J. Vis. Commun. Image R. 48 (2017) 340–352

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

A generic denoising framework via guided principal component analysis ☆

Tao Dai [a], Zhiya Xu [a], Haoyi Liang [b], Ke Gu [c], Qingtao Tang [a], Yisen Wang [a], Weizhi Lu [a,*], Shu-Tao Xia [a]

[a] Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, Guangdong, China
[b] University of Virginia, Department of ECE, Charlottesville, VA 22904, USA
[c] BJUT Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

ARTICLE INFO

Article history:
Received 28 June 2016
Revised 27 March 2017
Accepted 20 May 2017
Available online 26 May 2017

Keywords:
Image denoising
Principal component analysis
Back projection

ABSTRACT

Though existing state-of-the-art denoising algorithms, such as BM3D, LPG-PCA and DDF, obtain remarkable results, these methods are not good at preserving details at high noise levels, sometimes even introducing non-existent artifacts. To improve the performance of these denoising methods at high noise levels, a generic denoising framework is proposed in this paper, which is based on guided principle component analysis (GPCA). The propose framework can be split into two stages. First, we use statistic test to generate an initial denoised image through back projection, where the statistical test can detect the significantly relevant information between the denoised image and the corresponding *residual image*. Second, similar image patches are collected to form different patch groups, and local basis are learned from each patch group by principle component analysis. Experimental results on natural images, contaminated with Gaussian and non-Gaussian noise, verify the effectiveness of the proposed framework.

1. Introduction

In many applications like photography under low light and surveillance systems, prominent noise exists due to the physical acquisition process. Such noise could affect the following applications such as High Efficiency Video Coding (HEVC) [1,2], object recognition [3], image segmentation [4], contrast enhancement [5–7], video coding [8,9] and image interpolation [10].

Existing denoising methods adopt different approaches to address the problem. Much of the early work concentrates on processing the image in the spatial domain, due to the simplicity of such methods [11–13]. Instead, much work prefers to remove noise in the transform domain (e.g. wavelet), due to the good capability in noise suppression and edge-preserving for such methods [14–17]. Recently, most of state-of-the-art denoising methods [18–25] share a common patch-based framework. For example, block matching and 3-D (BM3D) [18] obtains satisfactory results by combining nonlocal means (self-similarity) and transform domain method. By contrast, dual domain filter (DDF) [19] is a simpler hybrid method that comprises the simple bilateral filter and Fourier transform. Another effective denoising method (LPG-PCA) [20] applies block matching to group similar patches and shrinks PCA transformation coefficients to suppress the noise. Other state-of-the-art methods also achieve remarkable results by using self-similarity and low rank approximation (e.g., spatially adaptive iterative singular-value thresholding method (SAIST) [21]), or by learning models from natural images [22–24].

Despite the impressive results achieved by state-of-the-art methods, most of these denoising methods, such as BM3D, LPG-PCA and DDF, tend to generate artifacts around the edges, especially at high noise levels. Moreover, it is shown that there is still room for improvement of existing methods due to the imperfect denoising that causes the loss of image details [26]. To preserve the image details better, much work [27–31] focuses on utilizing the residual information from method-noise (aka residual image),[1] based on the fact that lost details of the original image still exist in method-noise. Among them, back projection is an efficient method to exploit the residual information [28]. In later work, iterative back projection is used to extract the residual information with a no-reference image quality measurement [27]. The work in [29] extends this iterative way to a novel nonlocal iterative technique for image enhancement.

Inspired by the above observations, we propose a generic denoising framework, named GPCA, to improve the performance

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author.
E-mail address: wzlusd@sz.tsinghua.edu.cn (W. Lu).

[1] Method-noise (or residual image) is often defined as the difference between the noisy image and its denoised version.

http://dx.doi.org/10.1016/j.jvcir.2017.05.009
1047-3203/

of existing methods. In GPCA, we propose a new back projection based on statistical test to extract the useful information from method-noise, followed by the PCA-based denoising to improve the performance. Our GPCA framework can be divided into two stages. The first stage produces an initial denoised image (a guide image for the second stage) via the proposed back projection. Then Pearson's correlation coefficient test (Pearson-test) is applied locally to test the independence between the denoised image and the corresponding method-noise, which guarantees that only the significantly relevant counterparts are added to the denoised image. The second stage utilizes nonlocal self-similarity, i.e., similar patches are collected for the PCA-based denoising. In the PCA transform domain, we keep image structures while removing noise by only keeping the prominent significant eigenvalues and its corresponding eigenvectors. Experimental results demonstrate that our framework is able to reduce the artifacts produced by the state-of-the-art methods.

The rest of this paper is organized as follows. The existing literature on denoising is summarized in Section 2. In Section 3, we briefly review PCA transformation. The proposed framework is elaborated in Section 4, including how to obtain the guide image, and how to implement patch grouping and PCA-based denoising. Section 5 reports the experimental results. Finally, we draw the conclusion in Section 6.

2. Related works

The following part is a rough categorization of the plethora of image denoising methods that have been developed: spatial domain, transform domain and learning-based methods [32].

Spatial domain methods usually estimate each pixel by performing a weighted average of its local or nonlocal neighborhoods, thus leading to two types of filters: local and nonlocal filters. The local filters are widely used in the early stages due to the relatively low complexity. Among this type of work, the well-known bilateral filer (BF) [11], which computes the pixel similarity based on the spatial distances and range (radiometric) distances, has received much attention for its edge preserving and noise smoothing. To date, BF and its invariants have been widely used in different contexts [33,34,12]. However, the local filters usually cannot work well at high noise levels. To overcome this drawback, the so-called nonlocal means (NLM) filter, as a representative of the nonlocal filters, extends the pixel similarity in a local region to the patch similarity in the whole image [35,13]. Essentially, NLM makes use of the structural redundancy, named self-similarity which is inherent in natural images. So far, there have been so many variants of NLM about how to speed up NLM [36,37], further improve the performance of NLM [38,39], or choose the parameters of NLM [40,41]. These nonlocal filters can remove the noise effectively but often generate over-smooth images.

On the contrary, transform domain methods perform better at preserving the details of images. These methods assume that the image can be properly represented by the orthogonal basis (e.g., wavelets, curvelets and contourlets) with a series of coefficients. The smaller coefficients correspond to the high frequency part of the input image, which are related to noise. Hence, noise can be effectively removed using different coefficient shrinkage strategies, such as BayesShrink [14], ProbShrink [15], SUREShrink [16] and BLS-GSM [17]. Such transform domain methods are popular in various applications, due to the simplicity and efficiency in preserving the main structures of images (e.g., edges). However, such methods often produce the ringing artifacts around the edges, and fail to perform well on the images with more complex characteristics.

Recently, much work focuses on learning dictionary from noisy/clean images for restoration problems [24,42,43]. To obtain a sparse representation of image patches, K-clustering with singular value decomposition (K-SVD) is proposed to train an overcomplete dictionary in [42]. In later research work, some representative dictionary learning based methods attempt to learn a structured dictionary, such as learned simultaneous sparse coding (LSSC) [43]. Although most of learning-based denoising methods have obtained competitive performance compared to the state-of-the-art methods, such learning methods are computationally expensive.

3. PCA transformation

PCA is a well-known linear dimension reduction method, which aims to find a linear projection of high dimensional data into a lower dimensional subspace. Let $\mathbf{X}$ be a sample matrix related by a linear transformation $\mathbf{P}$, i.e., $\mathbf{Z} = \mathbf{PX}$, where $\mathbf{Z}$ is the transformed coefficients. The standard PCA transformation mainly consists of four steps as follows.

(1) Obtain the sample matrix $\mathbf{X} \in R^{m\times n}$, which contains n sample vectors in total and each *m-dimension* sample vector is arranged in columns.

(2) Obtain the centralized sample matrix $\overline{\mathbf{X}}$, i.e., each entry of *i-th row* of $\mathbf{X}$ subtracts the mean value of *i-th row* of $\mathbf{X}$.

(3) Diagonalize the covariance matrix $\mathbf{C} = (1/n)\overline{\mathbf{X}}\,\overline{\mathbf{X}}^T$. The goal of PCA is to find an orthonormal transformation matrix $\mathbf{P}$ to decorelate $\overline{\mathbf{X}}$, i.e., $\overline{\mathbf{Y}} = \mathbf{P}\overline{\mathbf{X}}$, so that the covariance matrix $\mathbf{C}$ is diagonal. Since the matrix $\mathbf{C}$ is symmetrical, it can be formulated as:

$$\mathbf{C} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T, \tag{1}$$

where $\mathbf{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m]$ is the $m \times m$ orthogonal eigenvector matrix, and $\mathbf{\Lambda} = diag\{\lambda_1, \ldots, \lambda_m\}$ is the diagonal eigenvalue matrix with eigenvalues in descending order, i.e. $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. The terms $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m$ ($\boldsymbol{\phi}_i \in R^{m\times 1}$) and $\lambda_1, \ldots, \lambda_m$ are the eigenvectors and eigenvalues of $\mathbf{C}$.
Thus, it can be observed that the PCA transformation matrix can be expressed as

$$\mathbf{P} = \mathbf{\Phi}^T. \tag{2}$$

(4) Decorrelate the centralized matrix $\overline{\mathbf{X}}$ by

$$\overline{\mathbf{Z}} = \mathbf{P}\overline{\mathbf{X}}, \tag{3}$$

where $\overline{\mathbf{Z}}$ is the decorrelated dataset of $\overline{\mathbf{X}}$.

The basic idea of using PCA to denoise images is that PCA can fully decorrelate the original dataset. In other words, the original signal and noise can be distinguished in the PCA domain, since the energy of a signal will generally concentrate on a small subset of the PCA transformed dataset, while the energy of noise will generally spread over the whole dataset evenly. Therefore, the noise can be removed by keeping those significant subset of the PCA transformed dataset. Based on this observation, some related works [20,44,45] based on PCA has been successfully applied to image denoising, and achieve competitive results over the state-of-the-art methods.

4. The proposed GPCA framework

The main problem of the PCA-based denoising methods is that they learn orthogonal basis of PCA transform from the noisy image directly, thus resulting in large estimation bias, particularly at high levels of noise. Instead, in this paper, an effective denoising framework is proposed to estimate noise-free images based on

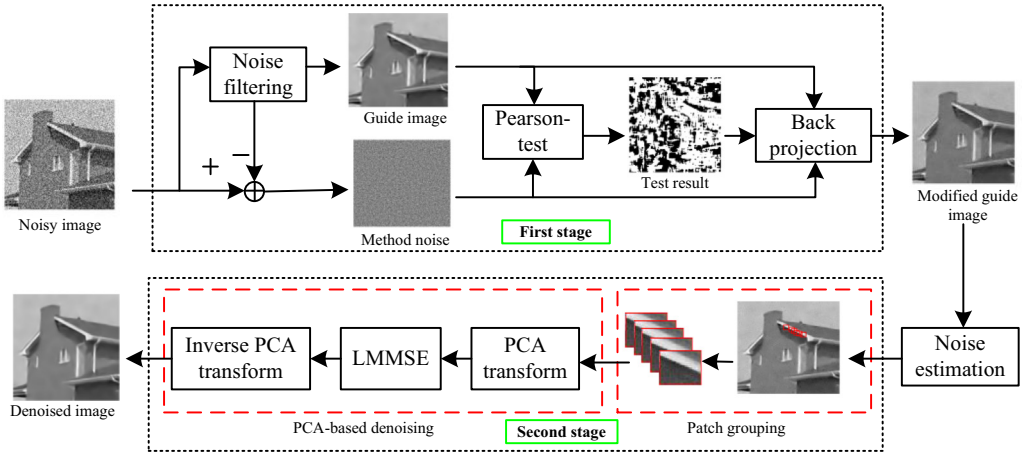


Fig. 1. Diagram of the proposed denoising framework.

back projection and guided PCA denoising. Specifically, the proposed GPCA framework can be divided into two stages, as illustrated in Fig. 1. The first stage generates a modified guide image by the proposed back projection, which uses *Pearson-test* to detect the significantly relevant counterparts between the guide image and the corresponding method-noise. The second stage collects similar image patches from the guide image, followed by the PCA-based denoising to further improve the performance. More details of our framework will be shown in the following parts.

4.1. The first stage based on back projection via statistical test

4.1.1. Noise filtering to produce an initial guide image

Let $\mathbf{F}_n$ denote a noisy image defined by

$$\mathbf{F}_n = \mathbf{F} + \mathbf{e}, \tag{4}$$

where $\mathbf{F}$ and $\mathbf{e}$ represent the clean image and additive white Gaussian noise (AWGN) with variance σ^2, respectively.

