Texture Variation Adaptive Image Denoising With Nonlocal PCA

Wenzhao Zhao, Qiegen Liu, *Member, IEEE*, Yisong Lv, and Binjie Qin, *Member, IEEE*

***Abstract*— Image textures, as a kind of local variations, provide important information for the human visual system. Many image textures, especially the small-scale or stochastic textures, are rich in high-frequency variations, and are difficult to be preserved. Current state-of-the-art denoising algorithms typically adopt a nonlocal approach consisting of image patch grouping and group-wise denoising filtering. To achieve a better image denoising while preserving the variations in texture, we first adaptively group high correlated image patches with the same kinds of texture elements (texels) via an adaptive clustering method. This adaptive clustering method is applied in an over-clustering-and-iterative-merging approach, where its noise robustness is improved with a custom merging threshold relating to the noise level and cluster size. For texture-preserving denoising of each cluster, considering that the variations in texture are captured and wrapped in not only the between-dimension energy variations but also the within-dimension variations of PCA transform coefficients, we further propose a PCA-transform-domain variation adaptive filtering method to preserve the local variations in textures. Experiments on natural images show the superiority of the proposed transform-domain variation adaptive filtering to traditional PCA-based hard or soft threshold filtering. As a whole, the proposed denoising method achieves a favorable texture-preserving performance both quantitatively and visually, especially for irregular textures, which is further verified in camera raw image denoising.**

***Index Terms*— Texture-preserving denoising, adaptive clustering, principal component analysis transform, suboptimal Wiener filter, LPA-ICI.**

I. Introduction

TEXTURE, as a systematic local variation of image values, is an essential component of natural visual information reflecting the physical properties of the surrounding environment [1]. There are two basic types of texture pattern: regular texture that consists of repeated texture elements (texels) and stochastic texture without explicit texels [2], [3]. Most of the real-world textures locate in-between these two extremes.

Preservation of texture variation is necessary for image preprocessing tasks such as image denoising [2], [4], so as to help make better use of natural feature details for image content interpretation. These feature-preserving image processing researches [2], [5]–[7] have attracted great attention in recent years. However, texture variation, especially the small-scale or stochastic texture variation often lies in high frequency bands. These high frequency variations are difficult to be preserved during noise removal and tend to be smoothed. The existing state-of-the-art denoising methods often adopt the nonlocal methodology [8]–[10], which firstly uses patch grouping (PG) techniques to exploit the nonlocal self-similarity (NSS) prior in natural images, and then uses denoising filters (DF) for group-wise denoising. Over-smoothness of the image textures is caused by the deficiencies in both PG and DF procedures.

PG techniques collect similar (high-correlated) patches together so that DF can exploit the NSS to boost the denoising performance. During the PG process, if dissimilar patches are gathered in the same patch group, it would be much more difficult for DF to preserve the texture variations. The most widely-used PG techniques are block matching and K-means clustering. Unfortunately, they perform poorly in gathering similar patches under noise interference due to their respective deficiencies.

(i) Block matching methods [8]–[12] are based on the computation of Euclidean distance between patch vectors, which are typically not robust to noise. Moreover, the size of patch groups in block matching is usually set manually so that some dissimilar patches may be grouped together.

(ii) K-means clustering algorithm divides the data points of a dataset into a fixed number of clusters such that a certain metric of the distance between clusters is minimized. On the one hand, the optimal cluster number cannot be determined easily [13]–[15]. On the other hand, applying K-means clustering to image patches can lead to heavy computational burden due to the high dimensionality of image data.

To overcome these problems, an efficient adaptive clustering method is designed in AC-PT [6], which not only determines the optimal cluster number automatically, but also accelerates the clustering without dimension reduction that can lead to the information loss. However, in case of high noise level, slight under-segmentation still can be observed.

Manuscript received April 14, 2018; revised January 3, 2019 and April 1, 2019; accepted May 5, 2019. Date of publication May 21, 2019; date of current version August 28, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61271320 and Grant 61871206, in part by the Young Scientist Training Plan of Jiangxi Province under Grant 20162BCB23019, in part by the Medical Engineering Cross Fund of Shanghai Jiao Tong University under Grant YG2014MS29, and in part by the Translational Medicine Cross Fund of Shanghai Jiao Tong University under Grant ZH2018ZDA19. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Khan M. Iftekharuddin. *(Corresponding author: Binjie Qin.)*

W. Zhao and B. Qin are with the School of Biomedical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: bjqin@sjtu.edu.cn).

Q. Liu is with the Department of Electronic Information Engineering, Nanchang University, Nanchang 330031, China.

Y. Lv is with the School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai 200240, China.

Digital Object Identifier 10.1109/TIP.2019.2916976

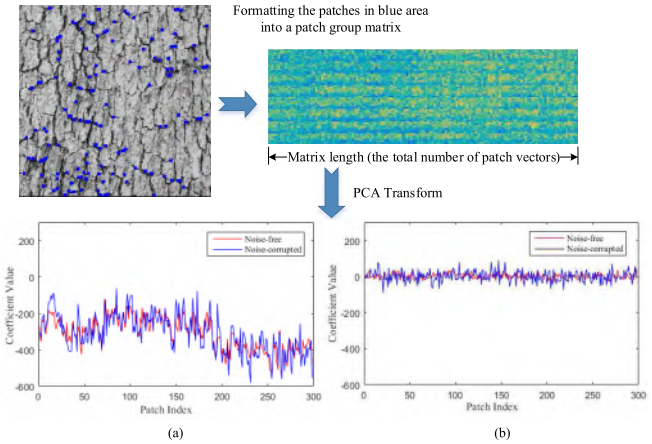

Fig. 1. The between- and within-dimension variations of PCA transform coefficients for a patch-group matrix consisting of similar patches. (a) The first dimension (signal-dominant) in PCA transform domain, (b) The last dimension (noise-dominant) in PCA transform domain. The difference between (a) and (b) shows the between-dimension variations, while the drastic fluctuation of coefficient value within (a) shows the within-dimension variations.

With similar patches collected by PG, it is important to find a suitable DF for the texture-preserving denoising of the patch groups. For the DF design, the state-of-the-art non-local methods usually incorporate NSS with transform domain methods to decorrelate the dimensions of patch vectors so that most of the variations among the highly correlated patches are preserved in some of the dimensions, with noise evenly distributed among all the dimensions, thus favors to improve denoising performance. Classical transform methods in DF usually use fixed bases such as discrete wavelet bases [16]–[19], discrete cosine transform bases [8], [11], [20]. One typical algorithm is BM3D [11] that uses fixed 3D transform to achieve an efficient denoising performance. However, fixed transform bases are not enough to represent the complex natural textures and often brings in artifacts.

Compared with fixed transform, adaptive transform, such as sparse representation and PCA, shows fewer artifacts. We see that the integration of NSS with sparse representation leads to excellent denoising performance, for example, [14] and [21]. However, stochastic texture variation that behaviors similarly to noise can not be represented sparsely. Thus the optimization based on sparsity prior can lead to the loss of stochastic texture information.

In the past decade, many PCA-based denoising methods have achieved state-of-the-art denoising performance. In the PCA transform domain, the energy of PCA coefficients among different dimensions corresponds to their respective eigenvalues and varies from each other; Meanwhile, within each dimension, especially the first few dimensions with the highest eigenvalues, both the noisy coefficients and their noise-free counterparts show a high variation, even though they come from the similar patches collected via PG. Fig. 1 shows an example of such variation in the first PCA dimension, where noisy and noise-free coefficients have a similar and severe up-and-down fluctuation (or variation). These drastic within-dimension variations of the transform coefficients come from the variations in natural image textures. Currently, many PCA-based denoising algorithms only consider the between-dimension energy variations and fail to recognize the within-dimension variations for the texture preservation [9], [12], [16], [22]. Specifically, the iterative and non-iterative singular value (eigenvalue) thresholding (SVT) algorithms specialize in shrinking the singular values (eigenvalues) of dimensions based on the low rank prior, while PCA transform domain filtering based methods such as [12] and [22] simply employ a "global" Wiener filter, where the filter parameters are estimated using a dimension-wise overall averaging. Recently, a novel detail-preserving denoising algorithm, AC-PT [6] is proposed to preserve these within-dimension variations via a combination of two thresholding operations: the hard thresholding of eigenvalues and the PCA-domain Wiener filtering with the filter parameters locally estimated, which are designed based on the consideration of between-dimension energy variations and within-dimension variations, respectively. The AC-PT method has been proven effective in denoising the optical coherence tomography (OCT) vessel images [23]. However, the simple Wiener filter adopted in AC-PT may cause over-smoothing and loss of texture detail as demonstrated in [24], and the fixed window width with which filter parameters are locally estimated also makes a not robust denoising performance.

In this work, an image denoising algorithm that preserves textures is proposed via PCA-transform-domain texture Variation Adaptive filtering for Adaptive Clustered patches (ACVA). We overcome the deficiencies of abovementioned algorithms and achieve a better denoising performance both quantitatively and visually. Generally, the contributions of this paper are embodied in four aspects:

- For PG, we improved the robustness of adaptive clustering method to high noise level by using a custom merging threshold (for iterative merging) that is a function of both noise level and cluster size instead of noise level only.
- For DF, considering the texture variations wrapped in the PCA coefficients of each signal-dominant dimension and the characteristic of over-smoothing of Wiener filter, we perform an improved denoising of the signal-dominant dimensions using the coefficient-wise suboptimal Wiener filter with the filter parameters tracking texture variations adaptively. Specifically, the improvement is mainly reflected in two aspects: a) The window width used for local parameter estimation is not fixed as done in AC-PT [6] but adaptively calculated by local polynomial approximation-intersection of confidence intervals (LPA-ICI) technique [25]; b) Instead of using the traditional Wiener filter, here for the first time, we propose to further preserve the texture variations in transform domain by applying the suboptimal Wiener filter [24] that is originally used for multichannel speech noise reduction.
- Another difference from AC-PT [6] is that to avoid the significant increase of computational burden of adaptive clustering along with the image size, we adopt a sliding-window-and-aggregation approach with fixed window size for better denoising performance.

- Besides of additive Gaussian noise reduction, the proposed denoising method is applied to remove Poisson-Gaussian noise in the camera raw image.

The rest of the paper is organized as follows. In Section II we introduce the noise model. Section III, IV and V are about the details of the adaptive patch clustering, texture variation adaptive filtering for PCA coefficients and the sliding window and aggregation technique, respectively. Experimental results are displayed in Section VI. Finally, conclusion is given in Section VII.

II. Noise Model

The additive white Gaussian noise (AWGN) is written as:

$$y = x + n, \tag{1}$$

where x is noise-free data, y is noisy, and n follows the normal distribution with zero mean and variance σ^2. AWGN is signal-independent.

Being different from AWGN, the Poisson-Gaussian noise corrupting the camera raw images that are acquired from digital cameras is typically signal-dependent noise. Let x be a noise-free signal at the position c. The observed data with Poisson-Gaussian noise can be written as:

$$y(\mathbf{c}) = \rho/\alpha + b\upsilon, \tag{2}$$

where $\rho \sim P(\alpha(x(\mathbf{c}) - p))$ is a Poisson variable with the parameter $\alpha(x(\mathbf{c}) - p)$, υ follows the normal distribution $N(0, 1)$, and α, b, p are parameters of the Poisson-Gaussian noise.

After applying a variance stabilization transform for the signal-dependent Poisson-Gaussian noisy signal, we can remove the noise using the denoising methods for additive white Gaussian noise. One well-known variance stabilization transform is called generalized Anscome transform (GAT) [26], [27]. GAT can approximately transform Poisson-Gaussian noise into additive white Gaussian noise with unitary variance:

$$f(y) = \begin{cases} 2\sqrt{y' + \frac{3}{8} + \sigma'^2}, & y' > -\frac{3}{8} - \sigma'^2 \\ 0, & y' \le -\frac{3}{8} - \sigma'^2 \end{cases} \tag{3}$$

where $y' = \alpha y$ and $\sigma' = \alpha b$.

Let x be the noise-free data, and the denoised data is treated as $E[f(y)|x]$. The exact unbiased inverse of the GAT is defined as:

$$T^{(IGAT)} : E[f(y)|x] \longmapsto E[y|x], \tag{4}$$

where $E[y|x] = x$, $E[f(y)|x] = 2\sum_{y=0}^{+\infty}\left(\sqrt{y + \frac{3}{8}} \cdot \frac{x^y e^{-x}}{y!}\right)$.

A better camera raw image denoising with GAT comes to require a better Gaussian denoising algorithm. Before we detail the proposed denoising algorithms, we want to make clear the symbol system. Generally, we use lower (upper) case bold-face letters to stand for column vectors (matrices). Denote by $\mathbf{X} = (\mathbf{x}) \in \mathbb{R}^{M\times L}$ a $M \times L$ matrix with column vectors $\mathbf{x}_i$, $1 \le i \le L$. Superscript ^{T} represents transpose of a vector or a matrix.

