An Algorithm for Improving Non-Local Means Operators via Low-Rank Approximation

Victor May, Yosi Keller, Nir Sharon, and Yoel Shkolnisky

***Abstract*— We present a method for improving a non-local means (NLM) operator by computing its low-rank approximation. The low-rank operator is constructed by applying a filter to the spectrum of the original NLM operator. This results in an operator, which is less sensitive to noise while preserving important properties of the original operator. The method is efficiently implemented based on Chebyshev polynomials and is demonstrated on the application of natural images denoising. For this application, we provide a comparison of our method with other denoising methods.**

***Index Terms*— Denoising, non-local means operator, Chebyshev polynomials.**

I. Introduction

DENOISING images is a classical problem in image processing. Representative approaches for solving this problem include local methods such as linear filtering and anisotropic diffusion [17], global methods such as total-variation [20] and wavelet shrinkage [6], and discrete methods such as Markov Random field denoising [23] and discrete universal denoiser [15].

As is often the case with inverse problems, many of the algorithms for image denoising involve priors on the solution. Commonly, the only prior knowledge assumed about the image is that it is of natural origin. In that case, the sought-after prior should represent the statistics of a natural image. Natural image statistics is a research topic on its own [12], [30], having importance for image processing problems other than denoising: segmentation, texture synthesis, image inpainting, super-resolution and more. An important observation in natural image statistics is that images contain repetitive local patterns. This observation is the basis of patch-based denoising methods such as Non-Local Means (NLM) [4] and block-matching and 3D filtering (BM3D) [7].

Manuscript received April 10, 2015; revised August 31, 2015 and December 2, 2015; accepted January 4, 2016. Date of publication January 18, 2016; date of current version February 8, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Karen Egiazarian. *(Corresponding author: Nir Sharon.)*

V. May and Y. Shkolnisky are with the School of Mathematical Sciences, Tel Aviv University, Tel Aviv 39040, Israel (e-mail: mayvic@gmail.com; yoelsh@post.tau.ac.il).

Y. Keller is with the Faculty of Engineering, Bar-Ilan University, Ramat Gan 52900, Israel (e-mail: yosi.keller@gmail.com).

N. Sharon is with the Program of Applied and Computational Mathematics, Princeton University, Fine Hall Washington Road, Princeton, NJ 08544-1000 USA (e-mail: nir.sharon@math.tau.ac.il).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes a supplementary reading file (PDF format), code (MATLAB package, including README file), and a database (two sets of images). The total size of the file is 12.6 M. Contact nir.sharon@math.tau.ac.il for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2016.2518805

In NLM, the denoising operator is constructed from patches of the corrupted image itself. As such, the denoising operator is affected by the noise. It had been noted [13], [22] that larger eigenvalues of the operator are less sensitive to noise than smaller ones, creating an opportunity to improve the operator. Furthermore, based upon numerical experiments, this improvement becomes more significant as one considers images contaminated with high levels of noise. The importance of this observation is related to the fundamental bounds of image denoising, as modern denoising methods have already reached these bounds for low noise levels [5]. Therefore, the method presented in the current paper addresses a range of noise levels for which the denoising problem has not yet been adequately solved.

We propose a method for denoising NLM operators by using a low-pass filter applied to their eigenvalues. In other words, we pose the problem of improving NLM operators as a filter design task, which is a classical task in signal processing. A similar concept of filtering the operator was recently proposed in [14] and [26], where a low rank approximation of the operator is determined by a statistical model. Other approaches of low rank approximation for denoising have been suggested in [18], [24], and [27]. In contrast to low rank approximation approaches, our method uses a chosen filter function which suppresses eigenvalues with small magnitude when applied to a matrix, and accurately approximates the remaining low-rank operator, while preserving the fundamental properties of the original operator. This filter is efficiently applied to the NLM operator based on Chebyshev polynomials for matrices. We also derive error bounds and analyse some of the required properties of the filter.

We further study the concept of low-rank NLM denoising operators by numerical experiments. In these experiments we investigate two key issues of our method. The first issue is the dependence of the low-rank operator on its two tuning parameters, which are the width of the kernel of the original NLM operator, and the rank of the low-rank operator. The second issue is the performance of our method compared to other advanced denoising algorithms. We provide a comparison which includes a few advanced methods including a two-stage denoising scheme based on our low-rank operator.

The paper is organized as follows. In Section II we present the notation, the problem's formulation and the NLM method. In Section III we introduce our method of constructing low-rank approximations for NLM operators, which is based on matrix filtering functions. Next, in Section IV, we show how to efficiently implement this low-rank approximation using Chebyshev polynomials. Section V provides numerical experiments with our algorithm, where we discuss its para-

meters and compare it to other advanced denoising methods. We summarize the paper in Section VI.

II. Preliminaries

We begin by introducing the notation and the model for the denoising problem. Let I be a discrete grid (typically of dimension 1 or 2) and denote by $X = \{x_i \mid i \in I\}$ a clean signal defined on I. Let $Y = \{y_i \mid i \in I\}$ be a signal contaminated with noise, that is

$$y_i = x_i + r_i, \quad r_i \sim \mathcal{N}\left(0, \sigma^2\right), \ i \in I, \tag{1}$$

are independently identically distributed Gaussian random variables. The goal is to estimate X given Y.