The first step in our denoising framework is to perform a noise filtering procedure to generate an initial guide image $\widehat{\mathbf{F}}$,

$$\widehat{\mathbf{F}} = \phi(\mathbf{F}_n), \tag{5}$$

where $\phi(\cdot)$ stands for a certain existing denoising filter (e.g., BF, NLM or BM3D filter).

Although the choice of denoising filter influences the final performance, this is not the core of our framework. The proposed framework concentrates on exploiting the image content from method-noise to further improve the performance of other methods. Therefore, our denoising framework is a post-processing technique for existing denoising methods.

4.2. The proposed back projection via Pearson-test

Many existing filtering methods (e.g., LPG-PCA, BM3D and DDF) are specifically designed to remove noise. However, many image details are also removed during the processing of filtering. Fortunately, the lost information still exists in method-noise. As a consequence, it is expected that the denoising performance can be further improved, if we can extract the image details from method-noise properly.

Let $\mathbf{R}$ denote the method-noise of the guide image $\widehat{\mathbf{F}}$ as follows

$$\mathbf{R} = \mathbf{F}_n - \widehat{\mathbf{F}} = \mathbf{F}_n - \phi(\mathbf{F}_n). \tag{6}$$

Many recent work [27–30,46,47] has attempted to exploit the residual information from method-noise to further improve the denoising results. Among them, back projection is a simple yet effective way to exploit the residual information [28,29]. The basic idea of back projection is to create a new noisy image $\widetilde{\mathbf{F}}_n$ by adding method-noise back to the denoised image, i.e.

$$\widetilde{\mathbf{F}}_n = \phi(\mathbf{F}_n) + \delta \cdot (\mathbf{F}_n - \phi(\mathbf{F}_n)), \tag{7}$$

where $\delta \in (0, 1)$ is a projection factor. Note that when $\delta \to 0, \widetilde{\mathbf{F}}_n = \phi(\mathbf{F}_n) = \widehat{\mathbf{F}}$; when $\delta \to 1, \widetilde{\mathbf{F}}_n = \mathbf{F}_n$.

Note that adding method-noise directly back to the denoised image can not always ensure an improvement, since method-noise contains much noise. As illustrated in Fig. 2(b), it can observed that most regions in method-noise behave like noise.

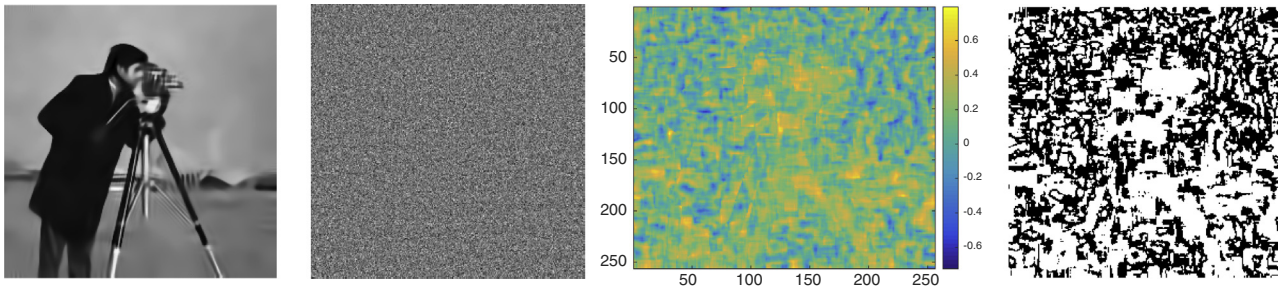


Fig. 2. (a): The denoised result of DDF for *Cameraman* at $\sigma = 50$. (b): The corresponding method-noise. (c): Pearson's correlation coefficient between denoised image and its method-noise. (The significantly relevant regions are colored in yellow.) (d): The result of the Pearson's correlation coefficient test (white: reject independence hypothesis, black: do not reject). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

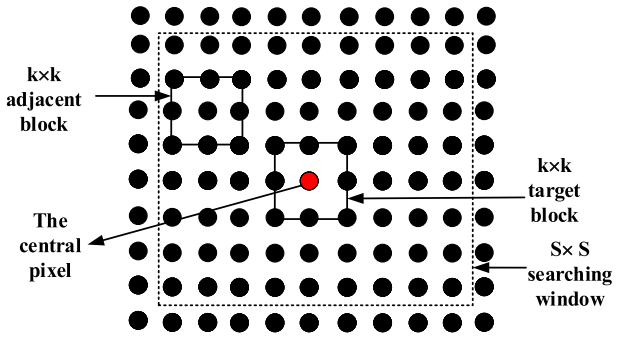


Fig. 3. The relationship between the central pixel, the target block, the adjacent block and the local searching window.

To weaken the effect of noise, we propose a new back projection via Pearson-test. To be specific, only the counterparts that have significant correlations with the guide image are added to the guide image, i.e.,

$$\widetilde{\mathbf{F}}_n = \phi(\mathbf{F}_n) + \delta \cdot \mathbf{R} * \mathbf{W} = \phi(\mathbf{F}_n) + \delta \cdot (\mathbf{F}_n - \phi(\mathbf{F}_n)) * \mathbf{W}, \tag{8}$$

where $*$ represents element-wise multiplication of two matrices; $\mathbf{W}$ is a binary mask matrix obtained by Pearson-test.

The subsequent problem is to obtain the mask matrix $\mathbf{W}$. Given L pixel-pairs (samples) (p, q) extracted from the guide image $\widehat{\mathbf{F}}$ and the corresponding method-noise $\mathbf{R}$, the biased correlation coefficient is defined by

$$r = \frac{s_{pq}}{s_p s_q}, \tag{9}$$

where s_p and s_q are the pixel standard deviation of, respectively, p and q, and s_{pq} is the sample covariance of p and q. The criterion we use is based on the value

$$t = r\sqrt{\frac{L-2}{1-r^2}}, \tag{10}$$

where the criterion t follows a Student-t distribution with $L-2$ degrees of freedom. The main merit of this test is that the joint distribution does not need to be calculated. As such, validity is achieved with smaller sample sizes.

As an example, we perform the Pearson-test between the denoised image and its method-noise using a sliding window of size 7×7. Fig. 2(c) shows the correlation coefficient between the denoised image and its method-noise, from which we can see that the denoised image and its method-noise are highly correlative in regions highlighted in yellow. In other words, method-noise contains the image structures of the original image. Fig. 2(d) shows the final result of Pearson-test, from which we can see that Pearson-test can detect the useful information from method-noise.

4.2.1. Noise estimation

As shown in [27,30], adding the method-noise directly to the guide image $\widehat{\mathbf{F}}$ as in (8) inevitably introduces additional noise. Thus, it is necessary to point out that the noise variance σ_r^2 of $\widetilde{\mathbf{F}}_n$ needs to be updated. The guide image $\widetilde{\mathbf{F}}_n$ can be denoted as $\widetilde{\mathbf{F}}_n = \mathbf{F} + \mathbf{e}_r$, where $\mathbf{F}$ is the clean image, and $\mathbf{e}_r$ is the residual in the guide image. Here we estimate σ_r based on the difference between the guide image $\widetilde{\mathbf{F}}_n$ and the noisy image $\mathbf{F}_n$:

$$\Delta = \mathbf{F}_n - \widetilde{\mathbf{F}}_n = (\mathbf{F} + \mathbf{e}) - (\mathbf{F} + \mathbf{e}_r) = \mathbf{e} - \mathbf{e}_r. \tag{11}$$

Then we have

$$\begin{aligned} E[\Delta^2] &= E[\mathbf{e}^2] + E[\mathbf{e}_r^2] - 2E[\mathbf{e} \cdot \mathbf{e}_r] \\ &= \sigma^2 + \sigma_r^2 - 2E[\mathbf{e} \cdot \mathbf{e}_r], \end{aligned} \tag{12}$$

where σ is the standard deviation of noise, and $E(\cdot)$ is the *expectation operator*.

In practice, $\mathbf{e}_r$ can be roughly regarded as the smoothed version of noise $\mathbf{e}$, which mainly contains the low frequency component of noise, and thus Δ has the main high frequency component of $\mathbf{e}$. Then we have $E[\mathbf{e} \cdot \mathbf{e}_r] = E[(\Delta + \mathbf{e}_r) \cdot \mathbf{e}_r] = E[\Delta \cdot \mathbf{e}_r] + E[\mathbf{e}_r^2]$. As shown in [20], $E[\Delta \cdot \mathbf{e}_r]$was much smaller than $E[\mathbf{e}_r^2]$, and thus $E[\Delta \cdot \mathbf{e}_r]$ can be neglected, which leads to $E[\mathbf{e} \cdot \mathbf{e}_r] \approx E[\mathbf{e}_r^2] = \sigma_r^2$. Therefore, (12) can be further formulated as

$$E[\Delta^2] = \sigma^2 + \sigma_r^2 - 2E[\mathbf{e} \cdot \mathbf{e}_r] \approx \sigma^2 + \sigma_r^2 - 2\sigma_r^2 = \sigma^2 - \sigma_r^2. \tag{13}$$

In practice, we can approximate the residual noise σ_r^2 as

$$\sigma_r = c_r\sqrt{\sigma^2 - E[\Delta^2]}, \tag{14}$$

where $c_r \in (0, 1)$ is a scaling factor determined empirically. In experiments, we have found that setting c_r around 0.55 can lead to satisfying denoising results for most of the test images.

4.3. The second stage based on guided principal component analysis

4.3.1. Patch grouping in the modified guide image

Given a guide image, the next step is to collect similar patches for PCA transform estimation. Grouping similar patches has different ways, such as block matching and K-means clustering. We adopt block matching for patch grouping due to its relative simplicity and efficiency.

Let $\mathbf{y}^c \in R^{m\times 1}(m = k^2)$ denote the central block of size $k \times k$. For convenience, $\mathbf{y}^i, i = (1, 2, \ldots, (S-k+1)^2)$ represents the candidate adjacent block of the same size $k \times k$ in the $S \times S$ searching window. The relationship between the central pixel, the target block, the adjacent block and the local searching window is shown in Fig. 3.

Since the observed image is noisy, we define

$$\mathbf{y} = \mathbf{x} + \mathbf{n}$$

as the noisy vector of $\mathbf{x}$, where $\mathbf{x} = [x_1, \ldots, x_m]^T$and $\mathbf{n} = [n_1, \ldots, n_m]^T$ represent the clean and noise counterpart, respectively.

In the $S \times S$ searching window of $\widetilde{\mathbf{F}}_n$, there are $(S-k+1)^2$ possible training blocks. The block matching is used to construct the patch group based on the similarity metric between the adjacent block and the target block, where the similarity metric can be computed using the Euclidean distance between the central block $\mathbf{y}^c$ and the adjacent block $\mathbf{y}^i$ as follows:

$$d_i = \frac{1}{m}\sum_{j=1}^{m}|y_j^c - y_j^i|^2 < T + 2\sigma^2. \tag{15}$$

By presetting a proper threshold T, and then we can select $\mathbf{y}^i$ as a sample vector, if d_i is smaller than the preset threshold. Following the practice in [20], we select n $(n \approx 5\,\mathrm{m})$ sample vectors in implementations.

Suppose we select n sample vectors including the central vector $\mathbf{y}^c$. Then the training dataset matrix $\mathbf{Y}$ is formed by

$$\mathbf{Y} = [\mathbf{y}^c, \mathbf{y}^1, \ldots, \mathbf{y}^{n-1}] = \begin{bmatrix} y_1^c & y_1^1 & \cdots & y_1^{n-1} \\ y_2^c & y_2^1 & \cdots & y_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ y_m^c & y_m^1 & \cdots & y_m^{n-1} \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix} \tag{16}$$

where $Y_i \in R^{1\times n}$ denotes the i-th *row* vector of $\mathbf{Y}$. Likewise, the clean counterpart $\mathbf{X}$ and the noise counterpart $\mathbf{N}$ are, respectively, expressed as

$$\mathbf{X} = [\mathbf{x}^c, \mathbf{x}^1, \ldots, \mathbf{x}^{n-1}] = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix}, \quad \mathbf{N} = [\mathbf{n}^c, \mathbf{n}^1, \ldots, \mathbf{n}^{n-1}] = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_m \end{bmatrix}, \tag{17}$$

where X_i and N_i denote the i-th *row* vectors of $\mathbf{X}$ and $\mathbf{N}$, respectively.

4.3.2. PCA-based denoising

After obtaining the sample matrix $\mathbf{Y} \in R^{m\times n}$, the next step is to perform PCA transform to decorrelate the dataset $\mathbf{Y}$ and remove the noise in the PCA domain. The PCA-based denoising procedure mainly consists of four steps as follows.