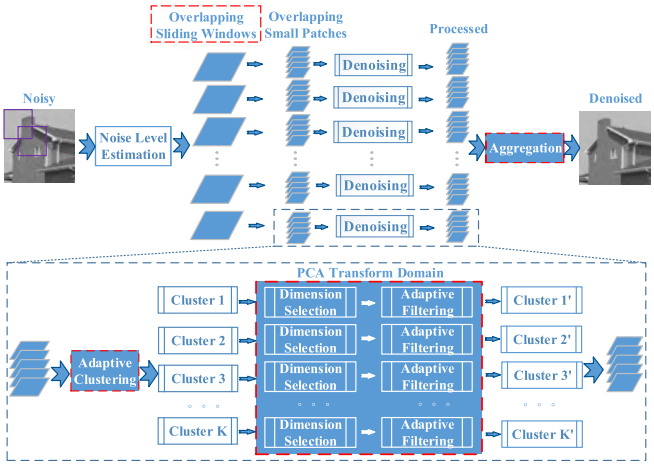

Fig. 2. Flowchart for the proposed algorithm.

Given an image $\Phi \in \mathbb{R}^{s\times t}$, the total number of all the possible $d \times d$ overlapping patches $\mathbf{P}_i \in \mathbb{R}^{d\times d}$ is $L = (s - d + 1) \times (t - d + 1)$ with $i \in \{1, 2, \cdots, L\}$. The observation vector $\mathbf{y}_i \in \mathbb{R}^{M\times 1}$ with $M = d^2$ is constructed by stretching the patch $\mathbf{P}_i$. So the image Φ can be represented as $\mathbf{Y}_\Phi \in \mathbb{R}^{M\times L}$ with each column being a stretched patch. A certain cluster can be represented with a matrix with each column representing a stretched patch. For any data matrix $\mathbf{Y}$, we add the subscript $_c$ to denote the centralized matrix $\mathbf{Y}_c = \mathbf{Y} - E(\mathbf{Y})$, where $E(\cdot)$ represents the expectation.

The proposed denoising method is depicted in Fig. 2. In the noise level estimation step, the noise level can be estimated as in [6] for Gaussian denoising. For Poisson-Gaussian denoising, we can estimate the noise parameters as in [26] and then transform Poisson-Gaussian noise into additive white Gaussian noise with unitary variance. In the following part of this paper, we focus on illustrating the novel part of the proposed denoising algorithm (in the dashed red boxes): (1) improved adaptive patch clustering for PG is discussed in Section III; (2) variation-adaptive filtering in PCA transform domain for DF is studied in Section IV; (3) sliding window and aggregation technique are discussed in Section V.

III. AC-Step: Adaptive Clustering of Patches

Many popular clustering algorithms have a common deficiency that an optimal number of clusters is difficult to be determined. However, it is easy for us to estimate an approximate range of the cluster number. Supposing the patch size is $d \times d$ and the image size is $s \times t$, in most cases, the maximal cluster number should be below $\frac{st}{d^2}$. Assuming each pixel to be the center of an image patch, we can obtain the maximal cluster number by separating the image into small non-overlapping segments, and each small segment with area approximately equal to d^2 represents a distinct cluster. Meanwhile, the minimal cluster number is 1.

Since we have the approximate range of cluster number (*i.e.*, from 1 to $\frac{st}{d^2}$), an intuitive idea is that we can determine the optimal number of clusters by first obtaining the maximal number of clusters, and then iteratively merging the similar

clusters according to a custom threshold. To this end, there are two problems that need to be solved:

a) clustering a huge number of clusters requires a huge computational burden due to the high dimensionality of image patches;
b) finding a way to calculate a suitable merging threshold for merging similar clusters.

For the first problem, we adopt the divide and conquer technique [28], [29]. The divide and conquer technique is a two-stage clustering scheme, which accelerates the K-means clustering with improved performance: It first clusters a small number of clusters using K-means, and then within each cluster it performs the K-means clustering again to further increase the cluster number.

For the second problem, we derive the merging threshold on the distance of any two similar clusters according to the noise level and cluster size. Specifically, we consider one special case, where we have two similar clusters $\mathbf{A} \in \mathbb{R}^{M\times L_a}$ and $\mathbf{B} \in \mathbb{R}^{M\times L_b}$ with very different sizes $L_a \gg 1$ and $L_b = 1$. Supposing the noise variance in the center of the large cluster $\mathbf{A}$ is small enough to be ignored, we further denote by $\mathbf{y}_a = \mathbf{x}$ and $\mathbf{y}_b = \mathbf{x} + \mathbf{n}$ the centers of $\mathbf{A}$ and $\mathbf{B}$ respectively, where $\mathbf{x}$ is noise-free, and the entries $n_i \sim N(0, 1), 1 \leq i \leq M$ of vector $\mathbf{n}$ are independent and identically distributed (i. i. d.). The between-cluster distance is

$$D(\mathbf{B}, \mathbf{A})^2 = \|\mathbf{y}_b - \mathbf{y}_a\|_2^2 = \|\mathbf{n}\|_2^2 = \sum_{i=1}^{M} n_i^2, \tag{5}$$

and $D(\mathbf{B}, \mathbf{A})^2$ follows the chi-squared distribution with M degrees of freedom. These two clusters obtained from K-means with such huge discrepancies in size usually have a very low probability of belonging to the same feature. Thus, the probability of merging the two clusters is $Prob(D(\mathbf{B}, \mathbf{A})^2 < \xi) = \varepsilon$, where ξ is the merging threshold and ε is a very small value. If we set $M = 64$ and $\varepsilon = 1.3 \times 10^{-10}$, we have $\xi \approx 16.0$. Furthermore, if $n_i \sim N(0, \sigma^2)$, $1 \leq i \leq M$, $\xi \approx 16.0\sigma^2$. When cluster $\mathbf{B}$ is enlarged to normal size, the noise variance in its center $\mathbf{y}_b$ will be much smaller. Therefore, the merging threshold ξ derived in that special case is essentially the largest acceptable dissimilarity between the two similar clusters that we want to merge together.

The merging threshold in AC-PT [6] is only related to noise level. However, the cluster size is also an important factor to be considered in the merging threshold design when a cluster center is calculated via an average of all the samples in the cluster and the noise variance in the cluster center decreases quickly as the cluster size increases. AC-PT only considers a special case in which a pair of clusters have huge difference in cluster size. However, it does not work well in the case of a pair of large clusters in which the noise's influences on both of these two large clusters' centers are small enough to be ignored. In this case, AC-PT shows a poor denoising performance at high noise levels where typically there exist more large clusters constructed via AC using an overlarge merging threshold. Therefore, setting a different merging threshold is necessary for the case of two large clusters (where the cluster sizes are above a certain value).

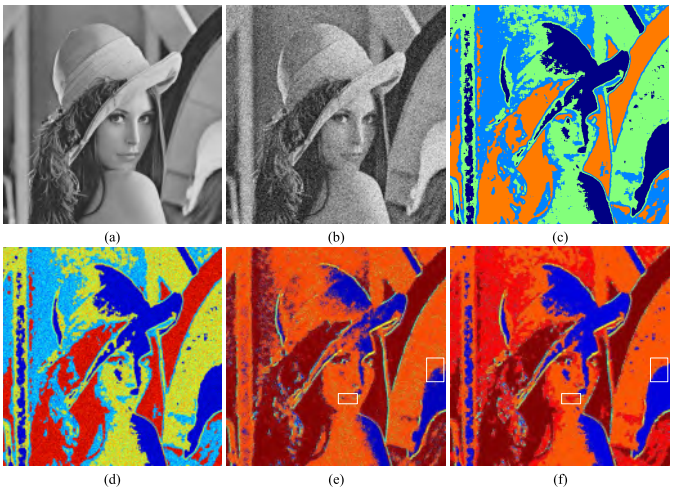

Fig. 3. The segmentation results on the noisy image with $\sigma = 50$ using K-means clustering and the adaptive clustering methods. (a) Lena; (b) Noisy image; (c) The clustering result of the first stage based on K-means; (d) Over-clustering based on divide-and-conquer technique; (e) Adaptive clustering as in AC-PT [6]; (f) Adaptive clustering considering cluster size.

Specifically, when the size of the smaller cluster in a pair of clusters is larger than a certain value L_T, we decrease the probability of merging the different clusters by amplifying the between-cluster distance with an amplification coefficient ρ: $\hat{D}(\mathbf{B}, \mathbf{A})^2 = D(\mathbf{B}, \mathbf{A})^2/\rho$. We empirically set $L_T = 200$ and $\rho = 0.7$ to get a satisfactory performance.

The comparison of different clustering methods displayed in Fig. 3 illustrates how the adaptive clustering methods segment the noisy images adaptively at high noise level $\sigma = 50$. The clustering result of the first stage based on K-means is somewhat under-segmented, while the segmentation by over-clustering is a typical over-segmentation. By comparison, the two results (Figs. 3(e)-(f)) by adaptive clustering methods are more favorable. Moreover, we see that the segmentation based on the improved adaptive clustering preserves better the edges of Lena's lip and the hat in the mirror in the white boxes.

In summary, we conclude the AC-step of the proposed ACVA method in Algorithm 1.

IV. VA-Step: Variation-Adaptive Filtering in PCA Domain

In PCA transform domain, inspired by the between-dimension energy variations and the within-dimension variations of PCA coefficients in the signal dominant dimensions with the highest eigenvalues (see Fig. 1), we use a two-step texture variation adaptive approximation strategy to achieve a texture-preserving denoising performance. First, a low rank approximation is implemented via dimension selection based on hard thresholding of eigenvalues to selectively preserve the energy variations of the signal dominant dimensions. Second, each signal dominant dimension is further denoised adaptively via a coefficient-wise adaptive filter with locally estimated filter parameters to protect the underlying within-dimension texture variations.

Algorithm 1 Adaptive Clustering Via Over-Clustering and Iterative Merging

Input: Gaussian noisy data $\Phi \in \mathbb{R}^{s\times t}$, noise level σ
Output: Cluster matrices

Patch extraction: partition the image into the collection of image patches Y with patch size $d = 8$
Initial clustering: clustering $K^1 = \max\{\frac{s\times t}{256\times 256}, 4\}$ clusters $\mathbf{Y}_k, 1 \le k \le K^1$ with K-means
for each cluster $\mathbf{Y}_k$ **do**
    re-clustering $K_k^2 = \max\{\frac{L_i}{d\times d}, 1\}$ clusters with K-means
end for
Collect all the clusters
while the minimum between-cluster distance is larger than threshold ξ **do**
    Compute between-cluster distance for all the possible cluster pairs and reorder the distance results in ascending order
    if there exists two clusters whose cluster sizes are larger than $L_T = 200$ **then**
        Amplify the distance with the coefficient ρ^{-1}
    end if
    Merge all the cluster pairs whose between-cluster-distance is below ξ
end while

A. Dimension Selection Considering the Between-Dimension Energy Variations

Since the texture information is hardly remained in the noise-dominant dimensions with the lowest eigenvalues, we discard the noise-dominant dimensions via dimension selection before the within-dimension filtering to reduce the computational cost and improve the denoising performance.

Considering the centralized noisy cluster matrix $\mathbf{Y}_c = (\mathbf{y}_c) \in \mathbb{R}^{M\times L}$, $\mathbf{Y}_c = \mathbf{X}_c + \mathbf{N}$, where $\mathbf{X}_c = (\mathbf{x}_c) \in \mathbb{R}^{M\times L}$ is noise-free and $\mathbf{N} = (\mathbf{n}) \in \mathbb{R}^{M\times L}$ is the noise matrix with each column vector $\mathbf{n}_i \sim N_M(0, \sigma^2\mathbf{I})$, where $\mathbf{I}$ is identity matrix. Suppose $\mathbf{Y}_c = \sqrt{L}\sum_{i=1}^{\min(M,L)} \sqrt{\lambda_i}\mathbf{u}_{y,i}\mathbf{v}_{y,i}^T$ and the low rank approximation (with rank R) $\mathbf{X}_c = \sqrt{L}\sum_{i=1}^{R} \sqrt{\lambda_{x,i}}\mathbf{u}_{x,i}\mathbf{v}_{x,i}^T$ with singular values $\sqrt{L\lambda_i}$ and $\sqrt{L\lambda_{x,i}}$, and singular vectors $\mathbf{u}_{y,i}$,$\mathbf{v}_{y,i}$, $\mathbf{u}_{x,i}$ and $\mathbf{v}_{x,i}$, $1 \le i \le \min(M, L)$.

The dimension selection can be done based on the Gaussian spiked population model [30], [31]. Suppose $\gamma = M/L$ is a constant. Letting $L \to \infty$ and $\lambda_{n\pm} = \sigma^2(1 \pm \sqrt{\gamma})^2$, there is:

$$\lim_{L\to\infty} \lambda_i = \begin{cases} \rho(\lambda_{x,i}), & if\ i \le R\ and\ \lambda_{x,i} > \sigma^2\gamma^{1/2}. \\ \lambda_{n+}, & otherwise. \end{cases} \quad (6)$$

where $\rho(\lambda) = \frac{(\sigma^2+\lambda)(\gamma\sigma^2+\lambda)}{\lambda}$ (for any $\lambda > 0$) is a real-valued function.

