the presence of noise. However, it has been observed in [1] that the steering weights computed in a neighborhood are robust to the presence of significant amounts of noise. These weights are roughly representative of the underlying local data structure. Moreover, the normalized weights (denoted by $\check{\mathbf{w}}_i = \mathbf{w}_i / \sum_j w_{ij}$) exhibit invariance to intensity difference between image patches. Hence, we use a vectorized version of these normalized weights as features to perform clustering. Thus, clustering is performed using feature vectors $\check{\mathbf{w}}_i$ of size $N \times 1$ for each local steering kernel computed over a $\sqrt{N} \times \sqrt{N}$ window[3] centered at pixel i in the image. That is to say, every pixel of the image is mapped to a feature vector of size $N \times 1$. This is a transformation that maps the pixel data into a much higher dimensional space in which the clustering takes place. At the end of this stage we expect the image to be divided into not necessarily contiguous regions (Ω_k), each containing patches of similar structure. Hence, the entire noisy image $\mathbf{Y}$ can be thought to be composed of a union of such clusters

$$\mathbf{Y} = \bigcup_{k=1}^{K} \{\mathbf{y}_i | i \in \Omega_k\}. \tag{11}$$

[3]In this paper, we will interchangeably refer to this window of pixels as a patch. The size of such a patch and the kernel support size are the same as the size of the window.

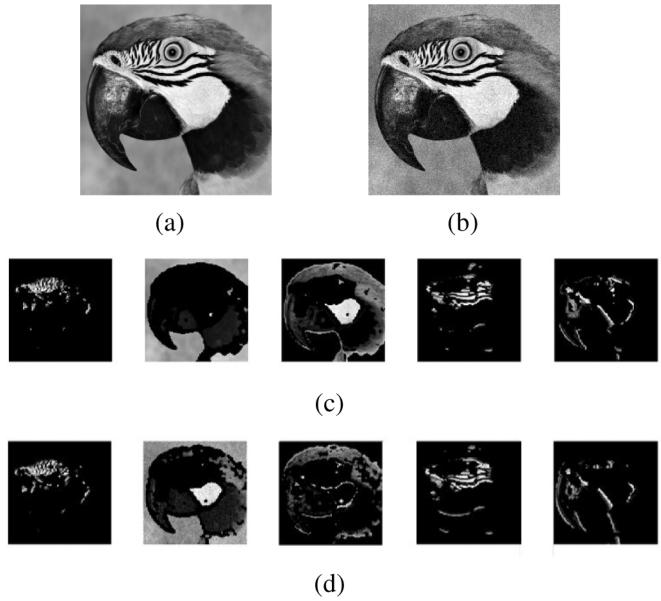


Fig. 3. Performance of clustering using the steering weights as features. Notice how the clustering takes into account the geometric structure of the underlying image even in the presence of additive white Gaussian noise of standard deviation 15. (a) Parrot image, (b) noisy image, (c) clustering of noise-free image, (d) clustering of noisy image.

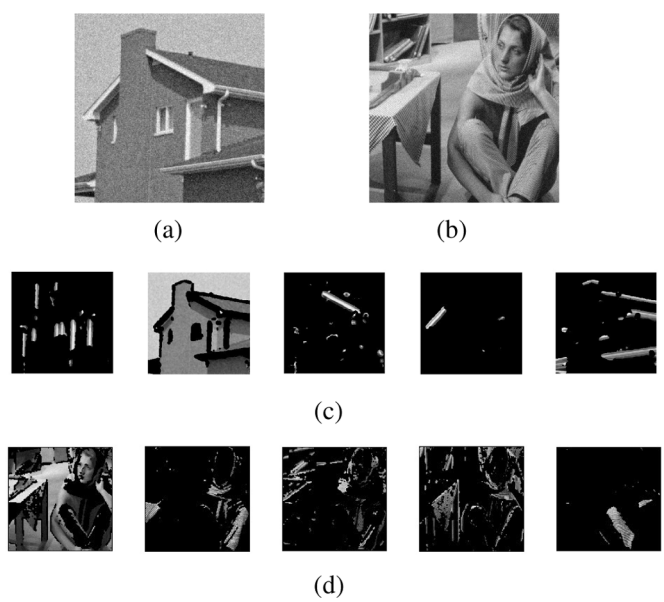


Fig. 4. Clustering results by K-Means algorithm on the house and Barbara images, corrupted by white Gaussian noise of standard deviation 15. Notice how edges and patterns of a certain kind are clustered together even when they have different intensities. (a) Noisy house image, (b) noisy Barbara image, (c) clustering of house image, (d) clustering of Barbara image.

Examples of such clustering can be seen in Figs. 3 and 4. While the clusters are formed of patches, for simplicity, we resort to an abuse of notation in (11) by denoting $i \in \Omega_k$ to signify $\mathbf{y}_i$ belongs to the cluster Ω_k. We can take this liberty since each patch in an image is uniquely identified by the index of its center pixel.

Distance Metric: Before we proceed to perform clustering on the weights, we need to specify a metric to calculate the distance between two weight functions (features). The easiest measure of distance between the steering weights of two patches would be to calculate the L_2 or the L_1 distance between them. In our work we use the L_2 norm as our distance metric. However, one may argue that such a choice may not be ideal as neither of these measures take into account the spatial location where the weights differ. One can then consider calculating the distance using a weighted L_2 or L_1 norm. Other measures of similarity such as taking the exponential of the negative weighted L_2 norm (where the measure is restricted to values between 0 and 1), or the Mahalanobis distance can also be considered.

The choice of the best distance metric to quantify the similarity between the kernel functions and the reasons to use such a metric remain open questions. In our case, for now we work with the simple L_2 distance which proves to be an effective distance metric to perform satisfactory clustering for our algorithm. Our experiments have shown that resorting to other metrics like the weighted L_2 norm and the Kullback-Leibler (K-L) divergence [31] have negligible effect on the final denoising result, while the latter proves to be much more computationally intensive. It is mainly due to this reason that we refrain from diverting much attention to the open problem of better distance metric selection.

Clustering Algorithm: While there are many clustering algorithms available in the literature, we need a method that is relatively fast and performs satisfactory segmentation of the image into geometrically similar regions based on the steering weights. For our method, we make use of a version of the standard K-means algorithm [32] by Lloyd [33]. This clustering algorithm is one of the simplest unsupervised methods where the motivation of clustering is to segment the image into some prefixed number (K) of clusters such that for each class, the squared distance of any feature (normalized weight) vector to the center of the class is minimized. If we use the L_2 distance as our metric, the method aims to minimize the intracluster variance. K-means then aims to find an iterative solution to compute the class centers $\overline{\mathbf{w}}^{(k)}$ (and, hence, the classes Ω_k) and the membership of each $\check{\mathbf{w}}_i$ that minimizes the objective function

$$J = \sum_{k=1}^{K} \sum_{i \in \Omega_k} d^2\left(\check{\mathbf{w}}_i, \overline{\mathbf{w}}^{(k)}\right) = \sum_{k=1}^{K} \sum_{i \in \Omega_k} \left\| \check{\mathbf{w}}_i - \overline{\mathbf{w}}^{(k)} \right\|^2 \tag{12}$$

where $\overline{\mathbf{w}}^{(k)}$ is the mean vector for the kth cluster. Figs. 3 and 4 display the discriminative capabilities of the normalized steering weights as features when used with the K-Means algorithm for clustering. We can see that the flat regions are clustered in the same class, irrespective of the difference in intensities in different regions. Moreover, edges along a particular direction (for the house image) and texture of a particular nature (for the parrot and Barbara images) are associated with the same cluster. This clearly shows how the clusters are formed by considering geometric similarity between image patches, even in the presence of considerable noise.

