Patch Group Based Nonlocal Self-Similarity Prior Learning for Image Denoising

Jun Xu[1], Lei Zhang[1,*], Wangmeng Zuo[2], David Zhang[1], and Xiangchu Feng[3]

[1]Dept. of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[3]School of Mathematics and Statistics, Xidian University, Xi'an, China

{csjunxu,cslzhang,csdzhang}@comp.polyu.edu.hk,wmzuo@hit.edu.cn,xcfeng@mail.xidian.edu.cn

Abstract

Patch based image modeling has achieved a great success in low level vision such as image denoising. In particular, the use of image nonlocal self-similarity (NSS) prior, which refers to the fact that a local patch often has many nonlocal similar patches to it across the image, has significantly enhanced the denoising performance. However, in most existing methods only the NSS of input degraded image is exploited, while how to utilize the NSS of clean natural images is still an open problem. In this paper, we propose a patch group (PG) based NSS prior learning scheme to learn explicit NSS models from natural images for high performance denoising. PGs are extracted from training images by putting nonlocal similar patches into groups, and a PG based Gaussian Mixture Model (PG-GMM) learning algorithm is developed to learn the NSS prior. We demonstrate that, owe to the learned PG-GMM, a simple weighted sparse coding model, which has a closed-form solution, can be used to perform image denoising effectively, resulting in high PSNR measure, fast speed, and particularly the best visual quality among all competing methods.

1. Introduction

As a classical problem in low level vision, image denoising has been extensively studied, yet it is still an active topic for that it provides an ideal test bed for image modeling techniques. In general, image denoising aims to recover the clean image $\mathbf{x}$ from its noisy observation $\mathbf{y} = \mathbf{x} + \mathbf{v}$, where $\mathbf{v}$ is assumed to be additive white Gaussian noise. A variety of image denoising methods have been developed in past decades, including filtering based methods [1], diffusion based methods [2], total variation based methods [3, 4], wavelet/curvelet based methods [5, 6, 7], sparse representation based methods [8, 9, 10], nonlocal self-similarity based methods [11, 12, 13, 14], etc.

*This work is supported by the HK RGC GRF grant (PolyU 5313/13E).

Image modeling plays a central role in image denoising. By modeling the wavelet transform coefficients as Laplacian distributions, many wavelet shrinkage based denoising methods such as the classical soft-thresholding [5] have been proposed. Chang *et al*. modeled the wavelet transform coefficients as generalized Gaussian distribution, and proposed the BayesShrink [6] algorithm. By considering the correlation of wavelet coefficients across scales, Portilla *et al*. [15] proposed to use Gaussian Scale Mixtures for image modeling and achieved promising denoising performance. It is widely accepted that natural image gradients exhibit heavy-tailed distributions [16], and the total variation (TV) based methods [3, 4] actually assume Laplacian distributions of image gradients for denoising. The Fields of Experts (FoE) [17] proposed by Roth and Black models the filtering responses with Student's t-distribution to learn filters through Markov Random Field (MRF) [18]. Recently, Schmidt and Roth proposed the cascade of shrinkage fields (CSF) to perform denoising efficiently [19].

Instead of modeling the image statistics in some transformed domain (e.g., gradient domain, wavelet domain or filtering response domain), another popular approach is to model the image priors on patches. One representative is the sparse representation based scheme which encodes an image patch as a linear combination of a few atoms selected from a dictionary [8, 20, 21]. The dictionary can be chosen from the off-the-shelf dictionaries (e.g., wavelets and curvelets), or it can be learned from natural image patches. The seminal work of K-SVD [8, 22] has demonstrated promising denoising performance by dictionary learning, which has yet been extended and successfully used in various image processing and computer vision applications [23, 24, 25]. By viewing image patches as samples of a multivariate variable vector and considering that natural images are non-Gaussian, Zoran and Weiss [26, 27] and Yu *et al*. [28] used Gaussian Mixture Model (GMM) to model image patches, and achieved state-of-the-art denoising and image

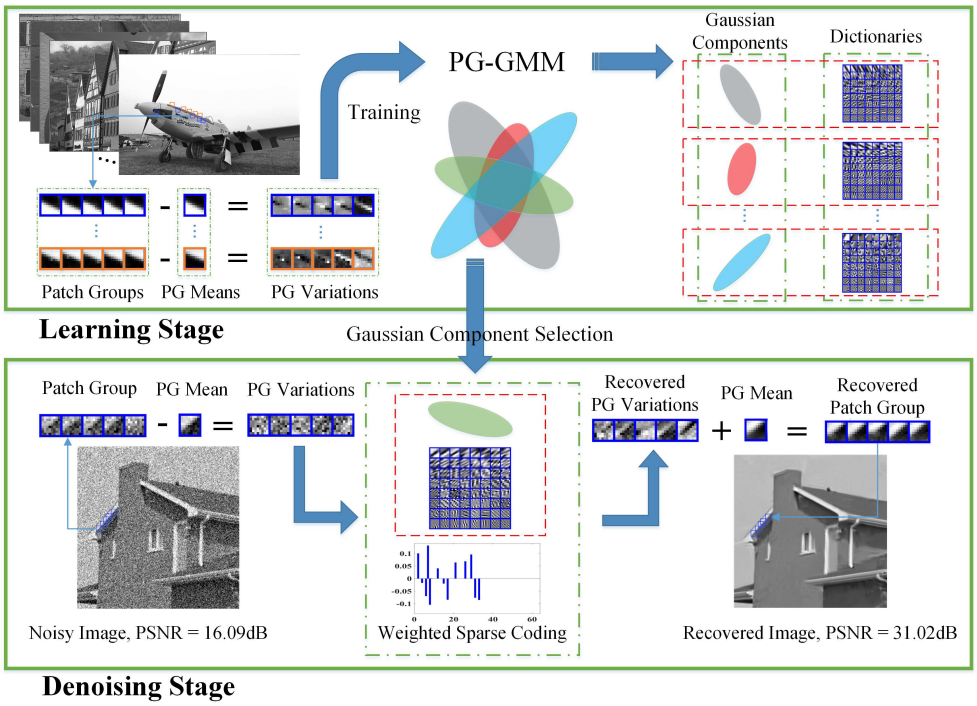


Figure 1. Flowchart of the proposed patch group based prior learning and image denoising framework.

restoration results, respectively.

Natural images often have many repetitive local patterns, and a local patch can have many similar patches to it across the whole image. The so-called nonlocal self-similarity (NSS) prior is among the most successful priors for image restoration. The nonlocal means [11] and nonlocal regularization [29] methods improve much the image denoising performance over the conventional local self-similarity based methods. Dabov *et al*. [12] constructed 3D cubes of nonlocal similar patches and conducted collaborative filtering in the sparse 3D transform domain. The so-called BM3D algorithm has become a benchmark in image denoising. Mairal *et al*. [9] proposed the LSSC algorithm to exploit NSS via group sparse coding. The NSP [30] method fits the singular values of NSS patch matrix by Laplacian distribution. Dong *et al*. [10] unified NSS and local sparse coding into the so-called NCSR framework, which shows powerful image restoration capability. By assuming that the matrix of nonlocal similar patches has a low rank structure, the low-rank minimization based methods [13, 14] have also achieved very competitive denoising results.

