Efficient Noise Filtration of Images by Low-Rank Singular Vector Approximations of Geodesics' Gramian Matrix

Kelum Gajamannage[a,*], Yonggi Park[a], Sunil Mathur[a]

[a]*Department of Mathematics and Statistics, Texas A&M University–Corpus Christi, Corpus Christi, TX–78412, USA.*

Abstract

Modern society is interested in capturing high-resolution and fine-quality images due to the surge of sophisticated cameras. However, the noise contamination in the images not only inferior people's expectations but also conversely affects the subsequent processes if such images are utilized in computer vision tasks such as remote sensing, object tracking, etc. Even though noise filtration plays an essential role, real-time processing of a high-resolution image is limited by the hardware limitations of the image-capturing instruments. Geodesic Gramian Denoising (GGD) is a manifold-based noise filtering method that we introduced in our past research which utilizes a few prominent singular vectors of the geodesics' Gramian matrix for the noise filtering process. The applicability of GDD is limited as it encounters $\mathcal{O}(n^6)$ when denoising a given image of size $n \times n$ since GGD computes the prominent singular vectors of a $n^2 \times n^2$ data matrix that is implemented by singular value decomposition (SVD). In this research, we increase the efficiency of our GGD framework by replacing its SVD step with four diverse singular vector approximation techniques. Here, we compare both the computational time and the noise filtering performance between the four techniques integrated into GGD.

Keywords: Efficient algorithm, noise filtration, singular vector approximation, geodesics' Gramian

1. Introduction

The surge of high-caliber imaging devices in the fields such as medical diagnosis, astronomy, and film industry produces high-resolution and fine-quality images [18]. However, due to unforeseen instrumental and environmental conditions, the images generated by them might be contaminated with noise [6]. Thus, filtration of noise to restore the natural image quality that includes all the image features including edges and texture is required as a prepossessing step. Especially, such noise filtration is essential if the images are used for any subsequent processes such as remote sensing or object tracking. Since noise, edges, and texture are high-frequency components of an image, discerning them to filter out noise is an arduous task [33]. This limitation frequently causes the loss of some vital features of the recovered image. Thus, restoration of the original quality of the recovered images without losing essential features is a crucial property that denoising methods should possess. Image denoising methods are two types, patch-based and pixel-based, where patch-based image denoising methods have significant advantages such as their efficiently smooth flat regions due to overlaps between patches, and their ability to preserve fine image details and sharp edges [2].

Deep learning based image denoising methods, such as [5], [36], and [37], learn a mapping function on a training set that contains clean image pairs by optimizing a loss function [7]. Recently, these methods have received surged attention as they have performed well in many computer vision tasks [7]. However, the deep learning image denoising frameworks suffer from major drawbacks, that are prominent in typical neural networks, such as difficulties in training when the noise contamination is high, vanishing gradient when the network is considerably deep, and high computational cost due to repeated training [16].

Geodesic Gramian Denoising, abbreviated as GGD, [14], is a novel patch-based denoising method that uses singular vectors of the Gramian matrix of geodesic distances between patches. In GGD, first, the noisy image of, say of size $n \times n$, is partitioned into partially overlapping moving square-shaped patches with a known length, say ρ, such that each patch is centered at one unique pixel of the image. Each of these n^2 patches is a point in a ρ^2-dimensional space where a low-dimensional manifold underlies [12, 13]. Then, GGD computes geodesic distances between each pair of points in the patch space. This geodesic distance matrix of size $n^2 \times n^2$ is converted into its Gramian matrix where the singular vectors of the Gramian matrix are utilized to generate the noise-filtered version of the given image. We have shown in Ref. [14] using five benchmark computer vision test images that GGD attains superior denoising performance. However, GGD suffers from high computational

---

complexity of $\mathcal{O}(n^6)$ when denoising an image of size $n \times n$ since it requires the computation of singular value decomposition (SVD) of a Gramian matrix of size $n^2 \times n^2$. Numerical implementation of SVD is carried out using the LAPACK routine available in Ref. [3] that includes two processes: first, the given matrix of size $m \times n$ is reduced to a bidiagonal matrix form using Householder transformations with $\mathcal{O}(mn \min(m, n))$ operations, and then use the QR algorithm in Ref. [9, 10], to compute the SVD of the bidiagonal matrix [19]. Thus, in this research, we approximate the singular vectors using four iterative schemes and replace the computationally expensive SVD scheme in GGD with those approximations.

Monte Carlo Low-Rank Approximation, abbreviated as MCLA, available in Ref. [11], iteratively approximates a L-rank surrogate for the conventional SVD of a data matrix A of size $m \times n$. Each iteration involves the reading of $\mathcal{O}(L)$ of columns or rows of A, and so the complexity of their algorithm is $O(kmn)$. Compared to other known algorithms, MCLA guarantees that each iteration is a better L-rank approximation than the previous iteration. Augmented Lanczos Bidiagonalization, abbreviated as ALB, available in Ref. [4], computes either a few largest or smallest singular triplets of a large matrix. For that, it makes a sequence of projections on low-dimensional Krylov subspaces that are determined similarly to the Lanczos bidiagonalization method. Preconditioned Iterative Multimethod Eigensolver, abbreviated as PIME, available in Ref. [35], provides a preconditioned, two-stage approach to effectively and accurately approximate a small number of extreme singular triplets. PIME is based on the state-of-the-art eigensolver package, named PRIMME, for both the largest and the smallest SVs. Randomized Singular Value Decomposition, abbreviated as RSVD, available in Ref. [22], is a powerful tool that computes partial matrix decompositions and then random samples of those decompositions to identify a subspace that captures most of the features of a matrix. Now, we replace the SVD step of GGD, with the above four approximation frameworks, MCLA, ALB, PIME, and RSVD, to create four hybrid versions of GGD.

This paper is organized as follows: first, we provide the technical details about GGD, MCLA, PRIME, and RSVD in Sec. 2; then, we analyze and compare the denoising performance of GGD and those four hybrid GGD frameworks with respect to their parameters, using three reconstruction metrics PSNR, SSIM, and RE in Sec. 3. Finally, we provide a summary along with conclusions in Sec. 4. Table 1 and Table 2 state nomenclature and abbreviations, respectively.

2. Method

Here, first, we state GGD algorithm [14], with details and then introduce four diverse algorithms for approximating SVD. These four algorithms are Monte Carlo Rank Approximation (MCLA) [11], Augmented Lanczos Bidiagonalization [4], Preconditioned Iterative Multimethod Eigensolver (PIME) [35], and Randomized Singular Value Decomposition (RSVD) [22].

2.1. Geodesic Gramian Denoising

GGD presented in Ref. [14] undergoes five steps including a step that requires it to compute the SVD of a data matrix of size $n^2 \times n^2$ for a given image of size $n \times n$. The computational complexity of this step is $O(n^6)$ for denoising an image of size $n \times n$ that we will replace with four singular vector approximation techniques.

First, GGD partitions the input noisy image, denoted as $\mathcal{U}_{n \times n}$, into equal-sized square-shaped overlapping patches, denoted as $\boldsymbol{u}(\boldsymbol{x}_{ij})$'s; $i, j = 1, \ldots, n$, of odd lengths, denoted as ρ, centered at each pixel of the image. For simplicity, we write $\boldsymbol{u}(\boldsymbol{x}_k)$ for $\boldsymbol{u}(\boldsymbol{x}_{ij})$ where $k = n(i-1) + j$ and $1 \leq k \leq n^2$ in some places. Each patch $\boldsymbol{u}(\boldsymbol{x}_k)$; $i = 1, \ldots, n^2$ can be represented as a ρ^2-dimensional point. All the patches of a given image represent a high-dimensional data cloud that can be projected onto a low-dimensional manifold. Second, we use the neighbor search algorithm in Ref. [1] to create a graph structure, denoted as $G(V, E)$ where V represents vertices and E represents edges, on this dataset by defining the points, $\{\boldsymbol{u}(\boldsymbol{x}_k) | k = 1, \ldots, n^2\}$, as the vertex set V. For a give neighborhood parameter, defined as δ, we define the edge set E by joining each point $\boldsymbol{u}(\boldsymbol{x}_k)$ to its δ nearest neighbors such that the weight of the edge between the point $\boldsymbol{u}(\boldsymbol{x}_k)$ and such neighbor point, defined as $\boldsymbol{u}(\boldsymbol{x}_{k'})$, is the Euclidean distance between them, denoted as $d(k, k')$ where

$$d(k, k') = \|\boldsymbol{u}(\boldsymbol{x}_k) - \boldsymbol{u}(\boldsymbol{x}_{k'})\|_2. \tag{1}$$

GGD approximates the geodesic distance between two patches in the patch-set as the shortest path distance between the corresponding two vertices in the graph $G(V, E)$ with the aid of Floyd's algorithm [8].

Third, GGD transforms the geodesic distance matrix, denoted as $\mathcal{D} \in \mathbb{R}_{\geq 0}^{n^2 \times n^2}$, into its Gramian matrix, denoted as $\mathcal{G}_{n^2 \times n^2}$, using

$$\mathcal{G}[i, j] = -\frac{1}{2} \left[\mathcal{D}[i, j] - \mu_i(\mathcal{D}) - \mu_j(\mathcal{D}) + \mu(\mathcal{D}) \right], \tag{2}$$

where $\mu_i(\mathcal{D})$, $\mu_j(\mathcal{D})$, and $\mu(\mathcal{D})$ are the means of the i-th row of the matrix $\mathcal{D}$, j-th column of that matrix, and the mean of the full matrix, respectively, [25]. Fourth, the noisy patches $\boldsymbol{u}_k$ are denoised using only a

Table 1: Nomenclature

Notation	Description
σ_i	i-th Singular value
λ_l	l-th eigenvalue
ϵ	Machine epsilon
η	Tolerance
α	Relative Norm of Noise
ζ	Relative noise
$d(k, k')$	Distance between patches $\boldsymbol{u}(\boldsymbol{x}_k)$ and $\boldsymbol{u}(\boldsymbol{x}_{k'})$
L	Singular triplets' threshold such that $l = 1, \ldots, L$
L'	Singular triplets' threshold of ALB such that $\{\hat{\sigma}_i, \hat{\boldsymbol{u}}_i, \hat{\boldsymbol{v}}_i\}, i = 1, ..., L'$
i, j, k	Indices for matrices
l	Index for singular triplets.
(t)	Index for iterations of an algorithm.
k	Index of the ij-th pixel such that $k = n(i - 1) + j$
m	Number of rows of a matrix or length of an image
n	Number of columns of a matrix or width of an image
h	Number of rows or columns of a matrix
δ	Nearest neighbor parameter
ρ	Patch size
U	$= [\boldsymbol{u}_1, ..., \boldsymbol{u}_n] \in \mathcal{R}^{m \times n}$
V	$= [\boldsymbol{v}_1, ..., \boldsymbol{v}_n] \in \mathcal{R}^{m \times n}$
$\tilde{V}$	$= [\boldsymbol{\nu}_1 | \ldots | \boldsymbol{\nu}_l | \ldots | \boldsymbol{\nu}_{n^2}]^T$
Λ	$= Diag(\lambda_1, \ldots, \lambda_l, \ldots, \lambda_{n^2})$
Σ	$= Diag(\sigma_1, ..., \sigma_n)$
$\tilde{U}_{\ell+1}$	$= [\boldsymbol{p}_1, ..., \boldsymbol{p}_{\ell+1}] \in \mathcal{R}^{m \times (\ell+1)}$, Matrix in ALB
$U_{\ell+1}$	$= [\boldsymbol{q}_1, ..., \boldsymbol{q}_{\ell+1}] \in \mathcal{R}^{n \times (\ell+1)}$, Matrix in ALB
$\hat{\tilde{U}}_{k+1}$	Predicted of $\tilde{U}_{\ell+1}$ for (k+1)-th singular triplets
$\hat{U}_{k+1}$	Predicted of $U_{\ell+1}$ for (k+1)-th singular triplets
$B_{\ell+1,\ell}$	$\in \mathbf{R}^{(\ell+1) \times \ell}$, Bidiagonal matrix in ALB
$\hat{B}_{k+1}$	Predicted of $B_{\ell+1}$ for (k+1)-th singular triplets
$U^\perp$	A basis for the orthogonal complement subspace of U.
$\tilde{\mathcal{U}}$	Denoised Image
$\mathcal{D}$	Geodesic distance matrix
$\mathcal{G}$	Gramian matrix
$\mathcal{I}$	Original image
I	Identity matrix
$\mathcal{U}$	Input image for the algorithm (often noisy)
Γ	Weights of Shepard's method
Q	Orthogonal matrix in randomized SVD
$\boldsymbol{u}_i$	i-th left singular vectors
$\boldsymbol{v}_i$	i-th right singular vectors
$\boldsymbol{\nu}_l$	l-th eigenvector
$\boldsymbol{w}_k$	Vectors in MGSA
$\boldsymbol{c}_k$	K columns which are randomly chosen in MGSA
$\boldsymbol{O}$	Eigenvectors corresponding to k largest eigenvalues of G in MCLA
$\tilde{\boldsymbol{u}}(\boldsymbol{x}_{ij})$	Denoised version of the patch $\boldsymbol{u}(\boldsymbol{x}_{ij})$
$\boldsymbol{x}_{ij}$	ij-th pixel of the image
$\boldsymbol{u}(\boldsymbol{x}_{ij})$	Patch centered at the point $\boldsymbol{x}_{ij}$
$\mathcal{K}_\ell(A, b)$	$= span\{\boldsymbol{b}, A\boldsymbol{b}, ..., A^{\ell-1}\boldsymbol{b}\}, \ell \geq 1$, Krylob subspace generated by A and $\boldsymbol{b}$
$G(V, E)$	Graph G with the vertex set V and edge set E
$\mathcal{N}(\boldsymbol{x}_{\tilde{k}})$	Neighborhood at the pixel $\tilde{k}$

