11-2015

Dictionary Pair Learning on Grassmann Manifolds for Image Denoising

Xianhua ZENG
Chongqing University of Posts and Telecommunications

Wei BIAN
University of Technology, Sydney

Wei LIU
Xidian University

Jialie SHEN
Singapore Management University, jlshen@smu.edu.sg

Dacheng TAO
University of Technology, Sydney

DOI: https://doi.org/10.1109/TIP.2015.2468172

Published in IEEE Transactions on Image Processing, Nov. 2015, Volume: 24, Issue: 11, pp. 4556 - 4569.
http://doi.org/10.1109/TIP.2015.2468172

Dictionary Pair Learning on Grassmann Manifolds for Image Denoising

Xianhua Zeng, Wei Bian, *Member, IEEE*, Wei Liu, *Member, IEEE*, Jialie Shen, and Dacheng Tao, *Fellow, IEEE*

***Abstract*— Image denoising is a fundamental problem in computer vision and image processing that holds considerable practical importance for real-world applications. The traditional patch-based and sparse coding-driven image denoising methods convert 2D image patches into 1D vectors for further processing. Thus, these methods inevitably break down the inherent 2D geometric structure of natural images. To overcome this limitation pertaining to the previous image denoising methods, we propose a 2D image denoising model, namely, the dictionary pair learning (DPL) model, and we design a corresponding algorithm called the DPL on the Grassmann-manifold (DPLG) algorithm. The DPLG algorithm first learns an initial dictionary pair (i.e., the left and right dictionaries) by employing a subspace partition technique on the Grassmann manifold, wherein the refined dictionary pair is obtained through a sub-dictionary pair merging. The DPLG obtains a sparse representation by encoding each image patch only with the selected sub-dictionary pair. The non-zero elements of the sparse representation are further smoothed by the graph Laplacian operator to remove the noise. Consequently, the DPLG algorithm not only preserves the inherent 2D geometric structure of natural images but also performs manifold smoothing in the 2D sparse coding space. We demonstrate that the DPLG algorithm also improves the structural SIMilarity values of the perceptual visual quality for denoised images using the experimental evaluations on the benchmark images and Berkeley segmentation data sets. Moreover, the DPLG also produces the competitive peak signal-to-noise ratio values from popular image denoising algorithms.**

***Index Terms*— Image denoising, dictionary pair, 2D sparse coding, Grassmann manifold, smoothing, graph Laplacian operator.**

Manuscript received December 2, 2014; revised May 18, 2015 and July 3, 2015; accepted August 4, 2015. Date of publication August 13, 2015; date of current version August 31, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61075019 and Grant 61379114, in part by the State Key Program of National Natural Science Foundation of China under Grant U1401252, in part by the Chongqing Natural Science Foundation under Grant cstc2015jcyjA40036, and in part by the Australian Research Council under Project FT-130101457, Project DP-140102164, and Project LP-140100569. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Nilanjan Ray.

X. Zeng is with the Chongqing Key Laboratory of Computational Intelligence, College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: xianhuazeng@gmail.com).

W. Bian and D. Tao are with the Centre for Quantum Computation and Intelligent Systems, and the Faculty of Engineering and Information Technology, University of Technology, Sydney, 81 Broadway Street, Ultimo, NSW 2007, Australia (e-mail: wei.bian, dacheng.tao@uts.edu.au).

W. Liu is with the School of Electronic Engineering, Xidian University, Xi'an 710071, China (e-mail: wliu.cu@gmail.com).

J. Shen is with the School of Information Systems, Singapore Management University, Singapore 178902 (e-mail: jlshen@smu.edu.sg).

I. Introduction

An IMAGE is usually corrupted by noise during the processes of being captured, recorded and transmitted. One general assumption is that an observed noisy image x is generated by adding a Gaussian noise corruption to the original clear image y, that is,

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

where υ is the additive white Gaussian noise with a mean of zero and a standard deviation σ.

Image denoising plays an important role in the fields of computer vision [1], [2] and image processing [3], [4]. Its goal is to restore the original clear image y from the observed noisy image x, which amounts to finding an inverse transformation from the noisy image to the original clear image. Over the past decades, many denoising methods have been proposed for reconstructing the original image from the observed noisy image by exploiting the inherently spatial correlations [5]–[12]. The image denoising methods are generally divided into three categories including (i) internal denoising methods (e.g., BM3D [5], K-SVD [11], NCSR [12]): using only the noisy image patches from a single noisy image; (ii) external denoising methods (e.g., SSDA [13], SDAE [14]): training the mapping from noisy images to clean images using only external clean image patches; and (iii) internal-external denoising methods (e.g. SCLW [15], NSCDL [16]): jointly using the external statistics information from a clean training image set and the internal statistics information from the observed noisy image. To the best of our knowledge, among these methods, BM3D [10] is considered to be the current state of the art in the image denoising area over the past several years. BM3D combines two classical techniques, non-local similarity and domain transformation. However, BM3D is a complex engineering method and has many tunable parameters, such as the choices of bases, patch-size, transformation thresholds, and similarity measures.

In recent years, machine learning techniques based on domain transformation have gained popularity and success in terms of a good denoising performance [11], [12], [14]–[16]. For example, K-SVD [11] is one of the most well-known and effective denoising methods that apply machine learning techniques. This method assumes that a clear image patch can be represented as a sparse linear combination of the atoms from an over-complete dictionary. Hence, the K-SVD method denoises a noisy image by approximating the noisy patch using a sparse linear combination of atoms, which is formulated as minimizing the following objective function:

$$\underset{D,\alpha_i}{argMin} \sum_i \{\|D\alpha_i - X_i\|^2 + \|\alpha_i\|_1\}, \tag{2}$$

where D is an over-complete dictionary and each column therein corresponds to an atom, and α_i is the sparse coding coefficient combination of all atoms for reconstructing the clean image patch from the noisy image patch X_i under the convex sparse priori regularization constraint $\|.\|_1$.

However, the above dictionary D is not easy to learn, and the corresponding denoising model uses a 1D vector, rather than the original 2D matrix to represent each image patch. Additionally, regarding the K-SVD basis, several effective, adaptive denoising methods, such as [11], [12], and [17]–[19] were also proposed in the theme of converting image patches into 1D vectors and clustering noisy image patches into regions with similar geometric structures. Taking the NCSR algorithm [12] as a classical example, it unifies both priors in image local sparsity and nonlocal similarity via a clustering-based sparse representation. The NCSR algorithm incorporates considerable prior information to improve the denoising performance through introducing sparse coding noise, (i.e., the third regularization term of the following model, which is an extension of the model in Eq.(2)) as follows:

$$\underset{D,\alpha_i}{argMin} \sum_i \{\|D\alpha_i - X_i\|^2 + \lambda\|a_i\|_1 + \gamma\,\|\alpha_i - \beta_i\|\}, \tag{3}$$

where β_i is a good estimation of the sparse codes α_i, and λ and γ are the balance factors of two regularization terms (i.e., the convex sparse regularization term and sparse coding noise term).

In the NCSR model, while enforcing the sparsity of coding coefficients, the sparse codes α_i's are also centralized to attain a good estimations β_i's. Dictionary D is acquired by adopting an adaptive sparse domain selection strategy, which executes K-Means clustering and then learns a PCA sub-dictionary for each cluster. Nevertheless, this strategy still needs to convert the noisy image patches into 1D vectors, so good estimations β_i's are difficult to obtain.

To summarize, almost all patch-based and sparse coding-driven image denoising methods convert raw, 2D matrix representations of image patches into 1D vectors for further processing, and thereby break down the inherent 2D geometric structure of the natural images. Moreover, the learned dictionary and sparse coding representations cannot capture the intrinsic position correlations between the pixels within each image patch. On the one hand, to preserve the 2D geometric structure of image patches in the transformation domain, a bilinear transformation is particularly appropriate (for image patches in the matrix representation) for extracting the semantic features of the rows and columns from the image matrixes [20], which is similar to 2DPCA [21] on two directions or can also be viewed as a special case of some existing tensor feature extraction methods such as TDCS [22], STDCS [23] and HOSVD [24]. On the other hand, we assume that image patches sampled from a denoised image lie on an intrinsic smooth manifold. However, the noisy image patches almost never exactly lie on the same manifold due to noise. A related work [26] shows that the manifold smoothing is a usual trick for effectively removing the noise. The weighted neighborhood graph, constructed from image patches, can approximate the intrinsic manifold structure. The graph Laplacian operator is the generator of the smoothing process on the neighborhood graph [25]. Therefore, the recent promising graph Laplacian operator, in [26]–[29] and [31], for approximating the manifold structure is leveraged as a generic smooth regularizer while removing the noise of 2D image patches based on the sparse coding model.

With the above considerations, we propose a Dictionary Pair Learning model (DPL model) for image denoising. In the DPL model, the dictionary pair is used to capture the semantic features of 2D image patches, and the graph Laplacian operator guarantees a disciplined smoothing according to the image patch geometric distribution in the 2D sparse coding space. However, we will face the NP-hardness of directly solving the dictionary pair and the 2D sparse coding matrixes for image denoising. In the NCSR model, the vectorized image patches are clustered into K subsets by K-means, and then one compact PCA sub-dictionary for each cluster is used. So, in our DPL model, 2D image patches can, of course, be clustered into some subsets with nonlocal similarities. The 2D patches in a subset are very similar to each other. Obviously, one needs only to extend the PCA sub-dictionary to a 2DPCA sub-dictionary for each cluster. However, the 2D image patches sampled from the noisy image with a multi-resolution and sliding window in our DPL model are of a high quantity and have a non-linear distribution, such that clustering faces a serious computational challenge. Fortunately, the literature [30] proposed a Subspace Indexing Model on Grassmann Manifold (SIM-GM) that can top-to-bottom partition the non-linear space into local subspaces with a hierarchical tree structure. Mathematically, a Grassmann manifold is the set of all linear subspaces with a fixed dimension [32], [33], and so an extracted PCA subspace in each leaf node of the SIM-GM model corresponds to a point on a Grassmann manifold. To obtaining the most effective local space, introducing the Grassmann manifold distances (i.e., the angles between linear subspaces [34]), the SIM-GM is able to automatically manipulate the leaf nodes in the data partition tree and build the most effective local subspace by using a bottom-up merging strategy. Thus, by extending the kind of PCA subspace partitioning on a Grassmann manifold to a 2DPCA subspace pair partitioning on two Grassmann manifolds, we propose a Dictionary Pair Learning algorithm on Grassmann-manifolds (DPLG algorithm in shorthand). Experimental results on benchmark images and Berkeley segmentation datasets show that the proposed DPLG algorithm is more competitive than the state-of-the-art image denoising methods including the internal denoising methods and the external denoising methods.

The rest of this paper is organized as follows: In Section II, we build a novel dictionary pair learning model for 2D image denoising. Section III first analyzes the learning methods of the dictionary pair and sparse coding matrixes, and then summarizes the dictionary pair learning algorithm on Grassmann-manifolds for image denoising. In Section IV, a series of experimental results are shown, and we present the concluding remarks and future work in Section V.

II. Dictionary Pair Learning Model

According to the above discussion and analysis, to preserve the original 2D geometric structure and to construct a sparse coding model for image denoising, the 2D noisy image patches are encoded by projections on a dictionary pair that correspond to left multiplying a matrix and right multiplying a matrix. Then by exploiting sparse coding and graph Laplacian operator smoothing to remove noises, we design a Dictionary Pair Learning model (DPL model) for image denoising in this section.