The Gaussian spiked population model implies that for the normal size cluster matrices the eigenvalues of noise-dominant dimensions are below λ_{n+} approximately. Since the dimensions with eigenvalues close to λ_{n+} are still noisy, we set a correction coefficient μ to estimate the rank R as:

$$R \approx \sum_{i=1}^{M} \mathbf{1}(\lambda_i > \mu\lambda_{n+}) \quad (7)$$

Thus the low rank approximation can be computed as:

$$\mathbf{Y}_R = \sqrt{L}\sum_{i=1}^{R} \sqrt{\lambda_i}\mathbf{u}_{y,i}\mathbf{v}_{y,i}^T. \quad (8)$$

According to [6], we set $\mu = 1.1$ to get a favorable denoising performance.

B. Within-Dimension Variation Adaptive Filtering

Consider the low-rank matrix $\mathbf{Y}_R = \mathbf{U}_R\mathbf{P}_R$ obtained in the previous section, where $\mathbf{U}_R$ consists of the selected eigenvectors $\mathbf{U}_R = [\mathbf{u}_{y,1}, \mathbf{u}_{y,2}, \cdots, \mathbf{u}_{y,R}]$, and $\mathbf{P}_R$ consists of the corresponding signal-dominant dimensions in PCA transform domain:

$$\mathbf{P}_R = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_R]^T, \quad (9)$$

where $\mathbf{p}_i = \sqrt{L\lambda_i}\mathbf{v}_{y,i}$ $(1 \le i \le R)$ is the selected PCA dimension.

For illustrative purpose, we further extract the coefficients in any dimension $\mathbf{p}_i = [p_{i,1}, p_{i,2}, \cdots, p_{i,L}]$ and denote these coefficients in the ith dimension as the noisy observations of a "signal sequence" containing L observation points: $y(n) = p_{i,n}$, $n = 1, 2, \cdots, L$. Let $y(n) = f(n) + w(n)$, where $w(n)$ is i.d.d Gaussian noise of zero mean and variance $\sigma_w^2 = \sigma^2$, and $f(n)$ (with variance σ_f^2) is the noise-free signal that we want to estimate. Thus for an observation at a point n, there is $y(n) \sim N(f(n), \sigma^2)$. Here, some specific observations are considered as being "similar" to each other when their respective means $f(n)$ are close to each other, and we say that there is a considerable "variation" between the observations when their respective means are different from each other.

Before we detail how to denoise $y(n)$ and obtain the estimate $\hat{f}(n)$ for each dimension, we must learn some useful characteristics on the extracted signal sequence $y(n)$ and its corresponding $f(n)$. One typical example of this kind of the signal sequence is displayed in Fig. 1(a), where the vertical axis corresponds to the signal value for the noisy signal $y(n)$ (blue line) and noise-free signal $f(n)$ (red line) and the horizonal axis is the sequence index n corresponding to different patches. We see that the signal values are aggregated densely in a certain interval, implying that their respective means are close to each other. This dense aggregation of all the means can be regarded as a global similarity between all the observations. Moreover, there is also a drastic and irregular fluctuation for both the noise-corrupted signal $y(n)$ and noise-free signal $f(n)$. This fluctuation of noise-free signal (*i.e.*, the sequence of the means) is called internal variation of the signal sequence. The global similarity and internal variation of the signal sequence in PCA transform domain can be interpreted in the corresponding spatial domain. Every transform-domain signal observation comes from a certain patch of specific patch group in spatial domain. All the pathes in the same patch group are similar to each other (*i.e.*, a global similarity for the whole patch group), while the patches within

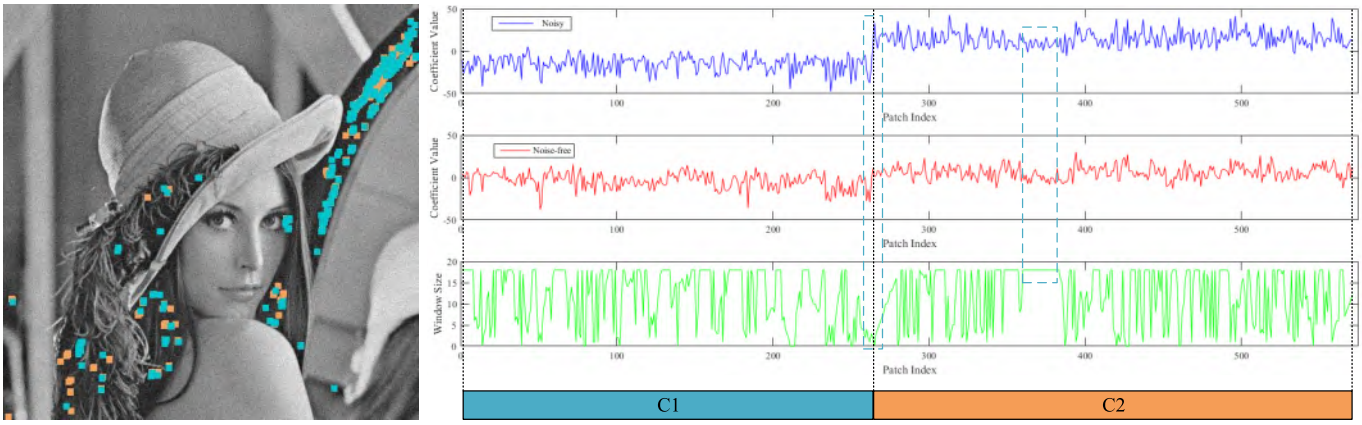

Fig. 4. Window-size determination based on the LPA-ICI method. A large patch cluster is built by the AC method from the noisy Lena image with $\sigma = 10$. This large cluster consists of 2 small clusters (C1 and C2) and the 2 clusters' corresponding patches are colored in the original noisy image. In the line charts, the blue and red lines represent the noisy and noise-free coefficients in the second principal component dimension respectively, while the green line shows the coefficients' corresponding window sizes computed via the LPA-ICI method.

the patch group are still different from each other to some extent (*i.e.*, an internal variation). Both the global similarity and the internal variations in spatial domain are transformed into the orthogonal PCA transform domain.

Considering the above characteristics in estimating each signal observation for PCA coefficient, we have three aspects of consideration: 1) The noise-free signal is rich in unsmooth and irregular variations that convey rich texture information. These unsmooth and irregular variations will be distorted by the denoising filters based on smoothness (continuity) constraints; 2) Based on the global similarity between the signal observations, we can find the sample observations with high similarity to the signal to be estimated and average them to get an optimal estimate for boosting denoising performance; 3) To preserve the internal variations within the observations, we must choose the sample observations adaptively according to the signal to be estimated, such that the highly similar observations around the signal to be estimated should be selected while other dissimilar observations need to be excluded.

For the consideration 1), we estimate $f(n)$ by applying an efficient Wiener filter that has been used for the favorable PCA domain filtering by many denoising algorithms, and we particularly use a suboptimal Wiener filtering to avoid signal distortion and preserve the variations. However, the satisfactory variation-preserving performance of suboptimal Wiener filter is highly dependent on an optimal estimate of its parameter, *i.e.*, the auto-covariance of $y(n)$: $R_y(n) = E(y(n)^2)$. Based on the consideration 2) and 3), we estimate the auto-covariance of $y(n)$ in a point-wise approach. For each signal its auto-covariance is estimated via an adaptive local average of similar observations. As a whole, this point-wise suboptimal Wiener filtering consists of two steps: locally adaptive estimation of filter parameter and suboptimal Wiener filtering using the estimated filter parameter.

1) Estimating the Filter Parameter Locally Based on LPA-ICI: To denoise the signal observation $y(n)$ at point n with Wiener filter, we need to estimate its filter parameter, *i.e.*, auto-covariance $R_y(n)$ to further compute the cross-covariance $R_{fy}(n) = R_y(n) - \sigma^2$ between $y(n)$ and its noise-free counterpart $f(n)$. The auto-covariance at point n can be written as: $R_y(n) = E(y(n)^2) = f(n)^2 + \sigma^2$, where the expected value can be approximated statistically via an average of several squared signal samples.

The remaining problem is how to accurately select the signal samples with high similarity to the signal (at point n) to be estimated. A simple practical solution is to select the observations in a local neighborhood of the signal point n because relatively higher similarity is more likely to happen in the local neighborhood due to the property of the applied PG technique. In particular, for AC, as shown in Fig. 4, a large cluster matrix consists of many small local segments that are actually small clusters matrices generated from the over-clustering stage in AC and typically have a higher similarity. Furthermore, the local neighborhood has been used for the parameter estimate of image filters [6], [32], [33] and proves to achieve a promising denoising performance.

Being different from [6] that chooses the local neighborhood in a fixed-size window for local estimate, the proposed method use LPA-ICI [25] to adaptively determine the window width for a better local estimate. Here, choosing the window width is essentially equivalent to choosing the estimate samples. A small width corresponds to a smaller moving window with fewer samples for the local parameter estimate and therefore to noisier estimate, with higher variance and typically decreased estimation bias, and vice versa. Therefore, the window width controls the trade-off between the bias and variance in the local estimate and the varying window width for the local estimate is very important. This assumption is confirmed by the fact that in density estimation [34] and signal reconstruction [35], [36] studies in the literature, almost all the adaptive-width windows [35] have been shown to be superior to fixed-width windows [36].

The local polynomial approximation (LPA) method, combined with intersection of confidence intervals (ICI) rule, is a method originally developed for pointwise adaptive estimation of 1-D signal (corrupted by Gaussian noise) [25]. In this work we only use it for the purpose of detecting variations and finding similar samples. The use of LPA-ICI for variation

detection has been used in some image denoising algorithms such as SADCT [20] and BM3DSAPCA [8], to adaptively detect the spatial variations of image value and collect similar pixel samples. However, the proposed algorithm uses LPA-ICI for signal variation detection in the PCA transform-domain.

Standard linear LPA tries to fit the signal $y(n)$ locally with polynomial functions of order m. Here, since we only use it to detect variations and find neighborhood with high internal similarity, we simply apply the zero-order polynomial fitting ($m = 0$) to find a suitable window of size h (a window containing $N_h = 2h + 1$ data points) where all the similar signal in the window can be approximated by a constant amplitude signal $\hat{y}(n, h) = C$. The computation of $\hat{y}(n, h)$ in LPA is related to the following loss function:

$$\Im_h(n) = \frac{1}{N_h} \sum_{s=1}^{N_h} \rho_h(n_s - n)(y(n_s) - \hat{y}(n, h))^2 \qquad (10)$$

where $y(n_s), 1 \le s \le N_h$ is the signal at the point in a window of size h with n as its center, $\rho(\cdot)$ is a basic window function, and $\rho_h(\cdot) = \rho(\cdot/h)/h$. For simplicity, we use the square uniform window, where $\rho(\cdot) = 1$ in $[-1, 1]$, and $\rho(\cdot) = 0$, otherwise. So there is $\rho_h(\cdot) = 1/h$ in $[-h, h]$, and $\rho(\cdot) = 0$, otherwise.

For a certain window size h, by minimizing the loss function, we have the estimate of $y(n)$: $\hat{y}(n, h) = \frac{1}{N_h} \sum_{s=1}^{N_h} y(n_s)$ and its standard deviation $std(n, h) = \frac{\sigma}{\sqrt{N_h}}$. So the confidence interval of the estimate can be

$$\begin{aligned} D &= [L, U] \\ U &= \hat{y}(n, h) + \Gamma \cdot std(n, h) \\ L &= \hat{y}(n, h) - \Gamma \cdot std(n, h) \end{aligned} \qquad (11)$$

where Γ is a threshold parameter.

Given a finite set of window size $H = h_1 < h_2 < \cdots < h_J$ starting from the minimum window size h_1, for each window we can use the LPA to get a estimate $\hat{y}(n, h_i)$ and a corresponding standard deviation $std(n, h_i)$, thereby determining a sequence of the confidence intervals $D(i)$, $1 \le i \le J$ of the biased estimates:

$$\begin{aligned} D(i) &= [L_i, U_i] \\ U_i &= \hat{y}(n, h_i) + \Gamma \cdot std(n, h_i) \\ L_i &= \hat{y}(n, h_i) - \Gamma \cdot std(n, h_i) \end{aligned} \qquad (12)$$

The ICI technique considers the optimal h to be the maximum window length satisfying $\underline{L}_i < \overline{U}_i$, where $\underline{L}_i = \max\{L_i, \underline{L}_{i-1}\}$ and $\overline{U}_i = \min\{U_i, \overline{U}_{i-1}\}$, $1 < i \le J$.

A simple illustration of the ICI rule is displayed in Fig. 5. We can see that as the window size increases from h_1 to h_4, the confidence intervals shrink and shift away because of the decrease of variance and the increase of bias, which leads to the shrinkage of the intersections between confidence intervals. According to the ICI rule, h_3 is the optimal window size that achieves a favorable balance between variance and bias.

Fig. 4 shows that LPA-ICI effectively detects the variations in the noisy signal, where LPA-ICI assigns adaptively a small window size to the drastically fluctuating segment and a large window size for the segment with small fluctuations (in the dashed boxes).

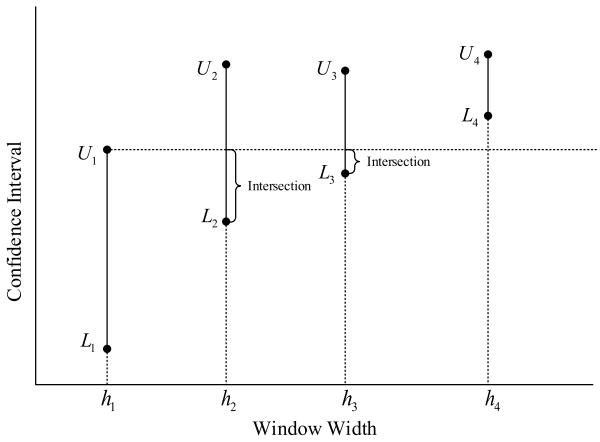

Fig. 5. The Intersection of Confidence Intervals (ICI) rule.