Table 2: Abbreviations

Abbreviation	Description
GGD	Geodesic Gramian Denoising
MCLA	Monte Carlo Low-Rank Approximation
ALB	Augmented Lanczos Bidiagonalization
PIME	Preconditioned Iterative Multimethod Eigensolver
RSVD	Randomized Singular Value Decomposition
RE	Relative Error
PSNR	Peak Signal to Noise Ratio
SSIM	Structural Similarity Index Measure

few, denoted as L, prominent right singular vectors, defined as $\boldsymbol{\nu}_l$'s where $l = 1, \ldots, L$, of the Gramian matrix computed using Def. 2.2 with $A = \mathcal{G}$ and $n_1 = n_2 = n$ because prominent singular vectors only represent image features excluding image noise. We denote the denoised version of the noisy patch $\boldsymbol{u}_k$ as $\tilde{\boldsymbol{u}}_k$ that GGD produces by

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}_k) = \sum_{l=1}^{L} \langle \boldsymbol{u}(\boldsymbol{x}_k), \boldsymbol{\nu}_l \rangle \boldsymbol{\nu}_l, \tag{3}$$

where $\boldsymbol{\nu}_l$ represents l-th prominent singular vector of the Gramian matrix where $l = 1, \ldots, L$, and $\langle \cdot, \cdot \rangle$ denotes the inner product.

Definition 2.1. *Consider that $Diag\,(\lambda_1, \ldots, \lambda_n)$ represents a diagonal matrix formed with the vector $(\lambda_1, \ldots, \lambda_n)$ as its diagonal. Let $A \in \mathbb{R}^{n \times n}$ be a matrix and $U_{n \times n}$ be a unitary matrix such that $U^T U = I$. Then, ED of A is $A = U \Lambda U^T$, where $\Lambda_{n \times n} = Diag\,(\lambda_1, \ldots \lambda_n)$, [19].*

Definition 2.2. *Consider that $Diag\,\big(\sigma_1, \ldots, \sigma_{\min(m,n)}\big)$ represents a diagonal matrix formed with the vector $\big(\sigma_1, \ldots, \sigma_{\min(m,n)}\big)$ as its diagonal. Let $A \in \mathbb{R}^{m \times n}$ be a matrix, and $U_{m \times m}$ and $V_{n \times n}$ are unitary matrices such that $U^T U = I$ and $V^T V = I$, respectively. Then, the SVD of A is $A = U \Sigma V^T$, where $\Sigma_{m \times n} = Diag\,\big(\sigma_1, \ldots \sigma_{\min(m,n)}\big)$ with $\sigma_1 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$. Here, for $l = 1, \ldots, n$, the column vector $\boldsymbol{\nu}_l$ represents l-th right singular vector of A such that $V = [\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_l, \ldots, \boldsymbol{\nu}_n]$, [19].*

Fifth, in order to construct the denoised image from the denoised patches GGD estimates the intensity of each pixel of the image using all the other pixels at the same location with respect to the location of the image but each exists in one of the ρ^2 overlapping patches. GGD combines all these estimations using a moving least square approximation given by using Shepard's method [29], as

$$\tilde{\mathcal{U}}(\boldsymbol{x}_k) = \sum_{\boldsymbol{x}_t \in \mathcal{N}(\boldsymbol{x}_k)} \Gamma(\boldsymbol{x}_k, \boldsymbol{x}_t) [\tilde{\boldsymbol{u}}(\boldsymbol{x}_t)]_{t_n}, \tag{4}$$

where the neighborhood of the target pixel $\boldsymbol{x}_k$ is

$$\mathcal{N}(\boldsymbol{x}_k) = \{\boldsymbol{x}_t \mid \ \|\boldsymbol{x}_k - \boldsymbol{x}_t\|_\infty \leq \rho/2\}, \tag{5}$$

[26], and the weight is given by

$$\Gamma(\boldsymbol{x}_k, \boldsymbol{x}_t) = \frac{e^{-\|\boldsymbol{x}_k - \boldsymbol{x}_t\|^2}}{\sum_{\boldsymbol{x}_{t'} \in \mathcal{N}(\boldsymbol{x}_k)} e^{-\|\boldsymbol{x}_k - \boldsymbol{x}_{t'}\|^2}}. \tag{6}$$

Here, for each pixel $\boldsymbol{x}_t \in \mathcal{N}(\boldsymbol{x}_k)$, there exists a new index t_n such that the extrinsic pixel location (i, j) at that new index of the patch $\tilde{\boldsymbol{u}}(\boldsymbol{x}_t)$, denoted by $[\tilde{\boldsymbol{u}}(\boldsymbol{x}_t)]_{t_n}$, is the same as the extrinsic pixel location of $\boldsymbol{x}_k$. The weighting term in Eqn. (6) weights close by pixels with more weight while the faraway pixels with less weight. Thus, according to Eqn. (4), merging assures that the pixel $\boldsymbol{x}_k$ of the reconstructed image is highly influenced by the pixels at the same location of the nearby patches. The main steps of GGD are summarized in Alg. 1.

For an input image of size $n \times n$, step 4 of Alg. 1 requires to compute the ED [see the computation of $\{\boldsymbol{\nu}_l | l = 1, \ldots L\}$ of Eqn. (3)] of a Gramian matrix $\mathcal{G}$ of size $n^2 \times n^2$. In GGD [14], we employed regular SVD that calculates the entire singular vector spectrum of $\mathcal{G}$ using Def. 2.2 and then we chose L prominent singular vectors for denoising. Especially, the implementation of SVD is done by two processes: first, the given matrix of size $m \times n$ is reduced to a bidiagonal matrix form using Householder transformations with $\mathcal{O}(mn \min(m, n))$ operations, and then use the QR algorithm[1] in Ref. [9, 10], to compute the SVD of the bidiagonal matrix with $\mathcal{O}(n)$ operations [19]. Thus, the computational complexity of GGD is dominated by step 4 of Alg. 1 with $\mathcal{O}(n^6)$

---

[1]QR algorithm is a procedure to calculate the eigenvalues and singular vectors of a given matrix using a Schur decomposition.

Algorithm 1 Geodesic Gramian Denoising (GGD) [14].

Inputs: noisy image ($\mathcal{U}_{n \times n}$), patch length (ρ), nearest neighborhood size (δ), and singular vector threshold (L).

Outputs: noise-reduced image ($\tilde{\mathcal{U}}_{n \times n}$).

1: Construct n^2 overlapping square-shaped patches each with the length ρ from the noisy image $\mathcal{U}_{n \times n}$ and denote the patch set as $\{\boldsymbol{u}(\boldsymbol{x}_k)|k = 1, \ldots, n^2\}$.

2: Produce the graph structure $G(V, E)$ from the patch set using the nearest neighbor search algorithm in [1]. Use Floyd's algorithm in [8], to approximate the geodesic distances in the patch space and then produce the geodesic distance matrix $\mathcal{D}$.

3: Construct the Gramian matrix $\mathcal{G}$ from the geodesic distance matrix $\mathcal{D}$ using Eqn. (2).

4: Compute the right singular vectors $\{\boldsymbol{\nu}_l|l = 1, \ldots L\}$ corresponding to the L biggest eigenvalues of the Gramian matrix $\mathcal{G}$

5: Produce the noise-free patches $\{\tilde{\boldsymbol{u}}(\boldsymbol{x}_k)|k = 1, \ldots, n^2\}$ using the right singular vectors $\{\boldsymbol{\nu}_l|l = 1, \ldots L\}$ as explained by Eqn. (3).

6: Merge noise-free patches using Eqns. (4) and (6), and generate the denoise image $\tilde{\mathcal{U}}_{n \times n}$.

---

as it computed SVD of Gramian matrix of the size $n^2 \times n^2$ for an input image of $n \times n$. However, GGD always performs superior denoising with a significantly small number of eigenvectors. Thus, in the Secs. 2.2-2.5, we present four singular vector approximation schemes that we can use to approximate the required number of prominent SVs and then replace SVD of step 4 of Alg. 1 to increase the computational efficiency of GGD.

2.2. Monte Carlo Low-Rank Approximation

Monte Carlo Low-Rank Approximation (MCLA) [11], uses an iterative Monte Carlo sampling approach to approximate the SVD of a data matrix A of size $m \times n$ by computing the L-rank approximation, for some $L \in \mathbb{N}$ where $L < m, n$. Each iteration involves the reading of $\mathcal{O}(L)$ of columns or rows of the data matrix that eventually attains a computational time complexity of $\mathcal{O}(Lmn)$. Note that, since the Gramian matrix $\mathcal{G}$ of GGD is of size $n^2 \times n^2$ for a given image of size $n \times n$, the computational time complexity of MCLA generating L-prominent singular vectors is $\mathcal{O}(Ln^4)$. Instead of precisely computing the singular vectors of $\mathcal{G}$ as stated in Step 4 of Alg. 1, we approximate those singular vectors using MCLA and then perform the denoising.

We formulate MCLA for a general data matrix A of size $m \times n$, and then we replace this $A_{m \times n}$ by $\mathcal{G}_{n^2 \times n^2}$. First, we start the MCLA algorithm by computing the initial L-rank approximation, denoted as $B_0 \in \mathbb{R}^{m \times n}$, of $\mathcal{G}$. For that, let $\boldsymbol{c}_1^{(0)}, \ldots, \boldsymbol{c}_n^{(0)} \in \mathbb{R}^m$ be the n columns of A. Choose L integers $1 \leq n_1 < \ldots, n_L \leq n$ randomly and let $\boldsymbol{\nu}_1^{(0)}, \ldots, \boldsymbol{x}_L^{(0)} \in \mathbb{R}^m$ be the orthonormal set obtained from $\boldsymbol{c}_{n_1}^{(0)}, \ldots, \boldsymbol{c}_{n_L}^{(0)}$ using the modified Gram-Schmidt algorithm given in Alg. 5.2.6 of Ref. [20]. Then, set

$$B^{(0)} = \sum_{l=1}^{L} \boldsymbol{x}_l^{(0)} \left(A^T \boldsymbol{x}_l^{(0)} \right)^T. \tag{7}$$