However, this clustering is still not perfect. The effect of noise on the clustering algorithm is brought to light in Fig. 3 by comparing the image segments obtained from the noisy and the noise-free parrot images. It can be seen that regions of fine texture such as the cheek area of the parrot are misclassified with the background in the noisy case. Such erroneous classifications

are also noticed in Fig. 4. Notice how the top of the table in the Barbara image, containing very fine periodic structure, is classified together with the flat background region. We observe a similar effect for the house image where the facade of the house, having fine structure, is grouped together with the smooth sky region. This failure can be attributed to the fact that in the presence of considerable noise, the steering weights may still render very fine structure indistinguishable from smoother regions corrupted by noise. This is demonstrated in Fig. 1 by the similarity in structure of the steering weights for pixels in the sky and the facade in the noisy house image. While this is certainly something that is not desired from the clustering stage in an ideal scenario, we must evaluate the clustering performance taking into consideration the effect of the noise. We find that the degeneration is quite graceful since the finer details are not easily distinguishable even by the human eye. In general, the advantages of our clustering scheme, especially under strong noise, easily outweigh these shortcomings. This can be readily seen from Figs. 3 and 4, where the majority of the clustering is true to the underlying image geometry.

B. Dictionary Selection

Once the clusters are formed, we proceed to form a dictionary best suited to each cluster independently. For each cluster we intend to find a dictionary that best describes the structure of the underlying data within that cluster. In other words, for each image patch in a cluster Ω_k, we want to find an estimate $\hat{\mathbf{y}}_i$ which best approximates the input vectorized patch $\mathbf{y}_i$. Mathematically, we intend to find the optimal $\mathbf{\Phi}^{(k)}$ and $\boldsymbol{\beta}_i$ to minimize the cost function

$$\sum_{i\in\Omega_k} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 = \sum_{i\in\Omega_k} \|\mathbf{y}_i - \left(\bar{\mathbf{y}}^{(k)} + \mathbf{\Phi}^{(k)}\boldsymbol{\beta}_i\right)\|^2 \quad (13)$$

where $\bar{\mathbf{y}}^{(k)}$ is the mean vector of the set $Y^{(k)} = \{\mathbf{y}_i | i \in \Omega_k\}$, $\mathbf{\Phi}^{(k)}$ denotes the dictionary which best represents the patches in cluster Ω_k and $\boldsymbol{\beta}_i$ denotes a vector of coefficients for the linear combination of dictionary atoms to compute $\hat{\mathbf{y}}_i$. Note that this allows us to learn a dictionary whose form is dictated by the underlying data and not restricted to be polynomial, as is the case for SKR [1]. This can in effect be thought of as progressively selecting one atom at a time such that the squared L_2 reconstruction error is minimized with each atom. This can be shown by rewriting (13) as

$$\begin{aligned}
&\sum_{i\in\Omega_k} \left\|\mathbf{y}_i - \bar{\mathbf{y}}^{(k)} - \mathbf{\Phi}^{(k)}\boldsymbol{\beta}_i\right\|^2 \\
&= \sum_{i\in\Omega_k} \left\|\left(\mathbf{y}_i - \bar{\mathbf{y}}^{(k)}\right) - \boldsymbol{\phi}^{(k)}_{(1)}\beta_{(1)i} - \boldsymbol{\phi}^{(k)}_{(2)}\beta_{(2)i} - \ldots\right\|^2 \\
&= \sum_{i\in\Omega_k} \left\|\tilde{\mathbf{y}}^{(k)}_i - \left[\boldsymbol{\phi}^{(k)}_{(1)}\boldsymbol{\phi}^{(k)}_{(2)}\cdots\boldsymbol{\phi}^{(k)}_{(M)}\right]\begin{bmatrix}\beta_{(1)i}\\ \vdots \\ \beta_{(M)i}\end{bmatrix}\right\|^2 \quad (14)
\end{aligned}$$

where $\tilde{\mathbf{y}}^{(k)}_i = \mathbf{y}_i - \bar{\mathbf{y}}^{(k)}$ performs centering of the data by mean subtraction and M is the total number of atoms needed to fully describe all $\tilde{\mathbf{y}}^{(k)}_i$. Choosing a dictionary $\mathbf{\Phi}^{(k)}$ to minimize this cost function is nontrivial since the $\boldsymbol{\beta}_i$ parameters are also unknown. One way to estimate these two unknowns is by using a numerical method of alternate minimization where we first assume one of the two variables (say $\mathbf{\Phi}^{(k)}$) to be fixed and minimize the cost function of (14) with respect to the other ($\boldsymbol{\beta}_i$ in our case). Once a minimum is obtained for $\boldsymbol{\beta}_i$, it is kept fixed and we minimize with respect to $\mathbf{\Phi}^{(k)}$. This process is carried on iteratively. However, such a method requires good initial guesses for each of the unknowns. To circumvent this difficulty, we appeal to the variable projection approach [34] where we first write down an analytical expression for the estimate of $\boldsymbol{\beta}_i$ assuming $\mathbf{\Phi}^{(k)}$ to be known. The estimate of $\boldsymbol{\beta}_i$ that minimizes the cost function in (13) is given by

$$\boldsymbol{\beta}_i = \left(\mathbf{\Phi}^{(k)T}\mathbf{\Phi}^{(k)}\right)^{-1}\mathbf{\Phi}^{(k)T}\tilde{\mathbf{y}}^{(k)}_i. \quad (15)$$

This solution is then plugged into (13) and we reformulate the problem as that of minimization with respect to $\mathbf{\Phi}^{(k)}$ alone. To simplify the problem further, we enforce the dictionary to be orthonormal, transforming the optimization problem of (13) into

$$\begin{aligned}
\hat{\mathbf{\Phi}}^{(k)} &= \arg\min_{\mathbf{\Phi}^{(k)}} \sum_{i\in\Omega_k} \left\|\tilde{\mathbf{y}}^{(k)}_i - \mathbf{\Phi}^{(k)}\boldsymbol{\beta}_i\right\|^2 \\
&= \arg\min_{\mathbf{\Phi}^{(k)}} \sum_{i\in\Omega_k} \Bigg\|\tilde{\mathbf{y}}^{(k)}_i \\
&\qquad - \mathbf{\Phi}^{(k)}\left(\mathbf{\Phi}^{(k)T}\mathbf{\Phi}^{(k)}\right)^{-1}\mathbf{\Phi}^{(k)T}\tilde{\mathbf{y}}^{(k)}_i\Bigg\|^2 \\
&= \arg\min_{\mathbf{\Phi}^{(k)}} \sum_{i\in\Omega_k} \left\|\tilde{\mathbf{y}}^{(k)}_i - \mathbf{\Phi}^{(k)}\mathbf{\Phi}^{(k)T}\tilde{\mathbf{y}}^{(k)}_i\right\|^2 \\
&= \arg\min_{\mathbf{\Phi}^{(k)}} \left\|\tilde{\mathbf{Y}}^{(k)} - \mathbf{\Phi}^{(k)}\mathbf{\Phi}^{(k)T}\tilde{\mathbf{Y}}^{(k)}\right\|_F^2 \\
&\qquad \text{such that } \mathbf{\Phi}^{(k)T}\mathbf{\Phi}^{(k)} = \mathbf{I}. \quad (16)
\end{aligned}$$

Here, $\tilde{\mathbf{Y}}^{(k)}$ is the matrix where each column is a vector from the set of all vectorized patches $\{\tilde{\mathbf{y}}^{(k)}_i | i \in \Omega_k\}$ and $\|\cdot\|_F$ denotes the Frobenius norm for matrices. This non-iterative approach of variable projections has been known to contain the same fixed points as solving the original optimization of (13) assuming quite general smoothness conditions described in [34] and [35]. Moreover, constraining the dictionary to be orthonormal satisfies the condition of $\mathbf{\Phi}^{(k)}$ having constant rank which is assumed by the variable projection method.[4]

The problem of dictionary learning, as expressed in (16), essentially boils down to finding the principal components of the (centered) data matrix $\tilde{\mathbf{Y}}^{(k)}$. However, we run the risk of overfitting to the noise since the data vectors $\mathbf{y}_i$ are noisy. To overcome this problem, we restrict the number of atoms to be the first m_k principal components so that we do not overfit to the noise. The optimization framework that we essentially work with is thus

$$m_k = \max(m) \text{ subject to } \sum_{i\in\Omega_k} \left\|\tilde{\mathbf{y}}^{(k)}_i - \sum_{j=1}^{m} \hat{\boldsymbol{\phi}}^{(k)}_{(j)}\hat{\boldsymbol{\phi}}^{(k)T}_{(j)}\tilde{\mathbf{y}}^{(k)}_i\right\|^2 \geqslant rN\sigma^2 \quad (17)$$

[4]The variable projection method assumes that the matrix $\mathbf{\Phi}^{(k)}$ be of constant rank, even if not full rank. In our case, since $\mathbf{\Phi}^{(k)}$ is orthonormal, the regularity condition is automatically satisfied.