Though NSS has demonstrated its great success in image denoising, in most existing methods only the NSS of noisy input image is used for denoising. For example, in BM3D [12] the nonlocal similar patches of a noisy image are collected as a cube for collaborative filtering. In NCSR [10], the nonlocal means are subtracted in the sparse domain to regularize the sparse coding of noisy patches. In WNNM [14], the low-rank regularization is enforced to recover the latent structure of the matrix of noisy patches. We argue that, however, such utilizations of NSS are not effective enough because they neglect the NSS of clean natural images, which can be pre-learned for use in the denoising stage. To the best of our knowledge, unfortunately, so far there is not an explicit NSS prior model learned from natural images for image restoration.

With the above considerations, in this work we propose to learn explicit NSS models from natural images, and apply the learned prior models to noisy images for high performance denoising. The flowchart of the proposed method is illustrated in Fig. 1. In the learning stage, we extract millions of patch groups (PG) from a set of clean natural images. A PG is formed by grouping the similar patches to a local patch in a large enough neighborhood. A PG based GMM (PG-GMM) learning algorithm is developed to learn the NSS prior for the PGs. In the denoising stage, the learned PG-GMM will provide dictionaries as well as regularization parameters, and a simple weighted sparse coding model is developed for image denoising. Our extensive experiments validated that the proposed PG prior based denoising method outperforms many state-of-the-art algorithms quantitatively (in PSNR) while being much more efficient. More importantly, it delivers the best qualitative denoising results with finer details and less artifacts, owe to the NSS prior learned from clean natural images.

2. Patch Group Based Prior Modeling of Nonlocal Self-Similarity

Image nonlocal self-similarity (NSS) has been widely adopted in patch based image denoising and other image

restoration tasks [9, 10, 11, 12, 14]. Despite the great success of NSS in image restoration, most of the existing works exploit the NSS only from the degraded image. Usually, for a given patch in the degraded image, its nonlocal similar patches are collected, and then the nonlocal means [11], or 3D transforms [12], or some regularization terms [9, 10, 14, 23] can be introduced for image restoration. However, how to learn the NSS prior from clean natural images and apply it to image restoration is still an open problem. In this work, we make the first attempt on this problem, and develop a patch group (PG) based NSS prior learning scheme.

2.1. Patch Group and Group Mean Subtraction

For each local patch (size: $p \times p$) of a given clean image, we can find the first M most similar nonlocal patches to it across the whole image. In practice, this can be done by Euclidean distance based block matching in a large enough local window of size $W \times W$. A PG is formed by grouping the M similar patches, denoted by $\{\mathbf{x}_m\}_{m=1}^M$, where $\mathbf{x}_m \in \mathbb{R}^{p^2 \times 1}$ is a patch vector. The mean vector of this PG is $\boldsymbol{\mu} = \frac{1}{M}\sum_{m=1}^M \mathbf{x}_m$, and $\overline{\mathbf{x}}_m = \mathbf{x}_m - \boldsymbol{\mu}$ is the group mean subtracted patch vector. We call

$$\overline{\mathbf{X}} \triangleq \{\overline{\mathbf{x}}_m\}, m = 1, ..., M \tag{1}$$

the group mean subtracted PG, and it will be used to learn the NSS prior in our work.

In Fig. 2, we show two different PGs, their group means, and the PGs after mean subtraction. One can see that before mean subtraction, the two PGs have very different local structures. After mean subtraction, the two PGs will have very similar variations. This greatly facilitates the prior learning because the possible number of patterns is reduced, while the training samples of each pattern are increased. We will discuss further the benefits of mean subtraction and the associated prior model learning in Section 2.4.

2.2. PG-GMM Learning

From a given set of natural images, we can extract N PGs, and we denote one PG as

$$\overline{\mathbf{X}}_n \triangleq \{\overline{\mathbf{x}}_{n,m}\}_{m=1}^M, n = 1, ..., N. \tag{2}$$

The PGs $\{\overline{\mathbf{X}}_n\}$ contain a rich amount of NSS information of natural images, and the problem turns to how to learn explicit prior models from $\{\overline{\mathbf{X}}_n\}$. Considering that Gaussian Mixture Model (GMM) has been successfully used to model the image patch priors in EPLL [26] and PLE [28], we propose to extend patch based GMM to patch group based GMM (PG-GMM) for NSS prior learning.

With PG-GMM, we aim to learn a set of K Gaussians $\{\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}$ from N training PGs $\{\overline{\mathbf{X}}_n\}$, while requiring that all the M patches $\{\overline{\mathbf{x}}_{n,m}\}$ in PG $\overline{\mathbf{X}}_n$ belong to the same Gaussian component and assume that the patches in the PG are independently sampled. Note that such an assumption

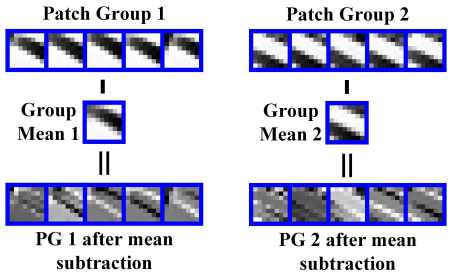


Figure 2. Different patch groups (PG) share similar PG variations.

is commonly used in patch based image modeling [8, 9]. Then, the likelihood of $\{\overline{\mathbf{X}}_n\}$ can be calculated as

$$P(\overline{\mathbf{X}}_n) = \sum_{k=1}^K \pi_k \prod_{m=1}^M \mathcal{N}(\overline{\mathbf{x}}_{n,m}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{3}$$

By assuming that all the PGs are independently sampled, the overall objective likelihood function is $\mathcal{L} = \prod_{n=1}^N P(\overline{\mathbf{X}}_n)$. Taking the log of it, we maximize the following objective function for PG-GMM learning

$$\ln \mathcal{L} = \sum_{n=1}^N \ln(\sum_{k=1}^K \pi_k \prod_{m=1}^M \mathcal{N}(\overline{\mathbf{x}}_{n,m}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)). \tag{4}$$

As in GMM learning [18], we introduce hidden variables $\{\Delta_{nk}|n = 1, ..., N; k = 1, ..., K\}$ to optimize (4). If PG $\overline{\mathbf{X}}_n$ belongs to the kth component, $\Delta_{nk} = 1$; and $\Delta_{nk} = 0$ otherwise. Then the EM algorithm [31] can be used to optimize (4) via two alternative steps. In the E-Step, by the Bayes' formula, the expected value of Δ_{nk} is

$$\gamma_{nk} = \frac{\pi_k \prod_{m=1}^M \mathcal{N}(\overline{\mathbf{x}}_{n,m}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^K \pi_l \prod_{m=1}^M \mathcal{N}(\overline{\mathbf{x}}_{n,m}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{5}$$

In the M-step, since for each PG $\overline{\mathbf{X}}_n$, $\sum_{m=1}^M \overline{\mathbf{x}}_{n,m} = \mathbf{0}$, we have

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N \gamma_{nk} \sum_{m=1}^M \overline{\mathbf{x}}_{n,m}}{\sum_{n=1}^N \gamma_{nk}} = \mathbf{0}, \tag{6}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^N \gamma_{nk} \sum_{m=1}^M \overline{\mathbf{x}}_{n,m} \overline{\mathbf{x}}_{n,m}^T}{\sum_{n=1}^N \gamma_{nk}}. \tag{7}$$

The calculations of $\pi_k = \frac{1}{N}\sum_{n=1}^N \gamma_{nk}$ are similar to [18].