Now, for $t = 0, \ldots, T$, we compute L-rank approximation of A, denoted as $B^{(t)} \in \mathbb{R}^{m \times n}$, which is improved from $B^{(t-1)} \in \mathbb{R}^{m \times n}$. Let $\boldsymbol{x}_1^{(t-1)}, \ldots, \boldsymbol{x}_L^{(t-1)} \in \mathbb{R}^m$ is the orthonormal set involved in the computation of $B^{(t-1)}$ such that $B^{(t-1)} = \sum_{l=1}^{L} \boldsymbol{x}_l^{(t-1)} \left(A^T \boldsymbol{x}_l^{(t-1)} \right)^T$. Now, we choose random L' columns of A where $1 \leq L' \leq n$ and denote them by $\boldsymbol{c}_1^{(t)}, \ldots, \boldsymbol{c}_{L'}^{(t)} \in \mathbb{R}^m$. Let, X be the subset spanned by $\boldsymbol{x}_1^{(t-1)}, \ldots, \boldsymbol{x}_L^{(t-1)}, \boldsymbol{c}_1^{(t)}, \ldots, \boldsymbol{c}_{L'}^{(t)}$. For $L < p \leq L + L'$, we generate orthonormal basis $\boldsymbol{w}_1^{(t)}, \ldots, \boldsymbol{w}_L^{(t)}, \ldots, \boldsymbol{w}_p^{(t)} \in \mathbb{R}^m$ of X obtained from $\boldsymbol{x}_1^{(t-1)}, \ldots, \boldsymbol{x}_L^{(t-1)}, \boldsymbol{c}_1^{(t)}, \ldots, \boldsymbol{c}_l^{(t)}$. For $L < p \leq L + L'$ using a modified Gram-Schmidt algorithm. Now, define a $p \times p$ non-negative definite matrix

$$C = \left(\left(A^T \boldsymbol{w}_i^{(t-1)} \right)^T \left(A^T \boldsymbol{w}_i^{(t-1)} \right) \right)_{i,j=1}^{p}. \tag{8}$$

We compute L-prominent eigenvectors of C that we denote as $\tilde{\boldsymbol{x}}_1^{(t)}, \ldots, \tilde{\boldsymbol{x}}_L^{(t)} \in \mathbb{R}^p$. We compute $\left\{ \boldsymbol{x}_l^{(t-1)} \in \mathbb{R}^m \mid l = 1, \ldots, L \right\}$ such that

$$\left(\boldsymbol{x}_1^{(t)}, \ldots, \boldsymbol{x}_L^{(t)} \right) = \left(\boldsymbol{w}_1^{(t)}, \ldots, \boldsymbol{w}_p^{(t)} \right) \left(\tilde{\boldsymbol{x}}_1^{(t)}, \ldots, \tilde{\boldsymbol{x}}_L^{(t)} \right) \tag{9}$$

Then,

$$B^{(t)} = \sum_{l=1}^{L} \boldsymbol{x}_l^{(t)} \left(A^T \boldsymbol{x}_l^{(t)} \right)^T, \tag{10}$$

5

is the improved rank L approximation of A from $B^{(t-1)}$. Now, while the approximated L prominent singular values of A are given by

$$\sigma_l = \sqrt{\left(A^T \boldsymbol{x}_i^{(t)}\right)^T \left(A^T \boldsymbol{x}_i^{(t)}\right)}, \tag{11}$$

corresponding right singular values, denoted by $\left\{\boldsymbol{\nu}_l^{(t)} \middle| l = 1, \ldots, L\right\}$, are given by

$$\boldsymbol{\nu}_l^{(t)} = \frac{1}{\sigma_l} A^T \boldsymbol{x}_l^{(t)}, \tag{12}$$

Algorithm 2 Monte Carlo Low-Rank Approximation (MCLA) [11].

Inputs: data matrix ($A \in \mathbb{R}^{m \times n}$), rank ($k$), number of iterations (T_I), tolerance (η).
Outputs: rank L approximation $B \in \mathbb{R}^{m \times n}$ of A, approximated L prominent singular values ($\{\sigma_l \mid l = 1, \ldots, L\}$), and their approximated right singular vectors ($\{\boldsymbol{\nu}_l \mid l = 1, \ldots, L\}$)

1: Compute L-rank approximation of A with L columns (or rows) of A using Eqn.(7).
2: **for** $t = 1$ to T_I **do**
3:    Select L' columns (or rows), $\boldsymbol{c}_1^{(t)}, \ldots, \boldsymbol{c}_{L'}^{(t)} \in \mathbb{R}^m$, from A at random. Consider $\left\{\boldsymbol{x}_l^{(t-1)} \in \mathbb{R}^m \middle| l = 1, \ldots, L\right\}$ is the orthonormal set involved in the computation of $B^{(t-1)}$.
4:    For $L \leq p \leq L + L'$, compute orthonormal basis $\boldsymbol{w}_1^{(t)}, \ldots, \boldsymbol{w}_L^{(t)}, \ldots, \boldsymbol{w}_p^{(t)}$ of L' columns and orthonormal set and compute C using the orthonormal basis as in Eqn. (8).
5:    Compute L-prominent eigenvectors $\left\{\tilde{\boldsymbol{x}}_l^{(t)} \in \mathbb{R}^p \middle| l = 1, \ldots, L\right\}$ of C in Eqn. (8). Convert these eigenvectors to $\left\{\boldsymbol{x}_l^{(t-1)} \in \mathbb{R}^m \middle| l = 1, \ldots, L\right\}$ using Eqn. (9)
6:    Compute $B^{(t)}$ using Eqn. (10).
7:    **if** $\|B^{(t-1)}\|/\|B^{(t)}\| > 1 - \eta$ **then**
8:        Goto 11.
9:    **end if**
10: **end for**
11: $B := B^{(t)}$;
    approximated singular values of A are $\left\{\sigma_l \middle| l = 1, \ldots, L\right\}$ where $\sigma_l := \sqrt{\left(A^T \boldsymbol{x}_i^{(t)}\right)^T \left(A^T \boldsymbol{x}_i^{(t)}\right)}$, and
    approximated singular vectors of A are $\left\{\boldsymbol{\nu}_l^{(t)} = \frac{1}{\sigma_l} A^T \boldsymbol{x}_l^{(t)} \in \mathbb{R}^m \middle| l = 1, \ldots, L\right\}$.

For a given image of size $n \times n$, we compute the Gramian matrix $\mathcal{G}_{n \times n}$ using Steps 1-3 in Alg. 1. Then, we approximate L prominent singular vectors using Alg. 2 with $A = \mathcal{G}_{n \times n}$. Finally, we use those singular vectors to compute the noise-free image as explained in Steps 5-6 of Alg. 1.

2.3. Augmented Lanczos Bidiagonalization

Augmented Lanczos Bidiagonalization (ALB) [4], is a framework for the computation of a few of the largest (or smallest) singular triplets of a large matrix, say A of order $n \times m$. ALB computes sequences of projections of A onto judiciously chosen low-dimensional Krylov subspaces. The method is implemented by augmentation of Krylov subspaces that are determined similarly as in the standard Lanczos bidiagonalization method where the augmentation is imposed by either Ritz vectors or harmonic Ritz vectors.

The first step of ALB is to approximate the data matrix $A \in \mathbb{R}^{m \times n}$ by a bidiagonal matrix, denoted as $B \in \mathbb{R}^{m \times n}$, using the Lanczos bidiagonalization process [4]. Bidiagonalization is a widely used kernel that transforms a full matrix into a bidiagonal form using orthogonal transformations. The bidiagonalization procedure is a preprocessing step that significantly lowers the cost of the implicit implementation of the SVD algorithm. For given data matrix A and the initial unit vector $\boldsymbol{p}^{(1)} \in \mathbb{R}^n$, this process yields the decomposition

$$A P^{(t)} = Q^{(t)} B^{(t)}, \tag{13}$$

$$A^T Q^{(t)} = P^{(t)} \left(B^{(t)}\right)^T + \boldsymbol{r}^{(t)} \left(\boldsymbol{e}^{(t)}\right)^T, \tag{14}$$

where $P^{(t)} \in \mathbb{R}^{n \times h}$, $Q^{(t)} \in \mathbb{R}^{m \times h}$, $\left(P^{(t)}\right)^T P^{(t)} = I^{(t)}$, $P^{(t)} \boldsymbol{e}^{(1)} = \boldsymbol{p}^{(1)}$, $\left(Q^{(t)}\right)^T Q^{(t)} = I^{(t)}$, $\boldsymbol{r}^{(t)} \in \mathbb{R}^n$, and $\left(P^{(t)}\right)^T \boldsymbol{r}^{(t)} = 0$. Lanczos bidiagonalization governing Eqns. (13) and (14) are numerically implemented in

Algorithm 3 Lanczos bidiagonalization [4].

Input: data matrix ($A \in \mathbb{R}^{m \times n}$), initial vector of unit length ($\boldsymbol{p}_1 \in \mathbb{R}^n$), and the number of bidiagonalization steps (T_B).

Output: upper bidiagonal matrix ($B \in \mathbb{R}^{h \times h}$) with entries α_j and β_j, matrices of orthonormal columns ($P = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n] \in \mathbb{R}^{n \times h}$ and $Q = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n] \in \mathbb{R}^{m \times h}$), and residual error ($\boldsymbol{r} \in \mathbb{R}^n$).

1: Initialization: $P_1 = \boldsymbol{p}_1$, $\boldsymbol{q}_1 = A\boldsymbol{p}_1$, $\alpha_1 = \|\boldsymbol{q}_1\|_2$, $\boldsymbol{q}_1 = \boldsymbol{q}_1/\alpha_1$, and $Q_1 = \boldsymbol{q}_1$
2: **for** $t = 1$ to T_B **do**
3:     $\boldsymbol{r}^{(t)} = A^T\boldsymbol{q}^{(t)} - \alpha^{(t)}\boldsymbol{p}^{(t)}$
4:     Reorthogonalization: $\boldsymbol{r}^{(t)} = \boldsymbol{r}^{(t)} - P^{(t)}\left(\left(P^{(t)}\right)^T \boldsymbol{r}^{(t)}\right)$
5:     **if** $t < T_B$ **then**
6:         $\beta^{(t)} = \|\boldsymbol{r}^{(t)}\|_2$; $\boldsymbol{p}^{(t+1)} = \boldsymbol{r}^{(t)}/\beta^{(t)}$, and $P^{(t+1)} = [P^{(t)}, \boldsymbol{p}^{(t+1)}]$
7:         $\boldsymbol{q}^{(t+1)} = A\boldsymbol{p}^{(t+1)} - \beta^{(t)}\boldsymbol{q}^{(t)}$
8:         Reorthogonalization: $\boldsymbol{q}^{(t+1)} = \boldsymbol{q}^{(t+1)} - Q^{(t)}\left(\left(Q^{(t)}\right)^T \boldsymbol{q}^{(t+1)}\right)$
9:         $\alpha^{(t+1)} = \|\boldsymbol{q}^{(t+1)}\|_2$; $\boldsymbol{q}^{(t+1)} = \boldsymbol{q}^{(t)}/\alpha^{(t+1)}$, and $Q^{(t+1)} = [Q^{(t)}, \boldsymbol{q}^{(t+1)}]$
10:     **end if**
11: **end for**
12: $B = B^{(t)}$, $P = P^{(t)}$, and $Q = Q^{(t)}$

Alg. 3. Alg. 3 is carried out in finite precision arithmetic; thus, the columns of P and Q are not necessarily orthogonal unless they are reorthogonalized.

Now, we perform SVD on this bidiagonal matrix ($B_{h \times h}$) according to Def. 2.2 and generate singular triplets $\{\sigma_l, \boldsymbol{u}_l, \boldsymbol{v}_l | l = 1, \ldots, h\}$ where $\sigma_1 \geq \cdots \geq \sigma_h \geq 0$. Entries of the singular triplet hold the relation

$$B\boldsymbol{v}_l = \sigma_l\boldsymbol{u}_l, \quad B^T\boldsymbol{u}_l = \sigma_l\boldsymbol{v}_l, \quad l = 1, \ldots, h. \tag{15}$$

We determine approximate singular triplets $\{\tilde{\sigma}_l, \tilde{\boldsymbol{u}}_l, \tilde{\boldsymbol{u}}_l\}$, $l = 1, \ldots, h$ of A from singular triplets of B by

$$\tilde{\sigma}_l = \sigma_l, \quad \tilde{\boldsymbol{u}}_l = Q\boldsymbol{u}_l, \quad \tilde{\boldsymbol{v}}_l = P\boldsymbol{v}_l \tag{16}$$

Combining Eqn. (15) with Eqns. (13) and (14) gives

$$A\tilde{\boldsymbol{v}}_l = \tilde{\sigma}_l\tilde{\boldsymbol{u}}_l, \quad A^T\tilde{\boldsymbol{u}}_l = \tilde{\sigma}_l\tilde{\boldsymbol{v}}_l + \boldsymbol{r}\boldsymbol{e}^T\boldsymbol{u}_j \tag{17}$$

Eqn. (17) suggests that the singular triplets $\{\tilde{\sigma}_l, \tilde{\boldsymbol{u}}_l, \tilde{\boldsymbol{v}}_l | l = 1, \ldots, h\}$ be qualified to be approximated singular triplets of A if $\boldsymbol{r}\boldsymbol{e}^T\boldsymbol{u}_j$ is sufficiently small. Especially, the ALM scheme accepts them as singular triplets if

$$\|\boldsymbol{r}\|_2|\boldsymbol{e}^T\boldsymbol{u}_l| \leq \delta\|A\|_2, \tag{18}$$

where $\delta \in \mathbb{R}^+$ and $\|A\|_2$ is approximated as the largest SV of B.