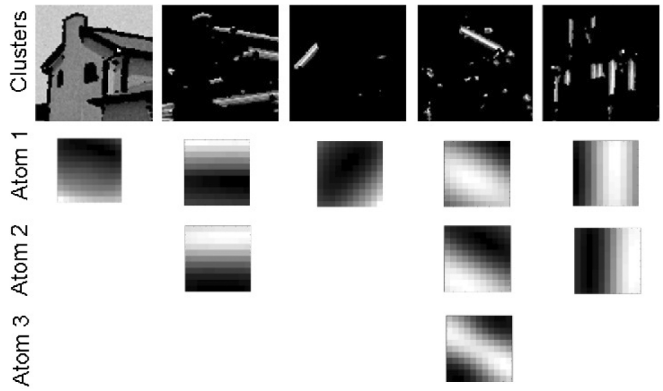


Fig. 5. Data adaptive dictionary for each of the clusters of the house image that is corrupted by additive white Gaussian noise of standard deviation 15. The patch size and hence size of each atom is 11 × 11. Note that different clusters may require different number of atoms and how the atoms represent the data structure of the corresponding cluster.

where N is the number of pixels in each patch, r is a constant and σ is the standard deviation of the corrupting noise which is assumed to be known or can be estimated [4], [36]. However, instead of solving (17) directly, we select the top m_k most informative principal components to reconstruct a rank m_k approximation of $\hat{\mathbf{Y}}_i^{(k)}$. The optimal choice of m_k can be easily obtained from the PCA framework as the solution of the constrained optimization problem

$$m_k = \max(m) \text{ such that } \sum_{j=m+1}^{M} s_j^2 \geqslant rN\sigma^2 \tag{18}$$

where $s_1 \geqslant s_2 \geqslant \ldots \geqslant s_M \geqslant 0$ are the singular values of the data matrix $\hat{\mathbf{Y}}^{(k)}$ obtained directly from the principal component analysis framework. The motivation for the constraint term of (18) is to find the principal components that contribute towards explaining the variance in the data that arises due to the presence of noise and discard them from the dictionary.

Once the dictionary is learned for each cluster, we can represent each of the vectorized versions of the noisy patches as a linear combination of the atoms in the dictionary in the form

$$\hat{\mathbf{y}}_i = \bar{\mathbf{y}}^{(k)} + \sum_{j=1}^{m_k} \hat{\boldsymbol{\phi}}_{(j)}^{(k)} \beta_{(j)i} \quad \forall i \in \Omega_k \tag{19}$$

where $\bar{\mathbf{y}}^{(k)}$ represents the mean patch of the kth cluster. As previously mentioned, the value of m_k that satisfies (18) may vary across clusters. This is illustrated in Fig. 5.

At this point, the coefficients $\beta_{(j)i}$ still have to be determined. One way of obtaining the coefficients would be by using (15). While such a formulation would stay true to the variable projection framework, we note that it does not take into account the local kernel weights $\mathbf{w}_i$ for each patch. This is necessary since the dictionary atoms that best describe the cluster need not necessarily be optimal for the patches that lie at the cluster boundaries, where important geometric transitions (such as changes in texture, occlusions, etc.) between regions appear. Moreover, patches that may have been misclassified due to the presence of noise will also not be reconstructed effectively. Local similarity information in the form of the kernels are thus useful for better restoration, especially at the cluster boundaries. We describe the coefficient calculation procedure using the local kernels in the next section.

C. Coefficient Calculation

Once the dictionary is formed for each cluster, we proceed to estimate the $\boldsymbol{\beta}_i$ parameters under a regression framework. We pose this as an optimization problem

$$\begin{aligned}\hat{\boldsymbol{\beta}}_i &= \arg\min_{\boldsymbol{\beta}_i} \left(\tilde{\mathbf{y}}_i^{(k)} - \hat{\boldsymbol{\Phi}}^{(k)}\boldsymbol{\beta}_i\right)^T \mathbf{W}_i \left(\tilde{\mathbf{y}}_i^{(k)} - \hat{\boldsymbol{\Phi}}^{(k)}\boldsymbol{\beta}_i\right) \\ &= \arg\min_{\boldsymbol{\beta}_i} \left\|\tilde{\mathbf{y}}_i^{(k)} - \hat{\boldsymbol{\Phi}}^{(k)}\boldsymbol{\beta}_i\right\|_{\mathbf{W}_i}^2 \quad \forall i \in \Omega_k\end{aligned} \tag{20}$$

where $\tilde{\mathbf{y}}_i^{(k)} = \mathbf{y}_i - \bar{\mathbf{y}}^{(k)}$ is the mean subtracted vectorized version of the ith patch in the image. Equation (20) is the same cost function for the SKR problem in [1], except that $\hat{\boldsymbol{\Phi}}^{(k)}$ is no longer a fixed polynomial dictionary of a certain order. The dictionary now is adapted to a specific class of image structure that is captured by each cluster. Furthermore, the number of principal components or dictionary atoms that will be needed to fit a prespecified percentage of data varies across the different clusters, as shown in Fig. 5. Here, for example, the smooth flat regions of the first cluster need only one dictionary atom to well describe each patch whereas regions with sharp edges and texture may require more atoms. This results in a varying order of regression across locations depending on which cluster each pixel belongs to. Equation (20) is simply a weighted least squares problem which has a solution in

$$\hat{\boldsymbol{\beta}}_i = \left(\hat{\boldsymbol{\Phi}}^{(k)T}\mathbf{W}_i\hat{\boldsymbol{\Phi}}^{(k)}\right)^{-1} \hat{\boldsymbol{\Phi}}^{(k)T}\mathbf{W}_i\tilde{\mathbf{y}}_i^{(k)}. \tag{21}$$

Once the $\hat{\boldsymbol{\beta}}_i$ parameters are estimated, we can reconstruct the target patch as

$$\begin{aligned}\hat{\mathbf{z}}_i &= \bar{\mathbf{y}}^{(k)} + \hat{\boldsymbol{\phi}}_{(1)}^{(k)}\hat{\beta}_{(1)i} + \hat{\boldsymbol{\phi}}_{(2)}^{(k)}\hat{\beta}_{(2)i} + \cdots \\ &\quad + \hat{\boldsymbol{\phi}}_{(m_k)}^{(k)}\hat{\beta}_{(m_k)i} \\ &= \bar{\mathbf{y}}^{(k)} + \hat{\boldsymbol{\Phi}}^{(k)}\hat{\boldsymbol{\beta}}_i \quad \forall i \in \Omega_k.\end{aligned} \tag{22}$$

The patches thus estimated are overlapping, so we should ideally optimally combine the overlapping regions somehow to form the final image. However, since the $\hat{\boldsymbol{\beta}}$ parameters are estimated taking into account the local weights, the pixels in each of the estimated patches have a high confidence in regions where the local weights are high. As a result, the patch reconstruction form of (22) is more accurate along the edge directions and towards the center of the patch under consideration (that is, wherever the local weights are strong). This is especially true for the patches that lie near the boundaries of a cluster,[5] or those that may have been misclassified. Hence, instead of combining the different estimates of the same pixel, we simply retain the intensity estimate for which the pixel is at the center of the patch being restored. This is done by simply using the center pixel of each estimated patch, as shown in (10). The combination of a

[5]Clustering using image patches allows us to associate with each cluster a region in the image formed by the center pixels of the member patches. For our work, any patch that has any part of it lying outside this region is considered a borderline patch (or equivalently borderline point).

local regression model to obtain the $\hat{\boldsymbol{\beta}}_i$ parameters and retention of the center pixel in each reconstructed patch builds into the method a tolerance to errors in the classification stage.