By alternating between the E-step and the M-step, the model parameters will be updated iteratively, and the update in each iteration can guarantee to increase the value of the log-likelihood function (5), and the EM algorithm will converge [18, 32]. Fig. 3 shows the convergence curve of the proposed PG-GMM algorithm by using the Kodak PhotoCD Dataset (http://r0k.us/graphics/kodak/) for training.

2.3. Complexity Analysis

In the training stage, there are N PGs, each of which has M patches, and hence we have $N \times M$ patches. In the M-step, we only need to calculate the covariance matrices since the mean of each Gaussian component is zero. The

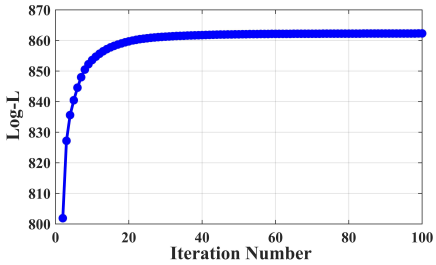


Figure 3. The convergence curve of log-likelihood in PG-GMM training on the Kodak PhotoCD Dataset.

cost of this step is $O(p^4MN)$. In the E-step, the cost is $O(p^6MN)$. Suppose that the number of iterations is T, the overall complexity of PG-GMM training is $O(p^6MNT)$.

2.4. Discussions

GMM has been used for patch based image prior learning and achieved promising results, e.g., EPLL [26] and PLE [28]. In this paper, we extend the patch based image prior learning to PG based prior learning to model the NSS information. The developed PG-GMM method has some important advantages over the patch based GMM method.

First, in patch based GMM, the mean value of each patch is subtracted before learning the Gaussian components. This is to remove the DC (direct current) of each patch but will not change the essential structure of a patch. However, in PG-GMM the mean vector of all patches in a group is calculated and subtracted from each patch, and hence the structure of each patch is changed. As a result, many patches which originally have different local patterns may become similar after group mean subtraction (please refer to Fig. 2 for an example). This makes the PG-GMM learning process easier and more stable.

Second, as can be seen in Eq. (9), the mean vector of each Gaussian component in PG-GMM is naturally a zero vector. This implies that we only need to learn the covariance matrix of each component without considering its mean. However, in patch based GMM [26], the mean vectors of Gaussians can only be forced to zero and there is no theoretical guarantee for this.

Third, due to reduction of possible patterns in PG-GMM and the reduced number of variables to learn, we do not need to set a large number of Gaussian components in PG-GMM learning. For example, in EPLL [26], 200 Gaussian components are learned to achieve competing denoising performance with BM3D [12], while in PG-GMM learning only 32 Gaussian components are enough to outperform BM3D (please refer to the experimental section for details).

3. Image Denoising by Patch Group Priors

3.1. Denoising Model

Given a noisy image $\mathbf{y}$, like in the PG-GMM learning stage, for each local patch we search for its similar patches in a window centered on it to form a PG, denoted by $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_M\}$. Then the group mean of $\mathbf{Y}$, denoted by $\boldsymbol{\mu}_y$, is calculated and subtracted from each patch, leading to the mean subtracted PG $\overline{\mathbf{Y}}$. We can write $\overline{\mathbf{Y}}$ as $\overline{\mathbf{Y}} = \overline{\mathbf{X}} + \mathbf{V}$, where $\overline{\mathbf{X}}$ is the corresponding clean PG and $\mathbf{V}$ contains the corrupted noise. The problem then turns to how to recover $\overline{\mathbf{X}}$ from $\overline{\mathbf{Y}}$ by using the learned PG-GMM priors. Note that the mean $\boldsymbol{\mu}_y$ of $\overline{\mathbf{Y}}$ is very close to the mean of $\overline{\mathbf{X}}$ since the mean vector of noise $\mathbf{V}$ is nearly zero. $\boldsymbol{\mu}_y$ will be added back to the denoised PG to obtain the denoised image.

3.1.1 Gaussian Component Selection

For each $\overline{\mathbf{Y}}$, we select the most suitable Gaussian component to it from the trained PG-GMM. As in [26], suppose that the variance of Gaussian white noise corrupted in the image is σ^2, the covariance matrix of the kth component will become $\boldsymbol{\Sigma}_k + \sigma^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The selection can be done by checking the posterior probability that $\overline{\mathbf{Y}}$ belongs to the kth Gaussian component:

$$P(k|\overline{\mathbf{Y}}) = \frac{\prod_{m=1}^{M}\mathcal{N}(\overline{\mathbf{y}}_m|\mathbf{0}, \boldsymbol{\Sigma}_k + \sigma^2\mathbf{I})}{\sum_{l=1}^{K}\prod_{m=1}^{M}\mathcal{N}(\overline{\mathbf{y}}_m|\mathbf{0}, \boldsymbol{\Sigma}_l + \sigma^2\mathbf{I})}. \quad (8)$$

Taking log-likelihood of (8), we have

$$\ln P(k|\overline{\mathbf{Y}}) = \sum_{m=1}^{M}\ln\mathcal{N}(\overline{\mathbf{y}}_m|\mathbf{0}, \boldsymbol{\Sigma}_k + \sigma^2) - \ln C \quad (9)$$

where C is the denominator in Eq. (9) and it is the same for all components. Finally, the component with the highest probability $\ln P(k|\overline{\mathbf{Y}})$ is selected to process $\overline{\mathbf{Y}}$.

3.1.2 Weighted Sparse Coding with Closed-Form Solution

Suppose that the kth Gaussian component is selected for PG $\overline{\mathbf{Y}}$. For notation simplicity, we remove the subscript k and denote by $\boldsymbol{\Sigma}$ the covariance matrix of this component. In PG-GMM, the PGs actually represent the variations of the similar patches in a group, and these variations are assigned to the same Gaussian distribution. By singular value decomposition (SVD), $\boldsymbol{\Sigma}$ can be factorized as

$$\boldsymbol{\Sigma} = \mathbf{D}\boldsymbol{\Lambda}\mathbf{D}^T, \quad (10)$$

where $\mathbf{D}$ is an orthonormal matrix composed by the eigenvectors of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. With PG-GMM, the eigenvectors in $\mathbf{D}$ capture the statistical structures of NSS variations in natural images, while the eigenvalues in $\boldsymbol{\Lambda}$ represent the significance of these eigenvectors. Fig. 4 shows the eigenvectors for 3 Gaussian components. It can be seen that these eigenvectors encode the possible variations of the PGs. For one Gaussian component, the first eigenvector represents its largest variation, while the last eigenvector represents its smallest variation. For different Gaussian components, we can see that their eigenvectors (with the same index) are very different. Hence, $\mathbf{D}$ can be used to represent the structural variations of the PGs in that component.