With the window size h and its center n being computed, the segment $y(n_s) = f(n_s) + w(n_s), 1 \le s \le N_h$ is available. The estimated auto-covariance of $y(n)$ at point n can be computed via a local average of this segment:

$$R_y(n) = \frac{1}{N_h} \sum_{s=1}^{N_h} y(n_s)^2. \qquad (13)$$

2) Suboptimal Wiener Filtering: Wiener filter has been widely used to remove the noise in transform domain efficiently [11], [12]. Given the estimated auto-covariance $R_y(n)$ at point n and auto-covariance of noise $R_w = \sigma^2$, the estimate of the observation $f(n)$ at n by Wiener filter is:

$$\hat{f}(n) = h_o y(n), \qquad (14)$$

where $h_o = R_y(n)^{-1} R_{fy}(n) = [1 - g_o]$, $g_o = R_y(n)^{-1} R_w$.

However, the optimal Wiener filter often results in signal distortion during noise reduction [24]. It is necessary for us to avoid the signal distortion as much as possible in reducing most of the noise for a satisfactory denoising performance. To achieve a better control of noise reduction and signal distortion, instead of using the optimal Wiener filter, we can use a suboptimal Wiener filter by manipulating the Wiener filter properly and automatically with an attenuation coefficient α of g_o as in [24]:

$$h_s = [1 - \alpha g_o], \qquad (15)$$

where $\alpha \in [0, 1]$ and $\alpha = 0$ and 1 correspond to the case of identity filter and Wiener filter, respectively. For simplicity, let $g_s = \alpha g_o$. Then the estimate of the observation $f(n)$ at n by the suboptimal Wiener filter is:

$$\hat{f}(n) = h_s y(n), \qquad (16)$$

The determination of α is based on the two indexes corresponding to the signal distortion and noise reduction, respectively. For the suboptimal Wiener filter corresponding to a certain α, the signal-distortion index can be defined as

$\upsilon_{sd}(g_s) \triangleq \frac{E\{[f(n)-h_s f(n)]^2\}}{\sigma_f^2}$, and the noise-reduction index is $\xi_{nr}(h_s) \triangleq \frac{\sigma_w^2}{E\{[h_s w(n)]^2\}}$.

Then, we can obtain the optimal α by maximizing the following discriminative cost function related to the noise-reduction and signal-distortion indexes [24],

$$J(\alpha) \triangleq \frac{\xi_{nr}(h_s)}{\xi_{nr}(h_o)} - \beta\frac{\upsilon_{sd}(g_s)}{\upsilon_{sd}(g_o)} = \frac{\sigma^2 + g_o R_w g_o - 2\sigma^2 g_o}{\sigma^2 + \alpha^2 g_o R_w g_o - 2\alpha\sigma^2 g_o} - \beta\alpha^2, \quad (17)$$

where β is an application-dependent constant and determines the relative importance between signal preservation and noise reduction. When β becomes larger, we have less signal distortion with less noise removal. We set $\beta = 0.7$ as in [24] to achieve a good balance.

With the suboptimal Wiener filter above, we tackle each $\mathbf{p}_i$ from $\mathbf{P}_R$ and obtain the corresponding processed result $\hat{\mathbf{p}}_i$, $i = 1, 2, \cdots R$. Then we have $\hat{\mathbf{P}}_R = [\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \cdots, \hat{\mathbf{p}}_R]^T$. We can further obtain the denoised cluster using the reverse PCA transform:

$$\hat{\mathbf{Y}}_R = \mathbf{U}_R \hat{\mathbf{P}}_R. \quad (18)$$

V. Overall of ACVA

Considering the high dimensionality of image data matrix, we use a sliding window approach to avoid a significant increase in the computational burden when implementing the adaptive clustering for the image of increasing size. By using a fixed-size sliding window, the computational burden of the proposed algorithm within each window is comparatively stabilized in a reasonable range. For a sliding window with fixed size 128×128, the runtime of the Matlab codes on a PC equipped with an Intel Core i5-4460 Quad-Core 3.2 GHz CPU ranges from $4.0s$ to $16.0s$ approximately for different noise levels.

As shown in Fig. 2, all the estimated patches from the sliding windows at different portions of the image are aggregated and averaged to obtain the final estimate. This sliding window and aggregation approach has also been used in most block matching based denoising algorithms and proves to be helpful to further remove the residual noise in the estimated patches thereby leading to the performance boost.

Overall of the proposed algorithm ACVA is summarized in Algorithm 2.

VI. Experimental Results

To validate all the algorithms comprehensively, we use three performance metrics: peak signal-to-noise ratio (PSNR) [39], structural similarity (SSIM) [40] and feature similarity (FSIM) [41]. The PSNR regards the structural information and the nonstructural information as the same in terms of the contribution towards the performance, whereas SSIM and FSIM put more emphasis on the structural information. The images used for denoising experiments are displayed in Fig. 6-7. We compare the proposed algorithm ACVA[1] with: BM3D [11], BM3DSAPCA (SAPCA for short) [8], WNNM [9], SLRD [10], SGHP [15], NCSR [42], and AC-PT [6]. Moreover, the most representative deep learning based denoising methods, *i.e.*, the denoising convolutional neural

[1]The Matlab source code is available at http://www.escience.cn/people/bjqin/research.html

Algorithm 2 Variation Adaptive Filtering Based on Adaptively Clustered Patches(ACVA)

Input: Gaussian noisy data Φ, noise level σ.
Output: The denoised result $\widehat{\Phi}$;

```
Patch extraction: shift a sliding window of size W_s = 128 by step size D_W = 32, and partition each image crop in the window into a collection of image patches Y with patch size d = 8.
Adaptive Clustering in Algorithm 1;
for each cluster do
    PCA transform;
    Dimension selection with a hard threshold of PCA eigenvalues;
    for each selected dimension do
        for each coefficient do
            Estimate the parameter of suboptimal Wiener filter with LPA-ICI;
            Suboptimal Wiener filter denoising;
        end for
    end for
    Reverse PCA transform;
end for
Reproject all the estimated stacked patches Ŷ into image Φ̂.
```

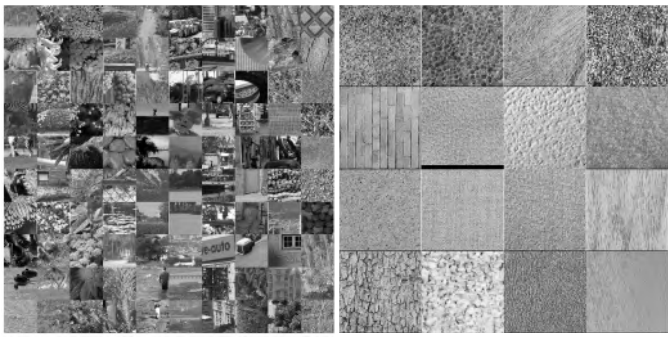

Fig. 6. The test datasets for Gaussian denoising experiments. Left:The 100 image samples from McGill dataset [37] (transformed from RGB into gray images); Right: The 16 texture images from USC-SIPI dataset [38].

Fig. 7. The standard test images. Left: six standard gray scale images for Gaussian denoising experiments; Right: four standard RGB images for camera raw image simulation.

TABLE I

THE AVERAGE DENOISING PERFORMANCE OF ACVA USING ADAPTIVE CLUSTERING WITH DIFFERENT PARAMETERS ON TEST IMAGES FROM THE MCGILL DATASET

σ	Index	AC(0,0.7)	AC(0,1.0)	AC(100,0.7)	AC(200,0.9)	AC(200,0.7)	AC(200,0.5)	AC(200,0.3)	AC(400,0.7)
10	PSNR	32.50	32.50	32.50	32.50	32.50	32.50	32.50	32.50
	SSIM	0.9195	0.9196	0.9196	0.9196	0.9196	0.9196	0.9197	0.9196
	FSIM	0.9564	0.9568	0.9566	0.9567	0.9567	0.9566	0.9566	0.9566
20	PSNR	28.88	28.91	28.91	28.91	28.91	28.91	28.91	28.91
	SSIM	0.8383	0.8409	0.8409	0.8409	0.8409	0.8410	0.8410	0.8409
	FSIM	0.9141	0.9158	0.9152	0.9154	0.9153	0.9152	0.9150	0.9154
50	PSNR	24.60	24.79	24.81	24.80	24.81	24.81	24.81	24.80
	SSIM	0.6516	0.6809	0.6810	0.6810	0.6812	0.6810	0.6809	0.6809
	FSIM	0.8310	0.8375	0.8362	0.8371	0.8364	0.8359	0.8348	0.8367
100	PSNR	21.66	22.05	22.08	22.06	22.09	22.09	22.11	22.07
	SSIM	0.4803	0.5347	0.5362	0.5345	0.5367	0.5363	0.5369	0.5351
	FSIM	0.7641	0.7561	0.7551	0.7555	0.7555	0.7548	0.7541	0.7552

network with noise level specified (DnCNN-S) [43], is also taken into consideration for comparison. All the algorithms are set with default parameters for their best performances.

A. Gaussian Denoising

A) Test on the adaptive clustering: Table I shows how the parameter L_T and ρ in the proposed adaptive clustering affect the denoising performance on 100 images from the McGill dataset [37] at different noise levels, where the DF described in the previous section is used. Denote the proposed adaptive clustering method as AC(L_T, ρ), where L_T is the maximum cluster size in a pair of clusters for computation of between-cluster distance and ρ is the amplification coefficient. Particularly, the adaptive clustering in AC-PT [6] can be denoted as AC(0, 1). When setting $L_T \geq 100$ and $\rho \leq 0.9$, we see the increase of PSNR especially at high noise levels. However, at high noise levels ($\sigma = 50$ and $\sigma = 100$), we can see that as ρ decreases, the FSIM results decline gradually, while the SSIM results peak at $\rho = 0.7$. When fixing $\rho = 0.7$ and changing the value of L_T from 0 to 400, we can observe that setting $L_T = 200$ can help achieve the highest PSNR and SSIM performances as well as a satisfactory FSIM performance. Thus, to achieve a robust denoising performance, we set $L_T = 200$ and $\rho = 0.7$.

B) Test on the variation adaptive denoising filter: With the adaptive clustering parameters fixed, we further test the proposed texture variation adaptive filter with different settings on the 100 image samples from McGill dataset. Denote the texture variation adaptive filter as VA(A, B), where $A = W$ for Wiener filter or $A = SW$ for suboptimal Wiener filter, and $B = N$ for fixed window width or $B = L$ for adaptive window width based on LPA-ICI. Particulary, the setting VA(W, N) corresponds to the DF in AC-PT [6]. Table II shows the superior performance of VA(SW, L) to other settings in terms of both PSNR and SSIM. As for the FSIM performance, VA(SW, L) is only inferior to VA(SW, N), but VA(SW, N) has the worst PSNR performance at high noise levels ($\sigma = 50$ and $\sigma = 100$) and therefore leads to a typical under-denoising. In this way, VA(SW, L) shows a better balance between noise reduction and feature preservation. In addition, it should be noted that compared with the settings using traditional Wiener filter (*i.e.*, VA(W, N) and VA(W, L)), the suboptimal approach VA(SW, L) can still lead to a boost of PSNR performance,

TABLE II

THE AVERAGE DENOISING PERFORMANCE OF ACVA USING TEXTURE VARIATION ADAPTIVE FILTER WITH DIFFERENT SETTINGS ON TEST IMAGES FROM THE MCGILL DATASET

σ	Index	VA(W, N)	VA(W, L)	VA(SW, N)	VA(SW, L)
10	PSNR	32.50	32.49	32.50	32.50
	SSIM	0.9179	0.9176	0.9195	0.9196
	FSIM	0.9556	0.9553	0.9567	0.9567
20	PSNR	28.87	28.85	28.90	28.91
	SSIM	0.8368	0.8361	0.8402	0.8409
	FSIM	0.9121	0.9109	0.9156	0.9153
50	PSNR	24.75	24.73	24.73	24.81
	SSIM	0.6767	0.6730	0.6807	0.6812
	FSIM	0.8334	0.8268	0.8407	0.8364
100	PSNR	22.01	22.00	21.96	22.09
	SSIM	0.5321	0.5283	0.5351	0.5367
	FSIM	0.7540	0.7421	0.7642	0.7555

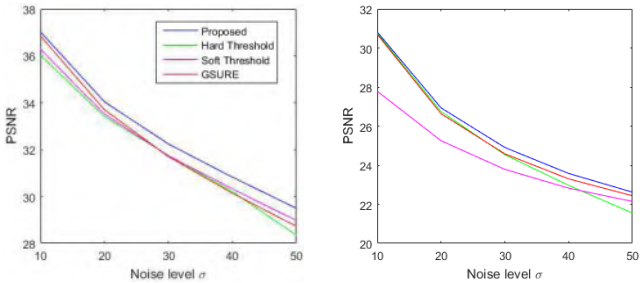

Fig. 8. PSNR (dB) results of the proposed filter compared with singular value thresholding methods at various noise levels σ. (a) and (b) are results for House and Bark respectively.

which may imply the characteristics of over-denoising by the traditional optimal Wiener filter.