As mentioned before, the outlined method results in different orders of regression in each cluster depending on the underlying image characteristics. The order of regression may be quite high at certain locations, especially in the texture regions. Moreover, the weight matrix can be singular, as can be seen in Fig. 1 where in some regions only few of the weights in the local kernels have any significant contribution. To make sure that the term $(\hat{\boldsymbol{\Phi}}^{(k)T}\mathbf{W}_i\hat{\boldsymbol{\Phi}}^{(k)})$ does not become ill-conditioned, we truncate the dictionary size such that the condition number of the term to be inverted remains below a certain threshold. Note that this would involve checking the condition number for each pixel location, making the process extremely inefficient. However, we make use of the fact that the weight matrices in each cluster should be quite similar in nature, and, hence, we can check the condition number of $(\hat{\boldsymbol{\Phi}}^{(k)T}\mathbf{W}_i\hat{\boldsymbol{\Phi}}^{(k)})$ for a much smaller subset of pixels in each cluster and determine the dictionary size. This truncated dictionary is now used as the dictionary for the whole cluster and the $\hat{\boldsymbol{\beta}}_i$ parameters are calculated using (21).

It is worth pointing out here that though we make use of kernel regression to estimate the coefficients for the linear combination shown in (19), this coefficient calculation process can very easily be replaced by another method. In our work, we have experimented with a version of orthogonal matching pursuit (OMP) [37] that takes into account the weight matrix to estimate the coefficients. While the resulting denoised image is similar in nature, OMP proves to be much slower than performing regression using the steering kernels.

V. Iterative Denoising

In this section, we explain how the clustering-based method described above can be iterated to improve the final denoised image. Fig. 6 shows the block diagram of the entire method. The motivation to iterate the method stems from the fact that presence of high amounts of noise adversely affects the calculation of the local image gradients, which in turn affects the calculation of the steering weights. This can hamper the ability of the clustering algorithm to properly segment the image into regions of similar structure. Furthermore, the dictionary selection process also becomes influenced by noise. Considering these effects, it makes sense to assume that the first two stages of the algorithm work better on images with less noise. Once all the three stages of the algorithm, namely *clustering*, *dictionary selection*, and *coefficient calculation stages* are run on the input noisy image, we get an output image where the noise is considerably reduced. The steering weights are then recalculated on this output image. These new weights are then used to perform clustering of the denoised image into similar regions as described in Section IV-A. Once the clusters are formed with the new weights, we use patches from the updated image to form dictionary atoms in the way outlined in Section IV-B. We thus choose the principal components from the updated image, which is a truer representation of the underlying data due to the relatively smaller amount of noise present. Finally, the new dictionary and the updated weights are used to perform the final stage

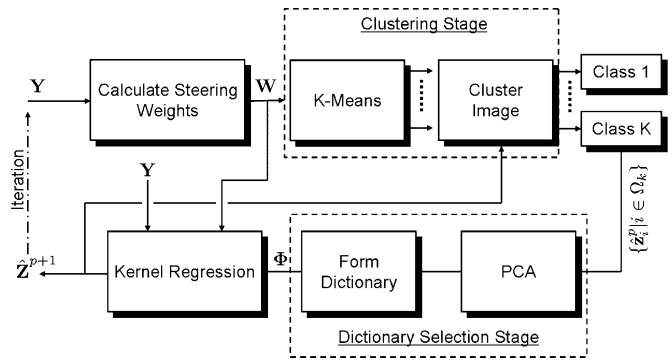


Fig. 6. Block diagram of the iterative version of our algorithm.

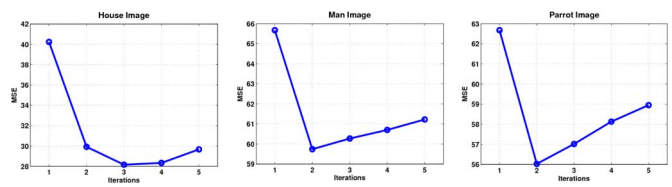


Fig. 7. Illustration of how the MSE varies with iterations for additive white Gaussian noise of standard deviation 15. The MSE of the input image was approximately 225 in each case.

of kernel regression as explained in Section IV-C. It is important to point out, however, that unlike the previous stages, we perform the final denoising using the original noisy image. In our experiments, we found that performing denoising with the updated weights on the original image better preserves edges and texture than denoising the output of the previous iteration. Fig. 7 illustrates the usefulness of iterating the current framework. Here we see that there is usually a sharp drop in mean squared error (MSE) from the initial MSE of 225 in the first couple of iterations. The minimum MSE was reached in the second iteration for the parrot and the man images and in the third iteration for the house image. On iterating further, the denoised image gets blurred in most cases, which results in a rise in MSE. This is much more noticeable in images containing texture, as in the parrot and man images. This necessitates a stopping criterion for the iterative framework. A resolution of this question based on Stein's unbiased risk estimator (SURE) [38] is presented later in Section VII.

VI. Data-Adaptive Kernel Support Size

In the previous sections, we motivated and presented an algorithm where the image is segmented into regions of similar geometric structure and an optimal dictionary is learned on each of the segments. A kernel regression is then performed to finally denoise the image. We also discussed how this process can be carried out in an iterative fashion. One factor which can strongly affect the performance of the resulting algorithm is the choice of the support size of the kernel. The size of the kernel has to be chosen keeping in mind the amount of noise present as well as the image structure that we are dealing with. Images consisting mainly of smooth regions will be denoised better when the kernel has a large support (i.e., when the number of pixels N considered for denoising is large) whereas texture regions are typically restored better by smoothing over smaller regions

(fewer number of pixels). This leads us to extend our method to incorporate a variable kernel support (which we also refer to as window size or patch size) in keeping with the underlying data structure. Thus, we want to adaptively control the number of pixels that will be considered in the denoising process of each pixel depending on the local geometric structure of the image. Such an optimization has been previously considered by researchers to obtain improved performance [4], [39]. In our approach, we readily take advantage of the fact that our method works by clustering the image into regions of similar underlying geometry. Since the optimal kernel size depends on the underlying structure, we can proceed to optimize the window size for each cluster. This leads to finding a data adaptive dictionary for each cluster, where even the size of the atoms differ in each cluster. In a sense, we now further optimize the dictionary for each cluster.

Before we move ahead with determining the optimal window size in each cluster, we need to first identify a measure of the denoising performance for each cluster. We choose to quantify the denoising performance in each cluster using the mean squared error (MSE) measure. Although, in the absence of ground truth calculating the MSE is not directly possible, an approximation of the MSE can be used to evaluate the denoising performance for a specific window size. In [4], Kervrann *et. al* approximate the local MSE by the variance of the intensity estimate at each pixel to determine the optimal neighborhood over which denoising is to be performed. They show that a function of the variance can be used as an indicator of an upper bound for the MSE. For our work, we estimate the MSE using SURE [38]. The use of SURE to effectively optimize parameters for denoising has also been reported in [40]–[43]. In our case the parameter to optimize here is the support size of the local kernels within a cluster.