The non-local means (NLM) estimator is given as follows. Denote by $N_i^Y = N_i^Y(p)$ the indices of the pixels within a cube of side length p centred around the pixel $i \in I$. Let $v\left(N_i^Y\right)$, $i \in I$ be the values of the pixels of Y at the indices N_i^Y, considered as a column vector. The NLM algorithm estimates x_i as

$$\hat{x}_i = \frac{1}{z_i} \sum_{j \in I} K_h\left(v\left(N_i^Y\right) - v\left(N_j^Y\right)\right) y_j,$$

where $K_h(\cdot) = \exp(-\frac{\|\cdot\|_2^2}{2h^2})$ is the Gaussian kernel function with width $h > 0$, and

$$z_i = \sum_{j \in I} K_h\left(v\left(N_i^Y\right) - v\left(N_j^Y\right)\right)$$

is a normalization factor. In matrix notation the NLM estimator is written as

$$\hat{x} = Ay, \tag{2}$$

where $y \in \mathbb{R}^n$ is the original noisy signal Y represented as a vector, $\hat{x}$ is the denoised signal (the estimations vector), and A is a NLM operator of Y, defined as follows.

Definition 1: A matrix $A \in \mathbb{R}^{n \times n}$ is a NLM operator of Y if $A = D^{-1}W$, where $W \in \mathbb{R}^{n \times n}$ is given by

$$W_{ij} = K_h\left(v\left(N_i^Y\right) - v\left(N_j^Y\right)\right)$$

with $K_h(\cdot) = \exp(-\frac{\|\cdot\|_2^2}{2h^2})$ and $h > 0$, and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix given by $D_{ii} = \sum_{j=1}^n W_{ij}$.

Remark 1: The pseudocode of all algorithms described in this paper is given in Appendix B. In these algorithms, we denote by $\mathrm{NLM}_{p,h}(Y)$ the NLM operator of an image Y with patch size p and kernel width h.

III. Denoising the NLM Operator

Since the image Y in (1) is noisy, Definition 1 implies that the NLM operator A in (2) is constructed from noisy data, see e.g., [13], and is thus noisy as well. We would like to replace A with an operator which is less noisy. In particular, we look for replacing A with its low-rank approximation.

A low-rank approximation obtained by a truncated singular-value decomposition (SVD) is optimal under the L_2 norm [9, Ch. 2.5], however, directly computing the SVD of A is computationally expensive and often impractical. Computing a truncated eigenvalues (spectral) decomposition is an alternative method, typically implemented by a variant of the power iterations algorithm, such as Lanczos iterations. Unfortunately, all existing spectral decomposition methods are often too slow to be applied to a NLM matrix. Low rank approximation of the NLM matrix via truncated spectral decomposition is given in Algorithm 1 in Appendix B.

Several approaches have been proposed to improve the NLM estimator (2). In [22], it has been proposed to replace A by the polynomial $2A - A^2$. This polynomial suppresses eigenvalues which are much smaller than 1, while preserving the eigenspaces corresponding to eigenvalues closer to 1. Alternatively, Awate and Whitaker [1], [2] have proposed an iterative method (known as "UINTA") which gradually minimizes an entropy function. This entropy serves as a measure of self-similarity of patches. These approaches essentially compute low rank approximations of A, which have been shown to outperform A. A low rank approximation of the NLM operator is also studied in [14] and [26], where soft thresholding of the spectrum is determined by a statistical model. Thresholding via SVD is used in [18] for stacks of similar patches.

Our approach is to construct a low-rank approximation of a NLM operator A (see Definition 1) by using a matrix function $f : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$, and computing $f(A)$. This matrix function approach also provides a different perspective on soft thresholding approaches such as [1] and [22], as for example, it explicitly shows the relation between soft thresholding, the kernel width, and the cutoff point, as shown later in Section V. Our approach is essentially a derivative of [22] which allows to design arbitrary filtering functions, while controlling their properties and applying them in a numerically stable and efficient way.

Next, we characterize functions of NLM operators and suggest properties of such functions which are suitable for our purposes. Then, we present a particular family of functions with controlled low-rank that have these properties.

A. Matrix Functions of NLM Operators

Before studying matrix functions of NLM operators, we summarize a few properties of these NLM matrices.

Lemma 2: Let A be a NLM operator, as defined in Definition 1. Then A has the following properties:

1) *A is positive diagonalizable, namely there exists an invertible matrix Q that satisfies $A = Q^{-1}\Lambda Q$, where Λ is a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0$.*
2) *$\lambda_1 = 1$ is the maximal eigenvalue of A with a corresponding eigenvector $\mathbf{1}$ (the all-ones vector), that is, $A\mathbf{1} = \mathbf{1}$.*

The proof of Lemma 2 is given in Appendix A. As a first conclusion from Lemma 2 we have.

Corollary 3: In the notation of Lemma 2, any function f such that $f(A)$ is well-defined, satisfies

$$f(A) = Q^{-1} f(\Lambda) Q.$$

The proof follows directly from the diagonalizability of A and the definition of matrix functions, see e.g., [9, Corollary 11.1.2]. For more details on lifting a scalar function to matrices see [11, Ch. 1].