It is well known that the implicitly restarted Lanczos bidiagonalization methods can suffer from numerical instability due to propagated round-off errors that delays or prevent convergence of desired eigenvalues and eigenvectors [4]. To increase the numerical stability, we augment Krylov subspaces by certain Ritz vectors as explained in Ref. [30]. Consider that the approximated right singular vectors $\{\tilde{\boldsymbol{v}}_l, l = 1, \ldots, h\}$ of A in Eqn. (16) are Ritz vectors of $A^T A$ associated with Ritz values $\{\tilde{\sigma}_l^2, l = 1, \ldots, h\}$. Let the Ritz vectors $\{\tilde{\boldsymbol{v}}_l, l = 1, \ldots, L\}$ associated with L-largest Ritz values be available, assume that $\boldsymbol{r} \neq 0$, and introduce the matrix

$$\tilde{P}_{L+1} = \begin{bmatrix} \tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_L, \boldsymbol{p}_{h+1} \end{bmatrix} \tag{19}$$

Let $\tilde{\boldsymbol{r}}_L$ be the remainder orthogonal to the vectors $\{\boldsymbol{u}_l, l = 1, \ldots, h\}$ defined in Ref. [4] and introduce the matrices

$$\tilde{Q}_{L+1} = \left[\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_L, \frac{\tilde{\boldsymbol{r}}_L}{\|\tilde{\boldsymbol{r}}_L\|} \right] \tag{20}$$

and

$$\tilde{B}_{L+1} = \begin{bmatrix} \tilde{\sigma}_1 & 0 & \tilde{\rho}_1 \\ & \tilde{\sigma}_L & \tilde{\rho}_L \\ 0 & & \tilde{\alpha}_{L+1} \end{bmatrix} \in \mathbb{R}^{(L+1) \times (L+1)}, \tag{21}$$

where $\tilde{\rho}_l = (\tilde{\boldsymbol{u}}_l)^T A\boldsymbol{p}_{h+1}$, $l = 1, \ldots, h$. Then, an analogous representation of Eqns. (13) and (14) using Eqns. (19), (20), and (21) are

$$A\tilde{P}_{L+1} = \tilde{Q}_{L+1}\tilde{B}_{L+1}, \tag{22}$$

and

$$A^T \tilde{Q}_{L+1} = \tilde{P}_{L+1} \left(\tilde{B}_{L+1} \right)^T + \boldsymbol{r}_{L+1} \left(\boldsymbol{e}_{L+1} \right)^T, \tag{23}$$

respectively.

Augmenting by Ritz vectors as described above gives good approximations to the largest singular triplets of A. However, Ref. [24] observed that when seeking to compute the smallest singular triplets of A, augmenting by harmonic Ritz values can give faster convergence than augmenting by Ritz values. Let L' smallest singular triplets of $B_{h \times h+1}$ determine the matrices

$$\begin{aligned} U' &= [\boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_{L'}] \in \mathbb{R}^{h \times L'} \\ V' &= [\boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_{L'}] \in \mathbb{R}^{(h+1) \times L'} \\ \Sigma' &= Diag[\sigma'_1, \ldots, \sigma'_{L'}] \in \mathbb{R}^{L' \times L'} \end{aligned} \tag{24}$$

We now derive the relations analogous to Eqns. (13) and (14) for harmonic Ritz vectors. We introduce QR factorization

$$\begin{bmatrix} B_h^{-1} U'_L \Sigma'_L & -\beta_h B_h^{-1} \boldsymbol{e}_h \\ 0 & 1 \end{bmatrix} = Q'_{L+1} R'_{L+1}, \tag{25}$$

where $Q'_{L+1} \in \mathbb{R}^{(h+1) \times (L+1)}$ has orthonormal columns and $R'_{L+1} \in \mathbb{R}^{(L+1) \times (L+1)}$ is upper triangular. We denote the matrices

$$\hat{P}_{L+1} = \left[\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_{L+1} \right] = P_{h+1} Q'_{L+1} \tag{26}$$

$$\hat{Q}_{L+1} = \left[\hat{Q}_L, \hat{\boldsymbol{q}}_{L+1} \right] \in \mathbb{R}^{m \times (L+1)}, \quad \text{with} \quad \hat{Q}_L = Q_h U'_L, \tag{27}$$

and

$$\tilde{B}_{L+1} = \begin{bmatrix} \sigma'_1 & 0 & \tilde{\gamma}_1 \\ & \sigma'_L & \hat{\gamma}_L \\ 0 & & \hat{\alpha}_{L+1} \end{bmatrix} \left(R'_{L+1} \right)^{-1} \in \mathbb{R}^{(L+1) \times (L+1)}, \tag{28}$$

where $\hat{\gamma}_l = -\beta_h (\hat{\boldsymbol{q}}_l)^T \hat{\boldsymbol{q}}_h + (\hat{\boldsymbol{q}}_l)^T A \boldsymbol{p}_{h+1}, l = 1, \ldots, L$.

Then, an analogous representation of Eqns. (13) and (14) using Eqns. (26), (27), and (28) are

$$A \hat{P}_{L+1} = \hat{Q}_{L+1} \hat{B}_{L+1}, \tag{29}$$

and

$$A^T \hat{Q}_{L+1} = \hat{P}_{L+1} \left(\hat{B}_{L+1} \right)^T + \boldsymbol{r}_{L+1} \left(\boldsymbol{e}_{L+1} \right)^T. \tag{30}$$

respectively.

For a given image of size $n \times n$, we compute the Gramian matrix $\mathcal{G}_{n \times n}$ using Steps 1-3 in Alg. 1. Then, we approximate L prominent singular vectors using Alg. 4 with $A = \mathcal{G}_{n \times n}$. Finally, we use those singular vectors to compute the noise-free image as explained in Steps 5-6 of Alg. 1.

2.4. Preconditioned Iterative Multimethod Eigensolver

The singular vector approximation scheme that we are utilizing here is named Preconditioned Iterative Multimethod Eigensolver (PIME) [35], which is an extension of the state-of-the-art package PRIMME [31]. This state-of-the-art SVD package enables practitioners to solve a variety of large and sparse singular value problems with or without preconditioning that assures unprecedented efficiency, robustness, and accuracy, for both smallest and largest singular triplets.

PIME inputs a data matrix, denoted as $A \in \mathbb{R}^{m \times n}$, and two parameters, namely, the number of desired singular triplets, denoted as T_I, and tolerance, denoted as δ, which then outputs approximated L-many singular triplets. PIME relies on eigensolvers that work on the equivalent eigenvalue formulations on C, where

$$C = A^T A \in \mathbb{R}^{n \times n} \quad (\text{or} \quad C = A A^T \in \mathbb{R}^{m \times m}), \tag{31}$$

and on B, where

$$B = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \mathbb{R}^{(m+n) \times (m+n)}. \tag{32}$$

Algorithm 4 Augmented Lanczos Bidiagonalization (ALB).

Input: data matrix ($A \in \mathbb{R}^{m \times n}$), initial vector of unit length ($\boldsymbol{p}_1 \in \mathbb{R}^n$), number of bidiagonalization steps (T_B), number of desired singular triplets (L), tolerance (δ), machine epsilon (ϵ), and Boolean variable harmonic (h).

Output: approximated singular triplets $\left(\{\sigma_l, \boldsymbol{u}_l, \boldsymbol{v}_l\}_{l=1}^{L} \right)$.

1: Produce the bidiagonal matrix B by running the Lanczos bidiagonalization Algorithm, e.i., Alg. 3, on A.
2: Compute the SVD of B using Def. 2.2 and generate the singular triplets $\{\sigma_l, \boldsymbol{u}_l, \boldsymbol{v}_l | l = 1, \ldots, h\}$ where
   $\sigma_1 \geq \cdots \geq \sigma_m \geq 0$.
3: Check convergence:
   if all $l = 1, \ldots, L$ desired singular triplets satisfy the inequality in Eqn. (18) **then**
      Exit
   end if
4: Compute the augmenting vectors:
   if not harmonic or $\sigma_h / \sigma_1 > \epsilon^{-1/2}$ **then**
      Determine matrices $P = \tilde{P}_{L+1}$, $Q = \tilde{Q}_{L+1}$, and $B = \tilde{B}_{L+1}$ given in Eqns. 19, (20), and 21, respectively.
   end if
   if harmonic and $\sigma_h / \sigma_1 \leq \epsilon^{-1/2}$ **then**
      Compute the partial SVD of $B_{h \times h+1}$ using Eqn. (24) and QR-factorization using Eqn. (25). Determine matrices $P = \hat{P}_{L+1}$, $Q = \hat{Q}_{L+1}$, and $B = \hat{B}_{L+1}$ given in Eqns. (26), (27), and (28), respectively. **end if**
5: The available matrices P, Q, B, and the vector r satisfy

$$AP = QB \quad ; \quad A^T Q = PB^T + \boldsymbol{r}\boldsymbol{e}_L^T.$$

   Append $h - L$ columns to the matrices P, Q, and $h - L$ rows and columns to the matrix B. Denote the matrices so obtained by P_h, Q_h, and B_h, respectively. Determine a new residual vector and denote it by $\boldsymbol{r}_h$.
6: Goto 4.

PIME starts on C because without preconditioning the convergence in terms of iterations it is much faster than that on B where the cost per iteration on C is up to two times cheaper than that on B (because of dimension n versus $n + m$). PIME refers to the computations on C as the first stage of the method and switches to a second stage that it works on B if further accuracy is required.

First, PIME computes eigenpairs of C in Eqn. (31) using Def. 2.1. Let, $\left(\lambda_i^C, \boldsymbol{x}_i^C \in \mathbb{R}^n \right), i = 1, \ldots, n$ be eigenpairs of C computed using Def. 2.1. Then, Rayleigh-Ritz scheme is used to approximate the eigenpairs, denoted as $\left(\tilde{\lambda}_i^C, \tilde{\boldsymbol{x}}_i^C \right), i = 1, \ldots, n$, on the vector basis of $\left[\boldsymbol{x}_1^C, \ldots, \boldsymbol{x}_n^C \right]$. Set, the singular triplets $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i), i = 1, \ldots, n$ such that $\tilde{\sigma}_i = \sqrt{|\tilde{\lambda}_i^C|}$, $\tilde{\boldsymbol{v}}_i = \tilde{\boldsymbol{x}}_i^C$, and $\tilde{\boldsymbol{u}}_i = A\tilde{\boldsymbol{x}}_i^C \tilde{\sigma}_i$. The convergence criterion is set PIME to stop when the residual norm of the singular value triplets of A is less than $\|A\|_2 \delta$, i.e.,

$$\|\tilde{\boldsymbol{r}}_i\| = \sqrt{\|A\tilde{\boldsymbol{v}}_i - \tilde{\sigma}_i \tilde{\boldsymbol{u}}_i\|_2 + \|A\tilde{\boldsymbol{u}}_i - \tilde{\sigma}_i \tilde{\boldsymbol{v}}_i\|_2} < \|A\|_2 \delta \tag{33}$$

Eqn. (33) can be transformed into a convergence criterion for the eigensolver on C as

$$\|\tilde{\boldsymbol{r}}_i^C\| = \|C\tilde{\boldsymbol{x}}_i^C - \tilde{\lambda}_i^C \tilde{\boldsymbol{x}}_i\|_2 < \sqrt{|\tilde{\lambda}_i^C|} \|C\|_2 \delta \tag{34}$$

This eigensolver returns when all requested triplets satisfy the convergence criterion. However, the eigensolver may reach its maximum achievable accuracy before the residual norm reduces below the above convergence tolerance. If the tolerance is set below this limit the eigensolver may stagnate whereas if the limit is overestimated, then the number of iterations of the second stage will increase, making the whole solver more expensive.