A. Stein's Unbiased Risk Estimate (SURE)

In [38], Stein presented a method by which an unbiased estimate of the MSE can be obtained, considering the corrupting noise to be sampled from a normal distribution. We first need to formulate an analytical expression for our pointwise denoising process. To do this, we define a local operator $\mathbf{g}_i$ that quantifies the noise suppressed at each pixel location such that

$$\hat{z}_i = y_i + \mathbf{g}_i^T \mathbf{y}_i \quad \forall i \in \Omega_k. \tag{23}$$

Then, from the SURE framework [38], we can get an estimate of the MSE as

$$\begin{aligned} E\left[\left|z_i - \left(y_i + \mathbf{g}_i^T \mathbf{y}_i\right)\right|^2\right] &= \sigma^2 + \frac{1}{|\Omega_k|} \sum_{i \in \Omega_k} \Bigg[\left(\mathbf{g}_i^T \mathbf{y}_i\right)^2 \\ &\quad + 2\sigma^2 \left(\nabla \cdot \mathbf{g}_i^T\right)(\mathbf{y}_i) \Bigg] \end{aligned} \tag{24}$$

where $|\Omega_k|$ denotes the number of pixels (i.e., set cardinality) of the kth cluster, σ is the standard deviation of the additive noise and $(\nabla \cdot \mathbf{g}_i^T)(\mathbf{y}_i)$ denotes the gradient of $\mathbf{g}_i^T \mathbf{y}_i$ with respect to the ith pixel. From (22) and (23), we get the expression

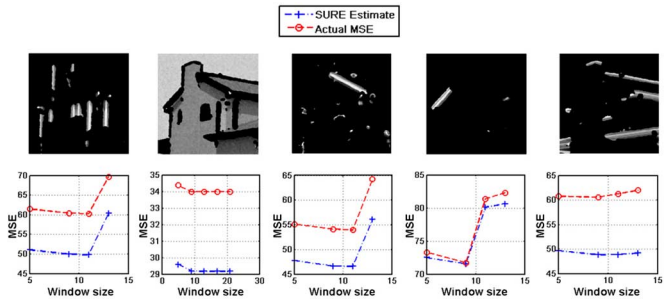


Fig. 8. Actual and estimated MSE as a function of the window size in different clusters for the house image with additive white Gaussian noise of standard deviation 15. The corresponding clusters are shown above.

$$\begin{aligned} \mathbf{g}_i^T \mathbf{y}_i &= \hat{z}_i - y_i = \mathbf{c}^T \hat{\mathbf{z}}_i - \mathbf{c}^T \mathbf{y}_i \\ &= \mathbf{c}^T \left(\left(\bar{\mathbf{y}}^{(k)} + \hat{\mathbf{\Phi}}^{(k)} \hat{\boldsymbol{\beta}}_i \right) - \left(\bar{\mathbf{y}}^{(k)} + \tilde{\mathbf{y}}_i^{(k)} \right) \right) \\ &= \mathbf{c}^T \left(\hat{\mathbf{\Phi}}^{(k)} \left(\hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i \hat{\mathbf{\Phi}}^{(k)} \right)^{-1} \right. \\ &\quad \left. \times \hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i \tilde{\mathbf{y}}_i^{(k)} - \tilde{\mathbf{y}}_i^{(k)} \right) \\ &= \mathbf{c}^T \left(\hat{\mathbf{\Phi}}^{(k)} \left(\hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i \hat{\mathbf{\Phi}}^{(k)} \right)^{-1} \hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i - \mathbf{I} \right) \\ &\quad \times \left(\mathbf{y}_i - \bar{\mathbf{y}}^{(k)} \right). \end{aligned} \tag{25}$$

Treating $\bar{\mathbf{y}}^{(k)}$ as a constant for each cluster, we calculate the gradient with respect to the ith pixel

$$\begin{aligned} &\left(\nabla \cdot \mathbf{g}_i^T\right)(\mathbf{y}_i) \\ &= \frac{\partial \mathbf{g}_i^T \mathbf{y}_i}{\partial y_i} \\ &= \mathbf{c}^T \left(\hat{\mathbf{\Phi}}^{(k)} \left(\hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i \hat{\mathbf{\Phi}}^{(k)} \right)^{-1} \hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i - \mathbf{I} \right) \mathbf{c} \\ &= \mathbf{c}^T (\mathbf{A}_i - \mathbf{I}) \mathbf{c} = \mathbf{c}^T \mathbf{A}_i \mathbf{c} - 1 \end{aligned} \tag{26}$$

where $\mathbf{A}_i = \hat{\mathbf{\Phi}}^{(k)} (\hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i \hat{\mathbf{\Phi}}^{(k)})^{-1} \hat{\mathbf{\Phi}}^{(k)T} \mathbf{W}_i$. In essence, pre and post multiplication with the vector $\mathbf{c}$ reduces the first part of the above expression to simply selecting the central element of the matrix $\mathbf{A}_i$. For each cluster, the estimated MSE from (24) then takes the form

$$\begin{aligned} \widehat{\mathrm{MSE}}^{(k)} &= \sigma^2 + \frac{1}{|\Omega_k|} \sum_{i \in \Omega_k} [(\hat{z}_i - y_i)^2 \\ &\quad + 2\sigma^2 (\mathbf{c}^T \mathbf{A}_i \mathbf{c} - 1)] \\ &= \frac{1}{|\Omega_k|} \sum_{i \in \Omega_k} [(\hat{z}_i - y_i)^2 + 2\sigma^2 (\mathbf{c}^T \mathbf{A}_i \mathbf{c})] - \sigma^2. \end{aligned} \tag{27}$$

B. Choosing the Optimal Kernel Support Size

Once we have a way of estimating the MSE for each cluster (assuming that σ is either known *a priori* or can be estimated [4], [36]), we can select an optimal kernel size for each cluster which leads to the best reconstruction (in terms of the MSE).

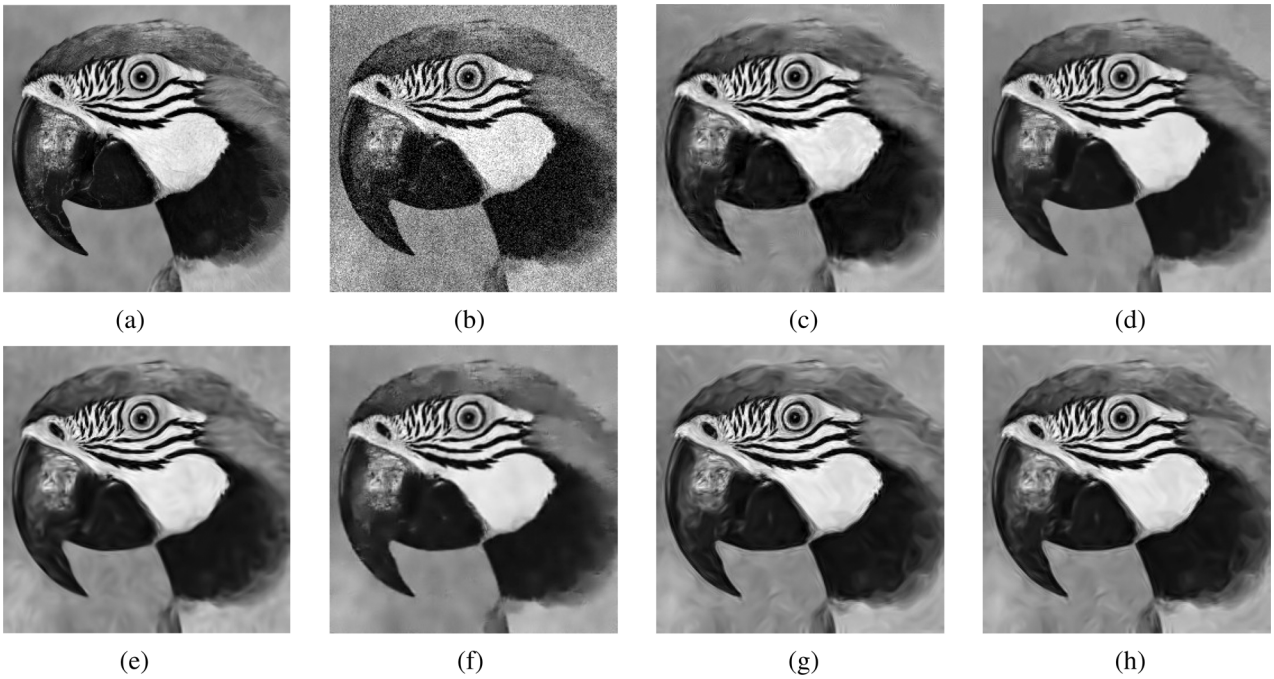


Fig. 9. Comparison of denoising results on noisy parrot image corrupted by additive white Gaussian noise of standard deviation 25. (a) Original image, (b) noisy image, (c) GSM [9] (MSE 93.87), (d) BM3D [7] (MSE 88.82), (e) iterative SKR [1] (MSE 99.96), (f) K-SVD [5] (MSE 101.54), (g) K-LLD (fixed kernel support size) (MSE 96.95), and (h) K-LLD (adaptive kernel support size) (MSE 94.57).