We would like to design f so that $f(A)$ is a low-rank approximation of the NLM operator A. Moreover, we wish to guarantee that $f(A)$ retains the properties of A listed in Lemma 2. The first property in Lemma 2 allows repeated applications of the operator $f(A)$. The advantages of repeated applications of a denoising operator have been demonstrated in [22]. The second property in Lemma 2, that is $f(A)\mathbf{1} = \mathbf{1}$ (row stochastic normalization), is useful because then the elements of $y = f(A)x$ are affine combinations (linear combinations whose coefficients sum to one) of the elements of x. In particular it is desired for a denoising operator to map a constant signal to itself, see e.g., [6].

Summarizing the above discussion of desired properties of the matrix $f(A)$, we define the notion of an extended NLM operator.

Definition 4: A matrix $B \in \mathbb{R}^{n\times n}$ is an extended NLM operator if B is diagonalizable, such that $B\mathbf{1} = \mathbf{1}$, with all its eigenvalues in $[0, 1]$.

While it is possible to define an extended NLM operator such that its eigenvalues are in $(-1, 1]$, we prefer for simplicity to use Definition 4 above, as this allows for a richer family of functions to be used in Section III-B below. This definition is also consistent with the Gaussian kernel used in Definition 1, which ensures that the resulting operator is non-negative definite. Equipped with the latter we have,

Theorem 5: Let A be a NLM operator. The following conditions on f are sufficient for $f(A)$ to be an extended NLM operator.

1) $f(1) = 1$.
2) *f is defined on* $[0, 1]$ *and* $\{f(x) \mid 0 \le x \le 1\} \subseteq [0, 1]$.

Proof: By Corollary 3, f acts solely on the diagonal matrix Λ, meaning that

$$f(A) = Q^{-1}\,\mathrm{diag}\left(f(\lambda_1), \ldots, f(\lambda_n)\right) Q,$$

where the eigenvalues of A are ordered such that $\lambda_1 = 1$. Therefore, the second condition guarantees that the eigenvalues of $f(A)$ are in $[0, 1]$. In addition, the first condition ensures that the maximal eigenvalue is indeed one, and is given by $f(\lambda_1)$, that is $f(\lambda_1) = 1$. The eigenvector corresponding to $f(\lambda_1)$ is the first column of Q, which is $\mathbf{1}$ (up to normalization), namely, $f(A)\mathbf{1} = \mathbf{1}$. □

The following corollary allows composition of functions satisfying the conditions of Theorem 5.

Corollary 6: Let B be an extended NLM operator, and let g be a function satisfying the conditions of Theorem 5. Then, $g(B)$ is also an extended NLM operator.

The proof of Corollary 6 is elementary and is thus omitted.

B. Constructing a Low-Rank Extended NLM Operator

We would like to construct a function $f(A)$ that satisfies the two conditions from Theorem 5, and moreover, suppresses the noise in the NLM operator A. A natural option is to choose a function that retains the eigenvalues above a given threshold. By Corollary 3, choosing such a function reduces to choosing an appropriate scalar function.

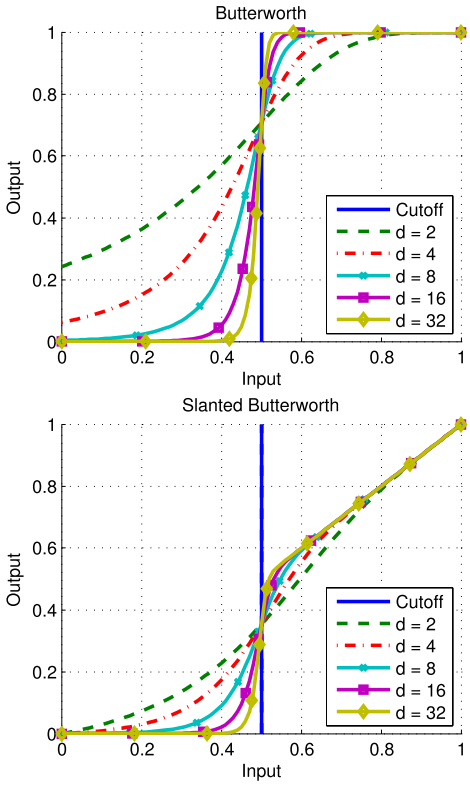


Fig. 1. Plots of the Butterworth and Slanted Butterworth functions, for cutoff set at 0.5 and different filter orders.

Our first prototype for a scalar thresholding function is

$$g_\omega(x) = \begin{cases} 0 & x < \omega, \\ x & \omega \le x. \end{cases} \tag{3}$$

This function satisfies the conditions of Theorem 5 for $0 \le \omega \le 1$, while zeroing values lower than ω and acting as the identity for higher values. However, this function is not smooth which eventually results in its slow evaluation for matrices (a detailed discussion is given in Section IV-B).