We compare the proposed DF with noniterative SVT methods in recent literature: GSURE [31], hard thresholding method [44], and soft thresholding method [45]. As in Fig. 8, our results show that the proposed DF is superior to these noniterative SVT algorithms in denoising Bark and House image.

C) Comparison with the state-of-the-art denoising algorithms: The images used for test include 16 textured images (as in Fig. 6) from the USC-SIPI Image Database [38], six standard test images (Fig. 7), and 100 image samples from McGill dataset (as in Fig. 6, already transformed into grayscale images). We test the considered algorithms at different noise levels $\sigma = 10, 20, 30, 40, 50$.

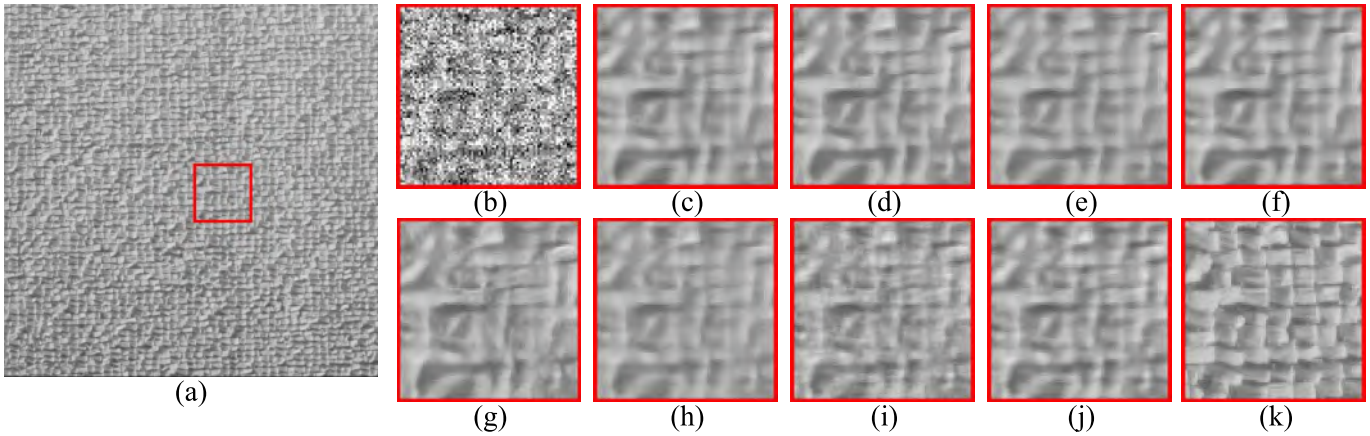

Fig. 9. Denoising the Grid image at $\sigma = 50$: (a) Grid image, (b) Noisy block, (c) BM3D, (d) BM3DSAPCA, (e) WNNM, (f) SLRD, (g) DnCNN-S, (h) SGHP, (i) AC-PT, (j) ACVA, (k) Noise-free block.

TABLE III

THE AVERAGE PSNR(dB), SSIM, FSIM RESULTS ON 16 IMAGES WITH IRREGULAR TEXTURES FROM THE USC-SIPI DATASET

Methods	σ	10	20	30	40	50
BM3D	PSNR	31.55	27.83	25.91	24.44	23.59
	SSIM	0.9293	0.8430	0.7710	0.7044	0.6469
	FSIM	0.9882	0.9667	0.9432	0.9177	0.8925
SAPCA	PSNR	31.62	27.88	25.92	24.62	23.65
	SSIM	0.9327	0.8500	0.7773	0.7141	0.6592
	FSIM	0.9884	0.9672	0.9436	0.9192	0.8961
SGHP	PSNR	31.55	27.91	25.91	24.71	23.67
	SSIM	0.9314	0.8495	0.7752	0.7108	0.6443
	FSIM	0.9874	0.9653	0.9399	0.9147	0.8859
NCSR	PSNR	31.58	27.79	25.80	24.50	23.54
	SSIM	0.9316	0.8453	0.7696	0.7004	0.6417
	FSIM	0.9879	0.9649	0.9381	0.9126	0.8842
WNNM	PSNR	31.65	27.85	25.92	24.60	23.74
	SSIM	0.9311	0.8450	0.7726	0.7042	0.6557
	FSIM	0.9884	0.9659	0.9415	0.9121	0.8892
SLRD	PSNR	31.64	27.90	25.98	24.70	23.78
	SSIM	0.9315	0.8504	0.7816	0.7164	0.6623
	FSIM	0.9883	0.9675	0.9438	0.9163	0.8894
DnCNN-S	PSNR	**31.70**	27.94	25.97	24.66	23.73
	SSIM	0.9340	0.8545	0.7835	0.7199	0.6659
	FSIM	0.9884	0.9673	0.9430	0.9161	0.8903
AC-PT	PSNR	31.65	27.89	25.96	24.63	23.59
	SSIM	0.9323	0.8499	0.7832	0.7257	0.6715
	FSIM	0.9884	0.9673	0.9455	0.9250	0.9047
ACVA	PSNR	31.66	**27.98**	**26.07**	**24.81**	**23.87**
	SSIM	**0.9345**	**0.8554**	**0.7894**	**0.7331**	**0.6848**
	FSIM	**0.9886**	**0.9686**	**0.9482**	**0.9286**	**0.9103**

TABLE IV

PSNR(dB), SSIM AND FSIM RESULTS OF GAUSSIAN DENOISING ON SIX WIDELY USED TEST IMAGES (THE GRAYSCALE IMAGES SHOWN IN FIG. 7)

	σ	10			20			30			40			50		
Method	Image	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM
BM3D	Fluocells	36.12	.9328	.9816	32.76	.8687	.9552	30.95	.8191	.9311	29.65	.7762	.9087	28.80	.7475	.8920
	House	36.71	.9218	.9549	33.77	.8726	.9225	32.09	.8480	.9065	30.65	.8256	.8917	29.69	.8122	.8763
	Lena	35.93	.9166	.9834	33.05	.8772	.9656	31.26	.8449	.9488	29.86	.8152	.9333	29.05	.7994	.9232
	Mandrill	33.14	.9327	.9813	29.07	.8503	.9508	26.85	.7741	.9214	25.27	.7028	.8929	24.38	.6433	.8682
	Stream	31.17	.9076	.9822	27.27	.7900	.9531	25.46	.6986	.9240	24.31	.6287	.8957	23.57	.5715	.8735
	Hill	33.67	.8851	.9784	30.76	.8061	.9522	29.18	.7525	.9304	28.03	.7098	.9101	27.22	.6770	.8941
	Average	34.46	.9161	.9770	31.12	.8441	.9499	29.30	.7895	.9270	27.96	.7431	.9054	27.12	.7085	.8879
SAPCA	Fluocells	36.16	.9334	.9818	32.78	.8665	.9548	30.95	.8133	.9295	29.68	.7686	.9066	28.71	.7329	.8867
	House	37.01	.9290	.9604	33.90	.8763	.9237	32.13	.8495	.9040	30.75	.8301	.8886	29.52	.8078	.8761
	Lena	36.07	.9183	.9838	33.20	.8809	.9668	31.40	.8505	.9518	30.10	.8247	.9378	29.08	.8014	.9236
	Mandrill	33.28	.9346	.9817	29.19	.8546	.9521	26.97	.7809	.9230	25.53	.7161	.8976	24.49	.6599	.8740
	Stream	31.36	.9119	.9829	27.51	.8028	.9550	25.64	.7117	.9267	24.50	.6410	.9018	23.70	.5861	.8786
	Hill	33.89	.8911	.9792	30.90	.8113	.9529	29.27	.7553	.9294	28.11	.7128	.9100	27.23	.6781	.8924
	Average	34.63	.9197	.9783	31.25	.8487	.9509	29.39	.7935	.9274	28.11	.7489	.9071	27.12	.7110	.8886
SGHP	Fluocells	35.79	.9209	.9799	32.68	.8701	.9540	30.69	.8176	.9287	29.54	.7823	.9051	28.54	.7479	.8836
	House	36.36	.9128	.9621	33.74	.8738	.9297	31.93	.8434	.9113	30.77	.8298	.8912	29.51	.8118	.8755
	Lena	35.65	.9089	.9823	32.88	.8737	.9641	30.94	.8379	.9463	29.82	.8211	.9341	28.70	.7970	.9206
	Mandrill	33.19	.9303	.9810	29.13	.8523	.9487	26.71	.7727	.9168	25.32	.7033	.8855	24.25	.6383	.8582
	Stream	31.22	.9084	.9818	27.38	.7998	.9513	25.50	.7128	.9220	24.36	.6386	.8944	23.56	.5772	.8704
	Hill	33.65	.8840	.9783	30.71	.8065	.9518	28.99	.7478	.9295	27.93	.7064	.9076	27.05	.6694	.8893
	Average	34.31	.9109	.9775	31.09	.8460	.9499	29.13	.7887	.9258	27.96	.7469	.9030	26.94	.7069	.8829
NCSR	Fluocells	36.07	.9329	.9811	32.70	.8713	.9539	30.80	.8234	.9284	29.54	.7824	.9029	28.58	.7516	.8809
	House	36.80	.9239	.9600	33.87	.8737	.9209	32.08	.8487	.8988	30.81	.8325	.8816	29.62	.8161	.8677
	Lena	35.85	.9157	.9821	32.95	.8768	.9629	31.06	.8455	.9442	29.92	.8239	.9318	28.90	.8035	.9182
	Mandrill	33.32	.9335	.9803	29.13	.8485	.9466	26.75	.7683	.9121	25.30	.6905	.8790	24.33	.6361	.8527
	Stream	31.20	.9059	.9813	27.32	.7890	.9496	25.52	.7024	.9190	24.32	.6198	.8915	23.54	.5677	.8669
	Hill	33.75	.8879	.9778	30.69	.8029	.9501	29.00	.7451	.9254	27.87	.6969	.9024	27.02	.6644	.8824
	Average	34.50	.9166	.9771	31.11	.8437	.9473	29.20	.7889	.9213	27.96	.7410	.8982	27.00	.7066	.8781
WNNM	Fluocells	36.17	.9337	.9820	32.74	.8687	.9537	30.93	.8213	.9271	29.61	.7806	.9014	28.75	.7539	.8818
	House	36.93	.9228	.9548	34.03	.8716	.9225	32.55	.8523	.9083	31.35	.8348	.8970	30.33	.8231	**.8836**
	Lena	36.05	.9177	.9827	33.12	.8787	.9634	31.43	.8502	.9468	30.11	.8220	.9303	29.25	.8059	.9183
	Mandrill	33.52	.9348	.9813	29.15	.8490	.9490	26.89	.7721	.9179	25.39	.6992	.8851	24.48	.6494	.8618
	Stream	31.26	.9080	.9826	27.38	.7925	.9529	25.59	.7053	.9240	24.46	.6315	.8939	23.69	.5815	.8719
	Hill	33.81	.8872	.9784	30.83	.8053	.9494	29.27	.7520	.9251	28.14	.7086	.9024	27.36	.6788	.8838
	Average	34.62	.9174	.9770	31.21	.8443	.9485	29.44	.7922	.9249	28.17	.7461	.9017	27.31	.7154	.8835
SLRD	Fluocells	36.12	.9324	.9815	32.84	.8720	.9542	31.01	.8273	.9268	29.80	.7910	.9002	28.87	**.7613**	.8787
	House	**37.06**	.9235	.9577	**34.14**	.8732	.9237	**32.68**	.8544	.9031	**31.58**	**.8435**	.8867	**30.64**	**.8333**	.8773
	Lena	36.10	.9192	.9831	33.27	.8818	.9648	31.51	**.8522**	.9461	**30.35**	**.8317**	.9297	**29.39**	**.8120**	.9167
	Mandrill	33.31	.9306	.9799	29.18	.8499	.9481	**27.01**	.7803	.9170	**25.58**	.7126	.8846	24.56	.6550	.8567
	Stream	31.25	.9061	.9818	27.44	.7957	.9526	25.62	.7097	.9236	24.48	.6367	.8945	23.68	.5801	.8663
	Hill	33.88	.8895	.9782	**30.92**	.8105	.9507	29.34	.7569	.9248	28.24	.7132	.8989	27.44	.6834	.8823
	Average	34.62	.9172	.9770	31.30	.8472	.9490	29.53	.7968	.9236	**28.34**	.7548	.8991	**27.45**	.7208	.8797
DnCNN-S	Fluocells	**36.28**	**.9363**	.9824	**32.95**	.8756	.9571	**31.10**	.8253	.9323	**29.89**	.7904	.9098	**28.87**	.7499	.8890
	House	36.52	.9115	.9556	33.89	.8702	.9225	32.30	.8509	.9061	31.02	.8341	.8907	30.02	.8200	.8794
	Lena	**36.20**	**.9195**	**.9842**	**33.40**	**.8836**	.9676	**31.59**	**.8546**	**.9525**	30.32	.8308	**.9389**	29.37	.8115	**.9277**
	Mandrill	**33.48**	**.9374**	.9823	**29.24**	**.8584**	.9533	26.98	**.7857**	.9243	25.55	.7213	.8961	24.57	.6685	.8714
	Stream	**31.49**	**.9159**	**.9831**	**27.73**	**.8166**	**.9563**	**25.91**	**.7354**	.9302	**24.77**	**.6673**	.9046	**23.97**	**.6133**	.8811
	Hill	**33.92**	.8923	.9797	30.91	.8103	.9537	**29.32**	.7556	.9309	**28.25**	.7149	.9110	**27.47**	**.6855**	.8947
	Average	**34.65**	.9188	.9779	**31.35**	.8524	.9518	**29.54**	**.8013**	.9294	28.30	**.7598**	.9085	27.38	**.7248**	.8906
AC-PT	Fluocells	36.03	.9307	.9816	32.62	.8658	.9566	30.58	.8001	.9286	29.29	.7633	.9103	27.92	.6852	.8777
	House	36.94	.9279	.9616	33.89	.8834	.9371	32.06	.8562	.9187	30.33	.8191	.9012	28.97	.7901	.8829
	Lena	35.76	.9124	.9830	32.58	.8630	.9630	30.62	.8218	.9423	29.13	.7787	.9208	27.86	.7360	.8936
	Mandrill	33.13	.9341	.9817	28.83	.8456	.9516	26.54	.7613	.9232	25.12	.6891	.9003	24.14	.6337	.8828
	Stream	30.93	.8989	.9814	27.17	.7856	.9525	25.37	.6977	.9262	24.32	.6376	.9048	23.53	.5899	.8852
	Hill	33.62	.8830	.9782	30.57	.8021	.9534	28.80	.7388	.9317	27.53	.6881	.9114	26.61	.6527	.8933
	Average	34.40	.9145	.9779	30.94	.8409	.9523	29.00	.7793	.9284	27.62	.7293	.9081	26.51	.6813	.8859
ACVA	Fluocells	36.21	.9353	**.9824**	32.88	**.8770**	**.9584**	31.03	**.8312**	**.9368**	29.74	**.7925**	**.9165**	28.76	.7603	**.8991**
	House	37.05	**.9309**	**.9631**	34.04	**.8859**	**.9359**	32.26	**.8580**	**.9132**	30.88	.8338	**.8981**	29.52	.8100	**.8835**
	Lena	36.07	.9190	.9841	33.15	.8808	**.9675**	31.32	.8490	.9517	29.94	.8208	.9365	28.89	.7960	.9223
	Mandrill	33.28	.9364	**.9824**	29.10	.8564	**.9543**	26.88	.7824	**.9278**	25.48	**.7220**	**.9064**	**24.64**	**.6815**	**.8926**
	Stream	31.26	.9119	.9826	27.53	.8063	.9556	25.69	.7202	**.9309**	24.56	.6585	**.9113**	23.76	.6084	**.8940**
	Hill	33.89	**.8925**	**.9796**	30.91	**.8150**	**.9558**	29.24	**.7596**	**.9362**	28.05	**.7152**	**.9183**	27.15	.6801	**.9027**
	Average	34.63	**.9210**	**.9790**	31.27	**.8536**	**.9546**	29.40	.8001	**.9327**	28.11	.7571	**.9145**	27.12	.7227	**.8990**