The algorithm then works by finding not just the most descriptive dictionary but also doing so at an optimal window size, both of which are dictated by the structure of the underlying image data. This leads to an optimal reconstruction (in terms of MSE) in each cluster. In this spatially adaptive method, the steering kernels are initially formed using a very small window size (e.g., 5×5). The initial kernel size is chosen taking into account the fact that too small a kernel size might result in failure to capture the underlying image structure. These initial steering kernels are used to segment the image into regions of similar geometric structure. A small kernel size (and, hence, smaller dimension of the feature) here also results in a relatively quicker clustering. Once the image is segmented we begin the process of finding the optimal window size for each cluster. For this, we start with the smallest window size and calculate the optimal dictionary at that particular size and estimate the MSE of the restored cluster. This process is then performed with a growing window size, terminating when the estimated MSE begins to increase. The optimal window size and the reconstruction which yields the best estimated MSE is thus chosen for each cluster. Fig. 8 shows the actual and estimated MSE of each of the clusters, with differing window sizes for the house image corrupted by additive Gaussian noise of standard deviation 15. It can be seen that the MSE changes with changing window sizes. We rollback the window size when the MSE begins to increase and, hence, an estimate of the MSE for all window sizes is not necessary. This spatially adaptive method introduces a feedback loop between the output of the *coefficient calculation* stage and the input of the *dictionary selection* stage.

While using a data adaptive window size allows us to better adapt to the underlying image structure, it also allows us to perform denoising without having to decide on the optimal patch size, thus reducing the number of parameters. Moreover, this adaptive framework allows us to decrease the time spent in the clustering stage. We can now perform the clustering based on features (normalized steering kernels) obtained with the smallest window size. This reduces the clustering time by an order of magnitude since K-Means performs clustering in $O(\text{NPK})$, where N is the size of each feature vector, P is the total number of features and K is the number of clusters.

VII. Stopping Criterion

Once our iterative denoising framework is in place, we need to determine a stopping criterion for the iterations. As illustrated in Fig. 7, the MSE of the denoised output reaches a minimum and then starts increasing with further iterations. We thus need a stopping rule which will allow us to determine the best result (in terms of MSE) and discontinue further iterations. To identify when the best result has been achieved, we make use of the fact that our framework already estimates the squared error for each pixel location in order to estimate the MSE for each cluster. One way to define a stopping criterion then is to estimate the MSE using the SURE framework for the entire image and stop iterating when the estimated MSE starts increasing. However, we can take further advantage of our framework by determining a stopping criterion for each cluster instead. We make use of the estimated MSE for each cluster and stop processing a cluster when the estimated MSE for that particular cluster increases. But this method cannot be used directly since there may be pixels whose cluster memberships change over iterations. Hence, we need to recalculate the estimated MSE from the previous iteration for each of the clusters formed in the current iteration. To do this we need to keep track of the pointwise error

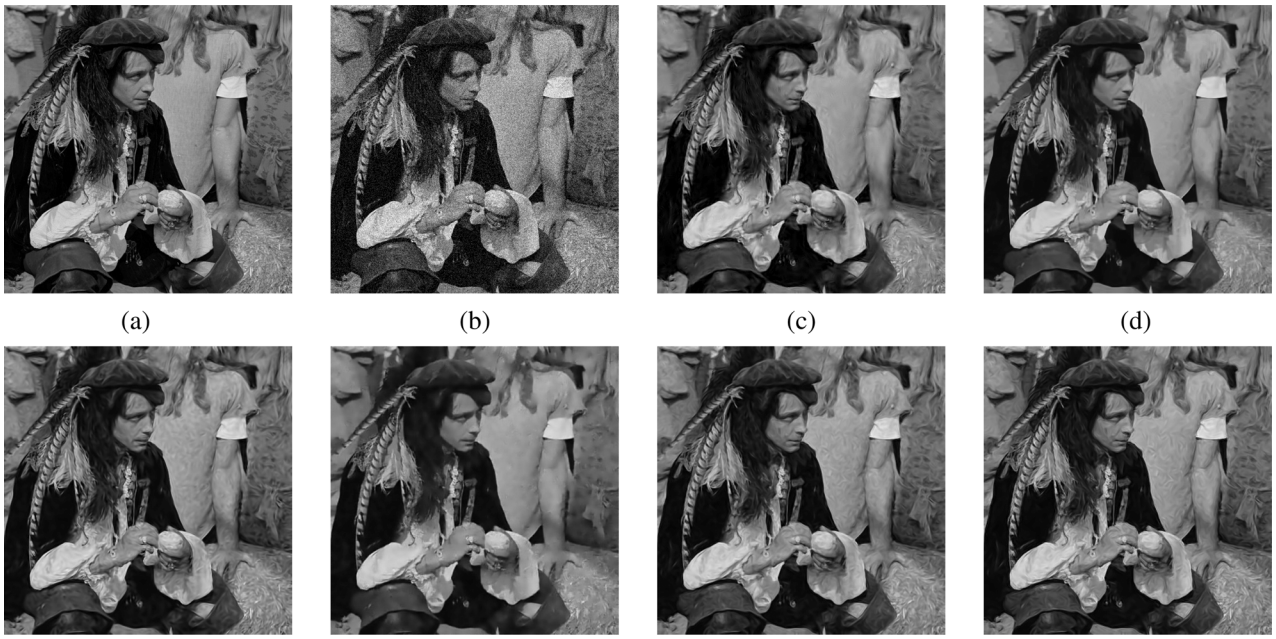


Fig. 10. Comparison of denoising results on noisy man image corrupted by additive white Gaussian noise of standard deviation 25. (a) Original image, (b) noisy image, (c) GSM [9] (MSE 93.12), (d) BM3D [7] (MSE 96.82), (e) iterative SKR [1] (MSE 104.75), (f) K-SVD [5] (MSE 113.64), (g) K-LLD (fixed kernel support size) (MSE 101.46), and (h) K-LLD (adaptive kernel support size) (MSE 99.98).

estimates from the previous estimate as we go into the next iteration. Once the image is segmented into clusters in a particular iteration, we recalculate the estimated MSE from the previous iteration for each cluster by taking the mean of the estimated squared errors of each cluster member. Once the cluster is denoised in the current iteration, we can obtain an estimate of the cluster MSE for the updated denoised result. The MSE of the updated result is then compared to that obtained from the previous iteration and the current denoising result is rejected if the estimated MSE is found to increase. This forms the stopping criterion for each cluster. Such a mechanism allows us to continue iterating over those clusters where an improvement in MSE might still be observed. Once all the clusters have reached their respective minimum estimated MSE, the iterations are stopped and the final denoised image is obtained. This stopping criterion proves to be more effective than a simple stopping rule based on the estimated MSE for the entire image since we allow the MSE to be minimized per cluster.

VIII. Results

To validate our method, we performed various experiments.[6] We artificially added zero mean white Gaussian noise of different standard deviations to produce noisy images. We compare the results obtained by two versions of our K-LLD method, in the case where the kernel support size is fixed and the case where the support size is allowed to vary across clusters depending on the underlying data geometry. For the fixed window size version, we used a patch size typically between 11×11 and 15×15 for our experiments, depending on the amount of noise present in the image. The parameters that can be tuned for our method are the number of clusters (K) for the clustering stage and the threshold on the condition number of the inverting term of (21). For the parrot image shown in Fig. 9, the method was found to give the best results when the image was divided into 10 classes. The man and house images of Figs. 10 and 11 were found to produce the best results using 5 clusters. Although K is a tunable parameter, our experiments with different images across various noise variances have revealed that for most images, denoising performance close to the least MSE can be achieved using anything between 5 to 10 clusters.[7] This, therefore, does away with the necessity of tuning K for most images. However, for illustrative purposes, in this paper we show results using a value of K that allowed us to achieve the least MSE. Moreover, to eliminate dependence on the random initialization of cluster centers for the K-Means algorithm, we perform clustering using K-Means multiple (typically 3) times and use the best clustering that produces the least cost from (12). This is done *only* for the first iteration of our algorithm. To reduce the running time, the cluster centers from an iteration are used as initialization for the clustering stage of the next iteration. For the dictionary selection stage, the parameter that needs to be specified is the constant of proportionality r from (18) that controls the number of atoms in the dictionary. The best results were obtained when r was fixed to be 2.5 for all images, across all noise variances. Apart from this, the bandwidth or the smoothing parameter for the steering kernel also needed to be tuned for optimality. This parameter was tuned heuristically to produce the best results in terms of MSE.