Inspired by the gain function of the Butterworth filter (also known as the maximal flat filter, see e.g., [16, Ch. 7])

$$f^b_{\omega,d}(x) = \left(1 + \left(\frac{1-x}{1-\omega}\right)^{2d}\right)^{-\frac{1}{2}}, \quad x \in [0, 1],$$

which approximates the ω-shifted Heaveside function on the segment $[0, 1]$, we propose to use

$$f^{sb}_{\omega,d}(x) = x\left(1 + \left(\frac{1-x}{1-\omega}\right)^{2d}\right)^{-\frac{1}{2}}, \quad x \in [0, 1]. \tag{4}$$

We term this function the Slanted Butterworth (SB) function. The SB function serves as an approximation to g_ω of (3). The response of both functions $f^b_{\omega,d}(x)$ and $f^{sb}_{\omega,d}(x)$ for various values of d is shown in Figure 1.

In its matrix version, the SB function becomes

$$f^{sb}_{\omega,d}(A) = A\left(I + \left(\frac{1}{1-\omega}(I - A)\right)^{2d}\right)^{-\frac{1}{2}}, \tag{5}$$

where $A \in \mathbb{R}^{n\times n}$ and I is the $n \times n$ identity matrix. Combining Theorem 5 and Corollary 6, one can verify that $f^{sb}_{\omega,d}(A)$, $0 \le \omega \le 1$, is indeed an extended NLM operator and thus $f^{sb}_{\omega,d}$ is applicable for our purposes. The parameter $0 \le \omega \le 1$ is termed the filter cutoff and $d \in \mathbb{N}$ is the order of the filter.

IV. Computing Low-Rank Approximation Based on Chebyshev Polynomials

The evaluation of the matrix function $f^{sb}_{\omega,d}(A)$ for a large matrix A can be challenging. While the function essentially operates on the eigenvalues of the NLM operator A, the spectral decomposition of A is assumed to be unavailable due to computational reasons. Furthermore, evaluating the function $f^{sb}_{\omega,d}$ directly according to its definition (5) requires computing the square root of a matrix. Evaluating the square root of a matrix is prohibitively computationally expensive, see e.g. [10], and thus, a direct evaluation of $f^{sb}_{\omega,d}$ is also infeasible. In addition, an important observation is that one does not need the resulting matrix $f^{sb}_{\omega,d}(A)$ but only the vector $\hat{x} = f^{sb}_{\omega,d}(A)\, y$, as can be seen from (2).

A. Evaluating the SB Function Based on Chebyshev Polynomials

The Chebyshev polynomials of the first kind of degree k are defined as

$$T_k(x) = \cos\left(k \arccos(x)\right), \quad x \in [-1,1],\ k = 0,1,2,\ldots$$

These polynomials satisfy the three term recursion

$$T_k(x) = 2xT_{k-1} - T_{k-2}, \quad k = 2,3,\ldots \tag{6}$$

with $T_0(x) = 1$ and $T_1(x) = x$, and form an orthogonal basis for $L_2([-1,1])$ with the inner product

$$\langle f, g\rangle_T = \frac{2}{\pi}\int_{-1}^{1} \frac{f(t)g(t)}{\sqrt{1-t^2}}\, dt. \tag{7}$$

The Chebyshev expansion for any $f \in L_2([-1,1])$ is thus given by

$$\begin{aligned} f(x) &= \sum_{j=0}^{\infty} \alpha_j T_j(x), \\ \alpha_0 &= \frac{1}{2}\langle f, T_0\rangle_T, \quad \alpha_n = \langle f, T_n\rangle_T, \ n \in \mathbb{N}, \end{aligned} \tag{8}$$

where the equality above holds in the L_2 sense for any $f \in L_2([-1,1])$, and becomes pointwise equality under additional regularity assumptions on f.

We will evaluate the function $f^{sb}_{\omega,d} : [0,1] \to \mathbb{R}$ by truncating the Chebyshev expansion (8), that is

$$f^{sb}_{\omega,d}(z) \approx \sum_{j=0}^{N-1} \alpha_j T_j(y),$$

where α_j are the corresponding Chebychev coefficients for $f^{sb}_{\omega,d}$, and $y = 2z - 1$ maps $f^{sb}_{\omega,d}$ from $[0,1]$ to $[-1,1]$, as required by the definition of α_j above. The truncated Chebyshev expansion provides a polynomial approximation which is the best least squares approximation with respect to the induced norm $\sqrt{\langle f, f\rangle_T}$. Remarkably, this least squares approximation is close to the best min-max polynomial approximation, measured by the maximum norm, $\max_{x\in[-1,1]} |f(x)|$. This phenomenon together with a fast evaluation procedure (which we describe shortly) motivate us to use Chebyshev polynomials.

As shown in Corollary 3, applying a Chebychev expansion to a NLM matrix A is reduced to applying it to the diagonal form of A. Thus,

$$\begin{aligned} f^{sb}_{\omega,d}(A) &= Q^{-1} f^{sb}_{\omega,d}(\Lambda) Q \\ &= Q^{-1}\left(\sum_{j=0}^{\infty} \alpha_j T_j(2\Lambda - I)\right) Q \end{aligned}$$

holds for any diagonal matrix since $f^{sb}_{\omega,d}$ is a continuous function [28, Ch. 8]. Therefore, $f^{sb}_{\omega,d}(A)$ is equal to

$$Q^{-1}\left(\sum_{j=0}^{\infty} \operatorname{diag}(\alpha_j T_j(2\lambda_1 - 1), \ldots, \alpha_j T_j(2\lambda_n - 1))\right) Q. \tag{9}$$

where the second equality holds for any diagonal matrix since $f^{sb}_{\omega,d}$ is a continuous function [28, Ch. 8]. In other words, one can see that the coefficients $\{\alpha_j\}$ required to evaluate the matrix function $f^{sb}_{\omega,d}(A)$ of (5) are the same as those required to evaluate the scalar function $f^{sb}_{\omega,d}(x)$ of (4).