In the second stage, PIME seeks eigenpairs of the augmented matrix $B \in \mathbb{R}^{(m+n) \times (m+n)}$ in Eqn. (39). If $U^{\perp} \in \mathbb{R}^{m \times (m-n)}$ is a basis for the orthogonal complement subspace of $U \in \mathbb{R}^{m \times n}$, we define the orthonormal matrix

$$X = \frac{1}{\sqrt{2}} \begin{bmatrix} V & -V & 0 \\ U & U & \sqrt{2}U^{\perp} \end{bmatrix} \in \mathbb{R}^{(m+n) \times (m+n)}, \tag{35}$$

where $V \in \mathbb{R}^{n \times n}$. The vectors $\frac{1}{\sqrt{2}}[\tilde{\boldsymbol{v}}_i; \tilde{\boldsymbol{u}}_i]$, for $i = 1, \ldots, L$, are set as initial guesses of the eigensolver as follows from Eqn. (35) and compute ED of B as

$$BX = X\, Diag(\lambda_1, ..., \lambda_n, -\lambda_1, ..., -\lambda_n, 0, ..., 0), \tag{36}$$

where 0 is repeated $(m - n)$-times in $Diag$. This approach can compute all eigenpairs accurately such that the residual norm is close to $\|A\|_2 \epsilon$. Let, $\left(\tilde{\lambda}_i^B, \tilde{\boldsymbol{x}}_i^B \in \mathbb{R}^{m+n} \right), i = 1, \ldots, (m+n)$, be eigenpairs of B approximated by

applying Rayleigh-Ritz scheme on the vector basis of $\left[\boldsymbol{x}_1^B, \ldots, \boldsymbol{x}_{(m+n)}^B\right]$. Set, the singular triplets $(\tilde{\sigma}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i)$, $i = 1, \ldots, (m+n)$ such that $\tilde{\sigma}_i = |\tilde{\lambda}_i^B|$, $\tilde{\boldsymbol{v}}_i = \tilde{\boldsymbol{x}}_i^B(1:n)$, and $\tilde{\boldsymbol{u}}_i = \tilde{\boldsymbol{x}}_i^B(n+1:m+n)$. The convergence criterion is set PIME to stop when the residual norm of the singular value triplets of B is less than $\sqrt{2}\|B\|_2\delta$, i.e.,

$$\|\tilde{\boldsymbol{r}}_i^B\| = \|B\tilde{\boldsymbol{x}}_i^B - \tilde{\lambda}_i^B\tilde{\boldsymbol{x}}_i\|_2 \approx \sqrt{2}\|\tilde{\boldsymbol{r}}_i\| < \sqrt{2}\|B\|_2\delta, \tag{37}$$

based on the assumption that $\|\tilde{\boldsymbol{x}}_i^B(1:n)\|_2 \approx \|\tilde{\boldsymbol{x}}_i^B(n+1:m+n)\|_2$.

Algorithm 5 Preconditioned Iterative Multimethod Eigensolver (PIME)
Input: data matrix ($A \in \mathbb{R}^{m \times n}$), number of desired singular triplets (L), tolerance (δ)
Output: approximated singular triplets $\{\tilde{\sigma}_l, \tilde{\mathbf{u}}_l, \tilde{\mathbf{v}}_l,\}, l = 1, ..., L$.

First stage: working on C:
1: Set the convergence criterion according to Eqn. (33), i.e., the residual norm of the singular value triplets of A is less than $\|A\|_2\delta$.
2: Compute eigenpairs $\left(\lambda_l^C, \boldsymbol{x}_l^C\right), l = 1, \ldots, L$ using ED in Def. 2.1 seeking L largest eigenvalues of $C = A^T A$ (or $C = AA^T$)
3: Perform Rayleigh-Ritz on the returned vector basis $\left[\boldsymbol{x}_1^C, \ldots, \boldsymbol{x}_n^C\right]$ and approximate eigenpairs $\left(\tilde{\lambda}_l^C, \tilde{\boldsymbol{x}}_l^C\right), l = 1, \ldots, L$
4: Set $\tilde{\sigma}_l = |\tilde{\lambda}_l^C|^{\frac{1}{2}}, \tilde{\mathbf{v}}_l = \tilde{\mathbf{x}}_l^C, \tilde{\mathbf{u}}_l = \mathcal{G}\tilde{\mathbf{v}}_l\tilde{\sigma_l}^{-1}$
5: **if** all the singular triplets of C converged, i.e., satisfy Eqn. (34) **then**
6:    Return $\{\tilde{\sigma}_l, \tilde{\mathbf{u}}_l, \tilde{\mathbf{v}}_l,\}, l = 1, ..., L$
7: **end if**

Second stage: working on B:
8: Set initial guesses as $\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{u}}_l \\ \tilde{\mathbf{v}}_l,\end{bmatrix} l = 1, ..., L$
9: Set the convergence criterion according to Eqn.(37), i.e., the residual norm of the singular value triplets of B is less than $\sqrt{2}\|B\|_2\delta$.
10: Approximate the eigenpairs $\left(\tilde{\lambda}_l^B, \tilde{\boldsymbol{x}}_l^B\right), l = 1, \ldots, L$ of B given in Eqn. (39) by applying the Rayleigh-Ritz scheme on the vector basis of X in Eqn. (35).
11: Set $\tilde{\sigma}_l = |\tilde{\lambda}_l^B|, \tilde{\boldsymbol{v}}_l = \tilde{\boldsymbol{x}}_l^B(1:n), \tilde{\boldsymbol{u}}_l = \tilde{\boldsymbol{x}}_l^B(n+1:m+n)$.
12: Return $\{\tilde{\sigma}_l, \tilde{\mathbf{u}}_l, \tilde{\mathbf{v}}_l\}, l = 1, ..., L$

2.5. Randomized Singular Value Decomposition

Randomized Singular Value Decomposition (RSVD) [22], combines probability theory with numerical linear algebra to develop an efficient, unbiased, and randomized algorithm to approximate SVD. RSVD offers an efficient construction of L number of orthonormal vectors that nearly span the range of given nonsquare data matrix $A \in \mathbb{R}^{m \times n}$ so that such vectors enable an efficient approximation for SVD of A. RSVD undergoes two stages computational process: First stage) computation of an approximate orthonormal basis, denoted as Q, for the range of A; and Second stage) approximating SVD of A using such orthonormal basis Q. Randomness only occurs in Step 1 and Step 2 is deterministic for a given Q.

- **First stage** Here, the goal is to produce an orthonormal matrix Q with as few columns as possible such that

$$A \approx QQ^*A, \tag{38}$$

where Q^* denotes the adjoint of Q. Draw L Gaussian random columns $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_L$ from $A_{m \times n}$. Then, project them sing the linear map A such that $Y = A\Omega$ where $\Omega = [\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_L]$. Finally, find the orthonormal matrix Q by using QR factorization of Y, i.e., $Y = QR$. While having as few columns as possible in the basis matrix Q increases efficiency, having more columns in it increases the accuracy of the approximation.

- **Second stage** Since the number L of columns of Q is substantially less than both dimensions of Q, we can efficiently compute

$$B = Q^*A, \tag{39}$$

10

where B has only L-many rows. We can then efficiently calculate the SVD of B as

$$B = W\Sigma V^* \tag{40}$$

where the columns of both W and V are orthonormal, and Σ is a diagonal $L \times L$ matrix whose entries are all non-negative. Let,

$$U = QW \tag{41}$$

and combining Eqns. (38)–(41) yields SVD of A such that

$$A \approx QQ^*A = Q(B) = QW\Sigma V^* = U\Sigma V^*. \tag{42}$$

If L is substantially less than both dimensions of A, then A has far more entries than any other matrix in the above calculations.

Algorithm 6 Randomized Singular Value Decomposition (RSVD) [22].
Inputs: data matrix ($A \in \mathbb{R}^{m \times n}$) and the number of desired singular triplets (L).
Outputs: approximated singular triplets $\{\tilde{\sigma}_l, \tilde{\mathbf{u}}_l, \tilde{\mathbf{v}}_l, \}, l = 1, ..., L$.

First stage:
1: Generate an $n \times L$ Gaussian matrix Ω by drawing columns of A randomly.
2: Produce $Y = A\Omega$.
3: Generate an orthonormal matrix Q using QR factorization, i.e., $Y = QR$.
4: Construct a matrix Q whose columns form an orthonormal basis for the range of Y.

Second stage:
5: Form $B = Q^*A$
6: Compute an SVD of the small matrix $B = W\Sigma V^*$
7: Set $U = QW$
8: Return $\{\tilde{\sigma}_i, \tilde{\mathbf{u}}_i, \tilde{\mathbf{v}}_i\}, i = 1, ..., L$, where $\tilde{\sigma}_i$, $\tilde{\mathbf{u}}_i$, and $\tilde{\mathbf{v}}_i$ are the i-th diagonal entry of Σ, i-th column of U, and i-th column of V^*, respectively.

---

3. Performance Analysis

Here, we analyze the performance of four hybrid GGD frameworks, made by integrating four SVD approximation techniques, namely, MCLA, ALB, PIME, and RSVD, in terms of accuracy and computational time by implementing them on image denoising tasks. The scope of our experimental procedure is applying GGD and its four hybrid variations to three test images, namely Barbara, Cameraman, and Mandrill, that are contaminated with three relative noise levels. The accuracy of the noise filtration is assessed by three performance metrics, relative error (RE), peak signal-to-noise ratio (PSNR), and structural similarity index measure (SSIM). Thus, this section includes three subsections, image preprocessing, performance metrics, and examples.

3.1. Image preprocessing

Gaussian noise is one of the most famous types of noise present in images taken by cameras; thus, we contaminate our test images with Gaussian noise and then attempt to filter it out using hybrid GGD frameworks presented in this paper. The probability density function (PDF) of Gaussian noise is given by

$$P_g(z, \mu, \gamma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}}, \tag{43}$$

where z is gray-level, μ is the mean of the distribution, σ is the standard deviation of the distribution.

Consider that we are given a noise-free image of size $n \times n$, denoted as $\mathcal{I}_{n \times n}$. We draw a noise sample of size $n \times n$, denoted as $\mathcal{N}_{n \times n}$, from the Gaussian PDF given in Eqn. (43) with $\mu = 0$ and some σ. Noisy image, denoted as $\mathcal{U}_{n \times n}$, is generated by additive rule as

$$\mathcal{U}_{n \times n} = \mathcal{I}_{n \times n} + \mathcal{N}_{n \times n}. \tag{44}$$

Since our analysis is based on gray images, we transform the colored images into gray images by taking the average across the three color channels. We compute the relative percentage noise, denoted as ζ, by

$$\zeta = \frac{\|\mathcal{U} - \mathcal{I}\|_2}{\|\mathcal{I}\|_2} 100\%. \tag{45}$$

The original noise-free image $\mathcal{I}_{100\times100}$ is imposed with diverse levels of noise intensities by changing σ of the PDF in Eqn. (43) to create three noisy images with $\zeta = 30\%$, 40%, and 50%. Since $\mathcal{U}$'s represent images, each entry of these matrices should be between 0–255 even after imposing the noise. Thus, we adjust $\mathcal{U}$ by replacing the values less than zero with zeros and the values more than 255 with 255s.

3.2. Performance metrics

The performance of the four hybrid GGD frameworks is assessed by three famous performance metrics RE, PSNR, and SSIM which we provide in Def. 3.1, Def. 3.2, and Def. 3.3, respectively. While RE and PSNR measure two aspects of the quality of the image reconstruction, RE and PSNR are related since the term $\|\mathcal{I} - \tilde{\mathcal{U}}\|_2$ in RE is related to RMSE in PSNR by $\|\mathcal{I} - \tilde{\mathcal{U}}\|_2 = n \times RMSE$. Thus, RE becomes a small value and PSNR becomes a large value in the case of $\mathcal{I} \approx \mathcal{U}$ that guarantees better denoising.