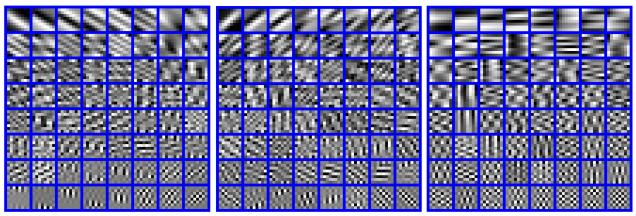


Figure 4. Eigenvectors of 3 Gaussian components from the learned PG-GMM, sorted by the values of corresponding eigenvalues.

For each patch $\overline{\mathbf{y}}_m$ in the PG $\overline{\mathbf{Y}}$, we propose to use $\mathbf{D}$ as the dictionary to sparsely encode $\overline{\mathbf{y}}_m$ as $\overline{\mathbf{y}}_m = \mathbf{D}\boldsymbol{\alpha} + \mathbf{v}$, where $\boldsymbol{\alpha}$ is the vector of sparse coding coefficients and $\mathbf{v}$ is the corrupted noise. Meanwhile, we propose to introduce a weighting vector $\mathbf{w}$ to weight the coding vector $\boldsymbol{\alpha}$ (we will see in (16) that $\mathbf{w}$ is related to the eigenvalues in $\boldsymbol{\Lambda}$), resulting in the following simple but highly effective weighted sparse coding model:

$$\min_{\boldsymbol{\alpha}} \|\overline{\mathbf{y}}_m - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \|\mathbf{w}^T\boldsymbol{\alpha}\|_1. \tag{11}$$

From the viewpoint of Maximum A-Posterior (MAP) estimation, the optimal solution of (11) is $\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}} \ln P(\boldsymbol{\alpha}|\overline{\mathbf{y}}_m)$. By Bayes' formula, it is equivalent to

$$\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}} \{\ln P(\overline{\mathbf{y}}_m|\boldsymbol{\alpha}) + \ln P(\boldsymbol{\alpha})\}. \tag{12}$$

The log-likelihood term $\ln P(\overline{\mathbf{y}}_m|\boldsymbol{\alpha})$ is characterized by the statistics of noise $\mathbf{v}$, which is assumed to be white Gaussian with standard deviation σ. Hence, we have

$$P(\overline{\mathbf{y}}_m|\boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}\|\overline{\mathbf{y}}_m - \mathbf{D}\boldsymbol{\alpha}\|_2^2). \tag{13}$$

We assume that the sparse coding coefficients in $\boldsymbol{\alpha}$ follow i.i.d. Laplacian distribution. More specifically, for entry $\boldsymbol{\alpha}_i$, which is the coding coefficient of patch $\overline{\mathbf{y}}_m$ over the ith eigenvector in $\mathbf{D}$, we assume that it follows distribution $\frac{c}{\sqrt{2}\lambda_i}\exp(-c\sqrt{2}|\boldsymbol{\alpha}_i|/\lambda_i)$, where $\lambda_i = \boldsymbol{\Lambda}_i^{1/2}$ and c is a constant. Note that we adjust the scale factor of the distribution by (square root of) the ith eigenvalue $\boldsymbol{\Lambda}_i$. This is because the larger the eigenvalue $\boldsymbol{\Lambda}_i$ is, the more important the ith eigenvector in $\mathbf{D}$ is, and hence the distribution of the coding coefficients over this eigenvector should have a longer tail (i.e., less sparse). Finally, we have

$$P(\boldsymbol{\alpha}) = \prod_{i=1}^{p^2} \frac{c}{\sqrt{2}\lambda_i} \exp(-\frac{c\sqrt{2}|\boldsymbol{\alpha}_i|}{\lambda_i}). \tag{14}$$

Putting (13) and (14) into (12), we have

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \|\overline{\mathbf{y}}_m - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \sum_{i=1}^{p^2} \frac{c * 2\sqrt{2}\sigma^2}{\lambda_i}|\boldsymbol{\alpha}_i|. \tag{15}$$

By comparing (15) with (11), we can see that the ith entry of the weighting vector $\mathbf{w}$ should be

$$\mathbf{w}_i = c * 2\sqrt{2}\sigma^2/(\lambda_i + \varepsilon), \tag{16}$$

where ε is a small positive number to avoid dividing by zero.

With $\mathbf{w}$ determined by (16), let's see what the solution of (11) should be. Since the dictionary $\mathbf{D}$ is orthonormal, it is not difficult to find out that (11) has a closed-form solution (detailed derivation can be found in the supplementary material):

$$\hat{\boldsymbol{\alpha}} = \text{sgn}(\mathbf{D}^\mathbf{T}\overline{\mathbf{y}}_m) \odot \max(|\mathbf{D}^\mathbf{T}\overline{\mathbf{y}}_m| - \mathbf{w}/2, 0), \tag{17}$$

where sgn(•) is the sign function, $\odot$ means element-wise multiplication, and $|\mathbf{D}^\mathbf{T}\overline{\mathbf{y}}_m|$ is the absolute value of each entry of vector $|\mathbf{D}^\mathbf{T}\overline{\mathbf{y}}_m|$. The closed-form solution makes our weighted sparse coding process very efficient.

Alg. 1: Patch Group Prior based Denoising (PGPD)

Input: Noisy image $\mathbf{y}$, PG-GMM model

1. Initialization: $\hat{\mathbf{x}}^{(0)} = \mathbf{y}, \mathbf{y}^{(0)} = \mathbf{y}$;

for $t = 1 : IteNum$ **do**

2. Iterative Regularization:
$\mathbf{y}^{(t)} = \hat{\mathbf{x}}^{(t-1)} + \delta(\mathbf{y} - \mathbf{y}^{(t-1)})$;
3. Estimate the standard deviation of noise;

for each PG $\mathbf{Y}$ **do**

4. Calculate group mean $\boldsymbol{\mu}_y$ and form PG $\overline{\mathbf{Y}}$;
5. Gaussian component selection via (9);
6. Denoising by Weighted Sparse Coding (15);
7. Recover each patch in this PG via $\hat{\mathbf{x}}_m = \mathbf{D}\hat{\boldsymbol{\alpha}} + \boldsymbol{\mu}_y$;

end for

8. Aggregate the recovered PGs to form the recovered image $\hat{\mathbf{x}}^{(t)}$;

end for

Output: The recovered image $\hat{\mathbf{x}}^{(IteNum)}$.