A. Dictionary Pair Learning Model for 2D Sparse Coding

To preserve the 2D geometrical structure with sparse sensing in the transformation domain, we need only to find two linear transformations for simultaneously mapping the columns and rows of image patches under the sparse constraint. Let the image patches set be $\{X_1, X_2, \ldots, X_i, \ldots, X_n\}, X_i \in \Re^{M \times N}$; our method computes the left and right 2D linear transformations to map the image patches into the 2D sparse matrix space. Thus, the corresponding objective function may be defined as follows:

$$\underset{A,B,S}{argMin} \sum_i \{\|A^T X_i B - S_i\|_F + \lambda \|S_i\|_{F,1}\}, \tag{4}$$

where $A \in \Re^{M \times M1}$ and $B \in \Re^{N \times N1}$ are respectively called the left coding dictionary and the right coding dictionary, $S = \{S_i\}, S_i \in \Re^{M1 \times N1}$ is the sparse coefficient matrix, λ is the regularization parameter, $\|.\|_F$ denotes the matrix Frobenious norm, and $\|.\|_{F,1}$ denotes the matrix L_1-norm which is defined as the sum of the absolute values of all its entries.

In this paper, the left and right coding dictionaries are combined and called as the dictionary pair $\langle A, B \rangle$. Once the dictionary pair and the sparse representations are learned, especially, the left and right dictionaries constrained by block orthogonality, each patch X_i can be reconstructed by multiplying the selected sub-dictionary pair $\langle A_{ki}, B_{ki} \rangle$ with its sparse representation, that is:

$$X_i \approx A_{ki} S_i B_{ki}^T, \tag{5}$$

where the orthogonal sub-dictionaries A_{ki}, B_{ki} are selected to code the image patch X_i, and ki is the index of the selected sub-dictionary pair. Note that the selection method of the $ki - th$ dictionary pair is described in Section III-B.

B. Graph Laplacian Operator Smoothing

Nonlocal smoothing and co-sparsity are the prevailing techniques for removing noises. Clearly, a natural assumption is that the coding matrixes of similar patches should be similar. If similar image patches are encoded only on a sub-dictionary pair of the learned dictionary pair, then, exploiting the graph Laplacian as a smoothing operator, both smoothing and co-sparsity can be simultaneously guaranteed while minimizing a penalty term on the weighted L_1-norm divergence between the coding matrix of a given image patch and those coding matrixes of its nonlocal neighborhood patches, as in:

$$\sum_{i,j} w_{ij} \|S_i - S_j\|_{F,1}, \tag{6}$$

where w_{ij} is the similarity between the $i - th$ patch and its $j - th$ neighbor.

According to our previous research in manifold learning, a patch similarity metric is selected to apply the generalized Gaussian kernel function in literature [31]:

$$w_{ij} = \begin{cases} \frac{1}{\Gamma} exp\left(-\left(\|X_i - X_j\|_F / 2\sigma_i\right)^\tau\right) \\ \quad if\ X_j\ is\ k-nearest\ neighbors\ of\ X_i, \\ 0, \quad otherwise. \end{cases} \tag{7}$$

where Γ is the normalization factor, σ_i is the variance of neighborhood distribution and τ is the generalization Gaussian exponent. In this paper, the neighborhood similarity is assumed to obey the super-Gaussian distribution:

$$w_{ij} = \frac{1}{\Gamma} exp\left(-\left(\|X_i - X_j\|_{F,1} / \sqrt{2}\sigma_i\right)\right). \tag{8}$$

C. The Final Objective Function

Combining the sparse coding term in Eq. (4) and the smoothing term in Eq. (6), the final objective function of the DPL model is defined as follows:

$$\begin{cases} \underset{A,B,S}{argMin} \sum_i \{\|A^T X_i B - S_i\|_F + \lambda \sum_i \|S_i\|_{F,1} \\ \qquad\qquad + \gamma \sum_{i,j} w_{ij} \|S_i - S_j\|_{F,1}\}, \\ S.t. \begin{cases} (1)\ \sum_j w_{ij} = 1 \\ (2)\ A_k^T A_k = I, \quad B_k^T B_k = I, \quad k = 1, \ldots, K \end{cases} \end{cases} \tag{9}$$

where $\|.\|_{F,1}$ denotes the matrix L_1-norm which is defined as the sum of the absolute values of all matrix elements, and A and B are constrained to be block orthogonal matrices in the following learning algorithm.

The above Eq. (9) is an accurate description of the Dictionary Pair Learning model (DPL model), and Fig. 1 shows an illustration of the DPL model. In the DPL model, two similar 2D image patches, X_i and X_j, extracted from the given noisy image are encoded on two dictionaries (i.e., the left dictionary A and the right dictionary B), which are respectively consisted of sub-dictionary sets $A = \{A_1, \ldots, A_k, \ldots, A_K\}$ and $B = \{B_1, \ldots, B_k, \ldots, B_K\}$ for computational simplicity, as analyzed in Section III-A. The left coding dictionary A is used to extract the features of the column vectors from the image patches, and the right coding dictionary B is used to extract the features of the row vectors from the image patches. For sparse response characteristics, the two learned dictionaries are usually required to be

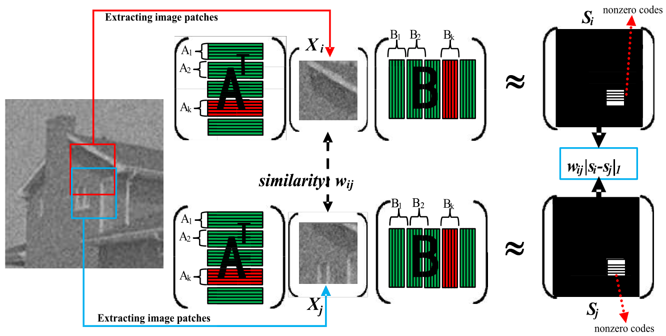


Fig. 1. Similar image patches encoded by the dictionary pair $\langle A, B\rangle$.

redundant such that they can represent the various local structures of 2D images. Unlike traditional sparse coding, the sparse coding of each image patch in our DPL model is a 2D sparse matrix. For sparsely coding each 2D image patch, a simple method is finding the most appropriate sub-dictionary pair from the learned dictionary pair $\langle A, B\rangle$ to carry out compact coding on it while constraining the zero coding coefficients on those un-selected sub-dictionary pairs. This method can ensure the attainment of a global sparse coding representation. As for the third term in Eq. (9), corresponding to the right of Fig. 1, it is expected to help realize as close and co-sparse as possible between the 2D sparse representations of nonlocal similar image patches (that is, the constraints of smoothing and nonlocal co-sparsity). Thus, the 2D sparse coding matrices with corresponding to nonlocal similar image patches are regularized under the manifold smoothing assumption with a L_1-norm metric.

III. Dictionary Pair Learning Algorithm on Grassmann-Manifold

In the DPL model (i.e., Eq. (9)), the dictionary pair $\langle A, B\rangle$ and the sparse coding matrixes S_i are all unknown, and their simultaneous solution is a NP problem. Therefore, our learning strategy is to decompose the problem into three subtasks: (1) learning the dictionary pair $\langle A, B\rangle$ from 2D noisy image patches by eigen-decomposition, as shown in Section III-A; (2) fixing the dictionary pair $\langle A, B\rangle$, and then updating the 2D sparse coding matrixes with smoothing, as shown in Section III-B; and (3) reconstructing the denoised image as shown in Section III-C. Thus, the so-called Dictionary Pair Learning algorithm on Grassmann-manifold (DPLG) is analyzed and summarized as follows.

A. Learning the Dictionary Pair

For solving Eq. (9), one important issue centers on how to learn the dictionary pair $\langle A, B\rangle$ for sparsely and smoothly coding the 2D image patches. Due to the difficulty and instability in the learned dictionary by directly optimizing the sparse coding model, the dictionaries can also be directly selected in conventional sparsity-based coding models (i.e., analytically designed dictionaries). Thus, we design the 2DPCA subspace pair partition on two Grassmann manifolds to implement the clustering-based sub-dictionary pair learning. Two sub-dictionaries for each cluster are computed, corresponding to decomposing the covariance matrix and its transposed matrix from 2D image patches (i.e., the sub-dictionary pair). All such sub-dictionary pairs construct two large over-complete dictionaries to characterize all the possible local structures of a given observed image. It is assumed that the $k-th$ subset is extracted to obtain the $k-th$ sub-dictionary pair $\langle A_k, B_k\rangle$, where $k = 1, \dots, K$. Then, in the dictionary pair $\langle A, B\rangle = \{\langle A_k, B_k\rangle\}_{k=1}^{K}$, the left dictionary $A = \{A_1, \dots, A_k, \dots, A_K\}$ is viewed as a point set on a Grassmann manifold, and the right dictionary $B = \{B_1, \dots, B_k, \dots, B_K\}$ is also viewed as a point set on other Grassmann manifold because a Grassmann manifold is the set of all linear subspaces with the fixed dimension [32]. In this paper, obtaining the dictionary pair $\langle A, B\rangle$ includes two basic stages: the initial dictionary pair $\langle A, B\rangle$ is obtained by the following Top-bottom 2D Subspace Partition (TTSP algorithm); next the refined dictionary pair $\langle A, B\rangle$ is obtained by the Sub-dictionary Merging algorithm (SM algorithm).

1) Obtaining the Initial Dictionary Pair by TTSP Algorithm: For overcoming the difficulty in directly learning the effective dictionary pair $\langle A, B\rangle$ under the nonlinear distribution characteristic of all of the 2D image patches, the entire training image patch set is divided into non-overlapping subsets with linear structures suited to the classical linear method, such as 2DPCA, and the sub-dictionary pair on each subset are easily learned by the eigen-decompositions of two covariance matrixes.[1] The literature [30] constructed a kind of data partition tree for subspace indexing based on the global PCA, but it is not suitable for our 2D subspace partition for learning the dictionary pair $\langle A, B\rangle$. We propose a Top-bottom 2D Subspace Partition algorithm (TTSP algorithm) for obtaining the initial dictionary pair $\langle A, B\rangle$. The TTSP algorithm recursively generates a binary tree, and each leaf node is used in learning a sub-dictionary pair by using an extended 2DPCA technique. The detailed steps of the TTSP algorithm are described in Algorithm 1.

2) Merging Sub-Dictionary Pairs by SM Algorithm: In the TTSP algorithm, each leaf node corresponds to two subspaces, namely, the left sub-dictionary and right sub-dictionary, called a sub-dictionary pair. However, as the number of levels in the partition increases, the number of training image patches in each leaf node decreases. Leaf nodes may not be the most effective local space for describing the image nonlocal similarity and local distribution because each leaf node may contain an insufficient number of samples. One reasonable method is to merge the leaf nodes that span almost the same left sub-dictionaries, and almost the same right sub-dictionaries. Because a Grassmann manifold is the set of all linear subspaces with a fixed dimension and any two points on a Grassmann manifold correspond to two subspaces. Therefore, to merge the very similar leaf nodes, we assume that all left sub-dictionaries from all leaf nodes lie on one Grassmann manifold and that all right sub-

[1] Two non-symmetrical covariance matrixes [21] of a matrix dataset $\{X_1, X_2, \dots, X_L\}$, $L_{cov} = \frac{1}{L}\sum_{i=1}^{L}(X_i - C_k)(X_i - C_k)^T$ and $R_{cov} = \frac{1}{L}\sum_{i=1}^{L}(X_i - C_k)^T(X_i - C_k)$ where $C_k = \frac{1}{L}\sum_{i=1}^{L} X_i$.