Definition 3.1. *Let, the two-dimensional matrix $\mathcal{I}$ represents a reference image of size $n \times n$ and $\tilde{\mathcal{U}}$ represents any other image of interest. The relative error, abbreviated as RE, of the image $\tilde{\mathcal{U}}$ with respect to the reference image $\mathcal{I}$ is defined as*

$$RE(\mathcal{I}, \tilde{\mathcal{U}}) = \frac{\|\mathcal{I} - \tilde{\mathcal{U}}\|_2}{\|\mathcal{I}\|_2}. \tag{46}$$

Here, $\max(\mathcal{I})$ represents the maximum possible pixel value of the image $\mathcal{I}$. RE ranges between zero and one such that zero ensures pure similarity between $\mathcal{I}$ and $\hat{\mathcal{U}}$ while 1 ensures perfect dissimilarity.

Definition 3.2. *Let, the two-dimensional matrix $\mathcal{I}$ represents a reference image of size $n \times n$ and $\tilde{\mathcal{U}}$ represents any other image of interest. Peak signal to noise ratio [23], abbreviated as PSNR, of the image $\tilde{\mathcal{U}}$ with respect to the reference image $\mathcal{I}$ is defined as*

$$PSNR(\mathcal{I}, \tilde{\mathcal{U}}) = 20 \log_{10} \left(\frac{\max(\mathcal{I})}{RMSE(\mathcal{I}, \tilde{\mathcal{U}})} \right). \tag{47}$$

Here, $\max(\mathcal{I})$ represents the maximum possible pixel value of the image $\mathcal{I}$. Since the pixels in our images of interest are represented in 8-bit digits, $\max(\mathcal{I})$ is 255. PSNR ranges between zero and ∞ such that zero ensures pure dissimilarity between $\mathcal{I}$ and $\hat{\mathcal{U}}$ while ∞ ensures perfect similarity.

Definition 3.3. *Let, a two-dimensional matrix $\mathcal{I}$ represents a reference image of size $n \times n$ and $\tilde{\mathcal{U}}$ represents an image of interest. Structural similarity index measure [34], abbreviated as SSIM, of the image $\tilde{\mathcal{U}}$ with respect to the reference image $\mathcal{I}$ is defined as the product of luminance distortion (I), contrast distortion (C), and loss of correlation (S), such as*

$$SSIM(\mathcal{I}, \tilde{\mathcal{U}}) = I(\mathcal{I}, \tilde{\mathcal{U}}) C(\mathcal{I}, \tilde{\mathcal{U}}) S(\mathcal{I}, \tilde{\mathcal{U}}), \tag{48}$$

where

$$
\begin{aligned}
I(\mathcal{I}, \tilde{\mathcal{U}}) &= \frac{2\mu_{\mathcal{I}}\mu_{\tilde{\mathcal{U}}} + c_1}{\mu_{\mathcal{I}}^2 + \mu_{\tilde{\mathcal{U}}}^2 + c_1}, \\
C(\mathcal{I}, \tilde{\mathcal{U}}) &= \frac{2\sigma_{\mathcal{I}}\sigma_{\tilde{\mathcal{U}}} + c_2}{\sigma_{\mathcal{I}}^2 + \sigma_{\tilde{\mathcal{U}}}^2 + c_2}, \\
S(\mathcal{I}, \tilde{\mathcal{U}}) &= \frac{\sigma_{\mathcal{I}\tilde{\mathcal{U}}} + c_3}{\sigma_{\mathcal{I}}\sigma_{\tilde{\mathcal{U}}} + c_3}.
\end{aligned}
\tag{49}
$$

Here, $\mu_{\mathcal{I}}$ and $\mu_{\tilde{\mathcal{U}}}$ are means of $\mathcal{I}$ and $\tilde{\mathcal{U}}$, respectively; $\sigma_{\mathcal{I}}$ and $\sigma_{\tilde{\mathcal{U}}}$ are standard deviations of $\mathcal{I}$ and $\tilde{\mathcal{U}}$, respectively; and $\sigma_{\mathcal{I}\tilde{\mathcal{U}}}$ is the covariance between $\mathcal{I}$ and $\tilde{\mathcal{U}}$. Moreover, c_1, c_2, and c_3 are very small positive constants to avoid the case of division by zero. SSIM ranges between -1 and 1 such that -1 indicates pure dissimilarity while 1 indicates perfect similarity.

3.3. Examples

Here, we implement the four hybrid GGD frameworks to denoise noisy versions of three benchmark computer vision test images, Barbara, cameraman, and mandrill available in Ref. [28]. In order to assess the computational time of GGD and its four hybrid versions, we implement these five frameworks on images of variable sizes on the same computer with the configuration of 11th Generation Intel Core i7-1165G7, 4 cores each with 4.70 GHz, 8 GB DDR4 3200MHz RAM. Here, we work on square-shaped images of size $n \times n$ for simplicity. First, we create six versions of the image Barbara with the lengths $n = 50, 60, 70, 80, 90, 100$ and impose an additive Gaussian noise sample with relative percentage noise of 30%. Then, we execute GGD and the other four of its hybrid frameworks over the images of $n = 50, 60, 70, 80, 90$, and 100 with some arbitrary parameter combination of $(\delta, \rho, L) = (10, 5, 15)$, and compute the computational time for denoising. Fig. 1 showing the computational time, in seconds (s), with respect to image length (n) justifies that the hybrid versions of GGD that are made
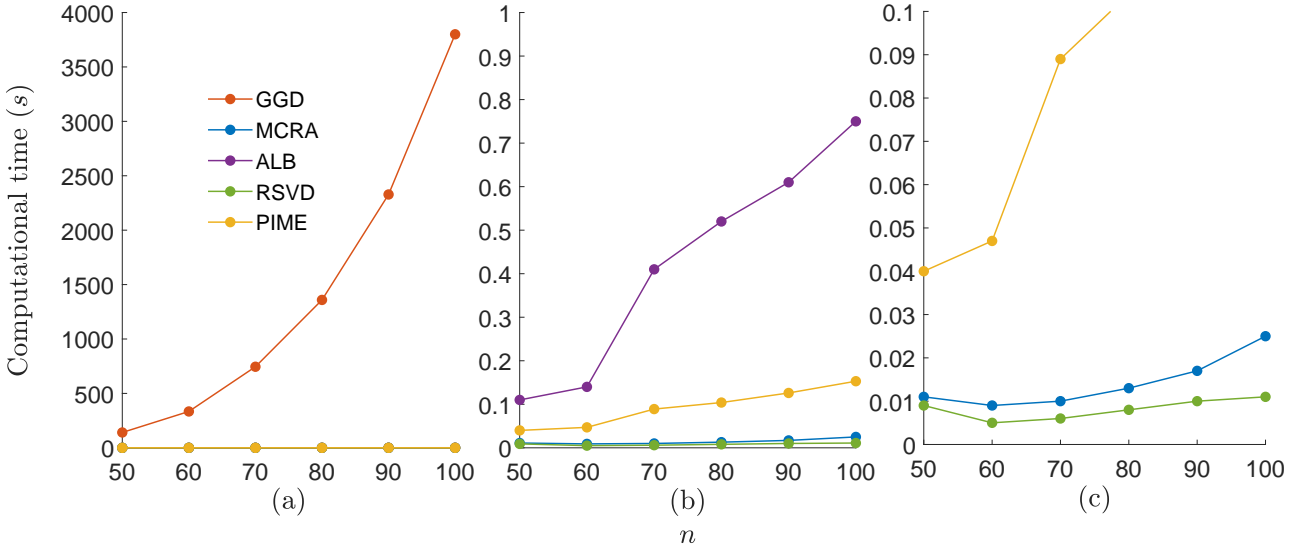
Figure 1: Computational time, in seconds (s), of GGD and its four hybrid versions with respect to the length of one side of a square-shaped image, i.e, n for an image of $n \times n$. First, we create six versions of the image Barbara with the sizes $n = 50, 60, 70, 80, 90$, and 100, and impose an additive Gaussian noise sample with relative percentage noise of 30%. We executed all five frameworks over these test images on a computer with the configuration of 11th Generation Intel Core i7-1165G7, 4 cores each with 4.70 GHz, 8 GB DDR4 3200MHz RAM. (a) Computational time attained by each framework of denoising six test images where the plots of MCRA, ALB, RSVD, and PIME are overlapped. While (b) shows the first cropped version of (a) which limits the vertical axis at $1s$, (c) shows the second cropped version of (a) which limits the verticle axis at $0.1s$.

by incorporating SVD approximation techniques into GGD attain significantly less computational time. We observe that the order from the smallest to the largest computational cost of the hybrid frameworks is RSVD, MCRA, PIME, and ALB.

Now, we asses the denoising performance of GGD and four of its hybrid frameworks by implementing them on three test images Barbara, cameraman, and mandrill. Our parameter space is $\delta = 8, 10, 12$, $\rho = 3, 5, 7$, and $L = 15, 20, 25$ for $\zeta =$20%; $\delta = 10, 12, 14$, $\rho = 5, 7, 9$, and $L = 15, 20, 25$ for $\zeta =$30%; and $\delta = 12, 14, 16$, $\rho = 7, 9, 11$, and $L = 15, 20, 25$ for $\zeta =$40%. First, we impose three noise levels, $\zeta = 20\%, 30\%$, and 40% into each of the three test images. Then, for each test image and the noise level, we implement GGD and the four of its hybrid frameworks with all the parameter combinations, i.e., δ's, ρ's, and ζ's. We compute the similarity between the noise-free original test image and its denoised version using the three performance metrics RE, PSNR, and SSIM. For each test image and noise level, we choose the best-denoised version with respect to each PSNR and SSIM. Table 3 presents both RE and PSNR along with optimized parameter values with respect to PSNR, as they are related so that the best denoising is predicated by small RE and big PSNR. Fig. 2 shows the images denoised by each method with their optimized parameter values. We observe that RSVD attains the best denoising performance with respect to PSNR while PIME attains the second best. The performance of GGD, MCLA, and ALB are mostly similar. The performance, assessed using PSNR, of all five methods decreases as the noise contamination increases. Table 4 presents SSIM and optimized parameter values with respect to SSIM. Fig. 3 shows the images denoised by each method with their optimized parameter values. We observe in Table 4 that while PIME attains the best denoising performance with respect to SSIM, all the other frameworks' performances are mostly similar. The structural similarity between the true image and the denoised image decreases for all five methods decreases as the noise contamination increases.

4. Conclusion

Filtering out noise from an image while preserving image features is an arduous task as noise blends into features. Our previous work presented in Ref. [14], named GGD, presented an image denoising framework that is capable of producing accurate denoising of a noisy image. However, the implementation of GGD for denoising of big images is inefficient as it requires computing singular vectors through conventional SVD which gets highly expensive when the image gets bigger. In this paper, we utilized four diverse singular vector approximation techniques to replace the conventional SVD computation step. The performance analysis is based on three test images each imposed with three nose levels. The performance of GDD and each of its hybrid frameworks are compared using visual perception and three similarity metrics, namely, RE, PSNR, and SSIM.