[6]Further experimental comparisons can be viewed at http://www.soe.ucsc.edu/~priyam/K-LLD/

[7]A graph showing the MSE obtained using different numbers of clusters on different images can be found at http://www.soe.ucsc.edu/~priyam/K-LLD.

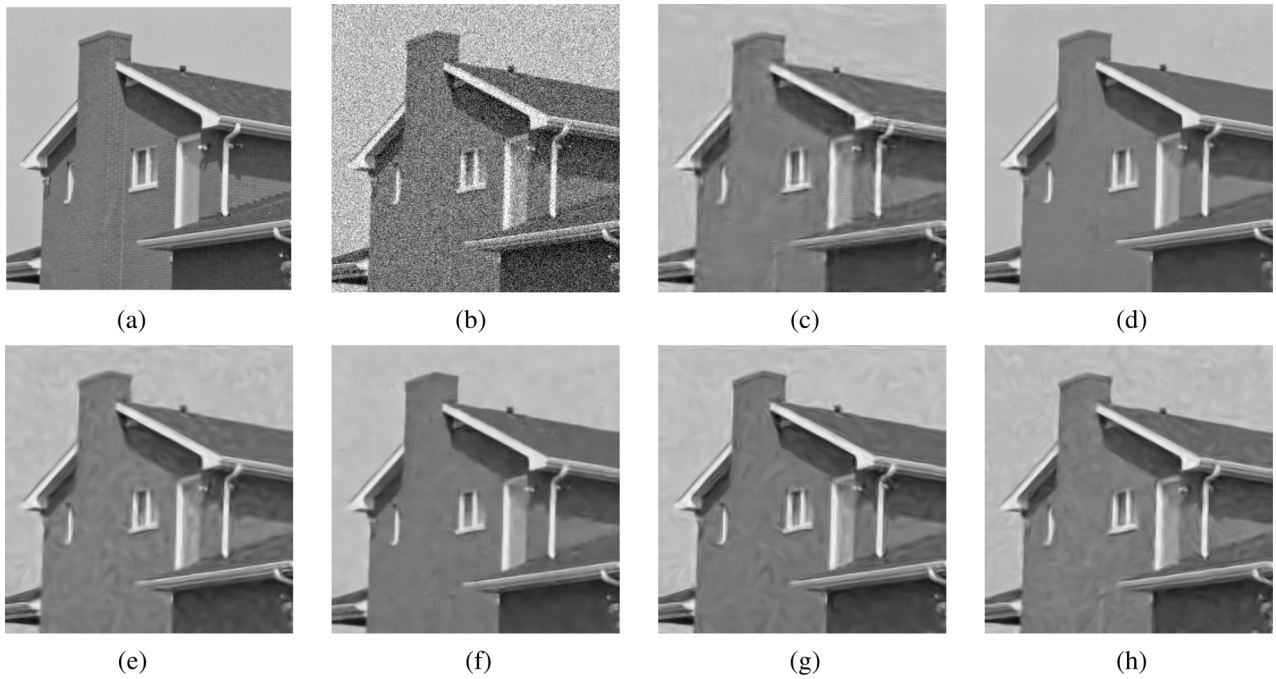


Fig. 11. Comparison of denoising results on noisy house image corrupted by additive white Gaussian noise of standard deviation 25. (a) Original image, (b) noisy image, (c) GSM [9] (MSE 49.16), (d) BM3D [7] (MSE 33.64), (e) iterative SKR [1] (MSE 47.69), (f) K-SVD [5] (MSE 41.09), (g) K-LLD (fixed kernel support size) (MSE 46.47), and (h) K-LLD (adaptive kernel support size) (MSE 45.18).

Apart from the smoothing parameter for the kernel calculation step, all other parameters were kept unchanged for the adaptive window size version of our method. For this method, we start off with an initial window size of 5×5 to calculate the steering weights and perform clustering. For each cluster, the window size is then increased gradually (by 2 pixels in both horizontal and vertical directions to 7×7, 9×9, etc.) and a new dictionary is learned. We bound the window size growth to a maximum of 21×21. Finally, an estimate of the MSE is obtained for the restored pixels in the cluster. We then retain the estimates that are obtained with the kernel support size at which the least estimated MSE is obtained. The results obtained for the parrot, man and house images corrupted by additive white Gaussian noise are shown in Figs. 9–11. Notice how the adaptive window size is able to restore more structure in the images, especially the cheek region for the parrot and hair and other texture regions for the man image. An improvement in terms of MSE is also achieved. This can be expected since we are locally minimizing the MSE for each cluster. In each of these experiments the noise variance was assumed to be known and this information was used to estimate the optimal kernel size using the SURE framework.

The time required by our method to denoise an image depends on the number of clusters (K) and the noise variance since stronger noise typically requires more iterations, as can be expected. Denoising the 256×256 pixels of the parrot image corrupted by additive white Gaussian noise of standard deviation 25 using 10 clusters requires 4 iterations. Our nonoptimized MATLAB implementation takes approximately 692 s. The house image of the same dimensions takes 540 s to denoise using 5 clusters. The clustering stage for our method accounts for approximately 8.5% and 6.6% of the time required to denoise the parrot and the house images respectively. The dictionary learning part using PCA proves to be the most time consuming stage with approximately 30% and 44% of the time being taken for the two images respectively. This can be expected since the dictionary needs to be learned at different scales for our method. The kernel computation takes up approximately 24% of the time in each of the images. While our method can take a relatively long time to run for large images, it should be noted that our method can be easily parallelized to take advantage of modern processors with multiple cores. This can be done because the last two stages of our algorithm, namely dictionary learning and coefficient calculation, are performed independently for each cluster.

We compare our denoising results to some recently proposed spatial and transform domain denoising methods, namely GSM [9], ISKR [1], K-SVD [5], and BM3D [7]. To be fair, we tune the required parameters to produce the best results in terms of MSE (or equivalently PSNR) for each of these methods. A comparison of these methods using five independent realizations of Gaussian white noise (for each choice of standard deviation) are tabulated in Table I where we present the average PSNR of the denoised outputs obtained from various methods. It can be seen that our method compares well or even betters, in some cases, the denoising result of the *spatial domain* methods (namely ISKR [1] and K-SVD [5]) in terms of PSNR, especially in higher noise cases. We note that the transform domain method of BM3D [7] consistently outperforms all the other methods in terms of PSNR. However, qualitatively, our method is capable of better restoration in the texture regions. This can be seen in the hair and the texture of the seat in Fig. 10. Further, we note that the patch-based method of K-SVD performs better denoising compared to our method for images that lack a lot of texture (e.g., the house image). However, in high noise cases, our method appears to perform better restoration of texture.

TABLE I
DENOISING PERFORMANCE OF SOME RECENT METHODS [1], [5], [7], [9] COMPARED TO OUR DATA ADAPTIVE WINDOW SIZE METHOD BASED ON RESULTS OBTAINED ON DIFFERENT IMAGES WITH ADDITIVE WHITE GAUSSIAN NOISE OF DIFFERENT STANDARD DEVIATIONS (σ). THE RESULTS NOTED ARE THE AVERAGE PSNR OBTAINED BY THE METHODS OVER FIVE INDEPENDENT NOISE REALIZATIONS FOR EACH σ

Image	σ	GSM	K-SVD	BM3D	ISKR	K-LLD
Parrot	5	37.49	37.56	37.92	37.09	37.28
	15	30.99	30.97	31.38	30.75	30.91
	25	28.44	28.12	28.74	28.20	28.40
Man	5	37.05	37.05	37.28	36.88	36.83
	15	30.95	30.46	30.98	30.61	30.65
	25	28.47	27.59	28.29	27.95	28.17
House	5	38.66	39.33	39.81	37.80	37.96
	15	33.60	34.30	34.94	33.53	33.81
	25	31.34	32.12	32.87	31.36	31.77

IX. CONCLUSION

In this paper, we presented a general framework for image denoising which works by learning a geometric descriptor using local kernels. The resulting approach falls in a class between methods that can be categorized as kernel regression based [1], [2], [4], [8] and those that aim to learn the best *global* dictionary [5], [25]. We go about our approach by clustering the image using meaningful features that are able to capture the underlying geometry in the presence of noise. A dictionary is learned for each of the clusters and a generalized kernel regression is performed to produce a denoised estimate for each pixel. Further degrees of freedom are introduced into the method by considering a varying kernel support size which is automatically learned from the image structure in each cluster. In this paper we present a particular way of carrying out each of the three stages of our method, namely *clustering*, *dictionary learning*, and *coefficient calculation*. However, each of these blocks can be replaced by alternate approaches that satisfy similar objectives. Our framework is evaluated experimentally and compared to some of the state of the art methods for denoising. It can be seen that the performance of our approach to denoising is competitive, qualitatively as well as quantitatively.