In Table III, we quantify the performances of nine competing algorithms for the textured images with different noise levels in terms of PSNR, SSIM and FSIM. From Table III, we can see that at low noise level ($\sigma = 10$) the proposed algorithm ACVA has a favorable PSNR performance that is only inferior to DnCNN-S, and shows the highest SSIM and FSIM performances. When the noise level increases, ACVA is superior to other algorithms in terms of all the metrics, which proves that ACVA quantitatively outperforms the other methods in denoising images with irregular textures.

In Table IV, we further compare quantitative performance of the nine algorithms on standard test images which have larger flat area and less textured area. Table IV shows that ACVA can still achieve competitive quantitative performance, especially in terms of FSIM, over other algorithms on denoising natural images.

In terms of visual texture-preserving performance, the proposed method is also superior to the state-of-the-art denoising methods. Fig. 9 shows the denoising results of Grid image with noise level $\sigma = 50$. As can be seen from it, the proposed method produces results closer to the noise-free image than do other methods and achieves a better balance between noise removal and detail preservation than AC-PT. Fig. 10 and 11 further illustrate the visual comparisons. From the zoom-in denoising results of the noisy image Mandrill and Stream,

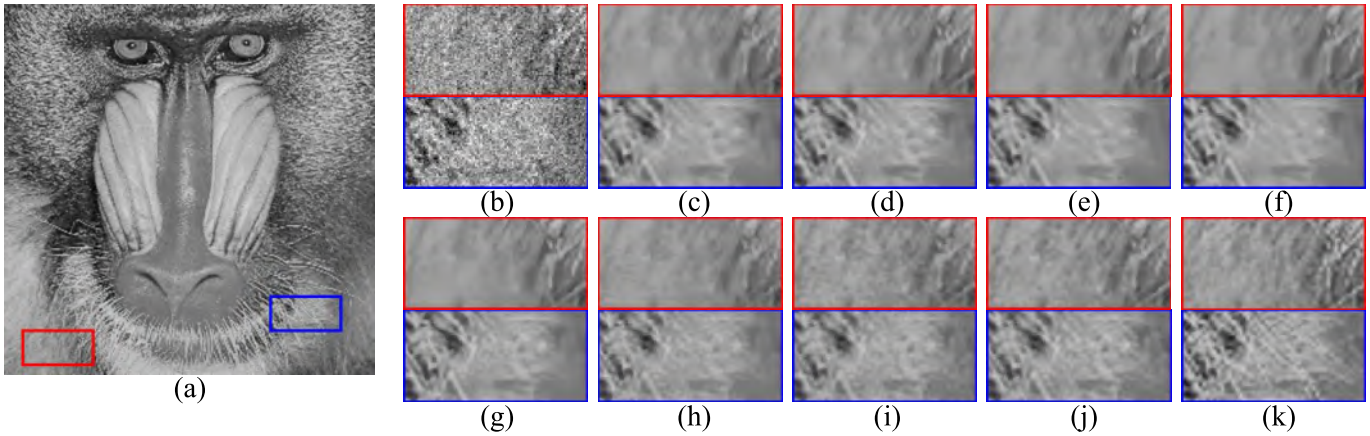

Fig. 10. Denoising the Mandrill image at $\sigma = 30$ with image details in the zoomed areas (red and blue boxes): (a) Mandrill image, (b) Noisy block, (c) BM3D, (d) BM3DSAPCA, (e) WNNM, (f) SLRD, (g) DnCNN-S, (h) SGHP, (i) AC-PT, (j) ACVA, (k) Noise-free block.

it can be observed that the proposed method outperforms other methods in restoring the textures of the fur (in Fig. 10) and water surface (in Fig. 11).

B. Camera RAW Image Denoising

1) Camera RAW Image Simulation: Four standard RGB test images, specifically, Peppers, Lena, Baboon, and House are selected for camera raw image denoising simulation. We adopt the simulation method used in [46]. To transform the RGB images into simulated raw images, we first scale the RGB images (within the bounds of $[0, 1]$) to the domain of raw images:

$$[r'_{i,j}, g'_{i,j}, b'_{i,j}]^T = R_{max} \times [r_{i,j}, g_{i,j}, b_{i,j}]^T \qquad (19)$$

where r, g, and b denote the signal in red, green, and blue channels, respectively, i and j are the x-coordinate and y-coordinate of any image pixel, respectively.

Then the pixels are further arranged into four subimages, R, $G1$, $G2$, and B for simulation of the color filter array (CFA) [47].

$$\begin{aligned} R &= \{r''_{i,j} = r'_{2i-1,2j-1}\}, \\ G1 &= \{g''_{i,j} = g'_{2i-1,2j}\}, \\ G2 &= \{g''_{i,j} = g'_{2i,2j-1}\}, \\ B &= \{b''_{i,j} = b'_{2i,2j}\}. \end{aligned} \qquad (20)$$

Finally, all the four subimages are arranged into a single image to simulate the RAW image.

To better simulate the noise in the real RAW image, we set the noise parameters $\alpha > 0$, $b = 0$, $p = 0$ as [46]. Based on Poisson-Gaussian noise reduction scheme described in Section II, we apply the proposed algorithm to denoise R, G1, G2, and B separately and then compute the average PSNR (SSIM and FSIM) results for evaluation.

Table V compares the quantitative performance of nine competing algorithms on simulated camera RAW images. We can see that the proposed algorithm is superior to all other algorithms in terms of SSIM and FSIM on average.

TABLE V

PSNR (dB), SSIM, FSIM RESULTS OF CAMERA RAW IMAGE SIMULATION ON THE RGB IMAGES SHOWN IN FIG. 7

	α	200			400			600			800			1000		
Method	Image	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM	PSNR	SSIM	FSIM
BM3D	House	32.91	.8797	.9428	34.59	.9109	.9609	35.66	.9272	.9691	36.44	.9377	.9739	37.07	.9470	.9779
	Baboon	28.09	.8883	.9412	30.53	.9323	.9624	32.06	.9498	.9717	33.13	.9595	.9769	33.98	.9657	.9801
	Peppers	32.40	.8736	.9431	34.08	.9040	.9585	35.17	.9213	.9672	36.00	.9332	.9722	36.61	.9404	.9758
	Lena	32.67	.8810	.9456	34.39	.9068	.9591	35.40	.9209	.9659	36.16	.9303	.9705	36.75	.9373	.9739
	Average	31.52	.8806	.9432	33.40	.9135	.9602	34.57	.9298	.9685	35.43	.9402	.9734	36.10	.9476	.9769
SAPCA	House	**33.31**	.8914	.9503	**35.04**	**.9211**	**.9663**	**36.09**	**.9356**	**.9728**	**36.85**	**.9438**	**.9767**	**37.44**	**.9511**	.9799
	Baboon	**28.29**	.8966	.9440	**30.69**	.9361	.9635	**32.19**	.9520	.9725	**33.25**	.9611	.9775	**34.08**	**.9669**	.9806
	Peppers	**32.67**	.8784	.9465	**34.35**	.9091	.9612	**35.40**	.9254	.9694	**36.22**	.9365	.9738	**36.83**	.9432	.9771
	Lena	32.90	.8854	.9485	**34.62**	.9110	.9615	**35.63**	.9252	.9681	**36.37**	.9341	.9723	**36.96**	.9409	.9756
	Average	**31.79**	.8879	.9473	**33.67**	.9193	.9631	**34.83**	.9345	.9707	**35.67**	.9439	.9751	**36.33**	.9505	.9783
NCSR	House	33.10	.8883	.9483	34.77	.9175	.9643	35.80	.9323	.9709	36.56	.9407	.9748	37.18	.9491	.9785
	Baboon	28.10	.8825	.9382	30.53	.9285	.9607	32.06	.9469	.9704	33.14	.9570	.9760	33.98	.9634	.9794
	Peppers	32.42	.8747	.9446	34.12	.9048	.9587	35.18	.9207	.9668	36.02	.9327	.9718	36.63	.9392	.9750
	Lena	32.69	.8809	.9451	34.40	.9064	.9586	35.40	.9210	.9658	36.16	.9302	.9700	36.75	.9372	.9736
	Average	31.58	.8816	.9440	33.46	.9143	.9606	34.61	.9302	.9685	35.47	.9402	.9732	36.13	.9472	.9766
SGHP	House	32.68	.8807	.9486	34.26	.9112	.9635	35.21	.9259	.9696	35.89	.9359	.9734	36.45	.9441	.9771
	Baboon	28.12	.8909	.9445	30.51	.9328	.9634	32.01	.9499	.9720	33.05	.9595	.9771	33.84	.9654	.9802
	Peppers	32.38	.8765	.9484	34.09	.9071	.9610	35.10	.9228	.9681	35.89	.9338	.9726	36.44	.9401	.9756
	Lena	32.62	.8817	.9493	34.37	.9099	.9614	35.39	.9244	.9685	36.13	.9330	.9721	36.71	.9396	.9752
	Average	31.45	.8825	.9477	33.31	.9152	.9623	34.43	.9307	.9696	35.24	.9405	.9738	35.86	.9473	.9770
SLRA	House	33.25	.8815	.9459	34.91	.9130	.9634	35.98	.9301	.9716	36.74	.9398	.9758	37.37	.9485	.9797
	Baboon	28.10	.8803	.9353	30.58	.9294	.9600	32.08	.9470	.9695	33.14	.9570	.9752	33.98	.9636	.9787
	Peppers	32.63	.8755	.9431	34.30	.9063	.9592	35.33	.9219	.9670	36.14	.9332	.9718	36.72	.9395	.9751
	Lena	**32.92**	.8845	.9464	34.59	.9090	.9593	35.59	.9219	.9656	36.33	.9317	.9704	36.91	.9381	.9739
	Average	31.72	.8804	.9427	33.60	.9144	.9605	34.75	.9302	.9684	35.59	.9404	.9733	36.24	.9474	.9768
WNNM	House	33.24	.8835	.9443	34.93	.9143	.9636	35.96	.9305	.9709	36.71	.9392	.9752	37.33	.9481	.9790
	Baboon	28.20	.8888	.9409	30.64	.9331	.9622	32.15	.9501	.9715	33.22	.9597	.9768	34.06	.9659	.9801
	Peppers	32.58	.8753	.9438	34.22	.9047	.9594	35.27	.9213	.9677	36.09	.9332	.9727	36.69	.9400	.9761
	Lena	32.85	.8827	.9453	34.52	.9075	.9593	35.53	.9216	.9665	36.25	.9307	.9708	36.84	.9375	.9743
	Average	31.72	.8826	.9435	33.58	.9149	.9611	34.73	.9309	.9692	35.57	.9407	.9739	36.23	.9479	.9774
DnCNN-S	House	32.46	.8622	.9428	33.88	.8873	.9468	34.19	.8856	.9451	34.06	.8784	.9389	34.17	.8805	.9399
	Baboon	28.15	.8897	.9418	29.44	.8928	.9434	29.82	.8916	.9441	30.05	.8920	.9452	30.39	.8963	.9476
	Peppers	32.36	.8672	.9378	32.51	.8571	.9293	32.44	.8523	.9262	32.50	.8524	.9269	32.44	.8500	.9252
	Lena	32.79	.8788	.9423	33.92	.8967	.9544	34.30	.8969	.9537	34.53	.8975	.9536	34.50	.8942	.9515
	Average	31.44	.8745	.9412	32.43	.8835	.9435	32.69	.8816	.9423	32.79	.8801	.9411	32.87	.8803	.9411
AC-PT	House	32.95	.8905	.9530	34.68	.9181	.9647	35.78	.9331	.9724	36.62	.9428	.9761	37.17	.9492	.9787
	Baboon	28.14	.8929	.9438	30.60	.9342	.9634	32.06	.9506	.9721	33.14	.9602	.9773	34.03	.9663	**.9809**
	Peppers	32.43	.8796	**.9483**	34.16	.9090	.9616	35.26	.9249	.9689	36.06	.9358	.9737	36.69	.9427	.9767
	Lena	32.63	.8803	.9492	34.41	.9093	.9620	35.46	.9235	.9682	36.22	.9335	.9728	36.85	.9405	.9757
	Average	31.54	.8858	.9486	33.46	.9177	.9629	34.64	.9330	.9704	35.51	.9431	.9750	36.18	.9497	.9780
ACVA	House	33.01	**.8915**	**.9520**	34.73	.9206	.9661	35.75	.9349	.9726	36.54	.9435	.9765	37.13	.9507	**.9803**
	Baboon	28.20	**.8994**	**.9464**	30.57	**.9369**	**.9646**	32.06	**.9522**	**.9728**	33.12	**.9611**	**.9777**	33.96	.9669	.9807
	Peppers	32.45	**.8803**	.9481	34.16	**.9110**	**.9626**	35.23	**.9268**	**.9703**	36.06	**.9377**	**.9746**	36.66	**.9441**	**.9778**
	Lena	32.79	**.8860**	**.9504**	34.49	**.9118**	**.9630**	35.51	**.9262**	**.9696**	36.26	**.9349**	**.9735**	36.84	**.9414**	**.9765**
	Average	31.61	**.8893**	**.9492**	33.49	**.9201**	**.9641**	34.64	**.9350**	**.9713**	35.49	**.9443**	**.9756**	36.15	**.9508**	**.9788**

