Globalized BM3D Using Fast Eigenvalue Filtering

Koki Suwabe, Masaki Onuki, Yuki Iizuka, and Yuichi Tanaka
Graduate School of BASE, Tokyo University of Agriculture and Technology
Koganei, Tokyo, 184-8588 Japan
{suwabe0126k, masaki.o, iizuka}@msp-lab.org, ytnk@cc.tuat.ac.jp

Abstract—**In this paper, we propose a progressive image denoising method using iterative filtering with Chebyshev polynomial approximation (CPA). It is known that a non-local/local image denoising method can be represented as matrix notation, and its denoising performance is improved by filtering the eigenvalues of the filter matrix. However, the eigenvalue filtering requires much computation time for eigendecomposition. To filter eigenvalues effectively, we proposed a fast eigenvalue filtering method using CPA [1]. The method drastically reduces the computation time but it still requires to construct a large sparse matrix. It often leads to much computational complexity. To overcome the problem, we propose an eigenvalue filtering method which does not construct a filter matrix by using the characteristic of the CPA. Experimental results show that our method is fast and applicable to large-size images. Additionally, the denoising performance of our method is almost better than those of the previous methods both in visual qualities and objective measures.**

Index Terms—**Chebyshev polynomial approximation, BM3D, denoising, eigendecomposition**

I. Introduction

In image processing and computer vision, image restoration is a very traditional problem but is still an important task [2]. A corrupted image is represented as

$$\mathbf{z} = \mathbf{y} + \mathbf{n}, \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^N$ and $\mathbf{z} \in \mathbb{R}^N$ denote vectors of the original and the observed images, respectively, and $\mathbf{n} \in \mathbb{R}^N$ denotes i.i.d. zero-mean additive white Gaussian noise with a standard deviation σ. The goal of image denoising is to restore $\mathbf{y}$ from $\mathbf{z}$. Many image denoising methods have been proposed so far [3]–[8].

In particular, BM3D presents the state-of-the-art performance [9]. It performs a redundant filtering to arrays of image blocks extracted from an input image under the assumption that an image has similarity among local blocks, i.e., we can find many similar blocks (patches) in an image. Both of the analysis and synthesis transforms in BM3D can be represented as very large matrices[1] [10]. This matrix representation enables BM3D to be used in different applications using an optimization approach e.g., an image deblurring [10].

Eigenvectors and eigenvalues of a filter matrix are considered as bases and frequencies such as the Fourier transform. From the characteristic, an input image can be transformed to the domain analogous to the frequency domain, and hence, eigenvalues are filtered to improve an image denoising performance. We call the technique *eigenvalue filtering*. Talebi et al. presented that the denoising performance could be improved by the eigenvalue filtering of a filter matrix [11]. However, a large matrix is difficult to perform eigendecomposition even in modern computers. To resolve the problem, global image denoising (GLIDE) [12] approximates full eigendecomposition of a filter matrix by using a subsampled pre-denoised image but it still needs much computation time for eigendecomposition.

[1]For example, matrix sizes of analysis and synthesis transforms are approximately $(1.18 \times 10^8) \times (1024^2)$ and $(1024^2) \times (1.18 \times 10^8)$ for an image with 1024×1024 pixels, respectively.

We proposed an eigenvalue filtering method without eigendecomposition [1]. In this method, we employ a very traditional signal processing and numerical analysis tool: Chebyshev polynomial approximation (CPA) [13]–[15]. It uses a shifted CPA often used in the context of graph signal processing [15]–[19]. To filter eigenvalues, our previous method using the CPA requires only multiplication of a large sparse matrix and an image vector. As a result, since eigendecomposition need not be calculated, its computation time is drastically reduced. However, it still requires to explicitly construct a filter matrix which leads to a heavy computational burden.

In this paper, we propose a progressive image denoising algorithm using the CPA while avoiding a construction of a filter matrix. In our method, the multiplication for the update of the Chebyshev polynomials is replaced by iteratively applying BM3D to an image vector. In our experiments, our method is faster than GLIDE[2], and its denoising performance is almost better than those of GLIDE and BM3D in visual qualities as well as objective measures. Additionally, its computational complexity is proportional to that of BM3D.

Our paper is structured as follows. Section II describes BM3D and its matrix notation. Section III presents previous methods using the eigenvalue filtering. We describe the proposed method in Section IV. Experimental results are presented in Section V to validate the effectiveness of the proposed approach. Finally, Section VI concludes the paper.