3.2. Denoising Algorithm

With the solution $\hat{\boldsymbol{\alpha}}$ in (17), the clean patch in a PG can be estimated as $\hat{\mathbf{x}}_m = \mathbf{D}\hat{\boldsymbol{\alpha}} + \boldsymbol{\mu}_y$. Then the clean image $\hat{\mathbf{x}}$ can be reconstructed by aggregating all the estimated PGs. In practice, we could perform the above denoising procedures for several iterations for better denoising outputs. In iteration t, we use the iterative regularization strategy [4] to add back to the recovered image $\hat{\mathbf{x}}^{(t-1)}$ some estimation residual in iteration $t-1$. The standard deviation of noise in iteration t is adjusted as $\sigma^{(t)} = \eta * \sqrt{\sigma^2 - \|\mathbf{y} - \mathbf{y}^{(t-1)}\|_2^2}$, where η is a constant. The proposed denoising algorithm is summarized in Algorithm 1 (Alg. 1).

In the proposed algorithm, there are N PGs in an image and M patches in each PG. Then the computational cost for Gaussian component selection is $O(p^6NMK)$. The cost for iterative regularization and noise estimation is negligible. The cost for closed-form weighted sparse coding is $O(p^4NM)$. Suppose that there are T iterations, the overall complexity of our denoising algorithm is $O(p^6NMKT)$.

4. Experiments

In this section, we perform image denoising experiments on 20 widely used natural images (shown in Fig. 5). More experiments on the Berkeley Segmentation Data Set [33] can be found in the supplementary file.

Figure 5. The 20 widely used test images.

As a common experimental setting in literature, additive white Gaussian noise with zero mean and standard deviation σ is added to the image to test the performance of competing denoising methods. We call our method *PG Prior based Denoising* (PGPD) in the following experiments. The Matlab source code of our PGPD algorithm can be downloaded at http://www4.comp.polyu.edu.hk/~cslzhang/code/PGPD.zip.

4.1. Implementation Details

Our proposed PGPD method contains two stages, the prior learning stage and the denoising stage. In the PG-GMM learning stage, there are 4 parameters: p, M, W and K. The patch size ($p \times p$) is set as $p = 6$ for $0 < \sigma \leq 20$, $p = 7$ for $20 < \sigma \leq 30$, $p = 8$ for $30 < \sigma \leq 50$, and $p = 9$ for $50 < \sigma \leq 100$. The window size (W) for PG searching is set to $W = 31$. The number (M) of patches in a PG is set to $M = 10$. The number (K) of Gaussian components is set to $K = 64$ for $p = 6$ and $K = 32$ otherwise. We extracted about one million PGs from the Kodak PhotoCD Dataset to train the PG-GMM.

In the denoising stage, there are 3 parameters: c, δ, and η. In our implementation, (c, δ, η) are set to $(0.33, 0.10, 0.79)$, $(0.29, 0.09, 0.73)$, $(0.19, 0.08, 0.89)$, $(0.15, 0.07, 0.98)$, $(0.12, 0.06, 1.05)$, $(0.09, 0.05, 1.15)$, $(0.06, 0.05, 1.30)$ when $\sigma = 10, 20, 30, 40, 50, 75, 100$, respectively. In addition, on all noise levels we stop Algorithm 1 in 4 iterations.

4.2. Comparison Methods

We compare the proposed PGPD algorithm with BM3D [12], EPLL [26], LSSC [9], NCSR [10], and WNNM [14], which represent the state-of-the-arts of modern image denoising techniques and all of them exploit image NSS. The source codes of all competing algorithms are downloaded from the authors' websites and we use the default parameter settings.

To more clearly demonstrate the effectiveness of PG based NSS prior learning, we also compare with an extreme case of PGPD, i.e., letting $M = 1$ in the PG-GMM learning stage[1]. Clearly, this reduces to a patch based prior learning scheme and no NSS prior will be learned. We call this extreme case as *Patch Prior based Denoising* (PPD). The number of Gaussian components in PPD is set to 64, and the weighted sparse coding framework in it is the same as that in PGPD. All the other parameters in PPD are tuned to achieve its best performance.

[1]Since there is only 1 patch in the PG, the group mean vector cannot be subtracted and we subtract the mean value of the patch from it.

4.3. Results and Discussions

We evaluate the competing methods from three aspects: PSNR, Speed, and Visual Quality.

PSNR. In Table 1, we present the PSNR results on four noise levels $\sigma = 30, 40, 50, 75$. The results on noise levels $\sigma = 10, 20, 100$ can be found in the supplementary material. From Table 1, we have several observations. Firstly, PGPD achieves much better PSNR results than PPD. The improvements are 0.24∼0.52dB on average. This clearly demonstrates the effectiveness of PG-GMM in NSS prior learning. Secondly, PGPD has higher PSNR values than BM3D, LSSC, EPLL and NCSR, and is only slightly inferior to WNNM. However, PGPD is much more efficient than WNNM (see next paragraph). This validates the strong ability of PG based NSS prior in image denoising.

Speed. Efficiency is another important factor to evaluate an algorithm. We then compare the speed of all competing methods. All experiments are run under the Matlab2014b environment on a machine with Intel(R) Core(TM) i7-4770K CPU of 3.50GHz and 12.0 GB RAM. The run time (s) of competing methods on the test images is shown in Table 2. One can easily see that BM3D is the fastest method. The proposed PGPD is the second fastest, and it is much faster than the other methods. For a 256×256 image, BM3D costs about 0.8s while PGPD costs about 10s. However, please note that BM3D is implemented with compiled C++ mex-function and with parallelization, while PGPD is implemented purely in Matlab. EPLL is about 4 times slower than PGPD. Both LSSC and NCSR are very slow since they need to train online dictionary. Though WNNM has the highest PSNR, it suffers from huge computational cost due to the many online SVD operations. It is 10∼16 times slower than PGPD.

Visual Quality. Considering that human subjects are the ultimate judge of the image quality, the visual quality of denoised images is also critical to evaluate a denoising algorithm. Fig. 6 and Fig. 7 show the denoised images of *Airplane* and *Cameraman* by the competing methods, respectively. Due to the page limit, the results of PPD are not shown here, and more visual comparisons can be found in the supplementary file. We can see that BM3D tends to over-smooth the image, while EPLL, LSSC, NCSR and WNNM are likely to generate artifacts when noise is high. Owe to the learned NSS prior, the proposed PGPD method is more robust against artifacts, and it preserves edge and texture areas much better than the other methods. For example, in image *Airplane*, PGPD reconstructs the numbers "01568" more clearly than all the other methods including WNNM. In image *Cameraman*, PGPD recoveries more faithfully the fine structures of the camera area.

In summary, the proposed PGPD method demonstrates powerful denoising ability quantitatively and qualitatively, and it is highly efficient.

Table 1. PSNR(dB) results of different denoising algorithms on 20 natural images.