For square matrices, substituting a matrix in a polynomial is well-defined (see e.g. [9, Ch. 11]) and so given a truncation parameter $N \in \mathbb{N}$, the matrix SB function (5) is approximated by

$$f^{sb}_{\omega,d}(A) \approx S_N(f^{sb}_{\omega,d}, A) = \sum_{j=0}^{N} \alpha_j T_j(2A - I). \tag{10}$$

The common practice for calculating Chebyshev expansions is by using the discrete orthogonality of the Chebyshev polynomials. Explicitly, the coefficients α_j of (8) are calculated using

$$\langle f, T_j\rangle_T = \frac{1}{N+1}\sum_{k=1}^{N+1} f(x_k) T_j(x_k), \quad j = 0,\ldots,N, \tag{11}$$

where

$$x_k = \cos\left(\frac{\pi(k - \frac{1}{2})}{N}\right).$$

For more details see [8, Ch. 5.8].

Having obtained the expansion coefficients $\{\alpha_j\}_{j=0}^{N}$ of (10), we turn to show how to efficiently evaluate $\hat{x} = S_N(f^{sb}_{\omega,d}, A)y$. We do the latter by using a variant of Clenshaw's algorithm (see e.g., [8, Ch. 5.5]), which is based on the three term recursion (6), as described in [8, p. 193]. This algorithm is adapted to matrices using the fact that each polynomial consists only of powers of A, which means that any two matrix polynomials commute. In addition, we exploit the fact that we need to compute only the vector $\hat{x}$ and not the matrix $S_N(f^{sb}_{\omega,d}, A)$ itself, which has much larger dimensions. The

pseudocode for evaluating $S_N(f^{sb}_{\omega,d}, A)y$ for some vector y is given in Algorithm 2 in Appendix B. Note that this algorithm does not require any matrix-matrix operations, and thus is applicable even for large matrix sizes.

The complete denoising algorithm, which computes a denoised image based on $S_N(f^{sb}_{\omega,d}, A)$ for a NLM operator A, is presented in Algoritm 3 in Appendix B.

B. Error Bounds for NLM Operators

We study the approximation error of the truncated Chebychev expansion for the case of matrix functions and for NLM operators, which are diagonalizable, non-symmetric matrices. The use of truncated Chebyshev expansions for matrices (10) and their approximation order has already been studied in the context of solutions of partial differential equations [25], [28]. However, most results assume that the approximated function is analytic, and so are not applicable in our case.

In this section we use the following notation. Denote by $\|X\|_F = \sqrt{\text{tr}(XX^T)}$ the Frobenius norm of X and by $\|X\|$ its spectral norm (or the induced 2-norm), that is the largest singular value of X. In addition, denote by $\kappa(X)$ the condition number of an invertible matrix X (with respect to the spectral norm), that is $\kappa(X) = \|X^{-1}\|\|X\|$.

Recall that if A is a NLM operator, then A can be decomposed as $A = Q^{-1}\Lambda Q$ (see Lemma 2). In addition, A is similar to a symmetric matrix via $D^{-\frac{1}{2}}$ with D being the diagonal matrix of Definition 1. Therefore, $Q = D^{-\frac{1}{2}}O$, where O is an orthogonal matrix.

The next theorem presents the main error bound.

Theorem 7: Let $f \in C^{m+1}([-1, 1])$ and let A be an $n \times n$ NLM operator. Denote by

$$e_N(f)(x) = f(x) - S_N(f, x)$$

the approximation error of the truncated Chebychev expansion of degree N. Then,

$$\|e_N(f)(A)\| \leq C\frac{1}{(N-m)^m}\kappa(D^{1/2}),$$

where $C = \frac{2}{\pi m}\|f^{(m+1)}\|_{T,1}$ is a constant that depends on the $m+1$ derivative of f but is independent of n and A, with

$$\|f^{(m+1)}\|_{T,1} = \int_{-1}^{1} \frac{|f^{(m+1)}(t)|}{\sqrt{1-t^2}}\, dt.$$