Algorithm 1 (**TTSP** Algorithm) Top-Bottom 2D Subspace Partition

Input: Training image patches, the maximum depth of the binary tree.

Output: the Dictionary pair $< A, B >$ and centers $\{C_k\}$ of all leaf nodes.

PROCEDURES:

Step1, The first node is the root node including all image patches.

Step2, For all image patches in the current leaf node, run the following 1)-4)steps:

1) Compute respectively the maximum eigenvectors u and v of the two covariance matrixes in the Footnote1.
2) Compute the one-dimensional projection representations of all image patches from this node, that is, $s_i = u^T X_i v, i = 1, ..., L$.
3) Partition the one-dimensional real number set $\{s_i\}$ into two clusters by K-means.
4) Partition the image patches corresponding to these two clusters into the left child and the right child. Simultaneously the depth of the node is added one.

Step3, IF the depth of the node is larger than the maximum depth or the number of image patches in this leaf node is smaller than the row number or column number of the image patches, THEN stop the partition. ELSE repeat Step2 recursively for the left child node and the right child node.

Step4, Compute the left sub-dictionary and the right sub-dictionary for each leaf node by the following 1)-4) steps:

1) Compute the center in the given leaf node k.
2) Compute the two covariance matrixes L_{cov} and R_{cov} in the Footnote1.
3) Compute respectively the corresponding eigenvectors $u_1, u_2, .., u_d$ and $v_1, v_2, .., v_d$ to the d largest eigenvalues; that is, to solve the two eigen-equations $L_{cov} u = \lambda u$ and $R_{cov} v = \widetilde{\lambda} v$.
4) Compute the left sub-dictionary $A_k = [u_1, u_2, .., u_d]$ and the right sub-dictionary $B_k = [v_1, v_2, .., v_d]$.

Step5, Collect the sub-dictionaries of K leaf nodes into the dictionary pair $< A, B >$ (i.e., the left dictionary $A = \{A_1, ..., A_k, ..., A_K\}$ and the right dictionary $B = \{B_1, ..., B_k, ..., B_K\}$).

dictionaries from all leaf nodes lie on the other Grassmann manifold.

The angles between linear subspaces have intuitively become a reasonable measure for describing the divergence between subspaces on a Grassmann manifold [32]. Thus, for computational convenience, the similarity metric between two subspaces is typically defined by taking the cosines of the principal angles. Taking the left sub-dictionaries for example, the cosines of the principal angles are defined as follows:

Definition 1: Let A_1 and A_2 be two m-dimensional subspaces corresponding to the two left sub-dictionaries. The cosine of the $t-th$ principal angle between the two subspaces $span(A_1)$ and $span(A_2)$ is defined by:

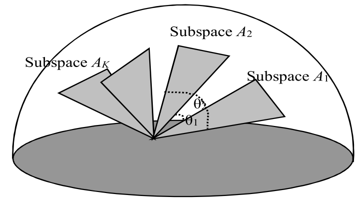


Fig. 2. Principal angles between sub-dictionaries.

$$\begin{cases} \cos(\theta_t) = \underset{u_t \in span(A_1)}{Max} \{ \underset{v_t \in span(A_2)}{Max} u_t^T v_t \} \\ S.t. \begin{cases} u_t^T u_t = v_t^T v_t = 1 \\ u_t^T u_r = v_t^T v_r = 0, \quad (t \neq r), \end{cases} \end{cases} \tag{10}$$

where $0 \leq \theta_t \leq \pi/2, t, r = 1, \ldots, m$, and u_t and v_t are the basis vectors from two subspaces, respectively.

In Eq. (10), the first principal angle θ_1 is the smallest angle among those between all pairs (each corresponds to two unit basis vectors), which are respectively from the two subspaces. The rest of the principal angles can be obtained by other basis vectors in each subspace, as shown in Fig. 2. The smaller the principal angles are, the more similar the two subspaces are (i.e., the closer they are on the Grassmann manifold). In fact, the cosines of all principal angles can be computed by a more numerically stable method, the Singular Value Decomposition (SVD) [34] solution, as described in Theorem 1, for which we provide a simple proof in Appendix A.

Let A_1 and A_2 be two m-dimensional column-orthogonal matrixes that respectively consist of orthogonal bases from two left sub-dictionaries. Then, the cosines of all principal angles between the two subspaces (i.e., the two sub-dictionaries) are computed by the following SVD equation:

Theorem 1: If A_1 and A_2 are two m-dimensional subspaces, then

$$A_1^T A_2 = U \Lambda V^T, \tag{11}$$

where the diagonal matrix $\Lambda = diag(\cos\theta_1, \ldots, cos\theta_m)$, $UU^T = I_m$ and $VV^T = I_m$.

In the following subspace merging algorithm, the similarity $Sim(A_1, A_2)$ between the two subspaces A_1 and A_2 is defined as the average of all principal angle cosine values:

$$Sim(A_1, A_2) = \frac{1}{m} \sum_{l=1}^{m} \cos\theta_l. \tag{12}$$

Therefore, the larger $Sim(A_i, A_j)$ are, the more similar the two subspaces are (i.e., the closer they are on the Grassmann manifold). Those almost same subspaces should be merge into a single subspace. On the other hand, the same situation should be considered for the right sub-dictionaries $B_i, i = 1, \ldots, K$. The similarity metric between the right sub-dictionaries is defined in the same manner as the above method. Therefore, simultaneously taking the left sub-dictionaries and the right sub-dictionaries into account, our Sub-dictionary Merging algorithm (**SM** algorithm) is described in Algorithm 2.

Algorithm 2 (**SM** Algorithm) Sub-Dictionary Merging Algorithm

Input: Sub-dictionary pairs $< A_i, B_i >, i = 1, ..., K1$, the pre-specified constant δ (empirical value 0.99).

Output: The reduced sub-dictionary pairs $< A_i, B_i >, k = 1, ..., K$, where $K <= K1$.

PROCEDURES:

Step1, Find the $subset_i$ and $subset_j$, if $Sim(A_i, A_j) > \delta$ and $Sim(B_i, B_j) > \delta$.

Step2, Delete A_i, A_j and B_i, B_j, and replace with the newly merged new left sub-dictionary and right sub-dictionary from updated the image patch set $subset_i \bigcup subset_j$.

Step3, Go Step1 until any $Sim(A_i, A_j) < \delta$ or $Sim(B_i, B_j) < \delta$.

Step4, Update the dictionary pair $< A, B >$ using the reduced sub-dictionary pairs.

B. Updating Sparse Coding Matrixes

Section III-A describes a method to rapidly learn the dictionary pair $\langle A, B\rangle$, where $A = \{A_1, \ldots, A_k, \ldots, A_K\}$, $B = \{B_1, \ldots, B_k, \ldots, B_K\}$. For sparsely coding each 2D noisy image patch and deleting noise, we need only to find the most appropriate sub-dictionary pair $\langle A_{ki}, B_{ki}\rangle$ from the learned dictionary pair $\langle A, B\rangle$ to represent the patch, and denoise the image patch by smoothing the sparse representation.

For the $i-th$ noisy image patch, we assume that the most appropriate sub-dictionary pair $\langle A_{ki}, B_{ki}\rangle$ is used to encode it and that the other sub-dictionary pairs are constrained to providing zero coefficient coding. According to the nearest center, the most appropriate sub-dictionary pair for the $i-th$ noisy image patch X_i can be selected by the smallest $L_1 - norm$ coding, that is:

$$ki = \underset{k}{argMin}\{\|A_k^T(X_i - C_k)B_k\|_{F,1}\}, \quad k = 1, \ldots, K, \tag{13}$$

where K is the total number of sub-dictionary pairs, C_k denotes the center of the $k-th$ leaf node, and $\|.\|_{F,1}$ denotes the matrix $L1 - norm$, which is defined as the sum of the absolute values of all matrix elements.

For obtaining sparse representations, we assume that any noisy image patch is only encoded by one sub-dictionary pair and that the coding coefficients on the other sub-dictionary pairs are constrained to zero. Therefore, for any noisy image patch X_i, we can simplify Eq. (9) to obtain the following objective function definition:

Definition 2: For image patch X_i, let the selected nearest sub-dictionary pair be $\langle A_{ki}, B_{ki}\rangle$ in Eq. (13). Then, the smoothing sparse coding is computed by the following formula:

$$\begin{cases} \underset{S_i}{argMin}\{\|A_{ki}^T X_i B_{ki} - S_i\|_F + \gamma\sum_j w_{ij}\|S_i - S_j\|_{F,1}\}, \\ S.t. \ \sum_j w_{ij} = 1 \end{cases} \tag{14}$$

where S_j is the sparse coding matrix of the $j-th$ nearest image patch on the sub-dictionary pair $\langle A_{ki}, B_{ki}\rangle$, w_{ij} is the non-local neighborhood similarity, and γ is the balance factor.

As for the balance factor γ, when the two terms of Eq. (14) are simultaneously optimized, we can reach the following conclusion (the proof is shown in Appendix B).

Theorem 2: If X_i is the corrupted image patch by noise $N(0, \sigma)$, and the non-local similarity obeys to the Laplacian distribution with the parameter σ_i, then the balance factor $\gamma = \frac{\sigma^2}{\sqrt{2}\sigma_i}$.

Clearly, the objective function of S_i in Eq. (14) is convex and can be efficiently solved. The first term is to minimize the reconstruction error on the sub-dictionary pair $\langle A_{ki}, B_{ki}\rangle$, and the second term is to ensure the smoothing and co-sparsity in coefficient matrix space. We initialize the coding matrix S_i and S_j by the projections of the image patch X_i and its neighbors X_j on the selected sub-dictionary pair $\langle A_{ki}, B_{ki}\rangle$, that is:

$$S_i(t) = A_{ki}^T X_i B_{ki}, \tag{15}$$

$$S_j(t) = A_{ki}^T X_j B_{ki}, \quad j = 1, .., k1, \tag{16}$$

where image patch X_j is one of the $k1$-nearest neighbors of image patch X_i.

Additionally, for computational convenience, we can reformat and relax Eq. (14) into the following objection function:

$$\begin{cases} \underset{S_i}{argMin}\{\|A_{ki}^T X_i B_{ki} - S_i\|_F + \gamma\|S_i - \sum_j w_{ij} S_j\|_{F,1}\} \\ S.t. \ \sum_j w_{ij} = 1. \end{cases} \tag{17}$$

According to the literature [35], a threshold-shrinkage algorithm is adopted to solve the Eq. (17) (i.e., using the gradient descent method and the threshold-shrinkage strategy). Therefore, the sparse coding matrix S_i on the sub-dictionary pair $\langle A_{ki}, B_{ki}\rangle$ is updated by the following formula:

$$\begin{cases} S_i(t+1) = f(S_i(t) - \sum_j w_{ij} S_j(t), \eta\gamma) + \sum_j w_{ij} S_j(t) \\ S.t. \ \|X_i - A_{ki} S_i B_{ki}^T\|_F < cN\sigma^2, \end{cases} \tag{18}$$

where σ is the noise variance, N is the number of image patch pixels, η is the gradient decent step, c is a scaling factor, which is empirically set 1.15, and $f(.,.)$ is the soft threshold-shrinkage function, that is:

$$f(z, \delta) = \begin{cases} 0, & if \ z < \delta \\ z - sgn(z)\delta, & otherwise, \end{cases} \tag{19}$$

where $sgn(z)$ is a sign function.