II. BM3D

BM3D estimates a restored image $\hat{\mathbf{y}}$ from an observation $\mathbf{z}$. The restored image can be represented as

$$\hat{\mathbf{y}} := \mathcal{F}_{\text{BM3D}}(\mathbf{z}), \tag{2}$$

where $\mathcal{F}_{\text{BM3D}}(\cdot)$ denotes the BM3D operator.

A. Algorithm

BM3D consists of the following three steps.

1) Grouping: For each reference block, a 3D array called group is constructed by collecting blocks which are similar to the reference block by block matching algorithm.
2) Collaborative filtering: Each group is transformed to the frequency domain by a 3D transform which can be realized by a 2D intra-block transform and a 1D inter-block transform. The group-wise estimate is obtained by processing coefficients then returning it to the spatial domain after the inverse 3D transform.
3) Aggregation: The estimated image is calculated by aggregation which is done by weighted averaging of all blocks.

BM3D has two phases. In each phase, the steps mentioned above are performed. In the first phase, hard-thresholding is applied to the coefficients in the frequency domain, whereas empirical Wiener filtering is used in the second phase.

[2]Since the matrix of BM3D is quite large and not easily constructed, we could not use our previous method in this experiment.

B. Matrix Notation of BM3D

Since BM3D has to be represented as a product of matrices that is required for using our approach, we formally introduce it [10].

Let $\mathbf{P}_j$ be an $N_{\text{bl}} \times N$ matrix to extract the jth block $\mathbf{y}_j \in \mathbb{R}^{N_{\text{bl}}}$ from $\mathbf{y}$, where N_{bl} is the number of pixels in a block. $\mathbf{D}_1$ and $\mathbf{D}_2$ are defined as $K_r \times K_r$ and $\sqrt{N_{\text{bl}}} \times \sqrt{N_{\text{bl}}}$ matrices representing the 1D and the 2D transforms, respectively, where K_r denotes the number of blocks in the rth group. By using these matrices, the group-wise coefficients are defined as

$$\boldsymbol{\omega}_r := \left(\sum_{j \in \mathbf{J}_r} \mathbf{d}_j \otimes \left[(\mathbf{D}_2 \otimes \mathbf{D}_2) \, \mathbf{P}_j \right] \right) \mathbf{y}, \tag{3}$$

where $\mathbf{d}_j$ denotes the jth column of $\mathbf{D}_1$, $\mathbf{J}_r$ denotes the set of indices of the blocks in the rth group, and $\otimes$ denotes the Kronecker product of matrices. Then, the rth group-wise analysis operator $\mathbf{\Phi}_r \in \mathbb{R}^{N_{\text{bl}} K_r \times N}$ can be written as

$$\mathbf{\Phi}_r := \textstyle\sum_{j \in \mathbf{J}_r} \mathbf{d}_j \otimes [(\mathbf{D}_2 \otimes \mathbf{D}_2)\mathbf{P}_j]. \tag{4}$$

BM3D analysis operator $\mathbf{\Phi} \in \mathbb{R}^{\sum_{r=1}^{R} (N_{\text{bl}} K_r) \times N}$ is obtained by arranging the group-wise analysis operator vertically, where R denotes the number of groups. It is expressed as

$$\boldsymbol{\omega} := \begin{bmatrix} \mathbf{\Phi}_1^T & \cdots & \mathbf{\Phi}_R^T \end{bmatrix}^T \mathbf{y} = \mathbf{\Phi} \mathbf{y}. \tag{5}$$

Similarly, the rth group-wise synthesis operator $\mathbf{\Psi}_r \in \mathbb{R}^{N \times N_{\text{bl}} K_r}$ can also be defined as

$$\mathbf{\Psi}_r := \textstyle\sum_{j \in \mathbf{J}_r} \mathbf{d}_j^T \otimes [\mathbf{P}_j^T (\mathbf{D}_2 \otimes \mathbf{D}_2)^T]. \tag{6}$$