Proof: Since A is diagonalizable, and similarly to Corollary 3 and (9),

$$\begin{aligned} e_N(f)(A) &= f(A) - \sum_{j=0}^{N} \alpha_j T_j(A) \\ &= Q^{-1}(f(\Lambda) - \sum_{j=0}^{N} \alpha_j T_j(\Lambda))Q = Q^{-1}E_\Lambda Q, \end{aligned}$$

where E_Λ is the diagonal matrix $E_\Lambda = f(\Lambda) - \sum_{j=0}^{N} \alpha_j T_j(\Lambda)$. For all submultiplicative norms, including the spectral norm, we have that

$$\|e_N(f)(A)\| \leq \|Q^{-1}\|\|E_\Lambda\|\|Q\|. \tag{12}$$

By the Chebychev approximation bound [29, Th. 4.3] for scalar functions,

$$\|(E_\Lambda)_{ii}\| \leq \frac{C}{(N-m)^m}, \quad 1 \leq i \leq n,$$

where $C = \frac{2}{\pi m}\|f^{(m+1)}\|_{T,1}$. The spectral norm of a diagonal matrix is given by its element on the diagonal having maximal absolute value and thus

$$\|E_\Lambda\| \leq \frac{C}{(N-m)^m}. \tag{13}$$

Now, it remains to find a bound on $\|Q^{-1}\|\|Q\|$. However,

$$\|Q^{-1}\|\|Q\| = \|O^* D^{1/2}\|\|D^{-1/2}O\|. \tag{14}$$

Using again the submultiplicativity property we get

$$\|Q^{-1}\|\|Q\| \leq \|O\|\|O^*\|\|D^{1/2}\|\|D^{-1/2}\| = \kappa(O)\kappa(D^{1/2}), \tag{15}$$

where in the last equality we have used the orthogonality of O. By the same property, we have that $\|O\| = 1$ and $\kappa(O) = 1$. Combining the latter with (12), (13), and (15) concludes the proof. □

The proof of Theorem 7 holds with minor adjustments to various submultiplicative norms. However, for one particular norm an explicit bound can be achieved, as explained in the following remark.

Remark 2: The error bound of the truncated Chebyshev expansion for matrices, as appears in Theorem 7, can be also expressed in terms of the Frobenius norm since $\|X\|_F \leq \sqrt{n}\|X\|$. Therefore, we immediately get

$$\|e_N(f)(A)\|_F \leq C\frac{\sqrt{n}}{(N-m)^m}\kappa_F(D^{1/2}),$$

where $\kappa_F(D^{1/2}) = \|D^{1/2}\|_F\|D^{-1/2}\|_F$.

The bound of Theorem 7 indicates that the approximation error decays as $\frac{1}{N^m}$, where m is related to the smoothness class of the function f, and N is the number of terms in the truncated Chebyshev expansion (10). However, there are two additional factors that appear on top of this decay rate. The first is the constant C, governed by $\|f^{(m+1)}\|_{T,1}$. This constant can be large for a function whose $m+1$ derivative has large magnitude. The second is the condition number of $D^{1/2}$. This condition number can be easily calculated numerically since D is a diagonal matrix and thus

$$\kappa(D^{1/2}) = \frac{\max_i \sqrt{D_{ii}}}{\min_i \sqrt{D_{ii}}}. \tag{16}$$

Similarly, for the Frobenius norm we have

$$\kappa_F(D^{1/2}) = \sqrt{\frac{\sum_i D_{ii}}{\sum_i D_{ii}^{-1}}}. \tag{17}$$

Moreover, we can bound (16) and (17) a priori, as seen next.

Lemma 8: Let D be the diagonal matrix from Definition 1. Then,

$$\kappa(D^{1/2}) \leq \sqrt{n} \quad \text{and} \quad \kappa_F(D^{1/2}) \leq n.$$

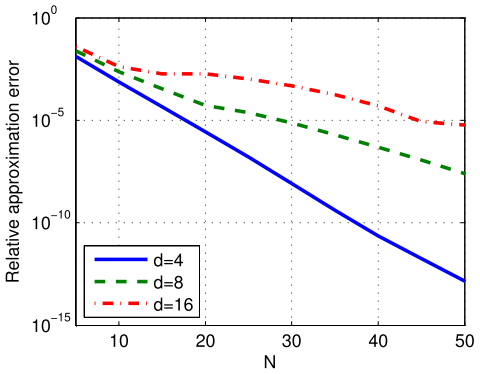


Fig. 2. Relative approximation errors for the truncated Chebyshev expansion of $f^{sw}_{\omega,d}$ with a fixed ω and different values of d.

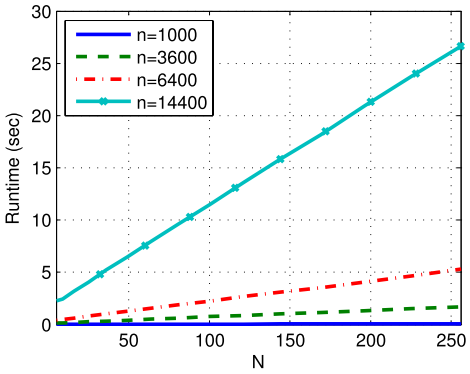


Fig. 3. Runtime, in seconds, of the evaluation of $S_N(f^{sb}_{\omega,d}, A)y$ for different lengths of the vector y.

Proof: K_h in Definition 1 is a Gaussian kernel and so $W_{ii} = 1$ and $0 \leq W_{ij} \leq 1$. Thus, $1 \leq D_{ii} \leq n$ and $1 \leq D_{ii}^{1/2} \leq \sqrt{n}$. Therefore, by (16) we have $\kappa_2(D^{1/2}) \leq \sqrt{n}$.