C. Reconstructing the Denoised Image

As a type of non-local similarity and transformation domain approach, a given noisy image needs to be divided into many overlapping small image patches. The corresponding denoised image is obtained by combining all of the denoised image patches. Let x denote a noisy image, and let the binary

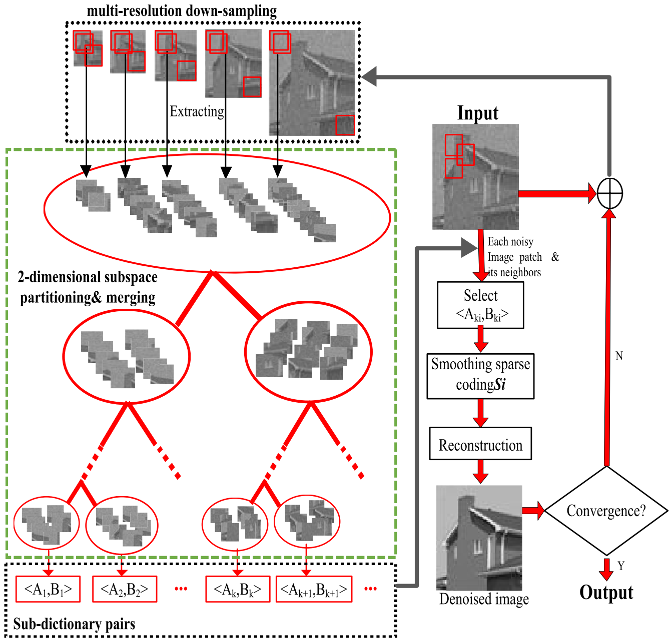


Fig. 3. The working flowchart of DPLG algorithm.

matrix R_i be used for extracting the $i-th$ image patch at the position i, that is:

$$X_i = R_i x, \quad i = 1, 2, \ldots, n, \tag{20}$$

where n denotes the number of possible image patches.

If we let S_i be the coding matrix, with smoothing and co-sparsity obtained by using the sub-dictionary pair $\langle A_{ki}, B_{ki}\rangle$, then the denoised image $\tilde{x}$ is reconstructed by:

$$\tilde{x} = \left\{ \sum_i (R_i^T A_{ki} S_i B_{ki}^T) \right\} \oslash \sum_i (R_i^T R_i \mathbf{1}), \tag{21}$$

where $\oslash$ denotes an element-wise division and $\mathbf{1}$ denotes a matrix of ones. That is, Eq. (21) puts all denoised patches together as the denoised image $\tilde{x}$ (the overlapped pixels between neighboring patches are averaged).

D. Summary of the DPLG Algorithm

1) The Description of the DPLG Algorithm: Summarizing the above analysis, for adaptively learning and denoising from a given noisy image itself, we put forward the Dictionary Pair Learning algorithm on Grassmann-manifold (DPLG). The DPLG algorithm allows the dictionary pair to be updated according to the last denoised result and then obtains better representations of the noisy patches. Thus, the DPLG algorithm is designed as an iterative image denoising method. Each iteration includes three basic tasks, namely, learning the dictionary pair $\langle A, B\rangle$ from the noisy image patches sampled from the current noisy image at a multi-resolution, updating the $2D$ sparse representations for image patches from the current noisy image, and reconstructing the denoised image, where the current noisy image is a slight translation from the current denoised image to the original noisy image. Fig. 3 shows the basic working flowchart of the DPLG algorithm, and the detailed procedures of the DPLG algorithm are described in the Algorithm 3.

Algorithm 3 (**DPLG** Algorithm) Dictionary Pair Learning on Grassmann-Manifold

---

Input: Noisy image N_Im0 and estimated noise variance σ_0.

Output: Denoised image D_Im.

PROCEDURES:

Step1, Set the initial parameters, including iterations, patch size, the maximum depth of leaf nodes, and the pre-specified constants $\mu 1$ and $\mu 2$.

Step2, Let the current denoised image and noisy image be $D\_Im = N\_Im = N\_Im0$.

Step3, Loop the following steps from 1) to 9) until the given iterations.

1) $N\_Im = D\_Im + \mu 1(N\_Im0 - D\_Im)$, $\sigma = \mu 2 \sqrt{\sigma_0^2 - \frac{1}{N}\sum\limits_{i,j}(N\_Im0 - N\_Im)_{ij}^2}$.
2) Extract the 2D noisy image patch set X from the given noisy image N_Im at multi-resolution.
3) Divide the 2D image patch set X into K subsets by using the Step1-3 of the TTSP algorithm.
4) Compute the two-dimensional sub-dictionary pairs $< A_k, B_k >$ and the center C_k for each 2D patch subset by using the Step4 of the above TTSP algorithm.
5) Merge those almost the same sub-dictionary pairs using the SM algorithm.
6) Select the corresponding sub-dictionary pair $< A_{ki}, B_{ki} >$ for each noisy image patch X_i from the current noisy image N_Im using the following formula: $ki = \underset{k}{argMin}\{||A_k^T(X_i - C_k)B_k||_{F,1}\}$
7) Compute the neighborhood similarity w_{ij} between these noisy image patches $\{X_i\}$ using Eq. (8).
8) Compute the smooth and sparse $2D$ representations for each image patch and its neighbors from the current noisy image by using Eq. (18).
9) Reconstruct the denoised image D_Im by integrating all denoised image patches $Y_i = A_{ki} S_i B_{ki}^T$ using Eq. (21).

---

2) Time Complexity Analysis: Our DPLG method preserves the original 2D structure of each image patch to un-change. If the size of the sampled image patches is $b \times b$, and the sub-dictionary pair $\langle A_k, B_k\rangle$ is computed by using 2DPCA on each image patch subset, then A_k and B_k are two $b \times b$ orthogonal matrices. Comparatively, NCSR needs to compute a more complex $b^2 \times b^2$ orthogonal matrix as the dictionary by using the PCA on 1D presentations of image patches. For example, in the NCSR method, the matrix size appears to be 64 times larger than our method when $b = 8$. Therefore, for DPLG, less time complexity is required to compute the eigenvectors.

Moreover, comparing our DPLG method with the NCSR method, the former is to rapidly top-bottom divide each leaf node into the left-child and right-child by the first principal component projection on the current sub-dictionary pair

TABLE I

TIME COMPLEXITY FOR ONE UPDATE OF TWO BASIC STEPS IN THREE DICTIONARY LEARNING ALGORITHMS: DPLG, NCSR AND K-SVD

Algorithm	Dictionary learning step	Sparse coding step
DPLG	$O(Knl) + O(Kb^3)$	$O(Knb) + O(nk1)$
NCSR	$O(Knlb^2) + O(Kb^6)$	$O(Knb^2)$
K-SVD	$O(Hn) + O(Hb^6)$	$O(HnM)$

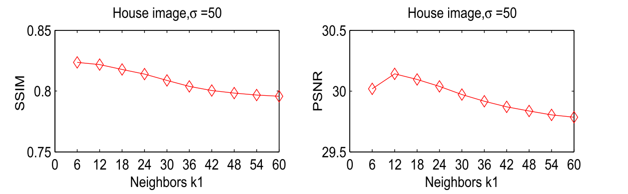


Fig. 4. The denoising performance of the DPLG at different k1-nearest neighbors.

(i.e., the two-way partition of 1D real numbers). The latter is to divide the whole training set (i.e., b^2-dimensional vectors) into the specified clusters by applying K-means with more time complexity. Compared with the K-SVD method, each atom of its single dictionary D needs to be updated by SVD decomposition. If the number of the dictionary atoms in K-SVD is equal to the amount of all sub-dictionary atoms in the DPLG or NCSR, then the computational complexity of K-SVD is the largest. However, the dictionary D of K-SVD in real-world applications is only ever empirically set to a smaller over-complete dictionary atom number than the DPLG and NCSR method, so that K-SVD has a faster computing speed. Additionally, in the sparse coding step, the three internal denoising methods DPLG, NCSR and K-SVD have slight differences in time complexity, as shown in Table I.

Without loss of generality, letting the number of clusters equal K, the number of image patches equal n, the size of each image patch equal $b \times b$, the iteration of K-means clustering equal l, the $k1$-nearest neighbors equal $k1$, the number of dictionary atoms in K-SVD equal H, and the max number of nonzero codes for each image patch in K-SVD equal M, we compare the computational complexity of the dictionary learning step and the sparse coding step in three iterative dictionary learning methods (internal denoising methods), namely, DPLG, NCSR and KSVD, as shown in the Table I. Due to computing the non-local neighborhood similarity within each cluster in our manifold smoothing strategy, computing the Laplacian similarity only needs linear computational time. Finally, the total time complexity of the DPLG is less than the NCSR and K-SVD algorithms with the same size of their dictionaries (that is, when $H = Kb$).

IV. EXPERIMENTS

In this section, we will verify the image denoising performance of the proposed DPLG method. We test the performance of the DPLG method on benchmark images [38], [39] and on 100 test images from the Berkeley Segmentation Dataset [40]. Moreover, these experimental results of the proposed DPLG method are compared with seven developed state-of-the-art denoising methods, including three internal denoising methods and four denoising methods using external information from clean natural images.

A. Quantitative Assessment of Denoised Images

An objective image quality metric plays an important role in image denoising applications. Currently, three classical image quality assessment metrics are typically used: the Root mean square error (RMSE), the Peak Signal-to-Noise Ratio (PSNR) and the measure of Structural SIMilarity (SSIM) [36]. The PSNR and RMSE are the simplest and most widely used image quality metrics. Common knowledge holds that the smaller the RMSE is, the better the denoising is. Equivalently, the larger the PSNR is, the better the denoising is. Moreover, the RMSE and PSRN have the same assessment ability, although they are not very well matched in the perceptual visual quality of denoised images. The third quantitative evaluation method, the Structural SIMilarity (SSIM), focuses on the perceptual quality metric, which compares normalized local patterns of pixel intensities. In our experiments, the PSNR and SSIM are used as objective assessments.

B. Experiments on Benchmark Images

To evaluate the performance of the proposed model, we exploit the proposed DPLG algorithm for denoising ten noisy benchmark images [38] and another difficult-to-be-denoised noisy image (named the 'ChangE-3' image [39]), which is significant. Several state-of-the-art denoising methods with default parameters are used for comparison with the proposed DPLG algorithm, including the internal denosing methods BM3D [10], K-SVD [11] and NCSR [12], the external denoising methods SSDA [13] and SDAE [14], SCLW [15], and NSCDL [16]. As for the parameter setting of our DPLG algorithm, the $k1$-nearest neighbor parameter, the maximum depth of leaf nodes and the number of iterations of the DPLG are empirically set to 6, 7 and 18, respectively, from a series of tentative test. Taking the k1-nearest-neighbor parameter as an example, we analyze the performance of our method at the different k1-nearest-neighbor parameters, as shown in Fig.4. Accordingly, when the size of neighbors is not large enough (for example the k1-nearest neighbors at [6, 60]), the performance of our DPLG method does not significantly change. However, the DPLG can obtain the largest SSIM value when the $k1$-nearest-neighbor parameter is set to 6.