Finally, the estimated image can be restored using $\mathbf{\Psi} \in \mathbb{R}^{N \times \sum_{r=1}^{R} (N_{\text{bl}} K_r)}$ as follows:

$$\mathbf{y} = \mathbf{\Psi} \boldsymbol{\omega} = \mathbf{W}^{-1} [g_1 \mathbf{\Psi}_1, \ldots, g_R \mathbf{\Psi}_R] \boldsymbol{\omega}, \tag{7}$$

where $\mathbf{W} := \sum_r g_r \sum_{j \in \mathbf{J}_r} \mathbf{P}_j^T \mathbf{P}_j$ is the normalization term and g_r is the group-wise weight. Therefore, by using $\mathbf{\Phi}$ and $\mathbf{\Psi}$, the transformation to the frequency domain and the inverse transformation can be performed, respectively. Thus, the following matrix $\mathbf{A}$ can be regarded as the overall filter matrix that is constructed based on a first phase estimate $\hat{\mathbf{y}}_{\text{ht}} = \mathbf{\Psi}_{\text{ht}} \mathbf{\Gamma}_{\text{ht}} \mathbf{\Phi}_{\text{ht}} \mathbf{z}$:

$$\mathbf{A} := \mathbf{\Psi}_{\text{wie}} \mathbf{\Gamma}_{\text{wie}} \mathbf{\Phi}_{\text{wie}}, \tag{8}$$

where $\cdot_{\text{ht}}$ and $\cdot_{\text{wie}}$ denote operators of the first and second phases, respectively. Additionally, $\mathbf{\Gamma}_{\text{ht}}$ and $\mathbf{\Gamma}_{\text{wie}}$ respectively denote the hard-thresholding and empirical Wiener filtering operators of the first and second phases. As a result, the BM3D algorithm can be represented as follows:

$$\hat{\mathbf{y}} = \mathcal{F}_{\text{BM3D}}(\mathbf{z}) = \mathbf{A}\mathbf{z}. \tag{9}$$

III. Eigenvalue Filtering of Denoising Filter Matrix

In this section, we introduce methods for improving a denoising performance using eigenvalue filtering.

A. Global Image Denoising

A denoising filter matrix $\mathbf{A}$ can be generally decomposed as

$$\mathbf{A} = \mathbf{V} \mathbf{S} \mathbf{V}^{-1}, \tag{10}$$

where $\mathbf{V} := [\mathbf{v}_1, \ldots, \mathbf{v}_N]$ is the matrix composed of eigenvectors of $\mathbf{A}$ and $\mathbf{S} := \operatorname{diag}(\lambda_1, \ldots, \lambda_N)$ is the diagonal matrix with corresponding eigenvalues λ on a diagonal. Let $\mathcal{H}(\mathbf{A}, l, m)$ be an eigenvalue filtered matrices

$$\mathcal{H}(\mathbf{A}, l, m) := \mathbf{V} \operatorname{diag}(h_{l,m}(\lambda_1), \ldots, h_{l,m}(\lambda_i), \ldots, h_{l,m}(\lambda_N)) \mathbf{V}^{-1}, \tag{11}$$

where $h_{l,m}(\cdot)$ is an arbitrary filter kernel, and parameters l and m control the shrinkage strength and truncation of eigenvalues, respectively.

The largest eigenvalue corresponds to the eigenvector whose all elements are constant. On the other hand, smaller ones correspond to eigenvectors whose elements are oscillated rapidly. Therefore, the eigenvectors can be regarded as frequency. Intuitively, small and large eigenvalues correspond to high and low frequencies, respectively [12]. Such intuition is also used in graph signal processing [15]. The filter kernel controls the denoising strength. Therefore, by filtering eigenvalues properly, the performance could be improved. Unfortunately, the exact eigendecomposition is hardly computed since the filter matrix is generally large.

In GLIDE, the following approximation is performed to overcome the problem mentioned above. At first, a pre-denoised image is calculated, and then, the symmetric filter matrix is approximated by using Nyström method [20], Sinkhorn algorithm [21], and the orthogonalization of a filter matrix. As a result, eigendecomposition is performed by using only a small fraction of pixels of the whole image in GLIDE. Unfortunately, it still requires much computation time for eigendecomposition.

To determine the optimal parameters l and m, a mean squared error (MSE) should be minimized. MSE is defined as

$$\text{MSE} := \frac{1}{N} \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2. \tag{12}$$

However, the ground-truth $\mathbf{y}$ is not given. Therefore, GLIDE estimates MSE inspired by SURE [22]–[24] represented as

$$\varepsilon := \frac{1}{N} \|\hat{\mathbf{y}} - \mathbf{z}\|_2^2 + \frac{2\sigma^2}{N} \operatorname{div}_{\mathbf{z}}(\hat{\mathbf{y}}) - \sigma^2, \tag{13}$$

where $\operatorname{div}_{\mathbf{z}}(\cdot)$ is the divergence operator over $\mathbf{z}$ [1], [22].