4.3.3. Centralize the training dataset $\mathbf{Y}$

The mean value of the row vector Y_i of the matrix $\mathbf{Y}$ is denoted by $\mu_i = (1/n)(y_i^c + \sum_{j=1}^{n-1} y_i^j)$, and then Y_i is centralized by $\overline{Y}_i = Y_i - \mu_i$. Since noise $\mathbf{N}$ is zero-mean, and $\mathbf{X}_i$ can be centralized by $\overline{X}_i = X_i - \mu_i$. Hence the centralized $\mathbf{Y}$ and $\mathbf{X}$ are described as

$$\overline{\mathbf{Y}} = \begin{bmatrix} \overline{Y}_1 \\ \overline{Y}_2 \\ \vdots \\ \overline{Y}_m \end{bmatrix}, \quad \overline{\mathbf{X}} = \begin{bmatrix} \overline{X}_1 \\ \overline{X}_2 \\ \vdots \\ \overline{X}_m \end{bmatrix}. \tag{18}$$

Thus the centralized sample matrix $\overline{\mathbf{Y}}$ can be formulated as

$$\overline{\mathbf{Y}} = \overline{\mathbf{X}} + \mathbf{N}. \tag{19}$$

Fig. 4. All experimental test images, which present a wide range of edges, texture and details.

Table 1
PSNR (dB) results of the proposed method on various images with different patch sizes and noise levels.

	Barbara				House			
σ	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9
$\sigma=10$	34.79	34.98	**35.05**	35.04	36.50	36.70	36.77	**36.80**
$\sigma=30$	29.64	29.94	**30.05**	30.04	31.73	32.00	**32.02**	**32.02**
$\sigma=50$	26.91	27.30	**27.50**	27.47	28.98	29.44	**29.51**	29.50
$\sigma=100$	23.23	23.75	**23.98**	23.96	24.98	25.71	**25.88**	25.85
	Pepper				Cameraman			
σ	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9
$\sigma=10$	34.72	**34.76**	34.74	24.71	34.26	**34.29**	34.27	34.25
$\sigma=30$	29.40	29.44	**29.50**	29.38	28.59	28.60	**28.61**	28.59
$\sigma=50$	26.79	26.93	**27.02**	26.91	26.15	26.30	**26.45**	26.29
$\sigma=100$	22.96	23.30	**23.54**	23.37	22.53	22.84	**23.12**	22.96

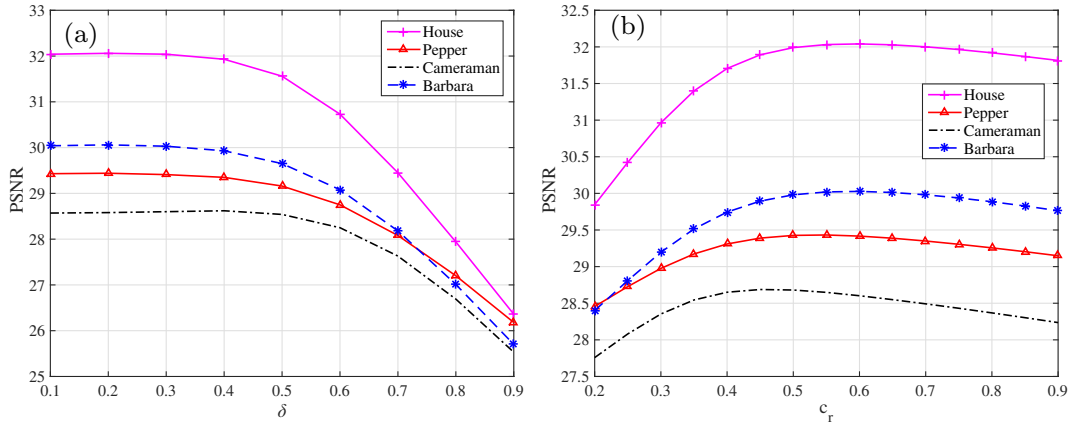


Fig. 5. PSNR results on various images ($\sigma = 30$) as a function of (a) varying δ with patch size 7×7 and $c_r = 0.55$, (b) varying c_r with patch size 7×7 and $\delta = 0.3$.

Table 2
Denoising results (PSNR and MSSIM) of various methods on natural images contaminated with $\sigma = 10$ and 30. The PSNR and MSSIM values with positive gains are highlighted in bold.

Image	BF-GPCA	BM3D-GPCA	LPG-GPCA	SAIST-GPCA	DDF-GPCA	PCLR-GPCA	PGPD-GPCA	DNN-GPCA
				$\sigma = 10$				
Lena	**34.12(+0.96)**	**36.13(+0.20)**	35.63(−0.09)	**35.91(+0.03)**	**36.01 (+0.20)**	**35.99(+0.04)**	**35.91(+0.13)**	36.13(−0.01)
	.9598(+.0072)	**.9707(+.0017)**	.9655(−.0017)	.9684(−.0001)	**.9701(+.0008)**	**.9697(+.0013)**	**.9690(+.0031)**	**.9708(+.0002)**
Barbara	**32.47(+1.33)**	**35.29(+0.31)**	34.88(−0.14)	**35.28(+0.09)**	**35.05(+0.41)**	**35.16(+0.08)**	**34.98(+0.23)**	**34.71(+0.17)**
	.9637(+.0093)	**.9778(+.0012)**	.9733(−.0019)	.9758(−.0003)	**.9769(+.0001)**	**.9770(+.0006)**	**.9763(+.0019)**	**.9762(+.0004)**
Boat	**32.49(+0.88)**	**34.01(+0.09)**	33.39(−0.23)	33.88(−0.01)	**33.89(+0.18)**	**34.04(+0.01)**	**33.79(+0.06)**	34.03(−0.02)
	.9473(+.0079)	**.9673(+.0012)**	.9591(−.0025)	.9656(−.0004)	**.9667(+.0006)**	**.9674(+.0003)**	**.9661(+.0019)**	.9687(−.0001)
Peppers	**33.37(+0.92)**	**34.89(+0.21)**	33.83(−0.23)	34.78(−0.01)	**34.74(+0.19)**	34.83(−0.05)	34.50(+0)	34.88(−0.13)
	.9154(+.0247)	.9283(−.0014)	.9189(−.0027)	.9261(−.0018)	**.9250(+.0015)**	.9272(−.0010)	**.9241(+.0011)**	.9285(−.0018)
Cameraman	**33.26(+0.80)**	**34.47(+0.29)**	33.43(−0.17)	**34.28(+0.05)**	**34.28(+0.32)**	**34.36(+0.02)**	**34.09(+0.09)**	34.54(−0.02)
	.9076(+.0400)	**.9329(+.0035)**	.9197(−.0044)	.9316 (−.0018)	**.9318(+.0035)**	**.9325(+.0002)**	**.9289(+.0050)**	.9351(−.0009)
House	**34.34(+0.87)**	**36.89(+0.18)**	35.96 (−0.26)	**36.57(+0.02)**	**36.77(+0.34)**	**36.75(+0.05)**	**36.59(+0.16)**	**36.56(+0.12)**
	.8931(+.0249)	**.9234(+.0007)**	**.9164(+.0034)**	.9134(−.0033)	**.9235(+.0019)**	**.9206(+.0005)**	**.9157(+.0045)**	**.9122(+.0023)**
Fingerprint	**30.28(+1.19)**	**32.86(+0.40)**	32.15(−0.47)	**32.70(+0.02)**	**32.41(+0.58)**	**32.65(+0.01)**	32.57(−0.01)	32.51(−0.10)
	.9849(+.0011)	**.9909(+.0006)**	.9881(−.0021)	.9905(−.0001)	**.9903(+.0001)**	**.9907(+.0001)**	**.9905(+.0001)**	.9906(+0)
Couple	**32.41(+1.04)**	**34.17(+0.13)**	33.30 (−0.26)	33.90 (−0.01)	**33.96(+0.12)**	34.07(−0.02)	33.96(+0)	34.23(−0.04)
	.9538(+.0063)	**.9679(+.0006)**	.9562 (−.0048)	.9632 (−.0005)	**.9661(+.0007)**	**.9669(+.0002)**	**.9665(+.0006)**	.9686(−.0001)
Hill	**32.50(+0.85)**	**33.63(+0.01)**	33.14 (−0.20)	**33.65(+0.01)**	33.59(+0.05)	33.67(−0.03)	**33.54(+0.02)**	33.79(−0.04)
	.9360(+.0061)	**.9576(+.0002)**	.9463 (−.0052)	.9555 (−.0032)	**.9571(+.0005)**	**.9581(+.0001)**	**.9563(+.0008)**	.9596(−.0003)
Man	**32.85(+0.98)**	**34.13(+0.15)**	33.39 (−0.24)	34.05 (−0.03)	**34.10(+0.12)**	34.13(−0.02)	**33.93(+0.01)**	34.31(−0.08)
	.9473(+.0081)	**.9647(+.0011)**	.9547 (−.0040)	.9618 (−.0012)	**.9669(+.0031)**	.9645(−.0001)	**.9635(+.0011)**	.9660(−.0004)
Montage	**35.88(+1.62)**	**37.44(+0.09)**	36.38 (−0.15)	**37.49(+0.05)**	**37.82(+0.39)**	**37.56(+0.18)**	**37.14(+0.39)**	**37.60(+0.06)**
	.9541(+.0408)	.9637 (−.0021)	**.9658(+.0022)**	**.9688(+.0007)**	**.9692(+.0034)**	**.9687(+.0020)**	**.9657(+.0085)**	**.9699(+.0012)**
Average	**33.08(+1.04)**	**34.90(+0.19)**	34.13 (−0.22)	**34.77(+0.02)**	**34.78(+0.27)**	**34.83(+0.02)**	**34.63(+0.09)**	**34.40(+0.01)**
	.9421(+.0160)	**.9587(+.0007)**	.9513 (−.0022)	.9564 (−.0008)	**.9585(+.0015)**	**.9585(+.0003)**	**.9566(+.0026)**	.9587(+0)
				$\sigma = 30$				
Lena	**28.94(+1.04)**	**31.44(+0.18)**	**30.80(+0.13)**	**31.39(+0.07)**	**31.50(+0.15)**	**31.48(+0.14)**	**31.40(+0.12)**	**31.70(+0.12)**
	.8752(+.0262)	**.9188(+.0069)**	.9056 (−.0008)	**.9167(+.0010)**	**.9214(+.0023)**	**.9201(+.0038)**	**.9170(+.0046)**	**.9242(+.0024)**
Barbara	**25.67(+0.78)**	**29.88(+0.07)**	**29.29(+0.19)**	**30.14(+0.06)**	**30.05(+0.21)**	**29.80(+0.20)**	**29.55(+0.20)**	**29.24(+0.40)**
	.8506(+.0245)	**.9303(+.0032)**	**.9160(+.0013)**	**.9310(+.0006)**	**.9322(+.0027)**	**.9284(+.0034)**	**.9237(+.0043)**	**.9220(+.0055)**
Boat	**26.60(+0.65)**	**29.29(+0.17)**	**28.37(+0.10)**	28.92 (+0)	**29.04(+0.11)**	**29.18(+0.01)**	**28.98(+0.01)**	**29.38(+0.05)**
	.8243(+.0208)	**.8933(+.0063)**	.8532 (−.0064)	.8719 (−.0008)	**.8833(+.0019)**	**.8878(+.0003)**	**.8800(+.0006)**	.8939(−.0002)
Peppers	**26.45(+0.63)**	**29.45(+0.17)**	**28.61**(+0.15)	**29.34**(+0.01)	**29.50**(+0.11)	29.54(−0.03)	**29.34(+0.02)**	29.82(−0.03)
	.6898(+.0139)	.8427 (−.0096)	**.8395(+.0045)**	**.8545(+.0016)**	**.8545**(+.0093)	**.8577(+.0009)**	**.8572(+.0043)**	**.8634(+.0016)**
Cameraman	**26.70(+1.07)**	**28.83**(+0.19)	**27.91**(+0.10)	**28.31**(+0.01)	**28.61**(+0.09)	28.68(−0.02)	28.38(−0.02)	29.08(−0.05)
	.7140(+.0561)	**.8424(+.0122)**	**.8177(+.0028)**	.8222 (−.0012)	**.8304**(+.0111)	**.8366(+.0018)**	**.8240(+.0027)**	.8505(−.0007)
House	**28.94(+1.64)**	**32.35**(+0.26)	**31.41**(+0.22)	**32.24**(+0.06)	**32.02(+0.25)**	**32.23(+0.10)**	**32.20(+0.04)**	**32.39(+0.20)**
	.7960(+.1071)	.8502(+.0009)	**.8415(+.0026)**	**.8508(+.0009)**	**.8480(+.0094)**	**.8513(+.0030)**	**.8494(+.0039)**	**.8528(+.0042)**
Fingerprint	**23.49(+1.30)**	**27.05**(+0.22)	26.16 (−0.11)	26.93 (−0.02)	**26.54(+0.11)**	26.90(+0)	**26.82(+0.03)**	**26.66(+0.08)**
	.8794(+.0273)	**.9528(+.0032)**	.9254 (−.0086)	**.9486(+.0002)**	.9429 (−.0004)	**.9493(+.0001)**	**.9467(+.0010)**	**.9490(+.0004)**
Couple	**25.97(+0.42)**	**29.08**(+0.21)	**28.07**(+0.15)	**28.68**(+0.01)	28.74(+0.12)	28.89(+0)	28.75(−0.01)	**29.18(+0.01)**
	.8185(+.0183)	**.8967(+.0056)**	.8556 (−.0034)	**.8757(+.0004)**	**.8827**(+.0019)	**.8873(+.0002)**	.8832(−.0001)	.8947(−.0006)
Hill	**27.32(+0.37)**	**29.31**(+0.16)	**28.50**(+0.10)	29.01 (−0.03)	**29.05**(+0.13)	**29.10(+0.02)**	29.03(−0.01)	**29.27(+0.02)**
	.8189(+.0165)	**.8695(+.0052)**	.8297 (−.0051)	.8505 (−.0016)	**.8532**(+.0019)	.8596(−.0004)	.8573(−.0008)	.8658(−.0018)
Man	**27.05(+0.42)**	**29.03**(+0.17)	**28.28**(+0.07)	28.72 (−0.03)	**28.84**(+0.11)	28.94(+0)	28.77(−0.03)	29.20(−0.04)
	.8289(+.0151)	**.8809(+.0058)**	.8420 (−.0080)	.8588 (−.0015)	**.8677**(+.0014)	.8729(−.0002)	.8670(−.0008)	.8798(−.0021)
Montage	**28.51(+1.15)**	**31.71**(+0.34)	**29.91**(+0.14)	**31.07**(+0.12)	**31.70**(+0.29)	**31.29(+0.09)**	**30.92(+0.25)**	**31.79(+0.05)**
	.7948(+.0740)	**.9176(+.0100)**	**.9100(+.0116)**	**.9221(+.0046)**	.9180(+.0132)	**.9210(+.0077)**	**.9154(+.0136)**	**.9275(+.0059)**
Average	**26.87(+0.86)**	**29.76(+0.19)**	**28.84(+0.11)**	**29.52(+0.03)**	**29.59(+0.15)**	**29.63(+0.04)**	**29.46(+0.05)**	**29.79(+0.07)**
	.8082(+.0363)	**.8905(+.0045)**	.8669 (−.0009)	**.8821(+.0004)**	**.8849(+.0049)**	**.8884(+.0019)**	**.8837(+.0030)**	**.8931(+.0013)**