1) Comparing With Internal Denoising Methods: The 20 different noisy versions of the 11 benchmark images, that is, corresponding to 220 noisy images, are denoised respectively by the previously mentioned four internal denoising methods: DPLG, NCSR, BM3D and K-SVD. The SSIM results of the four test methods are reported in Table II, and the highest SSIM values are displayed in black bold. The PSNR results are reported in Table III, and the highest PSNR values are displayed in black bold.

It is worth noting that our DPLG method preserves the 2D geometrical structure of the image patches and thus can significantly achieve the best visual quality, as shown

TABLE II

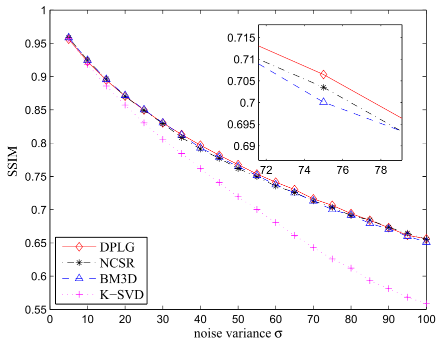
THE SSIM VALUES BY DENOISING 11 IMAGES AT DIFFERENT NOISE VARIANCE

SSIM \ σ Algorithm		5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Barbara	DPLG	0.962	0.940	0.920	0.904	0.886	0.865	**0.849**	**0.831**	0.812	0.795	0.777	**0.759**	**0.746**	0.728	**0.719**	0.696	**0.682**	0.671	**0.655**	**0.649**
	NCSR	0.964	0.941	0.921	**0.905**	**0.888**	**0.869**	0.847	0.823	0.811	0.790	0.775	0.741	0.743	0.718	0.705	0.688	0.686	0.663	0.649	0.643
	BM3D	**0.965**	**0.942**	**0.922**	0.905	0.885	0.866	0.846	0.820	**0.815**	**0.796**	**0.778**	0.758	0.744	**0.730**	0.713	**0.704**	0.674	**0.675**	0.653	0.642
	K-SVD	0.964	0.935	0.909	0.880	0.849	0.821	0.799	0.770	0.745	0.714	0.687	0.661	0.641	0.615	0.604	0.586	0.566	0.559	0.545	0.541
Boat	DPLG	0.935	0.885	0.850	0.823	0.797	0.773	0.753	**0.734**	**0.716**	0.700	0.683	**0.669**	0.657	**0.646**	**0.636**	0.621	0.609	0.603	0.592	0.587
	NCSR	**0.941**	**0.888**	0.851	0.818	0.793	0.772	0.742	0.723	0.705	0.688	0.675	0.661	0.651	0.637	0.631	0.619	0.613	0.604	0.594	0.590
	BM3D	0.939	0.888	**0.854**	**0.826**	**0.801**	**0.778**	**0.756**	0.735	0.716	**0.703**	**0.687**	0.669	**0.659**	0.644	0.633	**0.623**	**0.614**	**0.606**	**0.597**	**0.588**
	K-SVD	0.941	0.883	0.841	0.805	0.771	0.744	0.719	0.699	0.678	0.659	0.641	0.620	0.608	0.592	0.580	0.567	0.558	0.548	0.537	0.527
Camera Man	DPLG	0.959	0.926	0.897	0.863	0.849	0.831	0.817	**0.806**	**0.796**	**0.785**	**0.773**	**0.766**	0.750	0.741	0.731	0.726	0.719	0.710	0.694	0.694
	NCSR	0.961	0.930	**0.902**	0.870	**0.853**	0.827	0.811	0.799	0.791	0.781	0.769	0.762	**0.752**	**0.743**	**0.735**	**0.728**	**0.728**	**0.714**	**0.703**	**0.697**
	BM3D	**0.962**	**0.931**	0.899	**0.872**	0.852	**0.834**	**0.821**	0.803	0.788	0.779	0.767	0.759	0.745	0.741	0.726	0.720	0.699	0.696	0.693	0.689
	K-SVD	0.959	0.926	0.893	0.861	0.834	0.813	0.793	0.778	0.758	0.740	0.726	0.715	0.697	0.685	0.652	0.648	0.616	0.609	0.594	0.579
Couple	DPLG	0.947	0.907	0.873	0.842	0.816	0.790	**0.768**	**0.745**	**0.729**	**0.710**	**0.685**	0.668	**0.656**	0.638	0.622	0.610	0.594	0.581	0.573	0.561
	NCSR	0.950	0.907	0.871	0.838	0.809	0.780	0.758	0.732	0.712	0.690	0.673	0.654	0.638	0.624	0.608	0.598	0.583	0.578	0.568	0.555
	BM3D	**0.951**	**0.908**	**0.875**	**0.845**	**0.819**	**0.794**	0.768	0.743	0.723	0.705	0.685	**0.672**	0.652	**0.639**	**0.623**	**0.613**	**0.598**	**0.587**	**0.574**	**0.567**
	K-SVD	0.950	0.897	0.853	0.815	0.780	0.746	0.711	0.680	0.659	0.632	0.611	0.596	0.575	0.565	0.550	0.539	0.525	0.521	0.505	0.501
Finger print	DPLG	**0.988**	**0.969**	0.948	0.929	0.910	0.894	0.874	**0.858**	0.844	0.829	0.813	0.797	0.789	0.769	0.761	0.746	0.731	0.717	0.710	0.702
	NCSR	0.988	0.970	**0.950**	**0.932**	**0.913**	**0.896**	0.874	0.856	0.839	0.825	0.808	0.789	0.777	0.762	0.755	0.736	0.724	0.708	0.701	0.685
	BM3D	0.987	0.969	0.949	0.930	0.911	0.894	**0.878**	0.856	**0.847**	**0.832**	**0.822**	**0.806**	**0.793**	**0.781**	**0.772**	**0.762**	**0.746**	**0.739**	**0.726**	**0.718**
	K-SVD	0.988	0.968	0.946	0.923	0.897	0.871	0.846	0.817	0.791	0.753	0.721	0.686	0.647	0.605	0.572	0.546	0.505	0.480	0.458	0.447
House	DPLG	0.956	0.921	0.891	**0.876**	**0.860**	**0.853**	**0.846**	**0.837**	**0.827**	**0.819**	**0.811**	**0.806**	**0.801**	0.787	0.780	**0.775**	**0.771**	0.749	0.741	0.741
	NCSR	**0.958**	**0.924**	**0.894**	0.875	0.858	0.850	0.841	0.837	0.823	0.815	0.808	0.799	0.792	**0.789**	**0.783**	0.767	0.765	**0.756**	**0.751**	**0.743**
	BM3D	0.956	0.922	0.889	0.874	0.858	0.846	0.836	0.826	0.823	0.810	0.804	0.798	0.792	0.770	0.757	0.757	0.750	0.738	0.735	0.730
	K-SVD	0.953	0.906	0.877	0.860	0.843	0.828	0.817	0.795	0.777	0.764	0.750	0.731	0.711	0.688	0.678	0.665	0.649	0.622	0.617	0.618
Lena	DPLG	0.942	0.914	0.892	**0.877**	0.859	0.844	0.835	0.823	**0.814**	0.803	0.789	0.780	0.770	0.761	0.752	0.746	0.740	0.728	0.724	0.715
	NCSR	0.945	0.915	0.893	0.876	**0.860**	**0.848**	**0.836**	**0.824**	0.814	**0.805**	**0.793**	**0.787**	**0.779**	**0.771**	**0.762**	**0.753**	**0.749**	**0.744**	**0.734**	**0.727**
	BM3D	0.945	**0.917**	**0.895**	0.875	0.860	0.845	0.829	0.814	0.807	0.798	0.788	0.777	0.766	0.758	0.751	0.741	0.733	0.723	0.720	0.704
	K-SVD	**0.946**	0.911	0.885	0.862	0.843	0.825	0.807	0.791	0.773	0.758	0.745	0.733	0.720	0.707	0.697	0.684	0.671	0.660	0.656	0.639
Man	DPLG	0.951	0.906	**0.867**	**0.832**	**0.805**	0.777	0.752	0.732	0.718	**0.702**	0.686	0.676	0.666	**0.656**	**0.645**	0.633	**0.627**	0.618	**0.609**	0.601
	NCSR	0.954	**0.907**	0.866	0.831	0.804	0.776	0.750	0.730	0.714	0.699	0.683	0.672	0.662	0.653	0.644	**0.634**	0.625	**0.621**	0.608	**0.605**
	BM3D	**0.955**	0.907	0.865	0.832	0.803	**0.778**	**0.754**	**0.734**	**0.719**	0.702	**0.689**	**0.677**	**0.667**	0.654	0.642	0.633	0.625	0.612	0.607	0.598
	K-SVD	0.952	0.899	0.852	0.813	0.781	0.752	0.724	0.702	0.681	0.664	0.647	0.635	0.619	0.607	0.598	0.586	0.573	0.567	0.558	0.551
Monarch_full	DPLG	**0.976**	**0.959**	0.939	**0.923**	0.901	0.886	**0.871**	**0.859**	**0.836**	0.821	**0.820**	0.803	0.789	**0.777**	**0.770**	0.748	0.740	0.723	0.717	**0.713**
	NCSR	0.976	0.958	**0.940**	0.922	**0.902**	**0.889**	0.867	0.853	0.834	**0.822**	0.815	**0.810**	**0.794**	0.775	0.765	**0.752**	**0.744**	0.726	**0.720**	0.705
	BM3D	0.975	0.957	0.938	0.919	0.900	0.881	0.871	0.849	0.833	0.818	0.808	0.787	0.779	0.760	0.757	0.746	0.741	**0.727**	0.701	0.698
	K-SVD	0.972	0.949	0.928	0.908	0.885	0.865	0.853	0.831	0.812	0.796	0.782	0.755	0.745	0.724	0.710	0.694	0.676	0.661	0.633	0.627
Peppers	DPLG	0.952	0.925	0.905	0.885	0.867	**0.852**	**0.838**	**0.824**	0.810	**0.798**	**0.786**	**0.777**	**0.769**	0.750	0.743	0.733	0.721	**0.723**	0.696	0.695
	NCSR	**0.955**	0.927	0.907	0.886	0.868	0.851	0.835	0.817	**0.816**	0.795	0.786	0.772	0.767	**0.751**	**0.745**	**0.734**	**0.725**	0.718	**0.716**	**0.703**
	BM3D	0.955	**0.929**	**0.908**	**0.887**	**0.870**	0.852	0.835	0.820	0.807	0.792	0.779	0.763	0.751	0.750	0.727	0.718	0.708	0.700	0.688	0.680
	K-SVD	0.954	0.924	0.898	0.877	0.857	0.840	0.826	0.806	0.789	0.772	0.757	0.740	0.716	0.707	0.687	0.676	0.651	0.650	0.641	0.627
ChangE-3	DPLG	0.949	0.899	0.860	0.822	**0.796**	**0.770**	0.740	0.718	**0.699**	**0.681**	**0.667**	**0.654**	**0.639**	**0.627**	**0.613**	**0.601**	**0.592**	**0.579**	0.565	**0.562**
	NCSR	0.955	**0.905**	**0.866**	0.823	0.795	0.766	0.730	0.710	0.689	0.670	0.659	0.644	0.630	0.621	0.605	0.594	0.586	0.576	**0.569**	0.557
	BM3D	0.956	0.903	0.863	**0.826**	0.795	0.768	**0.743**	**0.719**	0.694	0.677	0.659	0.644	0.631	0.616	0.602	0.594	0.585	0.577	0.567	0.550
	K-SVD	**0.956**	0.904	0.863	0.826	0.793	0.762	0.733	0.708	0.681	0.659	0.634	0.615	0.595	0.577	0.557	0.543	0.533	0.518	0.507	0.489
Average	DPLG	0.956	0.923	0.895	0.871	**0.850**	**0.830**	**0.813**	**0.797**	**0.782**	**0.768**	**0.754**	**0.741**	**0.730**	**0.716**	**0.706**	**0.694**	**0.684**	0.673	0.661	**0.656**
	NCSR	**0.959**	**0.925**	**0.896**	0.870	0.849	0.829	0.809	0.791	0.777	0.762	0.749	0.735	0.726	0.713	0.704	0.691	0.684	**0.674**	**0.665**	0.655
	BM3D	0.959	0.925	0.896	**0.872**	0.850	0.830	0.812	0.793	0.779	0.765	0.751	0.737	0.725	0.713	0.700	0.692	0.679	0.671	0.660	0.651
	K-SVD	0.958	0.918	0.886	0.857	0.830	0.806	0.784	0.762	0.741	0.719	0.700	0.681	0.661	0.643	0.626	0.612	0.593	0.581	0.568	0.559