The conventional SVD encounters a computational complexity of $\mathcal{O}(mn \min(m, n))$ for a matrix of order $m \times n$. Denoising an image of size $n \times n$ requires GGD to compute some, say L, prominent singular values of patches' geodesic distance matrix of size $n^2 \times n^2$ using SVD that encounters computational complexity of $\mathcal{O}(n^6)$, which makes the complexity of GGD to be $\mathcal{O}(n^6)$. The approximation methods that we incorporated were

13

Table 3: PSNR and RE for the denoising of GGD and its four hybrid frameworks, MCLA, ALB, PIME, and RSVD. Three test images, namely, Barbara, cameraman, and mandrill, are imposed with three Gaussian relative noise levels, $\zeta = 20\%$, 30%, and 40%. Each noisy image is denoised using each method along with its best parameter combination (δ, ρ, L) that is optimized with respect to PSNR. The best parameter combinations are shown in parenthesis, the REs for those combinations are shown in square brackets, and the PSNRs for those combinations are shown in other values. For each noisy image, we used the coloring red, blue, and green to indicate the method with the best, second best, and third best performance. The parameter space for the optimization is $\delta = 8, 10, 12$, $\rho = 3, 5, 7$, and $L = 15, 20, 25$ for $\zeta = 20\%$; $\delta = 10, 12, 14$, $\rho = 5, 7, 9$, and $L = 15, 20, 25$ for $\zeta = 30\%$; and $\delta = 12, 14, 16$, $\rho = 7, 9, 11$, and $L = 15, 20, 25$ for $\zeta = 40\%$.

data	$\zeta(\%)$	GGD	MCLA	ALB	PIME	RSVD
Barbara	20	22.84 [0.1358] (10, 5, 20)	22.22 [0.1459] (10, 7, 15)	22.88 [0.1353] (10, 7, 15)	22.88 [0.1351] (10, 7, 15)	22.89 [0.1351] (8, 7, 15)
	30	22.32 [0.1443] (10, 5, 15)	21.39 [0.1605] (10, 7, 20)	22.09 [0.1481] (10, 7, 15)	22.09 [0.1481] (10, 7, 15)	22.17 [0.1468] (12, 9, 15)
	40	20.80 [0.1719] (12, 7, 15)	20.27 [0.1826] (14, 7, 25)	21.02 [0.1676] (16, 9, 15)	21.02 [0.1676] (16, 9, 15)	21.26 [0.1629] (16, 9, 15)
Cameraman	20	20.94 [0.1721] (10, 3, 25)	22.91 [0.1371] (8, 3, 15)	21.07 [0.1695] (12, 7, 15)	22.28 [0.1474] (10, 3, 15)	22.28 [0.1474] (10, 3, 15)
	30	19.61 [0.2005] (12, 7, 15)	20.78 [0.1753] (14, 5, 25)	20.54 [0.1801] (14, 7, 15)	21.20 [0.1670] (14, 5, 15)	20.97 [0.1714] (12, 5, 15)
	40	17.91 [0.12439] (16, 11, 25)	19.29 [0.2081] (14, 7, 15)	19.60 [0.2007] (16, 7, 15)	19.60 [0.2007] (16, 7, 15)	19.78 [0.1968] (16, 9, 20)
Mandrill	20	21.07 [0.1530] (8, 3, 25)	21.36 [0.1481] (8, 7, 15)	20.29 [0.1675] (12, 7, 15)	21.38 [0.1477] (8, 3, 15)	21.15 [0.1517] (8, 3, 15)
	30	19.02 [0.1939] (12, 7, 20)	18.80 [0.1989] (10, 7, 25)	18.98 [0.1947] (10, 7, 15)	18.98 [0.1947] (10, 7, 15)	19.03 [0.1937] (14, 7, 15)
	40	14.98 [0.1948] (14, 7, 25)	18.93 [0.1959] (14, 9, 20)	19.20 [0.1899] (16, 9, 15)	19.20 [0.1899] (16, 9, 15)	19.23 [0.1891] (14, 7, 15)

Table 4: SSIM for the denoising of GGD and its four hybrid frameworks, MCLA, ALB, PIME, and RSVD. Three test images, namely, Barbara, cameraman, and mandrill, are imposed with three Gaussian relative noise levels, $\zeta = 20\%$, 30%, and 40%. Each noisy image is denoised using each method along with its best parameter combination (δ, ρ, L) that is optimized with respect to SSIM. The best parameter combinations are shown in parenthesis and the SSIMs for those combinations are shown in other values. For each noisy image, we used the coloring red, blue, and green to indicate the method with the best, second best, and third best performance. The parameter space for the optimization is $\delta = 8, 10, 12$, $\rho = 3, 5, 7$, and $L = 15, 20, 25$ for $\zeta = 20\%$; $\delta = 10, 12, 14$, $\rho = 5, 7, 9$, and $L = 15, 20, 25$ for $\zeta = 30\%$; and $\delta = 12, 14, 16$, $\rho = 7, 9, 11$, and $L = 15, 20, 25$ for $\zeta = 40\%$.

| data	$\zeta(\%)$	GGD	MCLA	ALB	PIME	RSVD
Barbara	20	0.6395 (12, 5, 15)	0.6527 (8, 5, 25)	0.6398 (12, 7, 15)	0.6638 (8, 5, 15)	0.6536 (12, 5, 15)
	30	0.5848 (10, 5, 15)	0.5864 (10, 5, 25)	0.5858 (14, 9, 15)	0.5858 (14, 9, 15)	0.5882 (14, 9, 15)
	40	0.5069 (12, 7, 15)	0.5168 (16, 7, 20)	0.5310 (16, 9, 15)	0.5310 (16, 9, 15)	0.5309 (14, 9, 15)
Cameraman	20	0.6359 (8, 7, 25)	0.5991 (8, 5, 25)	0.6256 (8, 7, 15)	0.6256 (8, 7, 15)	0.6221 (12, 7, 15)
	30	0.5784 (12, 7, 15)	0.5063 (10, 7, 20)	0.5170 (10, 7, 15)	0.5170 (10, 7, 15)	0.5207 (12, 7, 15)
	40	0.4820 (12, 7, 15)	0.4542 (16, 11, 20)	0.4550 (12, 11, 15)	0.4550 (12, 11, 15)	0.4485 [12, 9, 15]
Mandrill	20	0.5847 (10, 3, 15)	0.6326 (8, 7, 15)	0.5239 (8, 7, 15)	0.6050 (8, 3, 15)	0.5825 (10, 3, 15)
	30	0.4144 (14, 5, 20)	0.4579 (12, 5, 20)	0.4250 (10, 7, 15)	0.4435 (12, 5, 15)	0.4427 (12, 5, 15)
	40	0.3643 (16, 7, 20)	0.4536 (12, 7, 15)	0.4173 (16, 9, 15)	0.4173 (16, 9, 15)	0.4503 (16, 7, 15)
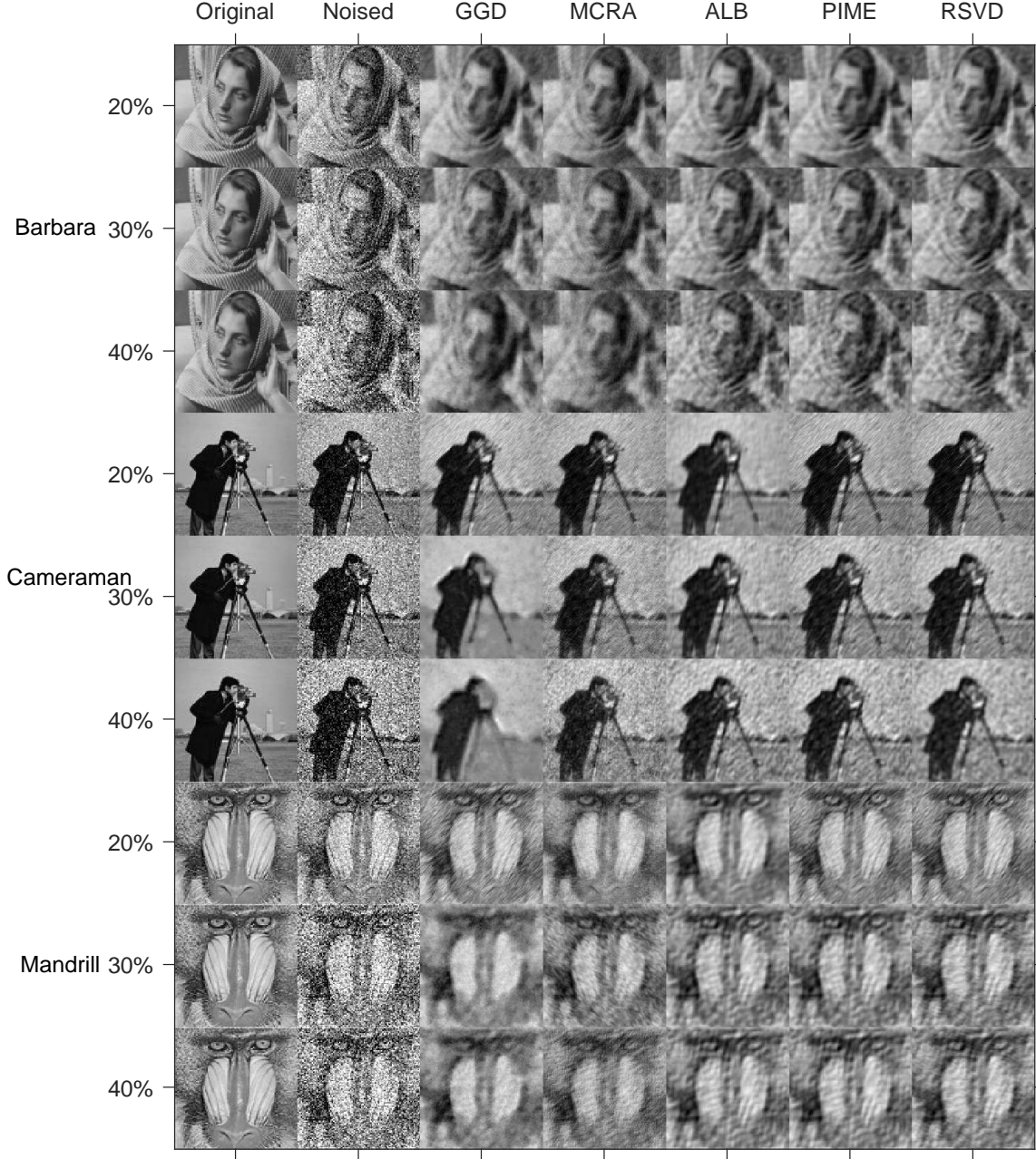
Figure 2: Denoising images using GGD and four of its hybrid frameworks, MCLA, ALB, PIME, and RSVD. Three test images, namely, Barbara, cameraman, and mandrill, are imposed with three Gaussian relative noise levels, $\zeta = 20\%$, 30%, and 40%. Each noisy image is denoised using each method along with its best parameter combination (δ, ρ, L), shown in Table 3, which is optimized with respect to PSNR.
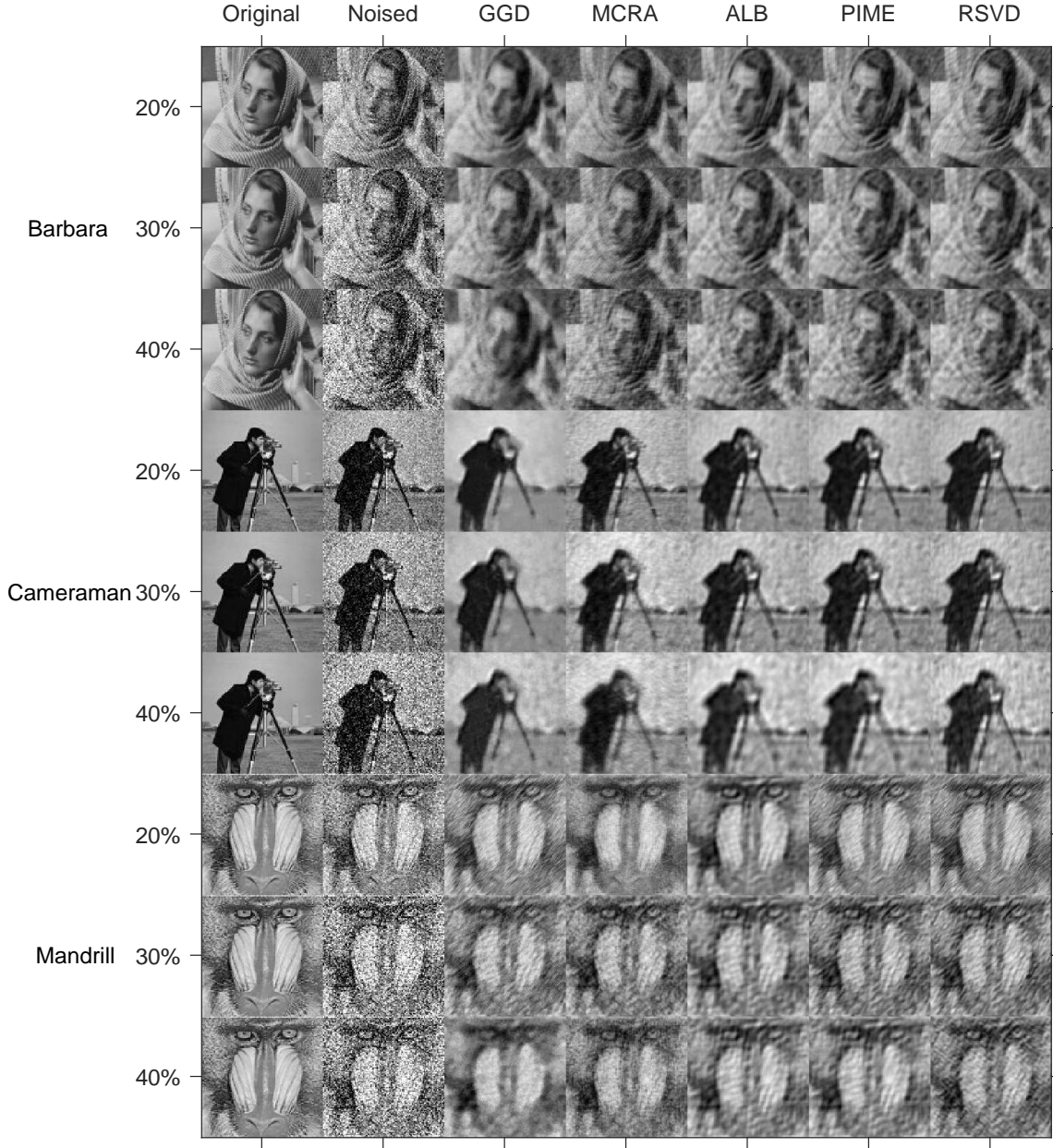
Figure 3: Denoising images using GGD and four of its hybrid frameworks, MCLA, ALB, PIME, and RSVD, with optimized parameters with respect to SSIM. Three test images, namely, Barbara, cameraman, and mandrill, are imposed with three Gaussian relative noise levels, $\zeta = 20\%$, 30%, and 40%. Each noisy image is denoised using each method along with its best parameter combination (δ, ρ, L), shown in Table 4, which is optimized with respect to SSIM.