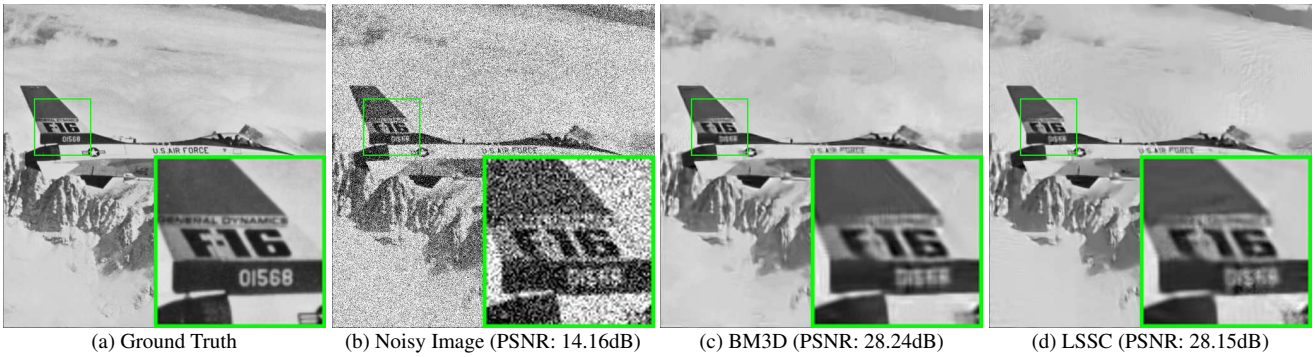
	$\sigma = 30$							$\sigma = 40$						
Images	**BM3D**	**LSSC**	**EPLL**	**NCSR**	**WNNM**	**PPD**	**PGPD**	**BM3D**	**LSSC**	**EPLL**	**NCSR**	**WNNM**	**PPD**	**PGPD**
Airfield	26.41	26.68	26.52	26.36	26.67	26.33	26.46	25.10	25.51	25.36	25.07	25.48	25.20	25.30
Airplane	30.71	30.62	30.68	30.70	30.97	30.62	30.80	29.20	29.21	29.28	29.28	29.58	29.21	29.44
Baboon	24.57	24.78	24.70	24.63	24.85	24.54	24.63	23.11	23.51	23.35	23.28	23.58	23.23	23.39
Barbara	29.81	29.60	27.64	29.62	30.31	27.97	29.38	27.99	28.17	26.06	28.20	28.76	26.29	27.97
Boat	29.12	29.06	28.97	28.94	29.24	28.80	29.05	27.74	27.77	27.72	27.65	27.96	27.51	27.82
C. Man	28.64	28.63	28.40	28.58	28.80	28.25	28.53	27.18	27.34	27.10	27.12	27.47	27.05	27.33
Carhouse	28.78	28.79	28.70	28.72	28.94	28.62	28.80	27.38	27.49	27.38	27.40	27.58	27.29	27.51
Couple	28.87	28.76	28.69	28.57	28.98	28.54	28.84	27.48	27.41	27.34	27.24	27.62	27.16	27.53
Elaine	30.45	30.54	30.26	30.26	30.46	30.24	30.37	29.52	29.55	29.46	29.59	29.60	29.42	29.62
Hat	29.37	29.22	29.22	29.16	29.44	29.05	29.31	27.74	27.60	27.73	27.66	27.85	27.43	27.90
Hill	29.16	29.09	28.94	28.97	29.25	28.85	29.09	27.99	28.00	27.86	27.83	28.12	27.76	28.06
House	32.09	32.40	31.48	32.07	32.52	31.62	32.24	30.65	31.10	30.20	30.80	31.31	30.32	31.02
Lake	28.34	28.36	28.41	28.31	28.59	28.30	28.38	26.98	27.13	27.19	26.99	27.34	27.03	27.15
Leaves	27.81	27.65	27.36	28.14	28.60	27.51	27.99	25.69	26.04	25.80	26.24	26.95	25.88	26.29
Lena	31.26	31.18	30.98	31.06	31.43	30.98	31.27	29.86	29.91	29.69	29.92	30.11	29.67	30.10
Man	28.86	28.87	28.87	28.78	29.00	28.72	28.86	27.65	27.64	27.68	27.54	27.80	27.53	27.73
Monarch	28.36	28.20	28.50	28.46	28.91	28.27	28.49	26.72	26.87	27.05	26.85	27.47	26.81	27.02
Paint	28.29	28.29	28.45	28.10	28.58	28.39	28.42	26.69	26.77	27.00	26.50	27.10	26.88	26.94
Peppers	31.26	31.17	31.10	31.11	31.38	31.13	31.25	29.97	30.00	29.93	30.07	30.18	29.95	30.18
Zelda	30.45	30.27	30.44	30.16	30.48	30.35	30.43	29.10	28.91	29.18	28.94	29.12	29.07	29.23
Average	29.13	29.11	28.92	29.03	29.37	28.85	29.13	27.69	27.80	27.62	27.71	28.05	27.53	27.88
	$\sigma = 50$							$\sigma = 75$						
Images	**BM3D**	**LSSC**	**EPLL**	**NCSR**	**WNNM**	**PPD**	**PGPD**	**BM3D**	**LSSC**	**EPLL**	**NCSR**	**WNNM**	**PPD**	**PGPD**
Airfield	24.20	24.58	24.46	24.18	24.51	24.33	24.44	22.71	22.85	22.85	22.57	22.94	22.69	22.90
Airplane	28.24	28.15	28.19	28.18	28.55	28.10	28.38	26.40	26.16	26.14	26.10	26.68	25.90	26.39
Baboon	22.35	22.60	22.35	22.43	22.73	22.30	22.47	21.11	21.18	20.85	21.03	21.36	20.71	21.09
Barbara	27.23	27.03	24.83	26.99	27.79	24.94	26.81	25.12	25.01	22.94	24.72	25.81	22.84	24.84
Boat	26.78	26.77	26.74	26.67	26.97	26.52	26.85	25.12	25.03	25.01	24.87	25.29	24.72	25.19
C. Man	26.12	26.35	26.10	26.15	26.42	26.13	26.46	24.33	24.41	24.29	24.22	24.55	24.36	24.64
Carhouse	26.53	26.48	26.39	26.41	26.67	26.27	26.53	24.89	24.85	24.65	24.53	25.04	24.44	24.85
Couple	26.46	26.35	26.30	26.19	26.65	26.07	26.50	24.70	24.51	24.51	24.33	24.85	24.22	24.70
Elaine	28.94	28.75	28.77	28.85	28.97	28.69	28.90	27.41	27.27	27.38	27.16	27.53	27.26	27.47
Hat	26.77	26.41	26.62	26.51	26.78	26.28	26.76	24.77	24.31	24.65	24.48	24.77	24.19	24.79
Hill	27.19	27.14	27.04	26.99	27.34	26.91	27.22	25.68	25.57	25.60	25.40	25.88	25.34	25.73
House	29.69	29.99	29.12	29.62	30.32	29.17	29.93	27.51	27.75	27.09	27.22	28.25	26.81	27.81
Lake	26.13	26.15	26.24	26.02	26.41	26.05	26.20	24.49	24.25	24.50	24.26	24.66	24.19	24.49
Leaves	24.68	24.78	24.55	24.96	25.47	24.56	25.03	22.49	22.17	22.12	22.60	23.06	21.94	22.61
Lena	29.05	28.95	28.68	28.90	29.25	28.61	29.11	27.26	27.22	26.88	27.00	27.54	26.68	27.40
Man	26.81	26.72	26.79	26.67	26.94	26.63	26.86	25.32	25.10	25.26	25.10	25.42	25.01	25.36
Monarch	25.82	25.88	25.94	25.76	26.31	25.66	26.00	23.91	23.66	23.88	23.67	24.31	23.51	24.00
Paint	25.67	25.59	25.87	25.36	25.98	25.70	25.82	23.80	23.52	23.88	23.44	24.07	23.50	23.89
Peppers	29.12	29.06	28.98	29.07	29.34	28.99	29.22	27.28	27.14	27.15	26.96	27.55	27.04	27.42
Zelda	28.25	27.90	28.22	27.97	28.21	28.06	28.24	26.60	26.09	26.55	26.21	26.44	26.37	26.56
Average	26.80	26.78	26.61	26.69	27.08	26.50	26.89	25.04	24.90	24.81	24.79	25.30	24.59	25.11