For the Frobenius norm, it follows that $\frac{1}{n} \leq D_{ii}^{-1} \leq 1$ and thus $\sum_i D_{ii} \leq n^2$ and $\sum_i D_{ii}^{-1} \geq 1$. Thus, by (17), $\kappa_F(D^{1/2}) \leq n$. □

Equipped with Lemma 8, we conclude that

Corollary 9: In the notation of Theorem 7 we have

$$\|e_N(f)(A)\| \leq C \frac{\sqrt{n}}{(N-m)^m}.$$

Note that by Remark 2, a bound on the Frobenius norm of the approximation error $e_N(f)(A)$ is given by

$$\|e_N(f)(A)\|_F \leq C \frac{n^{1.5}}{(N-m)^m}.$$

To conclude the above discussion, we evaluate the approximation error numerically, depicted in Figure 2, where we use 50 random, positive diagonalizable, and non-symmetric matrices, whose rank equals to 1000. We average the relative approximation errors defined as $\|E_N(f^{sw}_{\omega,d}, A)\| / \|f^{sw}_{\omega,d}(A)\|$ over all 50 matrices, where the truncated Chebyshev series is evaluated by Algorithm 2. The cutoff parameter ω of $f^{sw}_{\omega,d}$ has been set to $\omega = 0.7$. The plotted curves correspond to the orders 4, 8 and 16 of the function $f^{sw}_{\omega,d}$. The results clearly show that as the derivative grows, the error rate increases, since as d gets larger so are the derivatives of $f^{sw}_{\omega,d}$.

C. Runtime Analysis

We start by analysing the runtime of $S_N(f^{sb}_{\omega,d}, A)y$ as calculated by Algorithm 2 in Appendix B. The main loop performs N iterations which are dominated by a single matrix to vector multiplication. The parameter N is chosen a priori, according to the error analysis given in Subsection IV-B. Therefore, the total runtime complexity is $\mathcal{O}(Nn^2)$, where n is the number of pixels (see (2)). Experimental timings of Algorithm 2 are presented in Figure 3, for different n and N values. As illustrated in Figure 2, for low order filters, a reasonable value of N should not exceed 50, which means less than 7 seconds for a small image (or a block) of 120×120 pixels. These measurements where taken on a 3.4 GHz Intel Core i7 processor.

To further illustrate the applicability of our method, we measure the runtime for a practical image of size 512×512, that is about $250,000$ pixels. Recall that the NLM operator of an image of size $m_1 \times m_2$ is a dense matrix of size $n \times n$ where $n = m_1 m_2$ is the number of pixels. Therefore, even constructing a matrix of the required size is infeasible. To overcome this problem, we apply clustering preprocessing on the set of patches. Then, we proceed to denoise each cluster separately. Based upon this preprocessing scheme, the average runtime for denoising a 512×512 image by the NLM operator is about 90 seconds and the runtime for applying our denoising method is 120 seconds. As we see next in Section V, these reasonably short execution times result in significant improvements for denoising images contaminated with high levels of noise.

V. Numerical Experiments

The advantage of improving the NLM operator by using its low-rank approximation has already been argued theoretically in Section III. In this section we demonstrate this advantage by numerical examples, done in three parts. In the first part, in Sections V-A and V-B, we study the effect of the kernel width h from Definition 1 and the filter cutoff ω from (5). As a reference, we use the naive approach for denoising a NLM operator, by computing its truncated eigenvalues decomposition. In the second part, in Section V-C, we apply our method to denoise images of practical sizes (see also Subsection IV-C) and compare its performance with the baseline NLM operator. Finally, in Sections V-D and V-E, we compare the performance of our algorithm with several other advanced denoising algorithms. Due to space constraints, the complete set of comparisons against other denoising algorithms is given in the supplementary material.

The denoising performance in all of the following experiments is measured by the peak signal-to-noise ratio (PSNR), which is given for our test images having values in the range $0 - 255$ by

$$\text{PSNR}\,(I_1, I_2) = 20 \cdot \log_{10}\left(\frac{255}{\sqrt{\sum_{i,j} |I_1(i,j) - I_2(i,j)|^2}}\right). \tag{18}$$