B. Eigenvalue Filtering with CPA

1) CPA for Scalar Function: Prior to eigenvalue filtering using CPA for filter matrices, we describe CPA of a scalar function of $y \in [-1, 1]$ [14].

By using Chebyshev series, every function (filter kernel) can be represented as

$$h(y) := \frac{1}{2} c_0 + \sum_{k=1}^{\infty} c_k T_k(y), \tag{14}$$

where c_k denotes a Chebyshev coefficient and $T_k(\cdot)$ denotes the kth order Chebyshev polynomial which is defined by the relation:

$$T_k(y) := \cos(k \arccos(y)). \tag{15}$$

It can also be expressed in the fundamental recurrence relation as follows:

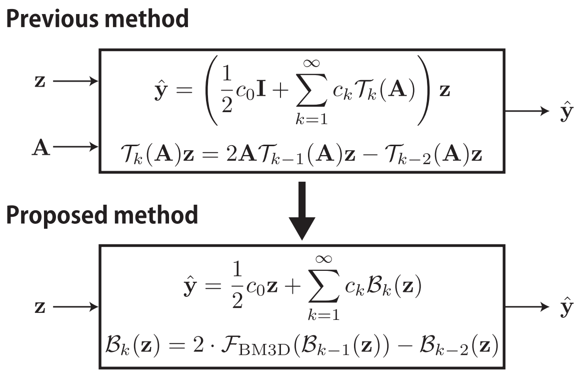
$$T_k(y) = 2y T_{k-1}(y) - T_{k-2}(y), \tag{16}$$

whose initial conditions are $T_0(y) = 1$, $T_1(y) = y$. Furthermore, by using the orthogonality of sinusoidal wave, c_k in (14) is defined as

$$c_k := \frac{2}{\pi} \int_{-1}^{1} \frac{T_k(y) h(y)}{\sqrt{1 - y^2}} dy = \frac{2}{\pi} \int_0^{\pi} \cos(k\theta) h(\cos\theta) d\theta. \tag{17}$$

2) Eigenvalue Filtering with CPA: Let us describe the matrix version of CPA [1]. We assume that the filter kernel $h_p(\lambda)$ has one parameter p. In general, eigenvalues of the filter matrix are located in $\lambda \in [-1, 1]$. Therefore, they can be filtered by using CPA. The matrix form of the filter kernel $\mathcal{H}(\mathbf{A}, p)$ is represented by using Chebyshev series:

$$\mathcal{H}(\mathbf{A}, p) := \frac{1}{2} c_0 \mathbf{I} + \sum_{k=1}^{\infty} c_k \mathcal{T}_k(\mathbf{A}), \tag{18}$$



Previous method
$\mathbf{z}$
$\mathbf{A}$
$\hat{\mathbf{y}} = \left(\frac{1}{2}c_0\mathbf{I} + \sum_{k=1}^{\infty} c_k \mathcal{T}_k(\mathbf{A})\right)\mathbf{z}$
$\mathcal{T}_k(\mathbf{A})\mathbf{z} = 2\mathbf{A}\mathcal{T}_{k-1}(\mathbf{A})\mathbf{z} - \mathcal{T}_{k-2}(\mathbf{A})\mathbf{z}$
$\hat{\mathbf{y}}$
Proposed method
$\mathbf{z}$
$\hat{\mathbf{y}} = \frac{1}{2}c_0\mathbf{z} + \sum_{k=1}^{\infty} c_k \mathcal{B}_k(\mathbf{z})$
$\mathcal{B}_k(\mathbf{z}) = 2 \cdot \mathcal{F}_{\mathrm{BM3D}}(\mathcal{B}_{k-1}(\mathbf{z})) - \mathcal{B}_{k-2}(\mathbf{z})$
$\hat{\mathbf{y}}$

Fig. 1. Comparison of the proposed method with the previous method.