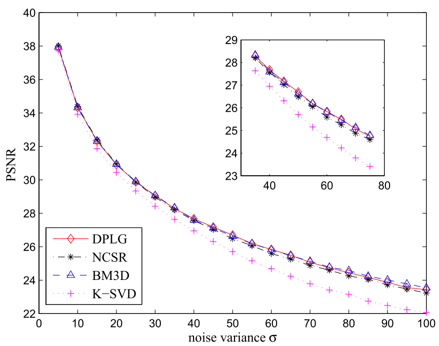
TABLE III

The PSNR Values by Denoising 11 Images at Different Noise Variance

Algorithm	PSNR \ σ	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Barbara	DPLG	38.290	34.921	32.935	31.621	30.575	29.633	28.933	**28.335**	27.681	27.169	26.625	26.160	25.759	25.306	24.983	24.504	24.199	23.913	23.628	23.442
	NCSR	**38.346**	**35.018**	33.033	**31.784**	**30.664**	29.670	28.909	28.170	27.633	27.036	26.502	25.726	25.515	24.985	24.737	24.296	24.269	23.762	23.431	23.198
	BM3D	38.284	34.977	**33.047**	31.772	30.651	**29.768**	**29.026**	27.989	**27.831**	**27.220**	**26.776**	**26.231**	**25.912**	**25.525**	**25.162**	**24.907**	**24.362**	**24.203**	**23.846**	**23.630**
	K-SVD	38.069	34.447	32.355	30.857	29.554	28.562	27.787	26.968	26.207	25.454	24.837	24.219	23.805	23.267	23.089	22.760	22.357	22.260	22.034	21.916
Boat	DPLG	37.169	33.856	32.055	30.783	29.780	28.971	28.279	**27.687**	**27.219**	**26.771**	26.256	25.838	25.502	25.090	24.894	24.511	24.188	24.053	23.734	23.670
	NCSR	**37.342**	33.877	32.049	30.705	29.653	28.859	28.104	27.469	26.976	26.423	26.085	25.684	25.408	25.019	24.741	24.466	24.287	24.017	23.749	23.604
	BM3D	37.285	**33.890**	**32.130**	**30.846**	**29.871**	**29.057**	**28.319**	27.635	27.130	26.705	**26.292**	**25.881**	**25.584**	**25.252**	**24.954**	**24.782**	**24.528**	**24.337**	**24.100**	**23.840**
	K-SVD	37.235	33.608	31.749	30.379	29.328	28.475	27.687	27.083	26.479	25.925	25.448	24.993	24.596	24.219	23.996	23.681	23.471	23.256	23.039	22.809
Camera Man	DPLG	38.211	34.024	31.893	30.192	29.346	28.596	27.776	27.148	**26.809**	**26.355**	**25.750**	**25.579**	**25.093**	**24.629**	**24.244**	24.101	**23.741**	**23.510**	**22.998**	**23.140**
	NCSR	38.292	34.147	**31.988**	30.360	29.398	28.388	27.654	27.016	26.594	26.001	25.689	25.278	24.812	24.475	24.095	23.698	23.656	23.278	22.888	22.673
	BM3D	**38.322**	34.153	31.884	**30.442**	**29.517**	**28.730**	**27.953**	**27.266**	26.618	26.134	25.597	25.286	24.832	24.627	24.170	**24.120**	23.662	23.470	23.350	23.047
	K-SVD	37.917	33.728	31.442	29.950	28.919	28.096	27.212	26.898	26.264	25.624	25.152	24.739	24.231	23.887	23.188	23.188	22.601	22.255	22.006	21.741
Couple	DPLG	37.350	33.975	32.078	30.631	29.604	28.769	28.045	**27.403**	**26.954**	**26.483**	25.915	25.561	**25.303**	24.863	24.522	24.301	23.948	23.661	23.579	23.228
	NCSR	37.480	33.941	31.970	30.584	29.441	28.518	27.922	27.187	26.625	26.153	25.720	25.261	24.925	24.667	24.336	24.090	23.810	23.534	23.376	23.213
	BM3D	**37.504**	**33.983**	**32.081**	**30.757**	**29.693**	**28.843**	**28.065**	27.391	26.915	26.412	**25.937**	**25.691**	25.281	**24.963**	**24.649**	**24.453**	**24.124**	**23.900**	**23.731**	**23.512**
	K-SVD	37.314	33.465	31.455	30.029	28.892	27.927	27.068	26.347	25.867	25.294	24.806	24.491	24.151	23.835	23.589	23.409	23.114	22.987	22.765	22.641
Finger print	DPLG	36.755	32.595	30.337	28.854	27.706	26.874	**26.117**	**25.509**	**24.974**	**24.520**	24.117	23.730	**23.432**	23.011	22.815	22.495	22.177	21.930	21.721	**21.635**
	NCSR	**36.782**	**32.672**	**30.453**	**28.980**	**27.817**	**26.975**	26.155	25.504	24.960	24.486	24.094	23.614	23.262	22.962	22.734	22.378	22.180	21.816	21.679	21.376
	BM3D	36.492	32.444	30.288	28.815	27.680	26.805	26.094	25.291	24.988	24.509	**24.164**	**23.757**	23.388	**23.102**	**22.817**	**22.598**	**22.262**	**22.028**	**21.808**	21.631
	K-SVD	36.630	32.382	30.069	28.493	27.251	26.283	25.495	24.717	24.009	23.161	22.492	21.810	21.171	20.481	19.965	19.596	19.101	18.778	18.525	18.374
House	DPLG	39.840	36.808	**35.100**	**33.960**	**33.132**	**32.332**	**31.736**	**31.020**	**30.483**	**29.953**	**29.265**	**29.035**	**28.529**	**28.057**	**27.522**	**27.226**	**27.226**	**26.693**	26.071	25.804
	NCSR	**39.852**	**36.821**	35.077	33.839	32.926	31.996	31.336	30.964	30.192	29.511	28.930	28.311	27.898	27.535	27.306	26.652	26.541	26.079	25.786	25.627
	BM3D	39.767	36.749	34.942	33.774	32.796	32.014	31.428	30.738	30.166	29.592	29.169	28.816	28.475	27.925	27.367	27.213	26.817	26.475	**26.228**	**26.033**
	K-SVD	39.308	35.981	34.312	33.077	32.078	31.185	30.431	29.503	28.639	28.032	27.334	26.854	26.324	25.582	25.187	25.044	24.473	23.907	23.793	23.835
Lena	DPLG	38.654	35.819	34.152	32.969	31.938	31.181	30.540	29.959	**29.545**	**29.070**	28.434	28.034	27.719	27.284	26.954	**26.831**	26.525	26.069	25.811	25.706
	NCSR	**38.742**	35.840	34.108	32.973	31.888	31.089	**30.594**	**29.983**	29.413	28.917	28.392	28.069	27.635	27.380	27.040	26.691	26.505	**26.310**	25.936	25.561
	BM3D	38.729	**35.929**	**34.240**	**32.998**	**32.077**	**31.266**	30.531	29.793	29.441	28.986	**28.515**	**28.221**	**27.805**	**27.534**	**27.138**	26.817	**26.621**	26.307	**26.189**	**25.900**
	K-SVD	38.625	35.519	33.737	32.377	31.364	30.475	29.711	29.015	28.316	27.789	27.233	26.871	26.481	26.094	25.734	25.459	25.166	24.841	24.713	24.434
Man	DPLG	37.694	33.935	31.893	30.541	**29.590**	28.776	28.089	27.535	27.119	26.717	26.298	26.013	25.743	25.458	25.203	24.821	24.656	24.345	24.192	24.043
	NCSR	**37.836**	**33.975**	**31.922**	30.542	29.584	28.744	28.047	27.497	27.021	26.623	26.251	25.838	25.588	25.314	25.071	24.778	24.616	24.432	24.050	23.950
	BM3D	37.811	33.918	31.863	**30.555**	29.566	**28.826**	**28.144**	**27.598**	**27.190**	**26.780**	**26.443**	**26.094**	**25.826**	**25.517**	**25.256**	**25.099**	**24.841**	**24.566**	**24.400**	**24.212**
	K-SVD	37.516	33.526	31.467	30.101	29.129	28.314	27.568	27.031	26.534	26.060	25.675	25.311	24.957	24.674	24.446	24.243	23.942	23.744	23.549	23.411
Monarch _full	DPLG	**38.548**	34.571	32.183	30.529	29.295	28.423	27.412	**26.919**	26.159	25.653	**25.433**	**24.966**	**24.603**	**24.229**	**23.890**	23.364	23.128	22.754	22.503	22.374
	NCSR	38.523	**34.655**	**32.367**	**30.722**	**29.401**	**28.565**	27.415	26.820	26.108	25.603	25.267	24.910	24.502	23.908	23.712	23.248	23.016	22.520	22.418	22.026
	BM3D	38.214	34.231	31.884	30.359	29.244	28.319	**27.762**	26.758	**26.258**	**25.713**	25.249	24.763	24.441	23.981	23.817	**23.577**	**23.250**	**23.006**	**22.517**	**22.423**
	K-SVD	37.771	33.710	31.432	29.980	28.767	27.830	27.214	26.389	25.838	25.312	24.725	24.081	23.667	23.134	22.811	22.352	21.916	21.581	20.944	20.836
Peppers	DPLG	38.022	34.592	32.646	31.162	30.054	29.217	28.333	27.673	27.205	**26.707**	26.014	25.712	**25.410**	24.886	24.432	24.282	23.742	23.756	23.134	22.967
	NCSR	38.070	34.650	**32.824**	31.212	**30.082**	29.139	28.355	27.527	**27.238**	26.428	25.967	25.342	25.182	24.582	24.327	24.195	23.510	23.291	23.169	22.843
	BM3D	**38.081**	**34.707**	32.803	**31.245**	30.232	**29.329**	**28.479**	**27.897**	27.220	26.689	**26.165**	**25.833**	25.378	**25.257**	**24.617**	**24.407**	**23.957**	**23.843**	**23.504**	**23.206**
	K-SVD	37.778	34.262	32.226	30.787	29.734	28.843	28.030	27.368	26.725	26.121	25.554	25.134	24.444	24.126	23.468	23.134	22.467	22.397	22.046	21.847
ChangE-3	DPLG	36.621	32.527	30.360	28.739	**27.636**	**26.769**	25.850	**25.220**	**24.741**	**24.298**	**23.887**	**23.575**	**23.270**	**22.954**	**22.686**	**22.435**	**22.203**	21.949	21.760	**21.650**
	NCSR	36.786	**32.596**	**30.420**	28.746	27.588	26.678	25.740	25.109	24.530	24.113	23.762	23.420	23.036	22.820	22.460	22.285	22.047	21.847	21.676	21.428
	BM3D	**36.791**	32.505	30.266	28.708	27.559	26.649	**25.854**	25.162	24.557	24.103	23.720	23.393	23.116	22.802	22.616	22.364	22.193	**21.991**	**21.836**	21.549
	K-SVD	36.751	32.452	30.285	**28.783**	27.624	26.652	25.812	25.133	24.546	23.970	23.480	23.064	22.723	22.354	22.041	21.776	21.571	21.296	21.143	20.830
Average	DPLG	37.923	34.329	32.330	30.907	29.878	29.049	28.283	**27.674**	**27.172**	**26.700**	26.181	**25.837**	**25.488**	25.070	24.740	24.443	24.157	23.876	23.557	23.423
	NCSR	**38.005**	**34.381**	**32.383**	**30.950**	29.858	28.965	28.203	27.568	27.026	26.481	26.060	25.587	25.251	24.877	24.596	24.252	24.040	23.717	23.469	23.227
	BM3D	37.934	34.317	32.312	30.934	**29.899**	**29.055**	**28.332**	27.592	27.119	26.622	**26.184**	25.815	25.458	**25.135**	**24.779**	**24.576**	**24.238**	**24.012**	**23.774**	**23.544**
	K-SVD	37.719	33.917	31.866	30.437	29.331	28.422	27.638	26.950	26.311	25.704	25.158	24.688	24.232	23.787	23.410	23.149	22.743	22.482	22.233	22.061