4.3.4. Diagonalize the covariance matrix of $\overline{\mathbf{Y}}$

For ease of presentation, we denote the covariance matrix of $\overline{\mathbf{X}}, \overline{\mathbf{Y}}$ and $\mathbf{N}$ by $\mathbf{C}_{\overline{\mathbf{X}}}, \mathbf{C}_{\overline{\mathbf{Y}}}$ and $\mathbf{C}_{\mathbf{N}}$, respectively. As Section 3 describes, the PCA transformation matrix $\mathbf{P}_{\overline{\mathbf{X}}}$ can be obtained by diagonalizing the covariance matrix $\mathbf{C}_{\overline{\mathbf{X}}}$. Since the dataset $\overline{\mathbf{X}}$ is unknown, we cannot directly compute $\mathbf{C}_{\overline{\mathbf{X}}}$. However, it can be proved that the PCA transformation matrix $\mathbf{P}_{\overline{\mathbf{X}}}$ associated with $\mathbf{C}_{\overline{\mathbf{X}}}$ is the same as the PCA transformation matrix associated with $\mathbf{C}_{\overline{\mathbf{Y}}}$ (shown in (23)). Then the PCA transformation matrix $\mathbf{P}_{\overline{\mathbf{X}}}$ can be estimated by computing the covariance matrix $\mathbf{C}_{\overline{\mathbf{Y}}}$ of $\overline{\mathbf{Y}}$. With linear model (19), $\mathbf{C}_{\overline{\mathbf{Y}}}$ can be expressed as[2]

$$\begin{aligned}\mathbf{C}_{\overline{\mathbf{Y}}} &= 1/n(\overline{\mathbf{X}}+\mathbf{N})(\overline{\mathbf{X}}+\mathbf{N})^T \\ &\approx 1/n\left(\overline{\mathbf{X}}\,\overline{\mathbf{X}}^T + \mathbf{N}\,\mathbf{N}^T\right) \\ &= \mathbf{C}_{\overline{\mathbf{X}}} + \mathbf{C}_{\mathbf{N}},\end{aligned} \tag{20}$$

where $\mathbf{C}_{\overline{\mathbf{X}}} = (1/n)\overline{\mathbf{X}}\,\overline{\mathbf{X}}^T$ and $\mathbf{C}_{\mathbf{N}} = (1/n)\mathbf{N}\mathbf{N}^T$.

Since $\mathbf{n}^i$ and $\mathbf{n}^j$ is uncorrelated (when $i \neq j$), we can see that $\mathbf{C}_{\mathbf{N}} \in R^{m\times m}$ is a diagonal matrix, which has all the diagonal components being σ^2. According to PCA transformation, $\mathbf{C}_{\overline{\mathbf{X}}}$ can be decomposed as

$$\mathbf{C}_{\overline{\mathbf{X}}} = \mathbf{\Phi}_{\overline{\mathbf{X}}}\mathbf{\Lambda}_{\overline{\mathbf{X}}}\mathbf{\Phi}_{\overline{\mathbf{X}}}^T, \tag{21}$$

where $\mathbf{\Phi}_{\overline{\mathbf{X}}}$ is the $m \times m$ orthogonal eigenvector matrix, i.e. $\mathbf{\Phi}_{\overline{\mathbf{X}}}\mathbf{\Phi}_{\overline{\mathbf{X}}}^T = \mathbf{I}$, where $\mathbf{I}$ is an $m \times m$ identity matrix; $\mathbf{\Lambda}_{\overline{\mathbf{X}}} = diag\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ is the diagonal eigenvalue matrix with $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. Then $\mathbf{C}_{\mathbf{N}}$ can be reformulated as

[2] Since dataset $\overline{\mathbf{X}}$ is uncorrelated with noise $\mathbf{N}$.

$$\mathbf{C}_{\mathbf{N}} = \sigma^2\mathbf{I} = \mathbf{\Phi}_{\overline{\mathbf{X}}}\mathbf{\Phi}_{\overline{\mathbf{X}}}^T(\sigma^2\mathbf{I}) = \mathbf{\Phi}_{\overline{\mathbf{X}}}(\sigma^2\mathbf{I})\mathbf{\Phi}_{\overline{\mathbf{X}}}^T = \mathbf{\Phi}_{\overline{\mathbf{X}}}(\mathbf{C}_{\mathbf{N}})\mathbf{\Phi}_{\overline{\mathbf{X}}}^T. \tag{22}$$

Based on (20)–(22), we have

$$\begin{aligned}\mathbf{C}_{\overline{\mathbf{Y}}} &\approx \mathbf{C}_{\overline{\mathbf{X}}} + \mathbf{C}_{\overline{\mathbf{N}}} \\ &= \mathbf{\Phi}_{\overline{\mathbf{X}}}\mathbf{\Lambda}_{\overline{\mathbf{X}}}\mathbf{\Phi}_{\overline{\mathbf{X}}}^T + \mathbf{\Phi}_{\overline{\mathbf{X}}}(\sigma^2\mathbf{I})\mathbf{\Phi}_{\overline{\mathbf{X}}}^T \\ &= \mathbf{\Phi}_{\overline{\mathbf{X}}}(\mathbf{\Lambda}_{\overline{\mathbf{X}}} + \sigma^2\mathbf{I})\mathbf{\Phi}_{\overline{\mathbf{X}}}^T,\end{aligned} \tag{23}$$

which implies that $\mathbf{C}_{\overline{\mathbf{Y}}}$ and $\mathbf{C}_{\overline{\mathbf{X}}}$ contain the same orthogonal eigenvector matrix $\mathbf{\Phi}_{\overline{\mathbf{X}}}$. Since $\mathbf{C}_{\overline{\mathbf{X}}}$ is not available, we can directly estimate $\mathbf{\Phi}_{\overline{\mathbf{X}}}$ by decomposing $\mathbf{C}_{\overline{\mathbf{Y}}}$. According to Section 3, the orthogonal PCA transformation matrix of $\overline{\mathbf{X}}$ is obtained by setting

$$\mathbf{P}_{\overline{\mathbf{X}}} = \mathbf{\Phi}_{\overline{\mathbf{X}}}^T. \tag{24}$$

Using $\mathbf{P}_{\overline{\mathbf{X}}}$, the sample matrix $\overline{\mathbf{Y}}$ can be decorrelated by the PCA transform

$$\begin{aligned}\overline{\mathbf{Z}}_{\mathbf{N}} &= \mathbf{P}_{\overline{\mathbf{X}}}\overline{\mathbf{Y}} = \mathbf{P}_{\overline{\mathbf{X}}}\overline{\mathbf{X}} + \mathbf{P}_{\overline{\mathbf{X}}}\mathbf{N} \\ &= \overline{\mathbf{Z}} + \mathbf{N}_{\mathbf{z}},\end{aligned} \tag{25}$$

where $\overline{\mathbf{Z}} = \mathbf{P}_{\overline{\mathbf{X}}}\overline{\mathbf{X}}$ is the decorrelated dataset for $\overline{\mathbf{X}}$, and $\mathbf{N}_{\mathbf{z}} = \mathbf{P}_{\overline{\mathbf{X}}}\mathbf{N}$is the transformed noise dataset for $\mathbf{N}$. Since $\overline{\mathbf{Z}}$ and $\mathbf{N}_{\mathbf{z}}$ are uncorrelated, the covariance matrix of $\overline{\mathbf{Y}}$ can be computed by

$$\begin{aligned}\mathbf{C}_{\overline{\mathbf{Z}}_{\mathbf{N}}} &= (1/n)\ \overline{\mathbf{Z}}_{\mathbf{N}}\overline{\mathbf{Z}}_{\mathbf{N}}^T \\ &= (1/n)(\mathbf{P}_{\overline{\mathbf{X}}}\overline{\mathbf{Y}})(\mathbf{P}_{\overline{\mathbf{X}}}\overline{\mathbf{Y}})^T \\ &= \mathbf{P}_{\overline{\mathbf{X}}}((1/n)\overline{\mathbf{Y}}\ \overline{\mathbf{Y}}^T)\mathbf{P}_{\overline{\mathbf{X}}}^T \approx \mathbf{P}_{\overline{\mathbf{X}}}(\mathbf{C}_{\overline{\mathbf{X}}} + \mathbf{C}_{\mathbf{N}})\mathbf{P}_{\overline{\mathbf{X}}}^T \\ &= \mathbf{C}_{\overline{\mathbf{Z}}} + \mathbf{C}_{\mathbf{N}_{\mathbf{z}}},\end{aligned} \tag{26}$$

where $\mathbf{C}_{\overline{\mathbf{Z}}} = \mathbf{P}_{\overline{\mathbf{X}}}\mathbf{C}_{\overline{\mathbf{X}}}\mathbf{P}_{\overline{\mathbf{X}}}^T$and $\mathbf{C}_{\mathbf{N}_{\mathbf{z}}} = \mathbf{P}_{\overline{\mathbf{X}}}\mathbf{C}_{\mathbf{N}}\mathbf{P}_{\overline{\mathbf{X}}}^T$. Note that $\mathbf{C}_{\overline{\mathbf{Z}}}$ and $\mathbf{C}_{\mathbf{N}_{\mathbf{z}}}$ are the covariance matrix of decorrelated dataset $\overline{\mathbf{Z}}$ and $\mathbf{N}$, respectively. Based on (23), $\mathbf{C}_{\overline{\mathbf{Z}}}$ and $\mathbf{C}_{\mathbf{N}_{\mathbf{z}}}$ are diagonal matrices.