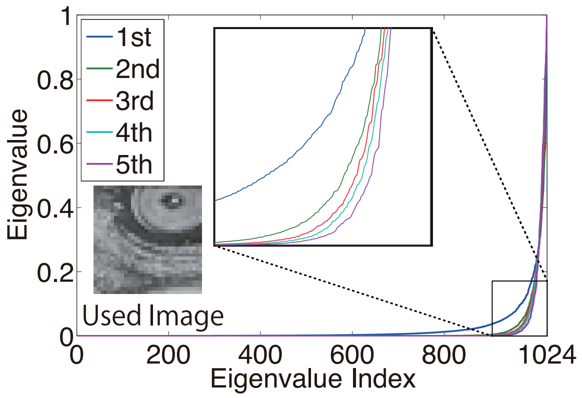


Fig. 2. Eigenvalue distribution of $\mathcal{B}_k(\cdot)$ according to the iteration number.

where $\mathcal{T}_k(\cdot)$ is the kth order polynomial of the matrix form defined as

$$\mathcal{T}_k(\mathbf{A}) := \mathbf{V}\mathrm{diag}(\cos k\theta_1, \ldots, \cos k\theta_i, \ldots, \cos k\theta_N)\mathbf{V}^{-1}. \tag{19}$$

Similar to (16), $\mathcal{T}_k(\mathbf{A})$ is calculated by using the recurrence relation:

$$\mathcal{T}_k(\mathbf{A}) := 2\mathbf{A}\mathcal{T}_{k-1}(\mathbf{A}) - \mathcal{T}_{k-2}(\mathbf{A}), \tag{20}$$

whose initial conditions are $\mathcal{T}_0(\mathbf{A}) = \mathbf{I}$ and $\mathcal{T}_1(\mathbf{A}) = \mathbf{A}$. Here, the estimated image with CPA is calculated as

$$\hat{\mathbf{y}}_p(\mathbf{z}) = \mathcal{H}(\mathbf{A}, p)\mathbf{z} = \left(\frac{1}{2}c_0\mathbf{I} + \sum_{k=1}^{\infty} c_k \mathcal{T}_k(\mathbf{A})\right)\mathbf{z}. \tag{21}$$

In practice, Equation (21) is calculated until the dth order approximation by using an arbitrary parameter d for an effective calculation.

IV. Proposed Method

Eigenvalue filtering itself will become efficient by using CPA. However, we still have a problem of constructing a large sparse matrix like the filter matrix of BM3D. Fig. 1 shows a comparison between the proposed and our previous methods [1]. By using the BM3D operator, the initial conditions in (21) can be replaced as

$$\mathcal{T}_0(\mathbf{A})\mathbf{z} := \mathcal{B}_0(\mathbf{z}) = \mathbf{z}, \tag{22}$$
$$\mathcal{T}_1(\mathbf{A})\mathbf{z} := \mathcal{B}_1(\mathbf{z}) = \mathcal{F}_{\mathrm{BM3D}}(\mathbf{z}), \tag{23}$$

where $\mathcal{B}_k(\cdot)$ is the BM3D operator in the kth iteration. Then, $\mathcal{T}_k(\mathbf{A})\mathbf{z}$ in (21) can be rewritten as

$$\mathcal{T}_k(\mathbf{A})\mathbf{z} \simeq \mathcal{B}_k(\mathbf{z}) = 2 \cdot \mathcal{F}_{\mathrm{BM3D}}(\mathcal{B}_{k-1}(\mathbf{z})) - \mathcal{B}_{k-2}(\mathbf{z}). \tag{24}$$

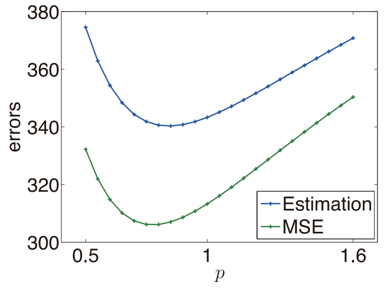


Fig. 3. MSE and its estimate according to the parameter p of the filter kernel $h_p(\lambda) = \mathrm{sign}(\lambda)|\lambda|^p$.