(a)



(b)

Fig. 5. The average SSIM values and average PSNR values of 11 denoised images at different noise variance σ. (a) Average SSIM values of 11 denoised images. (b) Average PSNR values of 11 denoised images.

in columns 5-17 in Table II. From Table III, we can see that when the noise level is not very high (seemly noise variance $\sigma < 30$), all of the four methods can achieve very good denoised images. When the noise level is high (seemly noise variance $30 \leq \sigma < 80$), our DPLG method can obtain basically the best denoising performance corresponding to columns 9-13 in Table III. Moreover, Fig. 5 shows the plots of the average PSNR and SSIM of the 11 images at different noise corruption levels.

Regarding the structural similarity (SSIM) assessment of restored images, our DPLG algorithm obtains the best denoising results for 87 noisy images, the NCSR method is best for 60 noisy images, the BM3D method is best for 71 noisy images, and the K-SVD method is best for 2 noisy images. Experiments show that the proposed DPLG algorithm has the best average performance for restoring the perceptual visual effect, as shown on the bottom of Table II and Fig. 5 (a). Under the PSNR assessment, our DPLG method obtains the best denoising results for 67 noisy images, while the NCSR method is best for 31 noisy images, the BM3D method is best for 105 noisy images, and the K-SVD method is best for 18 noisy images. The DPLG also has a competitive performance in reconstructing the pixel intensity, as shown in Table II and Fig. 5 (b).

2) Comparing With External Denoising Methods: In this experiment, we compare with several denoising methods that exploit the statistics information of external, noise-free natural images. Our DPLG method only exploits the internal statistics information of the tested noisy image itself.

TABLE IV

COMPARISON OF DPLG WITH SEVERAL DENOISING METHODS USING EXTERNAL TRAINING IMAGES

Methods	Internal Information	External Information	Combining (In-Ex)	PSNR (σ=25) Barbara	Boat	House	Average
DPLG	yes	no	no	30.58	29.78	33.13	31.16
SCLW [15]	yes	yes	yes	32.68	32.58	33.07	32.78
NSCDL [16]	yes	yes	yes	30.83	29.87	32.99	31.23
SSDA [13]	no	yes	no	\	\	\	30.52
SDAE [14]	no	yes	no	29.69	29.95	32.58	30.74

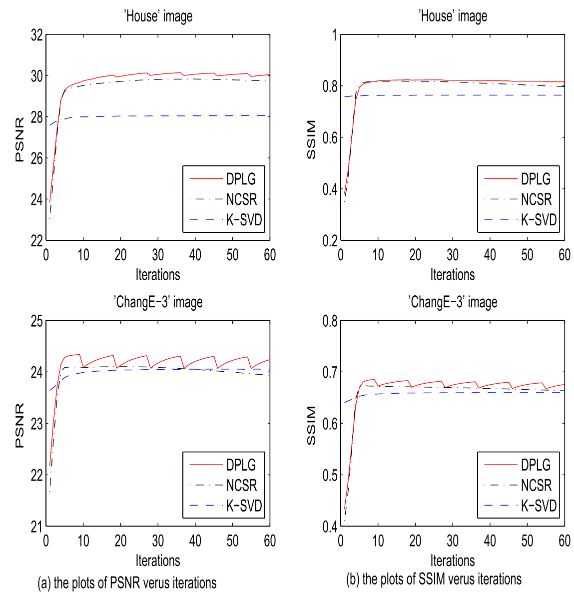


Fig. 6. (a) The PSNR values versus iterations by using DPLG, NCSR and KSVD when $\sigma = 50$; (b) The SSIM values versus iterations by using DPLG, NCSR and KSVD when $\sigma = 50$.

The SCLW and NSCDL denoising methods all exploit external statistics information from a clean training image set and the internal statistics from the observed noisy image. The SCLW learns the dictionary from external and internal examples, and the NSCDL learns the coupled dictionaries from clean natural images and exploits the non-local similarity from the test noisy images. The SSDA and SDAE adopt the same denoising technique, (i.e., learning a denoised mapping using a stacked Denoising Auto-encoder algorithm with sparse coding characteristics and a deep neural network structure [37]). Their aims are to find the mapping relations from noisy image patches to noise-free image patches by training on a large scale of external natural image set. Table IV shows the comparison of the DPLG with several internal-external denoising methods and external denoising methods, in terms of characteristics and the denoising performance on benchmark images. Our experiments show that the joint utilization of external and internal examples generally outperforms either stand-alone method, but no method is the best for all images. For example, our DPLG can obtain the best denoising result on the House benchmark image by using only the smoothing, sparseness and non-local self-similarity of the noisy image. Furthermore, our DPLG still maintains a better performance than the two external denoising methods SSDA and SDAE.

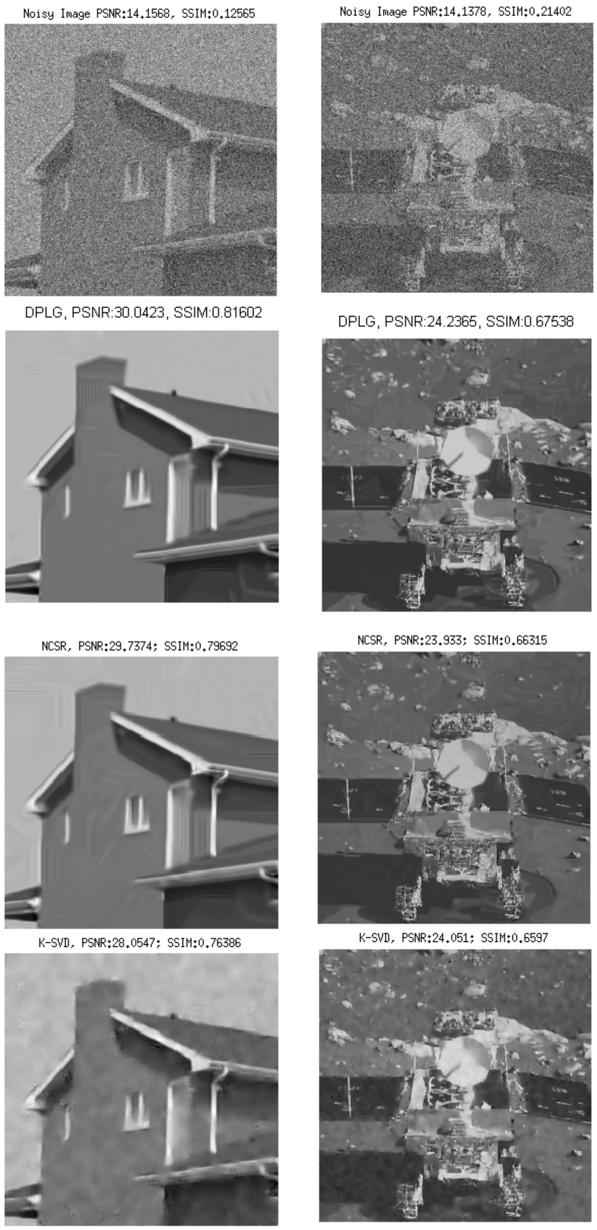
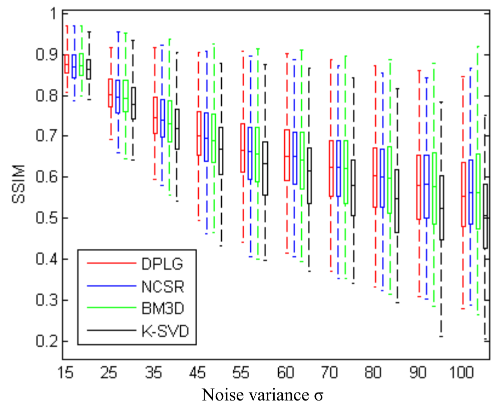
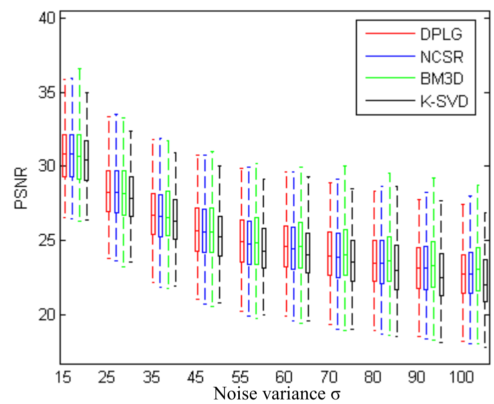


Fig. 7. The performance of the denoised 'House' image and 'ChangE-3' image at noise variance = 50 by using three iteration algorithms: DPLG, NCSR and K-SVD, which are iterated 60 times. Our DPLG method achieves the best denoised results (with corresponding to the second row, the denoised 'House' image, PSNR: 30.042, SSIM: 0.81602, and the denoised 'ChangE-3' image, PSNR: 24.2365, SSIM: 0.67538) of the several methods.