4.3.5. Denoising in the PCA transform domain

After PCA transform for the sample matrix $\overline{\mathbf{Y}}$, noise reduction is then converted to the PCA transform domain $\overline{\mathbf{Z}}_{\mathbf{N}}$. In general, most energy of the original signals will concentrate on the several most important components, while the energy of noise will spread much more evenly. Thus, the noise can be removed by using the linear minimum mean square-error estimation (LMMSE) estimator or hard-thresholding shrinkage in $\overline{\mathbf{Z}}_{\mathbf{N}}$. Here we adopt LMMSE estimator due to its simplicity and efficiency.

Let $\overline{Z}_n^i$ denote the *i-th row* vector of $\overline{\mathbf{Z}}_{\mathbf{N}}$. With LMMSE, the denoised result of $\overline{Z}_n^i$ can be formulated as

$$\widehat{\overline{Z}}_n^i = \widehat{W}_i \cdot \overline{Z}_n^i, \tag{27}$$

where the shrinkage operator $\widehat{W}_i = \frac{\mathbf{C}_{\overline{\mathbf{Z}}}(i,i)}{\mathbf{C}_{\overline{\mathbf{Z}}}(i,i)+\mathbf{C}_{\mathbf{N}_{\mathbf{z}}}(i,i)}$. In practice, $\mathbf{C}_{\overline{\mathbf{Z}}}(i,i)$ is much smaller than $\mathbf{C}_{\mathbf{N}_{\mathbf{z}}}(i,i)$ in flat zones, so that W_i is near 0. In other words, most of noise will be removed by LMMSE. Since $\mathbf{C}_{\overline{\mathbf{Z}}}(i,i)$ is not available, based on (26), $\mathbf{C}_{\overline{\mathbf{Z}}}(i,i)$ can be estimated by

$$\mathbf{C}_{\overline{\mathbf{Z}}}(i,i) = \mathbf{C}_{\overline{\mathbf{Z}}_{\mathbf{N}}}(i,i) - \mathbf{C}_{\mathbf{N}_{\mathbf{z}}}(i,i). \tag{28}$$

4.3.6. Inverse PCA transform and aggregation

Let $\widehat{\overline{\mathbf{Z}}}_{\mathbf{N}}$ denote the shrunk transformed matrix, and $\widehat{\overline{\mathbf{Y}}}$ denote the denoised result of $\overline{\mathbf{Y}}$. Then $\widehat{\overline{\mathbf{Y}}}$ can be obtained by inversing the PCA transform as

$$\widehat{\overline{\mathbf{Y}}} = \mathbf{P}_{\overline{\mathbf{X}}}^{-1}\ \widehat{\overline{\mathbf{Z}}}_{\mathbf{N}} = \mathbf{P}_{\overline{\mathbf{X}}}^T\ \widehat{\overline{\mathbf{Z}}}_{\mathbf{N}}. \tag{29}$$

Adding the mean value μ_i $(i = 1, \ldots, m)$ to the *i-th row* of $\widehat{\overline{\mathbf{Y}}}$ produces the denoised result of $\mathbf{Y}$, i.e., $\widehat{\mathbf{Y}} = [\hat{\mathbf{y}}^c, \hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^{n-1}]$. The final denoised result of the central block $\mathbf{y}^c$ is the weighted average of most similar l patches, i.e.

$$\hat{\mathbf{y}}_f^c = \alpha^c\hat{\mathbf{y}}^c + \sum_{i=1}^{l}\alpha^i\hat{\mathbf{y}}^i, \quad s.t.\ \alpha^c + \sum_{i=1}^{l}\alpha^i = 1, \tag{30}$$

where we set $l = 3$ empirically; α^c and α^i control the weights, which can be obtained by any other kernel function (e.g., Gaussian kernel), or by linear regression. In implementation, we use the average weight due to its simplicity, i.e., $\alpha^c = \alpha^1 = \cdots = \alpha^l$. Applying such procedures to each pixel, and then the whole image can be denoised.

In summary, the complete procedure of the proposed framework is sketched in the following Algorithm 1.

Algorithm 1. The proposed denoising framework

Input: Noisy image $\mathbf{F}_n$ with noise variance σ^2
Output: Denoised image $\widehat{\mathbf{F}}_n$

1: $\phi \leftarrow$ Choose a certain denoising filter, such as BM3D;
2: $\widehat{\mathbf{F}} \leftarrow$ Use ϕ to obtain a guide image via (5);
3: $\mathbf{R} \leftarrow$ Obtain method-noise using (6);
4: $\widetilde{\mathbf{F}}_n \leftarrow$ Generate a new guide image using the back projection based on Pearson-test by (8);
5: $\sigma_r \leftarrow$ Estimate the noise level of $\widetilde{\mathbf{F}}_n$ using (14);
6: **for**each pixel in $\widetilde{\mathbf{F}}_n$**do**
7: $\quad\overline{\mathbf{Y}} \leftarrow$ Obtain the training sample by selecting similar patches from $\widetilde{\mathbf{F}}_n$ via (15) and then centralize it;
8: $\quad\mathbf{P}_{\overline{\mathbf{X}}} \leftarrow$ Estimate the PCA transformation matrix via (24);
9: $\quad\widehat{\overline{\mathbf{Z}}}_{\mathbf{N}} \leftarrow$ Shrink the PCA transformed coefficients using LMMSE via (27);
10: $\quad\widehat{\overline{\mathbf{Y}}} \leftarrow$ Obtain the denoised result of $\overline{\mathbf{Y}}$ by inversing the PCA transform via (29);
11: $\quad\hat{\mathbf{y}}_f^c \leftarrow$ Obtain the final denoised result of the target block via (30).
12: **end for**
13: $\widehat{\mathbf{F}}_n \leftarrow$ Return the denoised image by weighted average.

5. Experiments

5.1. Parameter setup

To test the performance of the proposed framework comprehensively, all the experiments are performed on multiple test images from the standard image databases,[3] which are commonly used to evaluate the state-of-the-art denoising methods and are shown in Fig. 4. Each image is contaminated with AWGN at $\sigma \in [10, 30, 50, 100]$, ranging from low to high noise levels, and the intensity value for each pixel of the images ranges from 0 to 255.

Our GPCA framework can be seen as a post-processing technique for other denoising algorithms. In experiments, 8 representative denoising algorithms are used to verify the effectiveness of our framework, including spatial domain methods: BF [11], LPG-PCA [20] and SAIST [21], transform domain methods: DDF [19] and BM3D [18], and learning-based methods: PCLR [23], PGPD [24] and DNN [48]. All the parameters are consistent with the default parameters settings suggested by the respective authors. In our experiments, we empirically set $S = 41, T = 25, k = 7, \delta = 0.3$ and

[3] http://www.cs.tut.fi/foi/GCF-BM3D/index.html#ref_software.

Table 3
Denoising results (PSNR and MSSIM) of various methods on natural images contaminated with $\sigma = 50$ and 100. The PSNR and MSSIM values with positive gains are highlighted in bold.

Image	BF-GPCA	BM3D-GPCA	LPG-GPCA	SAIST-GPCA	DDF-GPCA	PCLR-GPCA	PGPD-GPCA	DNN-GPCA
				$\sigma = 50$				
Lena	**27.00(+0.95)**	**29.30**(+0.25)	**28.41**(+0.21)	**29.11**(+0.04)	**29.20**(+0.30)	**29.28(+0.20)**	**29.20(+0.08)**	**29.56(+0.18)**
	.8144(+.0419)	**.8780(+.0112)**	**.8559(+.0026)**	**.8722(+.0006)**	**.8774**(+.0049)	**.8779(+.0075)**	**.8725(+.0049)**	**.8832(+.0035)**
Barbara	**23.36(+0.28)**	**27.42**(+0.20)	**26.45**(+0.24)	**27.53**(+0.01)	**27.50**(+0.19)	**27.28(+0.18)**	**27.02(+0.15)**	**26.62(+0.46)**
	.7468(+.0136)	**.8804(+.0083)**	**.8490(+.0026)**	.8768 (−.0006)	**.8821**(+.0049)	**.8820(+.0052)**	**.8674(+.0048)**	**.8638(+.0094)**
Boat	**24.89(+0.85)**	**26.93**(+0.15)	**26.00**(+0.15)	26.52 (−0.02)	**26.71**(+0.16)	**26.95(+0.02)**	26.72(−0.04)	**27.16(+0.01)**
	.7481(+.0384)	**.8268(+.0093)**	.7662 (−.0071)	.7863 (−.0006)	**.8048**(+.0033)	**.8186(+.0002)**	.8046(−.0026)	.8234(−.0027)
Peppers	**24.11(+1.17)**	**27.00**(+0.32)	**26.04(+0.26)**	**26.91**(+0.02)	**27.02**(+0.20)	**27.13(+0.01)**	**26.95(+0.05)**	**27.3(+0.04)**
	.6829(+.0822)	**.7996(+.0061)**	**.7843(+.0158)**	**.8036(+.0040)**	.8019(+.0197)	**.8072(+.0014)**	**.7981(+.0067)**	**.8116(+.0055)**
Cameraman	**23.83(+1.52)**	**26.33**(+0.21)	**25.71**(+0.29)	**26.12**(+0.02)	**26.45**(+0.23)	26.53(+0)	26.30(−0.02)	26.92(−0.06)
	.6310(+.1482)	**.7823(+.0069)**	**.7716(+.0204)**	**.7795(+.0049)**	**.7835**(+.0232)	**.7905(+.0022)**	**.7756(+.0005)**	.7981(−.0025)
House	**26.59(+1.41)**	**30.05**(+0.36)	**28.90**(+0.41)	**30.13**(+0.04)	**29.51(+0.30)**	**29.96(+0.20)**	**29.89(+0.05)**	**30.19(+0.28)**
	.7488(+.1324)	**.8215(+.0075)**	**.8039(+.0191)**	**.8259(+.0020)**	.8143(+.0225)	**.8200(+.0030)**	**.8115(+.0020)**	**.8200(+.0050)**
Fingerprint	**21.01(+0.89)**	24.39 (−0.14)	23.47 (−0.05)	24.47 (−0.04)	**24.12(+0.16)**	**24.40(+0.03)**	24.34(+0)	**24.26(+0.17)**
	.7854(+.0343)	**.9084(+.0052)**	.8443 (−.0167)	**.8936(+.0001)**	**.8837(+.0014)**	.9001(−.0002)	**.8926(+.0006)**	**.8982(+.0008)**
Couple	**24.41(+0.56)**	**26.63**(+0.17)	**25.58**(+0.20)	26.24 (−0.03)	**26.36**(+0.17)	**26.56(+0.02)**	26.38(−0.02)	**26.87(+0.01)**
	.7229(+.0288)	**.8245(+.0085)**	.7604 (−.0014)	.7941 (−.0006)	**.8033**(+.0042)	.8145(−.0002)	.8040(−.0012)	.8260(−.0028)
Hill	**25.87(+0.35)**	**27.31**(+0.12)	**26.39**(+0.12)	26.97 (−0.04)	**27.05**(+0.15)	**27.20(+0.06)**	27.10(−0.02)	**27.40(+0.03)**
	.7421(+.0186)	**.7971(+.0060)**	.7372 (−.0057)	.7678 (−.0013)	.7754(+.0045)	.7870(−.0004)	.7797(−.0016)	.7943(−.0025)
Man	**25.44(+0.54)**	**26.93**(+0.12)	**26.20**(+0.16)	26.63 (−0.04)	**26.81**(+0.17)	**26.92(+0.01)**	26.77(−0.04)	27.16(−0.01)
	.7540(+.0224)	**.8102(+.0062)**	.7642 (−.0037)	.7806 (−.0031)	**.7922**(+.0043)	**.8016(+.0001)**	.7941(−.0019)	.8073(−.0027)
Montage	**24.44(+1.28)**	**28.29**(+0.39)	**26.90**(+0.36)	**27.99**(+0.16)	**28.50**(+0.27)	**28.21(+0.09)**	**27.98(+0.15)**	28.78(−0.06)
	.6847(+.1152)	**.8765(+.0192)**	**.8640(+.0273)**	**.8819(+.0105)**	**.8723**(+.0260)	**.8806(+.0091)**	**.8692(+.0019)**	**.8871(+.0082)**
Average	**24.63(+0.89)**	**27.32(+0.19)**	**26.36(+0.21)**	**27.14(+0.01)**	**27.20(+0.21)**	**27.31(+0.07)**	**27.15(+0.03)**	**27.47(+0.09)**
	.7328(+.0615)	**.8368(+.0085)**	**.8001(+.0048)**	**.8238(+.0015)**	**.8264(+.0108)**	**.8345(+.0025)**	**.8245(+.0013)**	**.8375(+.0017)**
				$\sigma = 100$				
Lena	**23.19(+0.55)**	**26.36(+0.41)**	**25.23(+0.39)**	**26.05(+0.05)**	**26.01**(+0.30)	**26.25(+0.18)**	**26.09(+0.05)**	**26.46(+0.14)**
	.6917(+.0553)	**.7950(+.0266)**	**.7656(+.0215)**	.7913 (−.0005)	**.7865**(+.0130)	**.7918(+.0104)**	**.7782(+.0061)**	**.7926(+.0024)**
Barbara	**20.79(+0.26)**	**23.90(+0.28)**	**22.85(+0.20)**	24.18(+0.03)	**23.98(+0.11)**	**23.85(+0.14)**	**23.60(+0.05)**	**22.97(+0.29)**
	.6129(+.0256)	**.7675(+.0294)**	**.7095(+.0137)**	.7689 (−.0012)	**.7644**(+.0086)	**.7589(+.0072)**	**.7394(+.0048)**	**.7220(+.0110)**
Boat	**21.93(+0.45)**	**24.16(+0.19)**	**23.21(+0.20)**	23.82 (−0.02)	**23.86**(+0.22)	**24.10(+0.03)**	23.91(−0.07)	24.23(−0.05)
	.6035(+.0411)	**.7010(+.0108)**	**.6432(+.0082)**	.6719 (−.0025)	**.6782**(+.0100)	.6910(−.0017)	.6767(−.0028)	.6935(−.0076)
Peppers	**20.81(+0.70)**	**23.81(+0.42)**	**22.16(+0.28)**	**23.60(+0.10)**	**23.54**(+0.29)	**23.79(+0.03)**	**23.55(+0.03)**	**23.92(+0.10)**
	.6278(+.1024)	**.7149(+.0321)**	**.6868(+.0583)**	**.7160(+.0136)**	**.6999**(+.0326)	**.7182(+.0003)**	**.6901(+.0117)**	**.7106(+.0077)**
Cameraman	**19.97(+0.91)**	**23.33(+0.26)**	**22.59(+0.47)**	**23.09(+0.06)**	**23.12**(+0.37)	**23.41(+0.02)**	23.08(−0.06)	23.59(−0.13)
	.5484(+.1688)	**.7159(+.0290)**	**.6935(+.0906)**	**.6987(+.0057)**	**.6951**(+.0456)	.7085(−.0001)	.6592(−.0167)	.6887(−.0279)
House	**22.94(+0.80)**	**26.36(+0.49)**	**25.11(+0.61)**	**26.75(+0.06)**	**25.88**(+0.48)	**26.40(+0.31)**	**26.27(+0.08)**	**26.73(+0.26)**
	.6692(+.0990)	**.7540(+.0318)**	**.7307(+.0811)**	**.7745(+.0150)**	**.7369**(+.0400)	**.7512(+.0032)**	.7074(−.0116)	.7326(−.0045)
Fingerprint	**18.61(+0.64)**	**21.78(+0.17)**	**20.14(+0.01)**	21.61 (−0.04)	**21.19(+0.20)**	**21.38(+0.06)**	21.40(−0.02)	**21.22(+0.47)**
	.5711(+.0414)	**.7945(+.0034)**	.6533 (−.0209)	.7705 (−.0046)	**.7456(+.0046)**	.7713(−.0029)	.7631(−.0050)	**.7577(+.0032)**
Couple	**21.86(+0.38)**	**23.69(+0.18)**	**22.62(+0.13)**	23.20 (−0.03)	**23.38**(+0.18)	23.65(−0.01)	23.46(−0.09)	23.72(−0.08)
	.5814(+.0320)	**.6702(+.0087)**	.5901 (−.0013)	.6270 (−.0060)	**.6439**(+.0076)	.6600(−.0052)	.6462(−.0082)	.6621(−.0109)
Hill	**22.53(+0.30)**	**24.79(+0.21)**	**23.81(+0.14)**	24.42(−0.02)	**24.54**(+0.19)	24.82(+0)	24.59(−0.04)	24.84(−0.04)
	.6151(+.0236)	**.6626(+.0082)**	.5918(−.0045)	.6300(−.0041)	**.6383**(+.0051)	.6554(−.0052)	.6440(−.0030)	.6578(−.0066)
Man	**22.26(+0.41)**	**24.59(+0.37)**	**23.49(+0.23)**	**23.95(+0.01)**	**24.20**(+0.22)	**24.41(+0.02)**	24.20(−0.05)	24.50(−0.04)
	.6257(+.0329)	**.6908(+.0128)**	.6373 (−.0068)	.6614 (−.0023)	**.6747**(+.0101)	**.6844(+.0001)**	.6734(−.0002)	.6874(−.0036)
Montage	**19.52(+0.98)**	**24.33(+0.45)**	**22.82(+0.45)**	**24.01(+0.13)**	**23.86**(+0.28)	**24.20(+0.07)**	**23.97(+0.05)**	**24.72(+0.01)**
	.5423(+.1588)	**.7938(+.0502)**	**.7733(+.0830)**	**.8012(+.0213)**	**.7778(+.0472)**	**.7986(+.0091)**	**.7538(+.0142)**	**.7965(+.0031)**
Average	**21.31(+0.58)**	**24.28(+0.31)**	**23.09(+0.28)**	**24.06(+0.03)**	**23.96(+0.27)**	**24.20(+0.07)**	24.10(+0)	**24.26(+0.08)**
	.6081(+.0710)	**.7327(+.0220)**	**.6796(+.0306)**	**.7192(+.0031)**	**.7128(+.0204)**	**.7263(+.0014)**	.7029(−.0009)	.7183(−.0031)