Table 2. Average run time (seconds) with standard deviation of different methods on images of size 256×256 and 512×512. BM3D uses parallelization and is implemented with compiled C++ mex-function while the other methods are implemented in Matlab.

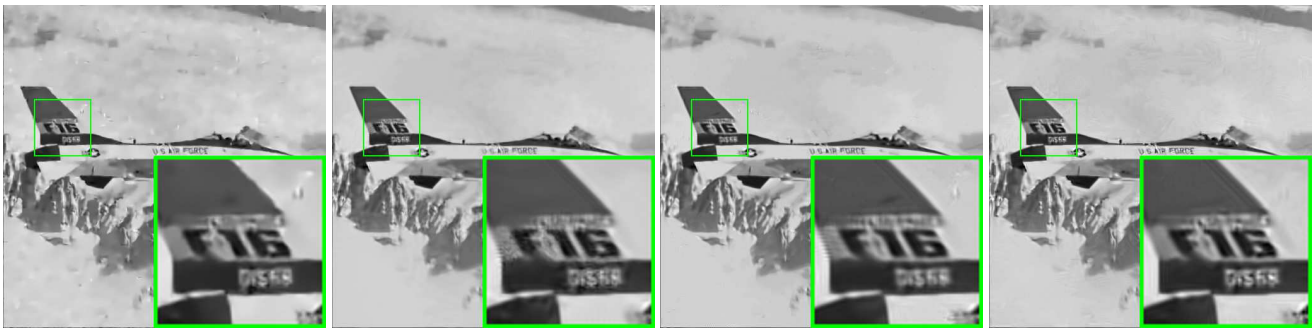
	256×256						
σ	**BM3D**	**LSSC**	**EPLL**	**NCSR**	**WNNM**	**PPD**	**PGPD**
10	0.67 ± 0.09	186.90 ± 4.02	38.47 ± 0.10	126.43 ± 3.84	84.34 ± 1.42	10.15 ± 0.07	8.00 ± 0.05
20	0.70 ± 0.09	184.21 ± 5.82	38.47 ± 0.13	156.14 ± 5.26	84.70 ± 1.71	10.18 ± 0.15	8.09 ± 0.09
30	0.70 ± 0.09	212.07 ± 8.72	38.55 ± 0.09	149.31 ± 4.19	155.75 ± 0.94	10.34 ± 0.25	8.47 ± 0.07
40	0.67 ± 0.11	209.13 ± 6.99	38.51 ± 0.08	346.91 ± 18.65	157.35 ± 1.48	10.47 ± 0.21	9.80 ± 0.08
50	0.87 ± 0.04	221.36 ± 6.27	40.21 ± 1.82	326.93 ± 9.64	119.47 ± 4.65	10.88 ± 0.05	9.91 ± 0.13
75	0.89 ± 0.03	240.75 ± 6.08	40.91 ± 1.33	258.04 ± 11.80	179.30 ± 5.08	10.87 ± 0.27	11.73 ± 0.08
100	0.90 ± 0.03	257.25 ± 6.01	42.80 ± 1.93	252.74 ± 8.50	191.32 ± 1.47	10.90 ± 0.19	11.78 ± 0.08
	512×512						
σ	**BM3D**	**LSSC**	**EPLL**	**NCSR**	**WNNM**	**PPD**	**PGPD**
10	3.16 ± 0.12	746.53 ± 24.96	160.93 ± 2.81	624.83 ± 40.24	352.34 ± 3.87	41.79 ± 0.32	33.03 ± 0.25
20	3.32 ± 0.11	762.62 ± 31.25	159.80 ± 0.37	751.09 ± 42.89	351.09 ± 3.14	42.09 ± 0.41	33.26 ± 0.29
30	3.32 ± 0.09	856.82 ± 40.32	160.21 ± 0.18	709.90 ± 31.62	650.54 ± 7.23	42.36 ± 0.99	35.45 ± 0.24
40	3.18 ± 0.18	865.83 ± 40.96	160.23 ± 0.17	1620.74 ± 104.59	652.49 ± 10.49	41.70 ± 0.47	40.13 ± 0.23
50	3.85 ± 0.09	891.53 ± 48.60	161.36 ± 3.08	1492.78 ± 65.87	476.50 ± 12.34	41.75 ± 0.64	40.40 ± 0.28
75	3.91 ± 0.05	983.05 ± 69.96	165.66 ± 2.62	1156.82 ± 66.37	784.92 ± 18.32	41.88 ± 0.78	50.00 ± 0.25
100	3.94 ± 0.04	1087.57 ± 68.76	177.51 ± 7.16	1100.00 ± 26.64	824.56 ± 34.41	42.80 ± 1.09	50.32 ± 0.31

5. Conclusion

How to learn explicit models of nonlocal self-similarity (NSS) prior for image restoration is an open problem, and we made a good attempt on this by lifting the patch based image modeling to patch group (PG) based image modeling. A PG is a group of similar patches in an image re-