For optimal performance, it is necessary to tune a few parameters of our framework. This is indeed undesirable and our ongoing research aims at addressing this issue. Although our method is not very sensitive to the number of clusters when it lies within a particular range, it may be useful to use variants of K-Means that converge to the optimal number of clusters automatically [44]. Use of other unsupervised clustering algorithms like the mean shift method [45] can be considered as well. Another important feature of the clustering stage is the selection of an informative distance metric. While all these factors influence the output of our method, we believe that the present work provides a generally interesting and intuitively appealing framework in which such relevant questions can be addressed in the future.

REFERENCES

[1] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 349–366, Feb. 2007.

[2] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Washington, DC, Oct. 2005, vol. 2, pp. 60–65.

[3] A. Buades, B. Coll, and J.-M. Morel, "The staircasing effect in neighborhood filters and its solution," *IEEE Trans. Image Process.*, vol. 15, no. 7, pp. 1499–1505, Jul. 2006.

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

[5] 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.

[6] 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.

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

[8] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Computer Vision*, Washington, DC, Jan. 1998, pp. 839–846.

[9] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using a scale mixture of Gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, Nov. 2003.

[10] C. Kervrann and J. Boulanger, "Local adaptivity to variable smoothness for exemplar-based image denoising and representation," *Int. J. Comput. Vis.*, vol. 79, no. 1, pp. 45–69, Aug. 2008.

[11] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3D transform-domain collaborative filtering," in *Proc. 15th Eur. Signal Processing Conf.*, Poznan, Poland, Sep. 2007, pp. 145–149.

[12] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[13] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *SIAM Multiscale Model. Simul.*, vol. 7, no. 1, pp. 214–241, Apr. 2008.

[14] H. Takeda, S. Farsiu, and P. Milanfar, "Higher order bilateral filters and their properties," in *Proc. SPIE Conf. Computational Imaging V*, Feb. 2007, vol. 6498, p. 64980S.

[15] P. Chatterjee and P. Milanfar, "A generalization of non-local means via kernel regression," in *Proc. IS&T/SPIE Conf. Computational Imaging VI*, San Jose, CA, Jan. 2008, vol. 6814, p. 68140P.

[16] H. J. Seo and P. Milanfar, "Video denoising using higher order optimal space-time adaptation," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Las Vegas, NV, Apr. 2008, pp. 1249–1252.

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

[18] G. Gilboa and S. Osher, "Nonlocal linear image regularization and supervised segmentation," *SIAM Multiscale Model. Simul.*, vol. 6, no. 2, pp. 595–630, Jul. 2007.

[19] D. Barash, "A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 844–847, Jun. 2002.

[20] D. Tschumperlé and R. Deriche, "Vector-valued image regularization with PDE's: A common framework for different applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 506–517, Apr. 2005.

[21] E. Candes and D. Donoho, "Curvelets—A surprisingly effective non-adaptive representation for objects with edges," in *Curves and Surfaces*, A. Cohen, C. Rabut, and L. Schumaker, Eds. Nashville, TN: Vanderbilt Univ. Press, 2000, pp. 105–120.

[22] M. N. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, Dec. 2005.

[23] E. L. Pennec and S. Mallat, "Sparse geometric image representation with bandelets," *IEEE Trans. Image Process.*, vol. 14, no. 4, pp. 423–438, Apr. 2005.

[24] H. Takeda, S. Farsiu, and P. Milanfar, "Deblurring using regularized locally-adaptive kernel regression," *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 550–563, Apr. 2008.

[25] L. Horesh and E. Haber, "Overcomplete dictionary design by empirical risk minimization," *Inv. Probl.,* submitted.

[26] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Comput.*, vol. 15, no. 2, pp. 349–396, 2003.
[27] J. F. Murray and K. Kreutz-Delgado, "Learning sparse overcomplete codes for images," *J. VLSI Signal Process.*, vol. 45, no. 1, pp. 97–110, Nov. 2006.
[28] W. Hong, J. Wright, K. Huang, and Y. Ma, "Multiscale hybrid linear models for lossy image representation," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3655–3671, Dec. 2006.
[29] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.
[30] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 839–842, Dec. 2005.
[31] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.
[32] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[33] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Secaucus, NJ: Springer-Verlag, 2007.
[34] G. H. Golub and V. Pereyra, "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate," *SIAM J. Numer. Anal.*, vol. 10, no. 2, pp. 413–432, 1973.
[35] G. Golub and V. Pereyra, "Separable non-linear least squares: The variable projection method and its applications," Dept. Comput. Sci., Stanford Univ., Tech. Rep. SCCM-02-07, Jul. 2002.
[36] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 299–314, Feb. 2008.
[37] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 1993, pp. 40–44.
[38] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *Ann. Statist.*, vol. 9, no. 6, pp. 1135–1151, Nov. 1981.
[39] V. Katkovnik, K. Egiazarian, and J. Astola, "Adaptive window size image denoising based on intersection of confidence intervals (ICI) rule," *J. Math. Imag. Vis.*, vol. 16, no. 3, pp. 223–235, May 2002.
[40] A. Benazza-Benyahia and J.-C. Pesquet, "Building robust wavelet estimators for multicomponent images using Stein's principle," *IEEE Trans. Image Process.*, vol. 14, no. 11, pp. 1814–1830, Nov. 2005.
[41] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *J. Amer. Statist. Assoc.*, vol. 90, no. 432, pp. 1200–1224, Dec. 1995.
[42] F. Luisier, T. Blu, and M. Unser, "A new SURE approach to image denoising: Interscale orthonormal wavelet thresholding," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 593–606, Mar. 2007.
[43] S. Ramani, T. Blu, and M. Unser, "Monte-Carlo SURE: A black- box optimization of regularization parameters for general denoising algorithms," *IEEE Trans. Image Process.*, vol. 17, no. 9, pp. 1540–1554, Sep. 2008.
[44] P. Bradley and U. Fayyad, "Refining initial points for K-means clustering," in *Proc. 15th Int. Conf. Machine Learning*, Jul. 1998, pp. 91–98.
[45] K. Fukunaaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975.

Priyam Chatterjee (S'07) received the B.Tech. degree in information technology from the University of Kalyani, India, and the M.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT), Bombay, in 2003 and 2006, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at the University of California, Santa Cruz.

His research interests are in image and video processing (denoising, interpolation, deblurring, and super-resolution).

Peyman Milanfar (SM'98) received the B.S. degree in electrical engineering/mathematics from the University of California, Berkeley, in 1988, and the M.S., E.E., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1990, 1992, and 1993, respectively.

Until 1999, he was a Senior Research Engineer at SRI International, Menlo Park, CA. He is currently a Professor of electrical engineering at the University of California, Santa Cruz. He was a Consulting Assistant Professor of computer science at Stanford University, Stanford, CA, from 1998–2000, and a visiting Associate Professor there in 2002. His technical interests are in statistical signal and image processing and inverse problems.

Dr. Milanfar won a National Science Foundation CAREER award in 2000. He is an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and was an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS from 1998 to 2001. He is a Senior member of the Signal Processing Society's Image, Video, and Multidimensional Signal Processing (IVMSP) Technical Committee.