$c_r = 0.55$ for various noise levels. The influence of the main parameters will be discussed in the following part.

5.2. Evaluation criteria

Two objective criteria, namely Peak Signal-to-Noise Ration (PSNR) and Structural Similarity Index (SSIM) [49], are used to evaluate the denoising results. Between them, PSNR is one of the most widely used measures of image quality. Given two images $\mathbf{X}$ and $\mathbf{Y}$, PSNR can be computed as follows:

$$\mathrm{PSNR}(\mathbf{X},\mathbf{Y}) = 10\log_{10}\frac{Q^2}{\mathrm{MSE}}, \tag{31}$$

where $\mathrm{MSE}(\mathbf{X},\mathbf{Y}) = 1/N\sum_{n=1}^{N}(x_n - y_n)^2$, where N is the number of pixels in a image; Q is the dynamic range of allowable image pixel intensities. For an 8-bit gray-level image, $Q = 255$.

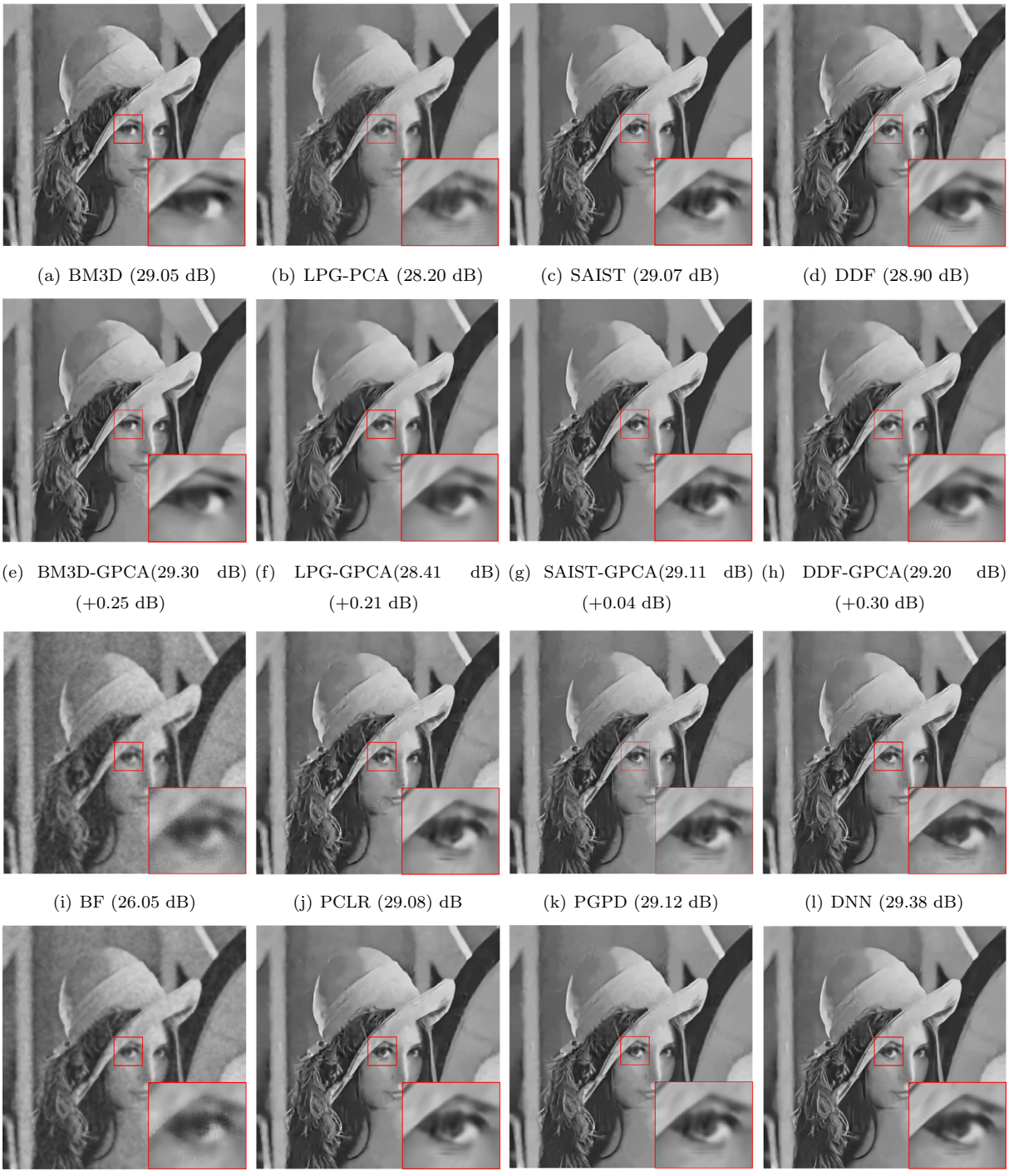
As a complementary metric to PSNR, SSIM is a metric conceived to simulate the response of the human visual system (HVS), which can better reflect the structure similarity between the reference image and the target image. Given two patches $\mathbf{x}_p$ and $\mathbf{y}_p$ taken from the same location of two images $\mathbf{X}$ and $\mathbf{Y}$, the structural similarity can be calculated as follows:

$$\mathrm{SSIM}(\mathbf{x}_p,\mathbf{y}_p) = \frac{(2\mu_{x_p}\mu_{y_p} + C_1)(2\sigma_{x_p y_p} + C_2)}{(\mu_{x_p}^2 + \mu_{y_p}^2 + C_1)(\sigma_{x_p}^2 + \sigma_{y_p}^2 + C_2)}, \tag{32}$$

where μ_{x_p} is the sample mean, σ_{x_p} is the sample standard deviation, and $\sigma_{x_p y_p}$ is the standard covariance; C_1 and C_2 are two constants. SSIM is bounded: $-1 < \mathrm{SSIM}(\mathbf{x}_p,\mathbf{y}_p) \leqslant 1$, achieving maximum value $\mathrm{SSIM}(\mathbf{x}_p,\mathbf{y}_p) = 1$ if and only if $\mathbf{x}_p = \mathbf{y}_p$. In practice, mean SSIM (MSSIM) is usually used to evaluate the overall image quality:

$$\mathrm{MSSIM}(\mathbf{x},\mathbf{y}) = 1/M\sum_{j=1}^{M}\mathrm{SSIM}(\mathbf{x}_j,\mathbf{y}_j), \tag{33}$$

where $\mathbf{x}_j$ and $\mathbf{y}_j$ are the image contents at the j-th local window; and M is the number of local windows in the image.



(a) BM3D (29.05 dB) (b) LPG-PCA (28.20 dB) (c) SAIST (29.07 dB) (d) DDF (28.90 dB)

(e) BM3D-GPCA(29.30 dB) (+0.25 dB) (f) LPG-GPCA(28.41 dB) (+0.21 dB) (g) SAIST-GPCA(29.11 dB) (+0.04 dB) (h) DDF-GPCA(29.20 dB) (+0.30 dB)

(i) BF (26.05 dB) (j) PCLR (29.08) dB (k) PGPD (29.12 dB) (l) DNN (29.38 dB)

(m) BF-GPCA (27.00 dB) (+0.95 dB) (n) PCLR-GPCA (29.28 dB) (+0.20 dB) (o) PGPD-GPCA (29.20 dB) (+0.08 dB) (p) DNN-GPCA (29.56 dB) (+0.18 dB)

Fig. 6. The denoised results for *Lena* at $\sigma = 50$. The first and third rows show the denoised images by different denoising methods, while the second and fourth rows show the corresponding denoised images by the proposed framework.