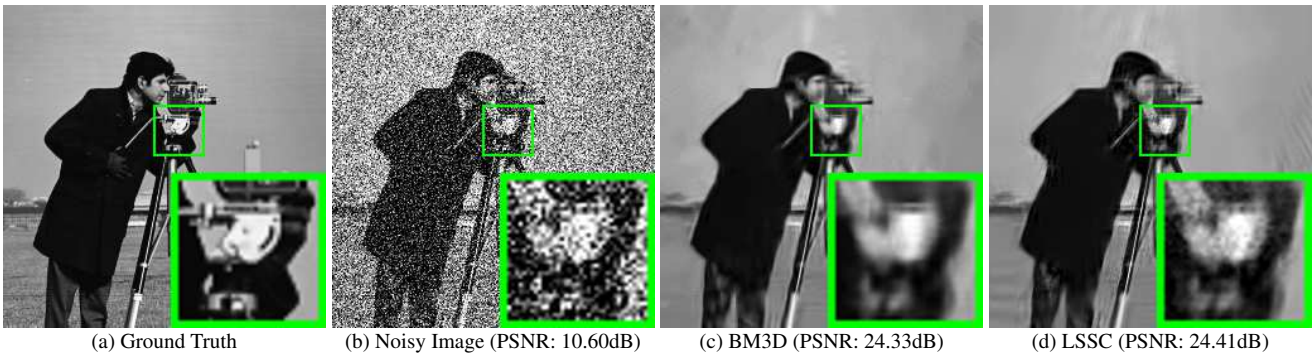


(a) Ground Truth (b) Noisy Image (PSNR: 14.16dB) (c) BM3D (PSNR: 28.24dB) (d) LSSC (PSNR: 28.15dB)

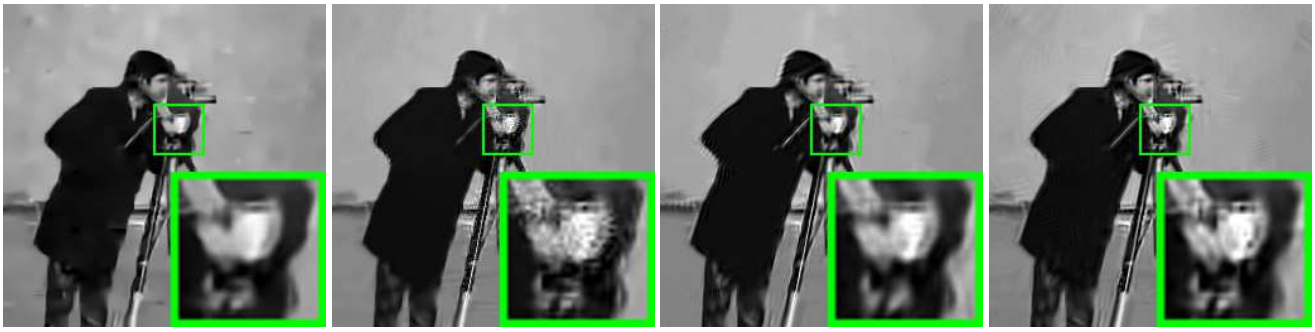


(e) EPLL (PSNR: 28.19dB) (f) NCSR (PSNR: 28.18dB) (g) WNNM (PSNR: 28.55dB) (h) PGPD (PSNR: 28.38dB)

Figure 6. Denoised images of *Airplane* by different methods (the standard deviation of noise is $\sigma = 50$).



(a) Ground Truth (b) Noisy Image (PSNR: 10.60dB) (c) BM3D (PSNR: 24.33dB) (d) LSSC (PSNR: 24.41dB)



(e) EPLL (PSNR: 24.29dB) (f) NCSR (PSNR: 24.22dB) (g) WNNM (PSNR: 24.55dB) (h) PGPD (PSNR: 24.64dB)

Figure 7. Denoised images of *Cameraman* by different methods (the standard deviation of noise is $\sigma = 75$).

gion. After group mean subtraction, a PG can naturally represent the NSS variations of natural images. A PG based Gaussian Mixture Model (PG-GMM) learning algorithm was developed to learned the NSS prior from natural images, and an associated weighted sparse coding algorithm was developed for high performance image denoising. The so-called *PG Prior based Denoising* (PGPD) algorithm not only achieves highly competitive PSNR results with state-of-the-art denoising methods, but also is highly efficient and preserves better the image edges and textures. The proposed method can be extended to other image processing tasks such as deblurring and super-resolution.

References

[1] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *ICCV*, pages 839–846, 1998. 1

[2] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990. 1

[3] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992. 1

[4] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005. 1, 5

[5] D. L. Donoho. De-noising by soft-thresholding. *IEEE Trans. Inf. Theor.*, 41(3):613–627, 1995. 1

[6] S. G. Chang, B. Yu, and M. Vetterli. Adaptive wavelet thresholding for image denoising and compression. *Image Processing, IEEE Transactions on*, 9(9):1532–1546, 2000. 1

[7] J. L. Starck, E. J. Candès, and D. L. Donoho. The curvelet transform for image denoising. *Image Processing, IEEE Transactions on*, 11(6):670–684, 2002. 1

[8] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, 2006. 1, 3

[9] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. *ICCV*, pages 2272–2279, 2009. 1, 2, 3, 6

[10] W. Dong, L. Zhang, G. Shi, and X. Li. Nonlocally centralized sparse representation for image restoration. *Image Processing, IEEE Transactions on*, 22(4):1620–1630, 2013. 1, 2, 3, 6

[11] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. *CVPR*, pages 60–65, 2005. 1, 2, 3

[12] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095, 2007. 1, 2, 3, 4, 6

[13] H. Ji, C. Liu, Z. Shen, and Y. Xu. Robust video denoising using low rank matrix completion. *CVPR*, pages 1791–1798, 2010. 1, 2

[14] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted nuclear norm minimization with application to image denoising. *CVPR*, pages 2862–2869, 2014. 1, 2, 3, 6

[15] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *Image Processing, IEEE Transactions on*, 12(11):1338–1351, 2003. 1

[16] Y. Weiss and W. T. Freeman. What makes a good model of natural images? *CVPR*, pages 1–8, 2007. 1

[17] S. Roth and M. J. Black. Fields of Experts. *International Journal of Computer Vision*, 82(2):205–229, 2009. 1

[18] C. M. Bishop. *Pattern recognition and machine learning*. New York: Springer, 2006. 1, 3

[19] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2774–2781, June 2014. 1

[20] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996. 1

[21] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997. 1

[22] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–4322, 2006. 1

[23] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on*, 17(1):53–69, 2008. 1, 3

[24] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010. 1

[25] Z. Jiang, Z. Lin, and L. S. Davis. Label consistent k-svd: Learning a discriminative dictionary for recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2651–2664, 2013. 1

[26] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. *ICCV*, pages 479–486, 2011. 1, 3, 4, 6

[27] D. Zoran and Y. Weiss. Natural images, Gaussian mixtures and dead leaves. *NIPS*, pages 1736–1744, 2012. 1

[28] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity. *Image Processing, IEEE Transactions on*, 21(5):2481–2499, 2012. 1, 3, 4

[29] G. Peyré, S. Bougleux, and L. D. Cohen. Non-local regularization of inverse problems. *Inverse Problems and Imaging*, 5(2):511–530, 2011. 2

[30] S. Wang, L. Zhang, and Y. Liang. Nonlocal spectral prior model for low-level vision. *ACCV*, pages 231–244, 2013. 2

[31] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977. 3

[32] C. F. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, pages 95–103, 1983. 3

[33] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011. 5