Note that $\mathcal{T}_k(\mathbf{A})\mathbf{z}$ and $\mathcal{B}_k(\mathbf{z})$ are not consistent strictly over k because the BM3D operator depends on the input image. For correctly using CPA in our approach, there must exist a strong assumption: Eigenvalue distributions of the denoising operator over k are the same. In this paper, we experimentally validate that the distributions roughly follow the assumption. Fig. 2 shows eigenvalue distributions of $\mathcal{B}_k(\cdot)$ according to the iteration number. We used a 32×32 portion of *Mandrill* and the fixed σ for every iteration. It is observed that, irrelevant to the iteration number, the eigenvalue distributions are very similar, especially from the second iteration. As a result, the estimated image can be calculated as

$$\hat{\mathbf{y}}_p(\mathbf{z}) = \mathcal{H}(\mathbf{A}, p)\mathbf{z} = \left(\frac{1}{2}c_0\mathbf{I} + \sum_{k=1}^{d-1} c_k \mathcal{T}_k(\mathbf{A})\right)\mathbf{z} \tag{25}$$
$$\simeq \frac{1}{2}c_0\mathbf{z} + \sum_{k=1}^{d-1} c_k \mathcal{B}_k(\mathbf{z}). \tag{26}$$

This means that, without constructing a filter matrix, the restored image $\hat{\mathbf{y}}_p$ is obtained by eigenvalue filtering with Chebyshev coefficients c_k.

Similar to the previous approaches [1], [12], the optimal parameter p_{opt} is estimated. For an effective calculation, $\mathrm{div}_{\mathbf{z}}(\hat{\mathbf{y}})$ of the second term in (13) is approximated as

$$\mathrm{div}_{\mathbf{z}}(\mathcal{H}(\mathbf{A}, p)\mathbf{z}) \simeq \mathbf{b}^T(\hat{\mathbf{y}}_p(\mathbf{z} + \mathbf{b}) - \hat{\mathbf{y}}_p(\mathbf{z})), \tag{27}$$

where $\mathbf{b} \in \mathbb{R}^N$ is a noise signal, which is i.i.d. zero-mean additive white Gaussian noise with unit variance. The detail for the derivation of (27) is described in [1], [25]. Fig. 3 shows the estimation and the actual MSE according to the parameter p when *Mandrill* ($\sigma = 40$) is denoised. By using (13) and (27), p_{opt} can also be calculated without the filter matrix.

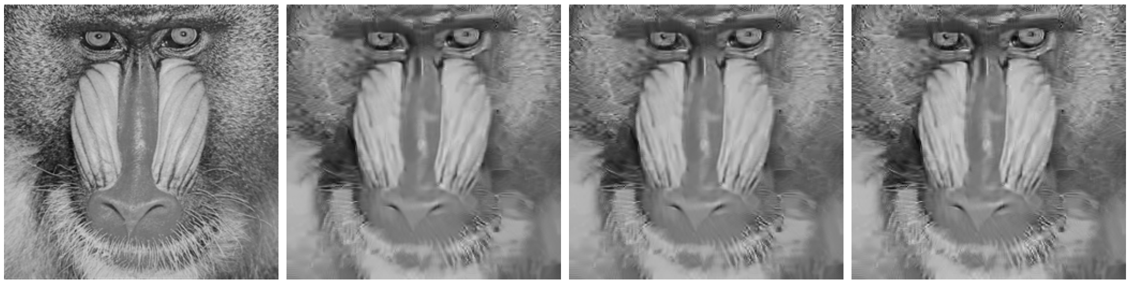
V. Experimental Results

In this section, some results to evaluate the proposed method are presented. Noisy grayscale images were denoised by using BM3D, GLIDE, and the proposed method. White Gaussian noise with standard deviation $\sigma \in \{10, 20, 30, 40, 50\}$ is used. The parameter p varied 0.5 to 1.6 in steps of 0.05, and the degree of truncated CPA is set to $d = 9$ experimentally. Furthermore, we experimentally set the filter kernel as $h_p(\lambda) = \mathrm{sign}(\lambda)|\lambda|^p$. As objective measures, peak signal to noise ratio (PSNR) and structural similarity (SSIM) [26] are used. For BM3D and GLIDE, MATLAB implementations by authors were used[3]. All experiments are performed on Intel Xeon E5-2690 2.9 GHz CPU and 62.9 GB RAM.

[3]Available at http://www.cs.tut.fi/~foi/GCF-BM3D/index.html (BM3D) and at https://users.soe.ucsc.edu/~htalebi/GLIDE.php (GLIDE).