Monte Carlo Low-Rank Approximation (MCLA), Augmented Lanczos Bidiagonalization (ALB), Preconditioned Iterative Multimethod Eigensolver (PIME), and Randomized Singular Value Decomposition (RSVD). While the conventional approximation techniques, MCLA, ALB, and PIME encounter computational complexity of $\mathcal{O}(mnL)$, the randomization-based approximation method RSDV encounter only $\mathcal{O}(mn\log(L))$. Compared to the original GGD, all of its hybrid methods significantly reduced the computational time and the order of the computational time from the best hybrid framework to the worst is RSVD, MCRA, PIME, and ALB. Moreover, we observed that the computational time differences of RSVD and MCRA are not much significant. While the PSNR values provide that RSVD performs the best and PIME performs the next best, SSIM values provide that all the four hybrid frameworks perform similarly. If we consider both the computational efficiency and denoising quality, while RSVD performs the best, MCRA performs next.

We have been developing deep learning techniques for data imputation in computer vision [17] and time series forecasting in financial markets [15]. We are planning to extend our data imputation and time series forecasting techniques along with this efficient image denoising technique so that the extended framework is capable of operating on partially observed noisy image sequences. Such a framework is capable of denoising a video that contains some objects which are partially observed due to natural phenomena such as occlusion or change of appearance. The applications of such techniques could include learning the dynamics of animal groups in wild, analyzing pedestrians on streets, counting vehicles on roads, and tracking sports players on courts [32]. Utilization of a single hybrid framework is favorable rather than three separate frameworks each for denoising, tracking, and trajectory imputation, as this hybrid framework increases both the computational efficiency and robustness. Future work also includes comparing these hybrid GGD frameworks with state-of-the-art deep learning image restoration tools such as Generative Adversarial Network [21], U-Net [27]. However, the use of deep learning techniques to learn image features requires the ground truth, and it could be a limitation of such a denoising method as ground truth is not always available. The hybrid GGD frameworks presented in this paper are only capable of processing gray-color images; thus, we are planning to extend these four methods in the future such that they are capable of processing color images. For that, first, we decompose the given noisy color image into its three color channels. Then, each channel is denoised with our hybrid GGD separately with its best parameter values. Finally, we merge the denoised color channels and produce the denoised color image.

In this paper, we extended our patch-based image denoising technique GGD that uses eigenvectors of the geodesics' Gramian matrix computed using SVD. The extension is focused on reducing the computational cost of GGD encountered during its SVD step, by integrating singular vector approximation frameworks, namely, MCLA, ALB, PIME, and RSVD. Both, the computational time comparison and denoising performance comparison evidence that the hybrid GGD framework RSVD is the most efficient and robust scheme among all the other techniques.

References

[1] Pankaj K Agarwal and Jeff Erickson. 1997. Geometric range searching and its relatives. *Advances in Discrete and Computational Geometry* 223 (1997), 1–56. https://doi.org/10.1.1.38.6261

[2] Monagi H. Alkinani and Mahmoud R. El-Sakka. 2017. Patch-based models and algorithms for image denoising: a comparative review between patch-based images denoising methods for additive noise reduction. *Eurasip Journal on Image and Video Processing* 2017, 1 (dec 2017), 1–27. https://doi.org/10.1186/s13640-017-0203-4

[3] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. 1999. LAPACK Users' Guide. https://doi.org/10.1137/1.9780898719604

[4] James Baglama and Lothar Reichel. 2005. Augmented Implicitly Restarted Lanczos Bidiagonalization Methods. *SIAM Journal on Scientific Computing* 27, 1 (jan 2005), 19–42. https://doi.org/10.1137/04060593X

[5] Sripad Krishna Devalla, Prajwal K. Renukanand, Bharathwaj K. Sreedhar, Giridhar Subramanian, Liang Zhang, Shamira Perera, Jean-Martial Mari, Khai Sing Chin, Tin A. Tun, Nicholas G. Strouthidis, Tin Aung, Alexandre H. Thiéry, and Michaël J. A. Girard. 2018. DRUNET: a dilated-residual U-Net deep learning network to segment optic nerve head tissues in optical coherence tomography images. *Biomedical Optics Express* 9, 7 (jul 2018), 3244. https://doi.org/10.1364/boe.9.003244 arXiv:1803.00232

[6] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. 2019. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art* 2, 1 (dec 2019), 1–12. https://doi.org/10.1186/s42492-019-0016-7

[7] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. 2019. Brief review of image denoising techniques. , 12 pages. https://doi.org/10.1186/s42492-019-0016-7

[8] Robert W. Floyd. 1962. Algorithm 97: shortest path. *Commun. ACM* 5, 6 (1962), 345. https://doi.org/10.1145/367766.368168

[9] John G. F. Francis. 1961. The QR Transformation A Unitary Analogue to the LR Transformation–Part 1. *Comput. J.* 4, 3 (jan 1961), 265–271. https://doi.org/10.1093/comjnl/4.3.265

[10] John G. F. Francis. 1962. The QR Transformation–Part 2. *Comput. J.* 4, 4 (jan 1962), 332–345. https://doi.org/10.1093/comjnl/4.4.332

[11] Shmuel Friedland, Amir Niknejad, Mostafa Kaveh, and Hossein Zare. 2006. Fast Monte-carlo low rank approximations for matrices. In *Proceedings 2006 IEEE/SMC International Conference on System of Systems Engineering.* 218–223. https://doi.org/10.1109/sysose.2006.1652299 arXiv:0510573 [math]

[12] Kelum Gajamannage and Randy Paffenroth. 2021. Bounded manifold completion. *Pattern Recognition* 111 (dec 2021), 107661. https://doi.org/10.1016/j.patcog.2020.107661

[13] Kelum Gajamannage, Randy Paffenroth, and Erik M. Bollt. 2019. A nonlinear dimensionality reduction framework using smooth geodesics. *Pattern Recognition* 87 (mar 2019), 226–236. https://doi.org/10.1016/j.patcog.2018.10.020

[14] Kelum Gajamannage, Randy Paffenroth, and Anura P Jayasumana. 2020. A Patch-based Image Denoising Method Using Eigenvectors of the Geodesics' Gramian Matrix. *arXiv preprint arXiv:2010.07769* (2020).

[15] Kelum Gajamannage and Yonggi Park. 2022. Real-time Forecasting of Time Series in Financial Markets Using Sequentially Trained Many-to-one LSTMs. (may 2022). https://doi.org/10.48550/arxiv.2205.04678 arXiv:2205.04678

[16] Kelum Gajamannage, Yonggi Park, Randy Paffenroth, and Anura P. Jayasumana. 2022. Reconstruction of fragmented trajectories of collective motion using Hadamard deep autoencoders. *Pattern Recognition* 131 (nov 2022), 108891. https://doi.org/10.1016/j.patcog.2022.108891 arXiv:2110.10428

[17] Kelum Gajamannage, Yonggi Park, Randy Paffenroth, and Anura P. Jayasumana. 2022. Reconstruction of Fragmented Trajectories of Collective Motion using Hadamard Deep Autoencoders. *Pattern Recognition* 131 (2022), 108891.

[18] Kelum Gajamannage, Yonggi Park, and Alexey Sadovski. 2022. Geodesic Gramian Denoising Applied to the Images Contaminated With Noise Sampled From Diverse Probability Distributions. *IET Image Processing* 00 (mar 2022), 1–13. https://doi.org/10.1049/ipr2.12623

[19] Gene H. Golub and Christian Reinsch. 1970. *Handbook Series Linear Algebra Singular Value Decomposition and Least Squares Solutions*.* Technical Report. 403–420 pages.

[20] Gene H. Golub and Charles F. Van Loan. 2013. *Matrix computations.* Vol. 3. JHU press.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[22] N. Halko, P. G. Martinsson, and J. A. Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* 53, 2 (may 2011), 217–288. https://doi.org/10.1137/090771806 arXiv:0909.4061

[23] Alain Horé and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. In *Proceedings - International Conference on Pattern Recognition.* hore2010, 2366–2369. https://doi.org/10.1109/ICPR.2010.579

[24] Effrosyni Kokiopoulou, Constantine Bekas, and Efstratios Gallopoulos. 2004. Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization. In *Applied Numerical Mathematics*, Vol. 49. North-Holland, 39–61. https://doi.org/10.1016/j.apnum.2003.11.011

[25] John Aldo Lee, Amaury Lendasse, and Michel Verleysen. 2004. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. *Neurocomputing* 57, 1-4 (2004), 49–76. https://doi.org/10.1016/j.neucom.2004.01.007

[26] François G. Meyer and Xilin Shen. 2014. Perturbation of the eigenvectors of the graph Laplacian: Application to image denoising. , 326–334 pages. https://doi.org/10.1016/j.acha.2013.06.004

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[28] Alireza Sepas-Moghaddam, Danial Yazdani, and Jalil Shahabi. 2014. A novel hybrid image segmentation method. *Progress in Artificial Intelligence* 3, 1 (apr 2014), 39–49. `https://doi.org/10.1007/s13748-014-0044-7`

[29] Donald Shepard. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference, ACM 1968*. ACM Press, New York, New York, USA, 517–524. `https://doi.org/10.1145/800186.810616`

[30] Danny C. Sorensen. 1992. Implicit Application of Polynomial Filters in a k -Step Arnoldi Method. *SIAM J. Matrix Anal. Appl.* 13, 1 (jul 1992), 357–385. `https://doi.org/10.1137/0613025`

[31] Andreas Stathopoulos and James R McCombs. 2010. PRIMME: PReconditioned Iterative MultiMethod Eigensolver—methods and software description. *ACM Transactions on Mathematical Software (TOMS)* 37, 2 (2010), 1–30.

[32] Tamás Vicsek and Anna Zafeiris. 2012. Collective motion. *Physics Reports* 517, 3-4 (aug 2012), 71–140. `https://doi.org/10.1016/j.physrep.2012.03.004`

[33] Qi Wang, Jing Ma, Siyuan Yu, and Liying Tan. 2020. Noise detection and image denoising based on fractional calculus. *Chaos, Solitons and Fractals* 131 (feb 2020), 109463. `https://doi.org/10.1016/j.chaos.2019.109463`

[34] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (apr 2004), 600–612. `https://doi.org/10.1109/TIP.2003.819861`

[35] Lingfei Wu, Eloy Romero, and Andreas Stathopoulos. 2017. Primme_svds: A high-performance preconditioned svd solver for accurate large-scale computations. *SIAM Journal on Scientific Computing* 39, 5 (2017), S248—-S271.

[36] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing* 26, 7 (jul 2017), 3142–3155. `https://doi.org/10.1109/TIP.2017.2662206` arXiv:1608.03981

[37] Kai Zhang, Wangmeng Zuo, and Lei Zhang. 2018. FFDNet: Toward a fast and flexible solution for CNN-Based image denoising. *IEEE Transactions on Image Processing* 27, 9 (sep 2018), 4608–4622. `https://doi.org/10.1109/TIP.2018.2839891` arXiv:1710.04026