3) Comparing With Iteration Denoising Methods: Our DPLG method is an iterative method that allows the dictionary pair to be updated using the last denoised result and then obtains better 2D representations of the noisy patches from the noisy image. Fig. 6 and Fig. 7 show the denoising results of two typical noisy images ("House" and "ChangE-3") with strong noise corruption (noise variance = 50) after 60 iterations. The experimental results empirically demonstrate the convergence of the DPLG, as shown in Fig. 6. As the number of iterations increases, the denoised results get better. Fig. 6(a)-(b) display the plots of their PSNR values and SSIM values versus iterations, respectively. Comparing with two known iterative methods: K-SVD and NCSR, Fig.6 shows that our DPLG has a more rapidly increasing speed of PSNR and SSIM versus the iterations. It shows that our algorithm can achieve the best denoising performance among several iterative methods. The DPLG has competitive performance for reconstructing the smooth, the texture and the edge regions, as shown in the second row of Fig. 7.



(a)



(b)

Fig. 8. The distribution of the SSIM and PSNR values of denoising 100 noisy images corrupted by different noises by using NCSR, BM3D, KSVD and our DPLG algorithm. (a) The distribution of SSIM values. (b) The distribution of PSNR values.

C. Experiments on BSD Test Images

To further demonstrate the performance of the proposed DPLG method, the image denoising experiments were also conducted on 100 test images from the public benchmark Berkeley Segmentation Dataset [40]. Aiming at 10 different noisy versions of these images, that is, corresponding to a total of 1000 noisy images, the comparison experiments were completed by respectively running the NCSR, BM3D, K-SVD and our DPLG method. In the experiments, the parameter settings for the DPLG are the same as in the above experiments. Under the different Gaussian noise corruption, the average PSNR and SSIM values of 100 noisy images are shown in Fig. 8. Our DPLG method can achieve the best total performance for restoring the perceptual visual effect, as shown in the distribution of the SSIM values of 100 denoised images by four methods in Fig. 8(a), and has competitive performance for reconstructing pixel intensity, as shown in the distribution of the PSNR values of 100 denoised images in Fig. 8(b).

V. Conclusion

In this paper, we proposed a DPLG algorithm which is a novel 2D image denoising method working on Grassmann manifolds and leading to state-of-the-art performance. The DPLG algorithm has three primary advantages: 1) the adaptive dictionary pairs are rapidly learned via subspace partitioning and sub-dictionary pair merging on the Grassmann manifolds; 2) 2D sparse representations are notably easy to obtain; and 3) a graph Laplacian operator makes 2D sparse code representations vary smoothly for denoising. Moreover, extensive experimental results achieved on the benchmark images and the Berkeley segmentation datasets demonstrated that our DPLG algorithm can obtain better-than-average performance for restoring the perceptual visual effect than the state-of-the-art internal denoising methods. In the future, we would consider several potential problems, such as learning 3D multiple dictionaries for video denoising and exploring the fusion of manifold denoising and multi-dimensional sparse coding techniques.

Appendix A
The Proof of Theorem 1 in Section III-A

We give a simple proof of Theorem 1.

Proof: $\because A_1$ and A_2 are two $D \times m$-dimensional column-orthogonal matrixes.

$\therefore A_1^T A_2$ is a m-dimensional matrix.
According to the SVD decomposition of the matrix $A_1^T A_2$, if $\lambda_1, \lambda_2, \ldots, \lambda_m$ are m eigen-values from the largest to the smallest, and U, V are two m-dimensional orthogonal matrixes corresponding to the eigen-values $\lambda_1, \lambda_2, \ldots, \lambda_m$, then

$$U^T A_1^T A_2 V = diag(\lambda_1, \lambda_2, \ldots, \lambda_m)$$

And $\because V$ is a $m \times m$ orthogonal matrix.

$\therefore A_2 V$ is a rotation transformation of subspace A_2 by the rotation matrix V, that is:

$$span(A_2) = span(A_2 V)$$

Similarly, the same equation $span(A_1) = span(A_1 U)$.

Let u_k and v_k of Eq.(11) respectively be the $k-th$ column of matrixes A_1 and A_2, that is:

$$[u_1, u_2, \ldots, u_k, \ldots, u_m] = A_1 U$$
$$[v_1, v_2, \ldots, v_k, \ldots, v_m] = A_2 V$$

then,

$$\begin{aligned} & diag(\cos\theta_1, \ldots, \cos\theta_k, \ldots, \cos\theta_m) \\ & = [u_1, u_2, \ldots, u_k, \ldots, u_m]^T [v_1, v_2, \ldots, v_k, \ldots, v_m] \\ & = U^T A_1^T A_2 V \\ & = diag(\lambda_1, \lambda_2, \ldots, \lambda_m). \end{aligned}$$

□

Appendix B
The Proof of Theorem 2 in Section III-B

Proof: Considering the first term of Eq. (14).

$\because X_i$ is the corrupted image patch by noise $N(0, \sigma)$, and S_i is the code of the corresponding clear patch.

And

$$\because E\left(\|A_{ki}^T X_i B_{ki} - S_i\|\right) = E\left(\|A_{ki}^T \left(X_i - A_{ki} S_i B_{ki}^T\right) B_{ki}\|_F\right)$$

According to Eq.(18), $A_{ki} S_i B_{ki}^T$ is the reconstruction of the clear image patch, $\{A_{ki}, B_{ki}\}$ are the orthogonal dictionary pair.

$\therefore E\left(\|A_{ki}^T X_i B_{ki} - S_i\|\right) = \sigma^2$

As for the second term of Eq. (14). According to that the F_1-norm $\|S_i - S_j\|_{F,1}$ obeys to the Laplacain distribution in Eq. (6)

$$\begin{aligned} \therefore E(\gamma \sum_j w_{ij} \|S_i - S_j\|_{F,1}) &= \gamma\, E(\sum_j w_{ij} E(\|S_i - S_j\|_{F,1})) \\ &= \gamma\, E(\sum_j w_{ij} \sqrt{2}\sigma_i) \\ &= \gamma\, E(\sqrt{2}\sigma_i \sum_j w_{ij}) \\ &= \gamma\, E(\sqrt{2}\sigma_i), \quad (\because \sum_j w_{ij} = 1) \\ &= \gamma \sqrt{2}\sigma_i \end{aligned}$$

For preserving the scaling consistency, the ratio of two terms should be equal to 1.

$\therefore \frac{\sigma^2}{\gamma\sqrt{2}\sigma_i} = 1 \Leftrightarrow \gamma = \frac{\sqrt{2}\sigma_i}{\sigma^2}$. □

Acknowledgment

The authors would like to thank the anonymous reviewers for their help.

References

[1] P. Chatterjee and P. Milanfar, "Is denoising dead?" *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 895–911, Apr. 2010.

[2] C. Sutour, C.-A. Deledalle, and J.-F. Aujol, "Adaptive regularization of the NL-means: Application to image and video denoising," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3506–3521, Aug. 2014.

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

[4] C.-H. Xie, J.-Y. Chang, and W.-B. Xu, "Medical image denoising by generalised Gaussian mixture modelling with edge information," *IET Image Process.*, vol. 8, no. 8, pp. 464–476, Aug. 2014.

[5] J.-L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Process.*, vol. 11, no. 6, pp. 670–684, Jun. 2002.

[6] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, Feb. 2005.

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

[8] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Comput. Soc. Conf. CVPR*, vol. 2. Jun. 2005, pp. 60–65.

[9] A. Rajwade, A. Rangarajan, and A. Banerjee, "Image denoising using the higher order singular value decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 4, pp. 849–862, Apr. 2013.

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

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

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

[13] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Adv. NIPS*, Dec. 2012, pp. 350–358.

[14] H. Li, "Deep learning for image denoising," *Int. J. Signal Process., Image Process. Pattern Recognit.*, vol. 7, no. 3, pp. 171–180, Mar. 2014.

[15] Z. Y. Wang, Y. Z. Yang, J. C. Yang, and T. S. Huang, "Designing a composite dictionary adaptively from joint examples," in *Proc. IEEE Comput. Soc. Conf. CVPR*, Jun. 2015. [Online]. Available: http://arxiv.org/pdf/1503.03621.pdf

[16] L. Chen and X. Liu, "Nonlocal similarity based coupled dictionary learning for image denoising," *J. Comput. Inf. Syst.*, vol. 9, no. 11, pp. 4451–4458, Nov. 2013.

[17] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2138–2150, Jun. 2012.

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

[19] W. Zuo, L. Zhang, C. Song, and D. Zhang, "Texture enhanced image denoising via gradient histogram preservation," in *Proc. IEEE Comput. Soc. Conf. CVPR*, Jun. 2013, pp. 1203–1210.

[20] J. Ye, "Generalized low rank approximations of matrices," *Mach. Learn.*, vol. 61, nos. 1–3, pp. 167–191, Nov. 2005.

[21] J. Yang, D. Zhang, A. F. Frangi, and J.-Y. Yang, "Two-dimensional PCA: A new approach to appearance-based face representation and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 131–137, Jan. 2004.

[22] S.-J. Wang, J. Yang, M.-F. Sun, X.-J. Peng, M.-M. Sun, and C.-G. Zhou, "Sparse tensor discriminant color space for face verification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 876–888, Jun. 2012.

[23] S.-J. Wang, J. Yang, N. Zhang, and C.-G. Zhou, "Tensor discriminant color space for face recognition," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2490–2501, Sep. 2011.

[24] J. Liang, Y. He, D. Liu, and X. Zeng,"Image fusion using higher order singular value decomposition," *IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2898–2909, May 2012.

[25] A. Elmoataz, O. Lezoray, and S. Bougleux, "Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1047–1060, Jul. 2008.

[26] M. Hein and M. Maier, "Manifold denoising," in *Proc. Adv. NIPS*, Jun. 2006, pp. 561–568.

[27] M. Zheng *et al.*, "Graph regularized sparse coding for image representation," *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1327–1336, May 2011.

[28] S.-J. Wang, S. Yan, J. Yang, C.-G. Zhou, and X. Fu, "A general exponential framework for dimensionality reduction," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 920–930, Feb. 2014.

[29] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[30] X. Wang, Z. Li, and D. Tao, "Subspaces indexing model on Grassmann manifold for image search," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2627–2635, Sep. 2011.

[31] X. H. Zeng, S. W. Luo, J. Wang, and J. L. Zhao, "Geodesic distance-based generalized Gaussian Laplacian eigenmap," *Ruan Jian Xue Bao/J. Softw.*, vol. 20, no. 4, pp. 815–824, Apr. 2009.

[32] J. Hamm, "Subspace-based learning with Grassmann manifolds," Ph.D. dissertation, Dept. Elect. Syst. Eng., Univ. Pennsylvania, Philadelphia, PA, USA, 2008.

[33] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa, "Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2273–2286, Nov. 2011.

[34] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.

[35] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004.

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

[37] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 3, pp. 3371–3408, Mar. 2010.

[38] *K-SVD*. [Online]. Available: http://www.cs.technion.ac.il/~elad/software/, accessed Mar. 14, 2014.

[39] *Lander and Rover Capture Photos of Each Other*. [Online]. Available: http://english.cntv.cn/20131215/104015_1.shtml, accessed May 20, 2014.

[40] *Berkeley Segmentation DataSet (BSDS)*. [Online]. Available: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/fg/, accessed Jun. 13, 2014.