TABLE I
DENOISING PERFORMANCE COMPARISON LEFT: PSNR (dB), RIGHT: SSIM

σ	Methods	*Bridge* (256×256)	*Goldhill* (256×256)	*Mandrill* (256×256)	*Building* (256×256)	*Mandrill* (512×512)	*Bridge* (512×512)
10	BM3D	29.84 / 0.911	31.80 / 0.880	30.56 / 0.905	**33.16** / **0.939**	**30.58** / 0.898	31.14 / 0.906
	GLIDE	29.81 / **0.913**	31.72 / 0.881	30.54 / 0.904	32.91 / 0.938	- / -	- / -
	Proposed	**29.86** / **0.913**	**31.86** / **0.884**	**30.57** / **0.906**	**33.16** / **0.939**	**30.58** / **0.899**	**31.16** / **0.908**
20	BM3D	25.46 / 0.765	28.50 / 0.775	26.39 / 0.773	29.35 / 0.862	26.60 / 0.793	27.25 / 0.788
	GLIDE	25.62 / 0.784	28.57 / **0.785**	26.55 / 0.788	29.30 / 0.865	- / -	- / -
	Proposed	**25.66** / **0.789**	**28.59** / 0.784	**26.56** / **0.791**	**29.40** / **0.866**	**26.65** / **0.801**	**27.36** / **0.802**
30	BM3D	23.55 / 0.647	26.91 / 0.706	24.33 / 0.651	27.32 / 0.790	24.57 / 0.703	25.44 / 0.697
	GLIDE	23.68 / 0.678	26.71 / 0.711	24.57 / 0.686	27.26 / 0.792	- / -	- / -
	Proposed	**23.73** / **0.679**	**26.96** / **0.714**	**24.58** / **0.689**	**27.37** / **0.794**	**24.65** / **0.720**	**25.55** / **0.713**
40	BM3D	22.51 / 0.572	**25.84** / 0.654	23.10 / 0.558	25.89 / 0.722	23.09 / 0.618	24.31 / 0.628
	GLIDE	22.43 / 0.584	25.70 / 0.640	**23.23** / 0.573	25.87 / **0.729**	- / -	- / -
	Proposed	**22.55** / **0.586**	25.83 / **0.655**	23.19 / **0.582**	**25.90** / 0.724	**23.13** / **0.633**	**24.33** / **0.637**
50	BM3D	21.81 / 0.509	**25.04** / 0.610	22.43 / 0.489	24.93 / 0.663	22.35 / 0.549	23.56 / 0.571
	GLIDE	21.81 / **0.547**	25.01 / **0.616**	**22.60** / 0.518	24.85 / **0.680**	- / -	- / -
	Proposed	**21.93** / 0.540	**25.04** / 0.615	22.59 / **0.525**	**24.95** / 0.673	**22.58** / **0.588**	**23.64** / **0.588**


(a) Original *Mandrill* (b) BM3D (c) GLIDE (d) Proposed

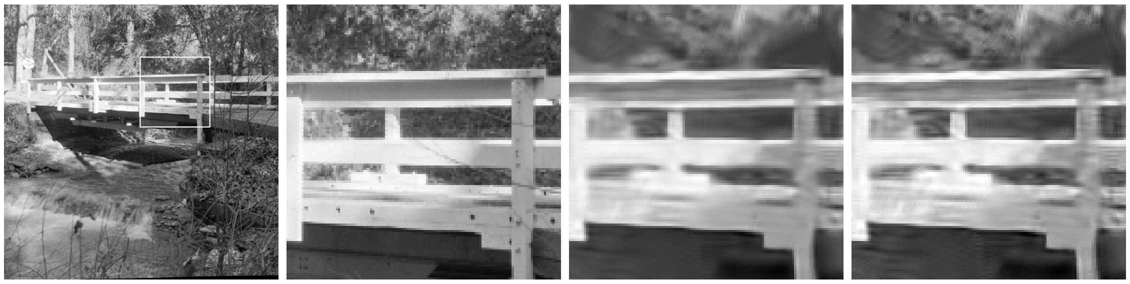

(e) Original *Bridge* (f) Zoomed-in part (g) BM3D (h) Proposed

Fig. 4. Comparison of denoised images. (Top row) *Mandrill* (256×256, $\sigma = 40$). (Bottom row) *Bridge* (512×512, $\sigma = 50$).

The objective performances are summarized in Table I, and the denoised *Mandrill* and *Bridge* images are shown in Fig. 4. The denoising performance of our method is comparable to or better than those of the previous methods. Moreover, it is also improved visually; textures are recovered more clearly.