5.3. Influence of parameters

In our denoising framework, there are three main tuning parameters: block size k, projection factor δ and scaling factor c_r. The patch size k plays an important role in our denoising framework. On one hand, a too large block size can capture the varying local geometry and also result in a high computational cost. On the other hand, a too small block size can deteriorate the denoising performance. In order to better illustrate the effects of the parameters, we use the results of DDF-GPCA to analyze specifically.

To consider the influence of the block size, we set $\delta = 0.3, c_r = 0.55$ and run our method on four typical images, which present a wide range of edges, textures and details, with different block sizes and noise levels. The PSNR results are reported in

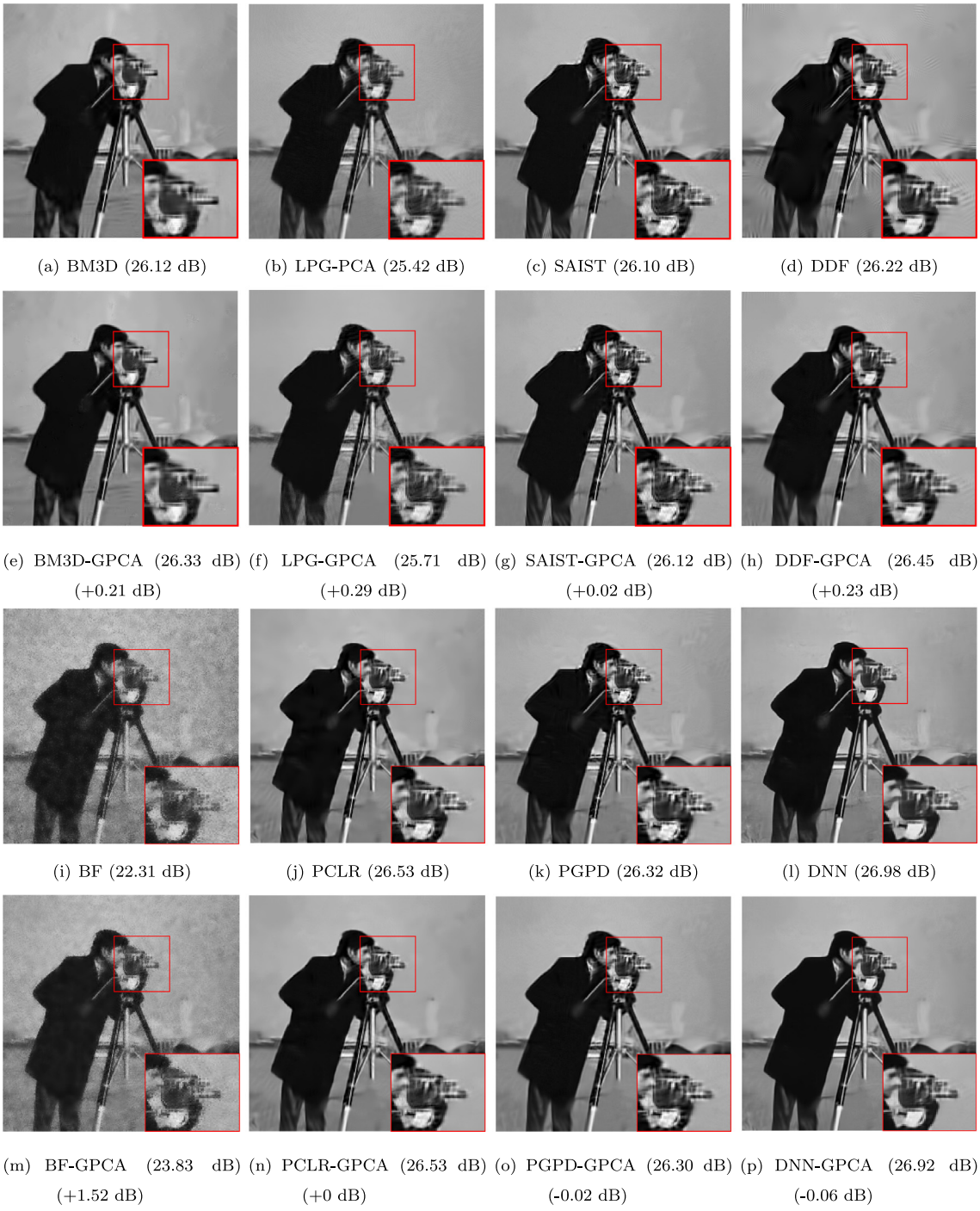


Fig. 7. The denoised results for *Cameraman* at $\sigma = 50$. The first and third rows show the denoised images by different denoising methods, while the second and fourth rows show the corresponding denoised images by the proposed framework.

Table 1. It can be seen that choosing the block size 7×7 in our tests is a reasonable tradeoff between accuracy and speed.

The projection factor δ determines the amount of the residual image added to the initial denoised image. To analyze the effect of δ, we perform our framework at 0.1 intervals at a range from 0.1 to 0.9. Fig. 5(a) displays the denoising performance of our framework, which is applied to the four typical images ($\sigma = 30$) as a function of the parameter δ. As can be observed in Fig. 5(a), the best results are achieved when δ ranges from 0.1 to 0.3. In implementation, we set $\delta = 0.2$. Similar curves for c_r are shown in Fig. 5(b). The best results are obtained when c_r is around 0.55 for most cases.

5.4. Denoising performance on Gaussian noise

5.4.1. Quantitative metrics

In order to quantitatively evaluate the performance of the proposed framework, we have extended it to 8 representative denoising methods. The source codes of these methods can be downloaded from the websites of the respective authors, and all of these methods are performed with the default parameters suggested by the respective authors.

We quantify the denoising performance on eleven test images with various noise levels in terms of both PSNR and MSSIM. All the results are reported in Tables 2,3, from which we can see that

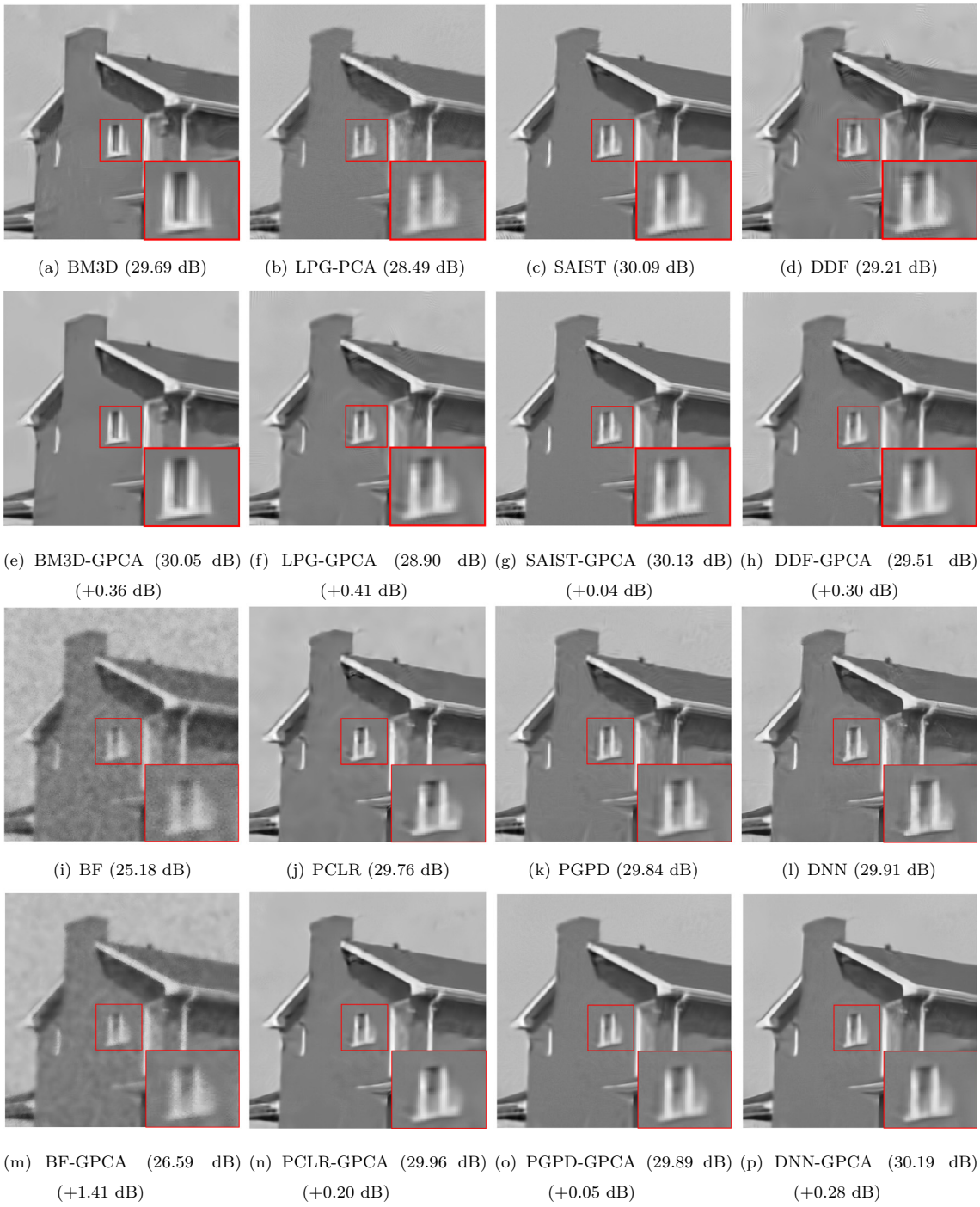


Fig. 8. The denoised results for *House* at $\sigma = 50$. The first and third rows show the denoised images by different denoising methods, while the second and fourth rows show the corresponding denoised images by the proposed framework.

the proposed framework improves the performance of other methods in most cases, including the recently developed deep-learning methods. For example, on average the PSNR gains of BF, BM3D and DDF are at least 0.58 dB, 0.19 dB, and 0.15 dB, respectively. For each individual image, it can be observed that our framework works better on images with more smooth regions and repeated patterns such as *Lena, Barbara, Peppers* and *Montage*. This is because such images often contain more prominent structures that can be utilized in method-noise. Note that our framework deteriorates the performance of some methods (e.g., LPG-PCA, SAIST and DNN) at $\sigma = 10$. This is because these methods preserves image fine structures well at low noise levels, thus resulting in less image structures to be utilized in method-noise.

5.4.2. Visual quality

To evaluate the visual quality of various methods, we show the denoised results of some noisy images under $\sigma = 50$ in Figs. 6–8, from which we can see that our framework improves the visual quality of baseline methods. For example, the proposed framework significantly enhances the edges of the denoised images produced by BF. Our framework also improves the visual effect of other state-of-the-art methods (e.g. BM3D and DDF) as shown in Figs. 6–8,

Table 4
The denoised results (PSNR) of different methods on non-Gaussian noise, including speckle noise and impulse noise. The PSNR values with positive gains are highlighted in bold.

Image	Speckle noise $\sigma = 10$			Impulse noise 4%		
	BF-GPCA	BM3D-GPCA	MED-GPCA	BF-GPCA	BM3D-GPCA	MED-GPCA
Lena	**26.08(+4.33)**	**24.53(+4.68)**	**28.46(+1.85)**	**19.65(+0.17)**	**19.69(+0.20)**	30.83(-0.12)
Barbara	**25.16(+3.49)**	**23.96(+3.82)**	**23.36(+1.10)**	**19.48(+0.15)**	**19.54(+0.18)**	**23.83(+0.77)**
Boat	**25.77(+4.52)**	**24.08(+4.59)**	**25.83(+1.23)**	**19.56(+0.15)**	**19.70(+0.18)**	27.03(-0.03)
Peppers	**25.12(+3.69)**	**23.76(+4.03)**	**25.92(+1.12)**	**19.46(+0.16)**	**19.50(+0.20)**	28.19(-0.08)
Cameraman	**25.68(+4.31)**	**24.08(+4.38)**	**23.38(+1.05)**	**19.22(+0.14)**	**19.28(+0.16)**	**24.16(+0.51)**
House	**25.61(+4.85)**	**24.31(+5.25)**	**27.12(+2.01)**	**19.85(+0.19)**	**19.90(+0.21)**	**30.54(+0.44)**
Fingerprint	**23.05(+3.15)**	**22.40(+3.40)**	**23.59(+0.89)**	**19.47(+0.14)**	**19.48(+0.16)**	**25.67(+0.28)**
Couple	**25.85(+3.90)**	**24.24(+4.19)**	**25.50(+1.07)**	**19.60(+0.16)**	**19.75(+0.19)**	**26.64(+0.06)**
Montage	**25.69(+3.07)**	**24.20(+3.20)**	**21.62(+1.10)**	**19.09(+0.13)**	**19.06(+0.15)**	**21.67(+0.70)**

from which we can see that the artifacts of the denoised images (e.g., in the eye regions of *Lena* and window area of *House*) have been significantly reduced by our framework.