Specially, for image House that has more flat area than other images, ACVA still shows the highest SSIM and FSIM results at the high noise level ($\alpha = 200$) and competitive quantitative performance at the low noise level ($\alpha >= 400$). For the PSNR results, ACVA outperforms BM3D, NCSR, SGHP and DnCNN-S on average. Since the source code of the denoising method EFBMD [46] is unavailable, we can only compare the proposed algorithm with EFBMD according to the results displayed in [46] which uses the same test dataset and simulation methods. As the denoising method EFBMD [46] is inferior to NCSR in terms of the average PSNR results, we can conclude that ACVA is also better than EFBMD in terms of PSNR results. For the SSIM results on the image Baboon, we see that while EFBMD only perform well at low noise level ($\alpha = 1000$), ACVA has a better SSIM performance at all the higher noise levels $\alpha < 1000$. We cannot compare the FSIM performance because [46] does not list the corresponding results.

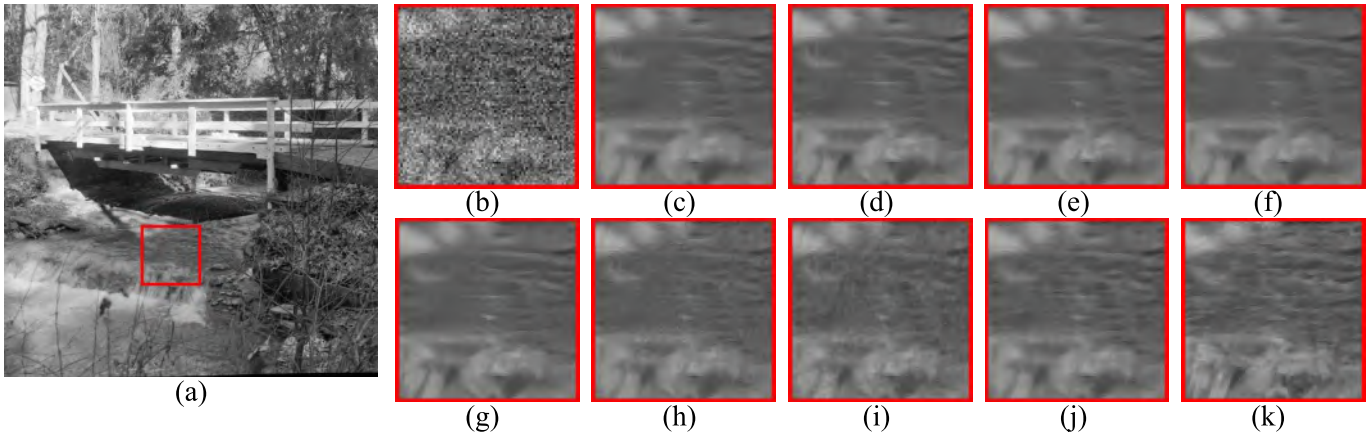

Fig. 11. Denoising the Stream image at $\sigma = 25$ with image details in the zoomed areas (red boxes): (a) Stream image, (b) Noisy block, (c) BM3D, (d) BM3DSAPCA, (e) WNNM, (f) SLRD, (g) DnCNN-S, (h) SGHP, (i) AC-PT, (j) ACVA, (k) Noise-free block.

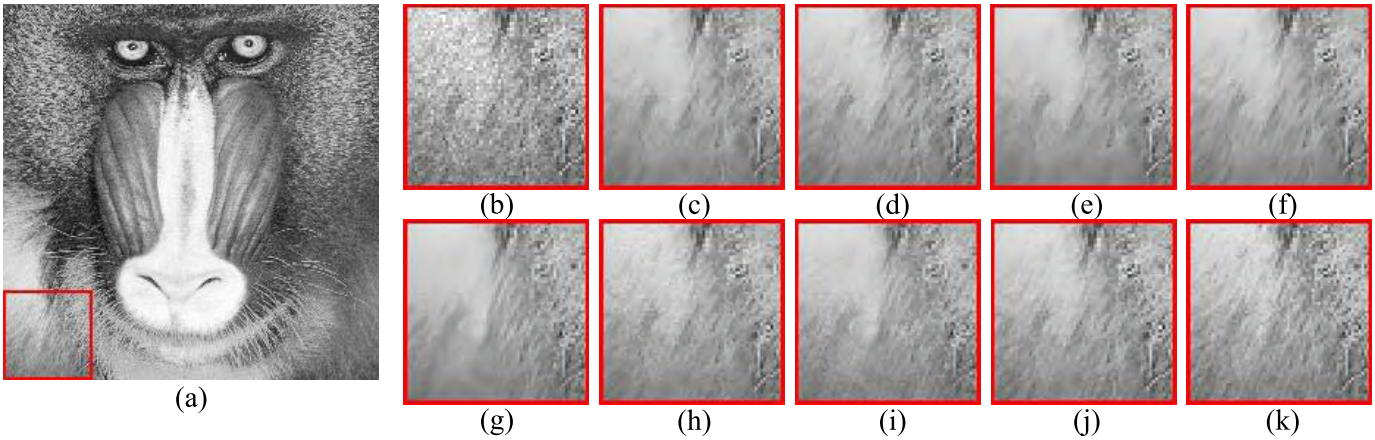

Fig. 12. Performance comparison on the red channel of Baboon image with image details in the zoomed areas (red boxes): (a) The red channel, (b) Noisy block ($\alpha = 200$), (c) BM3D, (d) BM3DSAPCA, (e) NCSR, (f) WNNM, (g) DnCNN-S, (h) SGHP, (i) AC-PT, (j) ACVA, (k) Noise-free block.

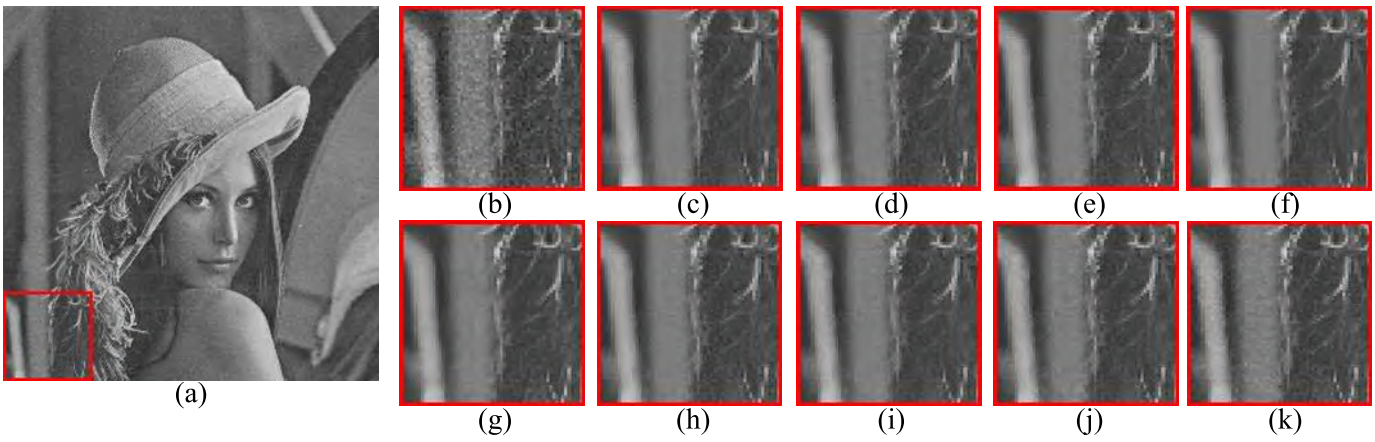

Fig. 13. Performance comparison on the blue channel of Lena image with image details in the zoomed areas (red boxes): (a) The blue channel, (b) Noisy block ($\alpha = 400$), (c) BM3D, (d) BM3DSAPCA, (e) NCSR, (f) WNNM, (g) DnCNN-S, (h) SGHP, (i) AC-PT, (j) ACVA, (k) Noise-free block.

As for the visual texture-preserving performance, ACVA also outperforms the state-of-the-art denoising algorithms. Figs. 12-13 compare the bottom-left corner of the denoising results of image Baboon and Lena. From the zoom-in area, the proposed method outperforms other methods in restoring the special textures of the fur (in Fig. 12) and doorframe (in Fig. 13). In addition, as shown in [46], EFBMD is also inferior to ACVA in preserving the fur texture at the bottom-left corner of image Baboon.

2) Denoising on Real RAW Images: The RAW image of size 3744×5616 is captured by a Canon EOS 5D Mark II. We cut down a 402×402 square from the raw image for denoising tests. The noise parameters (α and b) in Poisson-Gaussian noise model are estimated by the method in [26]. We assume

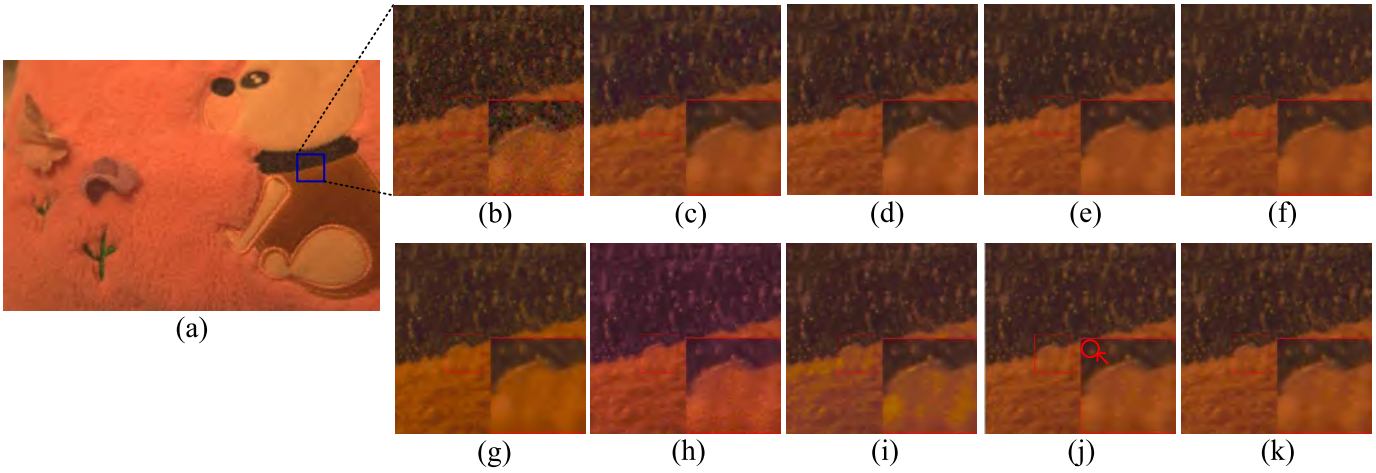

Fig. 14. Denoising the real camera raw image with image details in the zoomed areas (red boxes): (a) Noisy image, (b) Small sample cut down from the blue square, (c) BM3D, (d) BM3DSAPCA, (e) NCSR, (f) WNNM, (g) SLRD, (h) DnCNN-S, (i) SGHP, (j) AC-PT, (k) ACVA.

the noise level is invariant across the whole image. To avoid over-estimate of noise level, we select the top-left 200×200 flat area, estimate its R, G1, G2, B subimage separately, and adopt the minimum estimates of α and b, respectively. After applying the GAT on the RAW image based on the estimated parameters, we denoise the real camera raw image using the considered algorithms directly. To visualize the denoised image, we adopt the method in [47] to transform the results into RGB images.