Table II compares the averaged execution time of BM3D, GLIDE and the proposed method after 20 executions. Our method performs two times faster than GLIDE for 256×256 images. Moreover, GLIDE could not be performed for 512×512 or larger images. Since our method has to perform the BM3D algorithm iteratively to decide p_{opt}, its computational complexity is proportional to the number of iterations. Furthermore, the optimal parameter search can be easily parallelized for accelerating our method.

VI. CONCLUSION

In this paper, an image denoisng method using eigenvalue filtering with CPA was proposed. Eigenvalue filtering is realized by iterative filtering without constructing an explicit filter matrix. The denoising performance of our method is comparable to or better than those of the previous methods in objective measures, and visual qualities are also satisfactory. As a future work, we will further investigate and study eigenvalue distributions in each iteration, which leads to performance improvements.

TABLE II
EXECUTION TIME COMPARISON [SEC]

Image size	BM3D [9]	GLIDE [12]	Proposed
256×256	0.8	115.4	51.8
512×512	3.1	Out of memory	225.1
1024×1024	18.1	Out of memory	946.4

ACKNOWLEDGEMENT

This work was supported in part by MEXT Tenure-Track Promotion Program.

References

[1] M. Onuki, S. Ono, K. Shirai, and Y. Tanaka, “Non-local/local image filters using fast eigenvalue filtering,” in *Proc. ICIP*, 2015.

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[3] M. Petrou and C. Petrou, *Image Processing: The Fundamentals*. John Wiley & Sons, 2010.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing Third Edition*. Prentice Hall, 2008.

[5] A. Buades, B. Coll, and J. M. Morel, “A non-local algorithm for image denoising,” in *Proc. CVPR*, vol. 2, pp. 60–65, Jun. 2005.

[6] ——, “A review of image denoising algorithms, with a new one,” *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, Jul. 2005.

[7] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Proc. ICCV*, pp. 839–846, Jan. 1998.

[8] 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.

[9] 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.

[10] A. Danielyan, V. Katkovnik, and K. Egiazarian, “BM3D frames and variational image deblurring,” *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1715–1728, Apr. 2012.

[11] H. Talebi, X. Zhu, and P. Milanfar, “How to SAIF-ly boost denoising performance,” *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1470–1485, Apr. 2013.

[12] H. Talebi and P. Milanfar, “Global image denoising,” *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 755–768, Feb. 2014.

[13] G. M. Phillips, *Interpolation and Approximation by Polynomials*. Springer Science & Business Media, 2003.

[14] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*. CHAPMAN & HALL / CRC, Sep. 2002.

[15] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[16] Y. Iizuka and Y. Tanaka, “Depth map denoising using collaborative graph wavelet shrinkage on connected image patches,” in *Proc. ICIP*, pp. 828–832, Oct. 2014.

[17] Y. Tanaka and A. Sakiyama, “M-channel oversampled graph filter banks,” *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3578 – 3590, Jul. 2014.

[18] A. Sakiyama and Y. Tanaka, “Oversampled graph Laplacian matrix for graph signals,” in *Proc. EUSIPCO*, pp. 2225–2229, Sep. 2014.

[19] M. Onuki and Y. Tanaka, “Trilateral filter on graph spectral domain,” in *Proc. ICIP*, pp. 2046–2050, Oct. 2014.

[20] C. Williams and M. Seeger, “Using the Nyström method to speed up kernel machines,” in *Proc. NIPS*, pp. 682–688, Nov. 2001.

[21] P. Milanfar, “Symmetrizing smoothing filters,” *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 263–284, May 2013.

[22] C. Stein, “Estimation of the mean of a multivariate normal distribution,” *The Annals of Statistics*, vol. 9, no. 6, pp. 1135–1151, Jan. 1981.

[23] H. Kishan and C. S. Seelamantula, “SURE-fast bilateral filters,” in *Proc. ICASSP*, pp. 1129–1132, Mar. 2012.

[24] T. Qiu, A. Wang, N. Yu, and A. Song, “LLSURE: local linear SURE-based edge-preserving image filtering,” *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 80–90, Jan. 2013.

[25] S. Ramani, T. Blu, and M. Unser, “Monte-carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms,” *IEEE Trans. Image Process.*, vol. 17, no. 9, pp. 1540–1554, Sep. 2008.

[26] 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.