5.5. Denoising performance on non-Gaussian noise

To test the generality of our denoising framework, we test our framework on nine natural images contaminated with non-Gaussian noise, including speckle and impulse noise (salt and pepper noise). For each test image, the standard deviation of speckle noise is 10, and the density of salt-and-pepper noise is 4%. We choose three methods, i.e., BF, BM3D and Median filter (MED) as baselines. All the results are reported in Table 4, from which we can observe that our framework also works well on non-Gaussian noise, especially for speckle noise. For impulse noise, our framework only improves other methods to a certain degree. This is because many pixels have been damaged by the impulse noise, thus leaving less of the original structures in method-noise that can be utilized.

5.6. Computational complexity

Our GPCA framework is designed to improve the existing denoising algorithms, including the state-of-the-art algorithms. In other words, our framework can be considered as a post-processing technique for other denoising methods. Therefore, we only analyze the computational complexity of our framework except the noise filtering procedure. Then most of the computational cost relies in patch grouping and PCA-based denoising procedure. By contrast, the complexity of Pearson-test can be neglected. To be specific, given the parameters: the size k of the variable block, the size S of the searching window patch grouping requires $(2k^2-1)\cdot(S-k+1)^2$ additions, $k^2\cdot(S-k+1)^2$ multiplications and $(S-k+1)^2$ logic operations. Assume on average n variable blocks are selected, i.e., the dataset $\mathbf{Y}$ is of dimension $k^2\times n$. Then the PCA transformation of the dataset $\mathbf{Y}$ requires $k^2\cdot n+(n^2-1)\cdot k^4+(k^2-1)\cdot k^2\cdot n$ additions, $k^4\cdot(n+n^2)$ multiplications, and an SVD decomposition of an $k^2\times k^2$ definite covariance matrix in the PCA transformation.

In practice, our framework is performed on a MATLAB platform of an Intel Core i5 CPU 2.6 GHz with 8 GB memory. For a 256×256 grayscale image, we set $k=7$ and $S=41$, and our MATLAB implementation requires about 66 s on average, which means that our GPCA framework improves the performance of other denoising methods at the cost of reasonably additional computational burden. Besides, the main computational cost of the PCA-based denoising is the calculation of PCA transform for each patch group matrix. Since each group matrix could potentially be dealt with independently in parallel, our framework is appropriate to parallel processing. Therefore, in fact, our framework can be further sped up in a parallel implementation.

6. Conclusions

In this paper, we propose a generic image denoising framework to further improve the performance of existing denoising methods based on a new back projection and PCA-based denoising. To be specific, the back projection adopts Pearson-test to detect the useful information in method-noise, thus resulting in a trade-off between the loss of image details and the residual noise. Similar patches are then collected to construct patch group for the PCA-based denoising, which is able to remove the noise while reducing the artifacts. Experimental results on natural images, contaminated with Gaussian and non-Gaussian noise, show that our framework works well on other denoising methods in most cases, especially at high noise levels. Furthermore, our framework shows more flexibility, indicating a wide range of potential applications of our framework.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 61371078, and the R&D Program of Shenzhen under Grant Nos. JCYJ20140509172959977, JSGG20150512162853495, ZDSYS20140509172959989, JCYJ20160331184440545.

References

[1] C. Yan, Y. Zhang, J. Xu, F. Dai, J. Zhang, Q. Dai, F. Wu, Efficient parallel framework for hevc motion estimation on many-core processors, IEEE Trans. Circ. Syst. Video Technol. 24 (12) (2014) 2077–2089.
[2] C. Yan, Y. Zhang, F. Dai, J. Zhang, L. Li, Q. Dai, Efficient parallel hevc intra-prediction on many-core processor, Electron. Lett. 50 (11) (2014) 805–806.
[3] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: high confidence predictions for unrecognizable images, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, pp. 427–436.
[4] Y. Han, C. Xu, G. Baciu, M. Li, Lightness biased cartoon-and-texture decomposition for textile image segmentation, Neurocomputing 168 (2015) 575–587.
[5] Haoyi Liang, Daniel S. Weller, Comparison-based image quality assessment for selecting image restoration parameters, IEEE Trans. Image Process. 25 (11) (2016) 5118–5130.
[6] H.-T. Wu, J. Huang, Y.-Q. Shi, A reversible data hiding method with contrast enhancement for medical images, J. Visual Commun. Image Represent. 31 (2015) 146–153.
[7] K. Gu, D. Tao, J.F. Qiao, W. Lin, Learning a no-reference quality assessment model of enhanced images with big data, IEEE Trans. Neural Netw. Learning Syst. PP (99) (2017) 1–13.
[8] C. Yan, Y. Zhang, J. Xu, F. Dai, L. Li, Q. Dai, F. Wu, A highly parallel framework for hevc coding unit partitioning tree decision on many-core processors, IEEE Signal Process. Lett. 21 (5) (2014) 573–576.
[9] C. Yan, Y. Zhang, F. Dai, X. Wang, L. Li, Q. Dai, Parallel deblocking filter for hevc on many-core processor, Electron. Lett. 50 (5) (2014) 367–368.

[10] K.W. Hung, W.C. Siu, Robust soft-decision interpolation using weighted least squares, IEEE Trans. Image Process. 21 (3) (2012) 1061–1069.
[11] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: 1998 IEEE International Conference on Computer Vision (ICCV), IEEE, 1998, pp. 839–846.
[12] T. Dai, W. Lu, W. Wang, J. Wang, S.-T. Xia, Entropy-based bilateral filtering with a new range kernel, Signal Process. 137 (2017) 223–234.
[13] A. Buades, B. Coll, J.-M. Morel, A non-local algorithm for image denoising, in: 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2005, pp. 60–65.
[14] S.G. Chang, B. Yu, M. Vetterli, Adaptive wavelet thresholding for image denoising and compression, IEEE Trans. Image Process. 9 (9) (2000) 1532–1546.
[15] A. Pizurica, W. Philips, Estimating the probability of the presence of a signal of interest in multiresolution single-and multiband image denoising, IEEE Trans. Image Process. 15 (3) (2006) 654–665.
[16] F. Luisier, T. Blu, M. Unser, A new sure approach to image denoising: interscale orthonormal wavelet thresholding, IEEE Trans. Image Process. 16 (3) (2007) 593–606.
[17] J. Portilla, V. Strela, M.J. Wainwright, E.P. Simoncelli, Image denoising using scale mixtures of gaussians in the wavelet domain, IEEE Trans. Image Process. 12 (11) (2003) 1338–1351.
[18] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Image denoising by sparse 3-d transform-domain collaborative filtering, IEEE Trans. Image Process. 16 (8) (2007) 2080–2095.
[19] C. Knaus, M. Zwicker, Dual-domain image denoising, in: 2013 IEEE International Conference on Image Processing (ICIP), IEEE, 2013, pp. 440–444.
[20] L. Zhang, W. Dong, D. Zhang, G. Shi, Two-stage image denoising by principal component analysis with local pixel grouping, Pattern Recogn. 43 (4) (2010) 1531–1549.
[21] W. Dong, G. Shi, X. Li, Nonlocal image restoration with bilateral variance estimation: a low-rank approach, IEEE Trans. Image Process. 22 (2) (2013) 700–711.
[22] H.C. Burger, C.J. Schuler, S. Harmeling, Image denoising: can plain neural networks compete with bm3d?, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp 2392–2399.
[23] F. Chen, L. Zhang, H. Yu, External patch prior guided internal clustering for image denoising, in: 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, 2015, pp. 603–611.
[24] J. Xu, L. Zhang, W. Zuo, D. Zhang, X. Feng, Patch group based nonlocal self-similarity prior learning for image denoising, in: 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, 2015, pp. 244–252.
[25] T. Dai, C.-B. Song, J.-P. Zhang, S.-T. Xia, Pmpa: a patch-based multiscale products algorithm for image denoising, in: 2015 IEEE International Conference on Image Processing (ICIP), IEEE, 2015, pp. 4406–4410.
[26] P. Chatterjee, P. Milanfar, Is denoising dead?, IEEE Trans Image Process. 19 (4) (2010) 895–911.
[27] D. Brunet, E.R. Vrscay, Z. Wang, The use of residuals in image denoising, in: 2009 Image Analysis and Recognition (IAP), Springer, 2009, pp. 1–12.
[28] S. Dai, M. Han, Y. Wu, Y. Gong, Bilateral back-projection for single image super resolution, in: 2007 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2007, pp. 1039–1042.
[29] W. Dong, L. Zhang, G. Shi, X. Wu, Nonlocal back-projection for adaptive image enlargement, in: 2009 IEEE International Conference on Image Processing (ICIP), IEEE, 2009, pp. 349–352.
[30] H. Zhong, C. Yang, X. Zhang, A new weight for nonlocal means denoising using method noise, IEEE Signal Process. Lett. 19 (8) (2012) 535–538.
[31] Q. Guo, C. Zhang, Y. Zhang, H. Liu, An efficient svd-based method for image denoising, IEEE Trans. Circ. Syst. Video Technol. 26 (5) (2016) 868–880.
[32] S. Ling, Y. Ruomei, L. Xuelong, L. Yan, From heuristic optimization to dictionary learning: a review and comprehensive comparison of image denoising algorithms, IEEE Trans. Cybernet. 44 (7) (2014) 1001–1013.
[33] S. Paris, P. Kornprobst, J. Tumblin, F. Durand, Bilateral Filtering: Theory and Applications, Now Publishers Inc, 2009.
[34] K.W. Hung, W.C. Siu, Fast image interpolation using the bilateral filter, IET Image Process. 6 (7) (2012) 877–890.
[35] A. Buades, B. Coll, J.-M. Morel, A review of image denoising algorithms, with a new one, Multiscale Model. Simul. 4 (2) (2005) 490–530.
[36] T. Brox, O. Kleinschmidt, D. Cremers, Efficient nonlocal means for denoising of textural patterns, IEEE Trans. Image Process. 17 (7) (2008) 1083–1092.
[37] J. Orchard, M. Ebrahimi, A. Wong, Efficient nonlocal-means denoising using the svd, in: 2008 IEEE International Conference on Image Processing (ICIP), IEEE, 2008, pp. 1732–1735.
[38] G. Peyré, S. Bougleux, L. Cohen, Non-local regularization of inverse problems, in: European Conference on Computer Vision (ECCV), Springer, 2008, pp. 57–68.
[39] Y.C. Eldar, Generalized sure for exponential families: applications to regularization, IEEE Trans. Signal Process. 57 (2) (2009) 471–481.
[40] D. Van De Ville, M. Kocher, Nonlocal means with dimensionality reduction and sure-based parameter selection, IEEE Trans. Image Process. 20 (9) (2011) 2683–2690.
[41] C. Kervrann, J. Boulanger, Optimal spatial adaptation for patch-based image denoising, IEEE Trans. Image Process. 15 (10) (2006) 2866–2878.
[42] M. Aharon, M. Elad, A. Bruckstein, K-svd: an algorithm for designing overcomplete dictionaries for sparse representation, IEEE Trans. Signal Process. 54 (11) (2006) 4311–4322.
[43] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Non-local sparse models for image restoration, in: 2009 IEEE International Conference on Computer Vision (ICCV), IEEE, 2009, pp. 2272–2279.
[44] C.-A. Deledalle, J. Salmon, A.S. Dalalyan, Image denoising with patch based pca: local versus global, in: 2011 British Machine Vision Conference (BMVC), BMVA Press, 2011, pp. 25-1.
[45] Y. Zhang, J. Liu, M. Li, Z. Guo, Joint image denoising using adaptive principal component analysis and self-similarity, Inform. Sci. 259 (2014) 128–141.
[46] J. Chen, C.-K. Tang, J. Wang, Noise brush: interactive high quality image-noise separation, ACM Trans. Graph. 28 (5) (2009) 146.
[47] Y. Romano, M. Elad, Improving k-svd denoising by post-processing its method-noise, in: 2013 IEEE International Conference on Image Processing (ICIP), IEEE, 2013, pp. 435–439.
[48] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: residual learning of deep cnn for image denoising, IEEE Trans. Image Process. PP (99) (2017), pp. 1-1.
[49] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.