Fig. 14 shows that ACVA protects zoom-in details (such as singular points and textures) best compared with other algorithms. Specifically, we can also find that there is a noisy black dot mistakenly preserved by AC-PT. And serious color distortion can be observed in the results by SGHP and DnCNN-S, while BM3D, BM3DSAPCA, NCSR, SLRD, and WNNM just blur the isolated white points and brown texture. The serious color distortion by DnCNN-S implies that this state-of-the-art deep learning based denoising algorithm distorts heavily the special textures resulted from the CFA, and how to control this kind of distortion remains an unsolved problem.

VII. Conclusions

In this paper, we have proposed a texture-preserving nonlocal denoising algorithm ACVA. In ACVA, an adaptive clustering method is designed to adaptively and robustly cluster similar patches. A state-of-the-art PCA-based denoising filter is proposed in a transform-domain texture variation adaptive filtering approach to perform a texture-preserving denoising of each cluster. The denoising performance of ACVA is further improved via a sliding window and aggregation approach. When compared with the existing PG techniques (especially the adaptive clustering method in AC-PT), the proposed adaptive clustering method achieves more robust performance at the high noise level. Meanwhile, the proposed DF shows superior denoising performance to other PCA (or SVD) based DFs.

ACVA achieves satisfactory texture-preserving Gaussian denoising performance both quantitatively and visually. Especially on images with irregular textures, ACVA can outperform all the other denoising algorithms tested here in terms of PSNR, SSIM and FSIM results. The noise removal results for camera raw images containing special textures of CFA further verify ACVA's excellent texture-preserving Poisson-Gaussian denoising performance for real application, while the deep learning based denoising algorithm DnCNN works [43] poorly on the real images with CFA patterns that have irregular or stochastic textures. The future work will explore potential benefits of ACVA for improving the overall performance in processing low SNR and low contrast images with irregular textures, such as OCT vessel images [23], low-dose X-ray vessel images [48]–[50] and fluorescence microscopy images [51], [52].

Acknowledgment

The authors would like to thank all the cited authors for providing the source codes used in this work, and the reviewers for their valuable comments on the manuscript.

References

[1] M. Haidekker "Texture Analysis," in *Advanced Biomedical Image Analysis*. Hoboken, NJ, USA: Wiley, 2011, pp. 236–275.
[2] I. Zachevsky and Y. Y. J. Zeevi, "Statistics of natural stochastic textures and their application in image denoising," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2130–2145, May 2016.
[3] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, Sep. 1999, pp. 1033–1038.
[4] 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.
[5] X. Kang, X. Xiang, S. Li, and J. A. Benediktsson, "PCA-based edge-preserving features for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 12, pp. 7140–7151, Dec. 2017.
[6] W. Zhao, Y. Lv, Q. Liu, and B. Qin, "Detail-preserving image denoising via adaptive clustering and progressive PCA thresholding," *IEEE Access*, vol. 6, no. 1, pp. 6303–6315, 2018.
[7] K. Mei, B. Hu, B. Fei, and B. Qin. (2018). "Phase asymmetry guided adaptive fractional-order total variation and diffusion for feature-preserving ultrasound despeckling." [Online]. Available: https://arxiv.org/abs/1810.12538
[8] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, "From local kernel to nonlocal multiple-model image denoising," *Int. J. Comput. Vis.*, vol. 86, no. 1, p. 1, 2010.
[9] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *Int. J. Comput. Vis.*, vol. 121, no. 2, pp. 183–208, Jan. 2017.

[10] M. Nejati, S. Samavi, H. Derksen, and K. Najarian, "Denoising by low-rank and sparse representations," *J. Vis. Commun. Image Represent.*, vol. 36, no. C, pp. 28–39, Apr. 2016.

[11] 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.

[12] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern Recognit.*, vol. 43, no. 4, pp. 1531–1549, 2010.

[13] P. Chatterjee and P. Milanfar, "Patch-based near-optimal image denoising," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1635–1649, Apr. 2012.

[14] 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.

[15] W. Zuo, L. Zhang, C. Song, D. Zhang, and H. Gao, "Gradient histogram estimation and preservation for texture enhanced image denoising," *IEEE Trans. Image Process.*, vol. 23, no. 6, pp. 2459–2472, Jun. 2014.

[16] Q. Guo, C. Zhang, Y. Zhang, and H. Liu, "An efficient SVD-based method for image denoising," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 5, pp. 868–880, May 2016.

[17] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1532–1546, Sep. 2000.

[18] A. Pizurica and W. Philips, "Estimating the probability of the presence of a signal of interest in multiresolution single- and multiband image denoising," *IEEE Trans. Image Process.*, vol. 15, no. 3, pp. 654–665, Mar. 2006.

[19] 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.

[20] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image Process.*, vol. 16, no. 5, pp. 1395–1411, May 2007.

[21] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep./Oct. 2010, pp. 2272–2279.

[22] L. Xu, J. Li, Y. Shu, and J. Peng, "SAR image denoising via clustering-based principal component analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 11, pp. 6858–6869, Nov. 2014.

[23] Z. Chen, Y. Mo, P. Ouyang, H. Shen, D. Li, and R. Zhao, "Retinal vessel optical coherence tomography images for anemia screening," *Med. Biol. Eng. Comput.*, vol. 57, no. 4, pp. 953–966, 2018.

[24] J. Chen, J. Benesty, Y. Huang, and S. Doclo, "New insights into the noise reduction Wiener filter," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 4, pp. 1218–1234, Jul. 2006.

[25] V. Katkovnik, "A new method for varying adaptive bandwidth selection," *IEEE Trans. Signal Process.*, vol. 47, no. 9, pp. 2567–2571, Sep. 1999.

[26] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data," *IEEE Trans. Image Process.*, vol. 17, no. 10, pp. 1737–1754, Oct. 2008.

[27] M. Makitalo and A. Foi, "Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 91–103, Jan. 2013.

[28] M. Khalilian, F. Z. Boroujeni, N. Mustapha, and M. N. Sulaiman, "K-means divide and conquer clustering," in *Proc. Int. Conf. Comput. Autom. Eng.*, 2009, pp. 306–309.

[29] G. Ahirwar, "A novel K means clustering algorithm for large datasets based on divide and conquer technique," *Pradnyesh J. Bhisikar/Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 1, pp. 301–305, 2014.

[30] I. M. Johnstone, "On the distribution of the largest eigenvalue in principal components analysis," *Ann. Statist.*, vol. 29, no. 2, pp. 295–327, 2001.

[31] J. Bigot, C. Deledalle, and D. Féral, "Generalized SURE for optimal shrinkage of singular values in low-rank matrix denoising," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 4991–5040, 2016.

[32] H. Liu, P. Sun, Q. Du, Z. Wu, and Z. Wei, "Hyperspectral image restoration based on low-rank recovery with a local neighborhood weighted spectral–spatial total variation model," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 3, pp. 1409–1422, Mar. 2018.

[33] F. Zhu *et al.*, "Reducing Poisson noise and baseline drift in X-ray spectral images with bootstrap Poisson regression and robust nonparametric regression," *Phys. Med. Biol.*, vol. 58, no. 6, p. 1739, 2013.

[34] M. Köhler, A. Schindler, and S. Sperlich, "A review and comparison of bandwidth selection methods for kernel regression," *Int. Stat. Rev.*, vol. 82, no. 2, pp. 243–274, 2014.

[35] B. Qin, Z. Shen, Z. Fu, Z. Zhou, Y. Lv, and J. Bao, "Joint-saliency structure adaptive kernel regression with adaptive-scale kernels for deformable registration of challenging images," *IEEE Access*, vol. 6, no. 1, pp. 330–343, 2018.

[36] B. Qin, Z. Shen, Z. Zhou, J. Zhou, and Y. Lv, "Structure matching driven by joint-saliency-structure adaptive kernel regression," *Appl. Soft Comput.*, vol. 46, pp. 851–867, Sep. 2016.

[37] A. Olmos and F. A. A. Kingdom, "A biologically inspired algorithm for the recovery of shading and reflectance images," *Perception*, vol. 33, no. 12, pp. 1463–1473, 2004.

[38] A. G. Weber. *The USC-SIPI Image Database*. Accessed: Aug. 10, 2017. [Online]. Available: http://sipi.usc.edu/services/database/Database.html

[39] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment," *Electron. Lett.*, vol. 44, no. 13, pp. 800–801, Jun. 2008.

[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[41] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011.

[42] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1620–1630, Apr. 2013.

[43] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[44] M. Gavish and D. L. Donoho, "The optimal hard threshold for singular values is $4/\sqrt{3}$," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 5040–5053, Jun. 2014.

[45] M. Gavish and D. L. Donoho, "Optimal shrinkage of singular values," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2137–2152, Apr. 2014.

[46] C.-C. Yang, S.-M. Guo, and J. S.-H. Tsai, "Evolutionary fuzzy block-matching-based camera raw image denoising," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2862–2871, Sep. 2017.

[47] D. Khashabi, S. Nowozin, J. Jancsary, and A. W. Fitzgibbon, "Joint demosaicing and denoising via learned nonparametric random fields," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 4968–4981, Dec. 2014.

[48] M. Jin, R. Li, J. Jiang, and B. Qin, "Extracting contrast-filled vessels in X-ray angiography by graduated RPCA with motion coherency constraint," *Pattern Recognit.*, vol. 63, pp. 653–666, Mar. 2017.

[49] M. Jin, D. Hao, S. Ding, and B. Qin, "Low-rank and sparse decomposition with spatially adaptive filtering for sequential segmentation of 2D+t vessels," *Phys. Med. Biol.*, vol. 63, no. 17, 2018, Art. no. 17LT01.

[50] B. Qin *et al.*, "Accurate vessel extraction via tensor completion of background layer in X-ray coronary angiograms," *Pattern Recognit.*, vol. 87, pp. 38–54, Mar. 2019.

[51] M. Weigert *et al.*, "Content-aware image restoration: Pushing the limits of fluorescence microscopy," *Nature Methods*, vol. 15, no. 12, p. 1090, 2018.

[52] R. Li, Y. Wang, H. Xu, B. Fei, and B. Qin, "Micro-droplet detection method for measuring the concentration of alkaline phosphatase-labeled nanoparticles in fluorescence microscopy," *Sensors*, vol. 17, no. 11, p. 2685, 2017.

Wenzhao Zhao received the B.Sc. degree in bioengineering from Yangzhou University, Yangzhou, in 2013, and the M.Sc. degree in biomedical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2018. His current research interests include image processing and computer vision.

Qiegen Liu (M'16) received the B.S. degree in applied mathematics from Gannan Normal University, and the M.Sc. degree in computation mathematics and the Ph.D. degree in biomedical engineering from Shanghai Jiao Tong University (SJTU). From 2015 to 2017, he held a post-doctoral position with UIUC and the University of Calgary. Since 2012, he has been with School of Information Engineering, Nanchang University, Nanchang, China, where he is currently an Associate Professor. His current research interest is sparse representations, deep learning and their applications in image processing, computer vision, and MRI reconstruction.

Yisong Lv received the M.Sc. degree in automatic instruments from the Kunming University of Science and Technology, Kunming, in 1999, and the Ph.D. degree in biomedical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2003. His current research interests include image analysis, machine learning, and computer vision.

Binjie Qin (M'07) received the M.Sc. degree in measuring and testing technologies and instruments from the Nanjing University of Science and Technology, Nanjing, and the Ph.D. degree in biomedical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1999 and 2002, respectively. He was a Lecturer and an Associate Professor with the Department of Biomedical Engineering, School of Life Sciences and Biotechnology, Shanghai Jiao Tong University, Shanghai, China. From 2012 to 2013, he was a Visiting Professor with the Department of Computer Science, University College London, U.K. He is currently an Associate Professor with the School of Biomedical Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include biomedical imaging, image processing, machine learning, computer vision, and biomedical instrumentation.