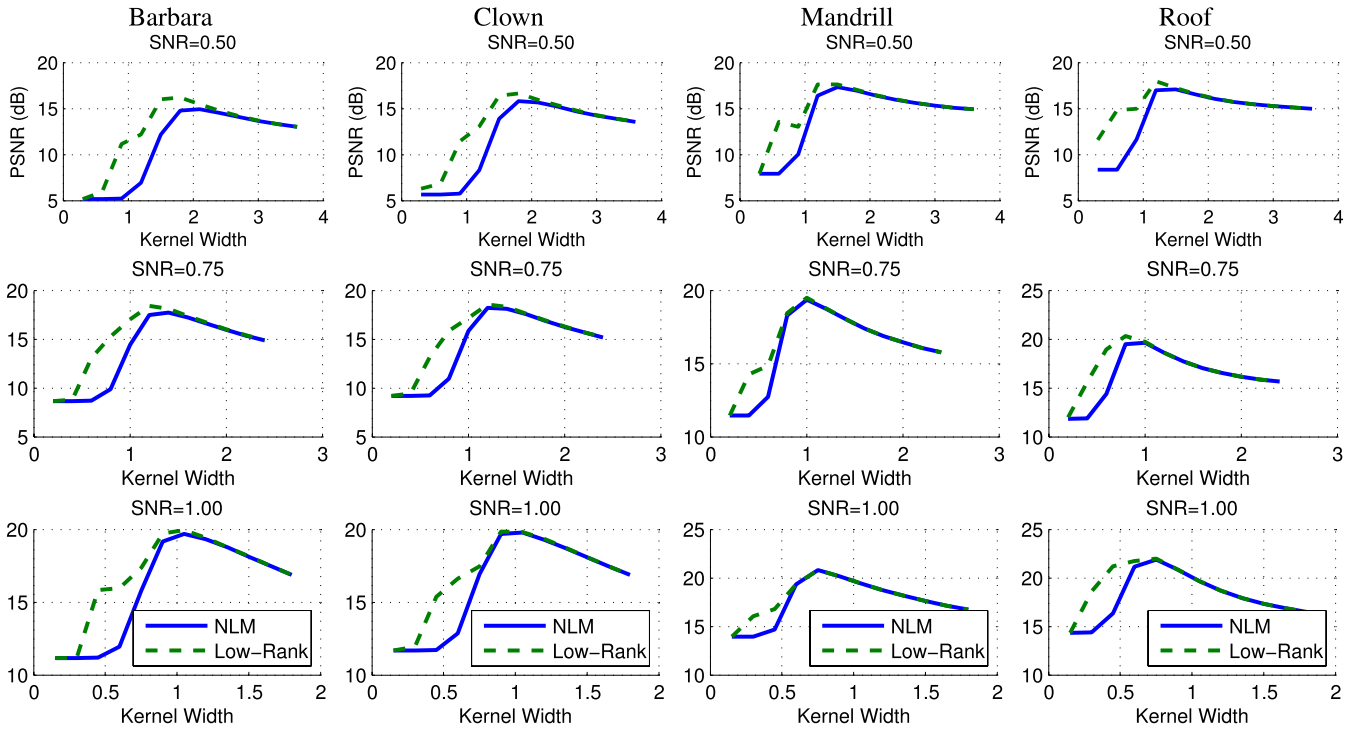


Fig. 4. Using a low-rank operator versus the NLM operator: the PSNR between a clean test image and its denoised version as a function of the kernel width, for different SNR (noise) levels.

In this metric, two similar images, for example an image and a good approximation of it, get high PSNR value, while two different images have low PSNR value.

In addition to the PSNR, which measures the difference between two images, we use the signal-to-noise ratio (SNR) to measure the noise level in a given (single) image. SNR is given by

$$\text{SNR}\,(I_c, \sigma_{noise}) = \frac{\sigma_{I_c}}{\sigma_{noise}},$$

where I_c is the clean image, σ_{I_c} is the standard deviation of the intensity values of I_c and σ_{noise} is the standard deviation of the noise added to I_c. In our experiments we use SNRs of 0.5, 0.75 and 1.

A. Effectiveness of Low-Rank NLM Operators

We explore the improvement achieved by low-rank approximations of the NLM operator compared to the original NLM operator, as a function of the noise level (SNR) of the image and the kernel width of the operator.

For this experiment we used four standard test images (Figure 1 in the supplementary material), resized to 60×60 pixels by bi-cubic interpolation, from which we computed NLM operators for a range of kernel widths, with patch size $p = 5$. For each operator corresponding to a given kernel width, we constructed several low-rank approximations of it using the method of eigenvalues truncation, as given in Algorithm 1 in Appendix B, named NLM-Eig. The low-rank values are given by the set

$$\mathcal{K} = \{1, 5, 10, 15, 20, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 400, 600, 1200, 2000, 3000, 3600\}.$$

For each of the four images, we present in Figure 4 three charts corresponding to three SNR levels. Each chart shows the PSNR between the clean image and its denoised version, as a function of the kernel width, for two operators. The two operators are the NLM operator and its best performing low-rank approximation, that is

$$\arg\max_{k_i \in \mathcal{K}} \{\text{PSNR}(\text{NLM-Eig}(I, p, k_i), I_c\}, \tag{19}$$

where I_c and I are the clean and corrupted images, respectively.

From the results of this experiment we can see that the performance gap between the optimal low-rank operator (19) and the NLM operator can be very large, in favor of the low-rank approximation, for kernel widths that are much smaller than the optimal width. This advantage diminishes when the kernel width approaches its best performing value. In addition, one observes that the performance gain of the low-rank operator naturally diminishes as the SNR increases since less noise is involved.

B. The Cutoff Point

We investigate the effect of the cutoff point on the performance of two denoising methods. For the naive method of truncated eigenvalues (Algorithm 1), the cutoff point corresponds to the number of leading eigenvalues preserved in the low-rank operator. For our method based upon the SB function (Algorithm 3), the cutoff point corresponds to the filter's cutoff parameter ω, which is a value between zero and one (Section III-B). In the experiment of this section, we measure the "PSNR gain", which is the difference between

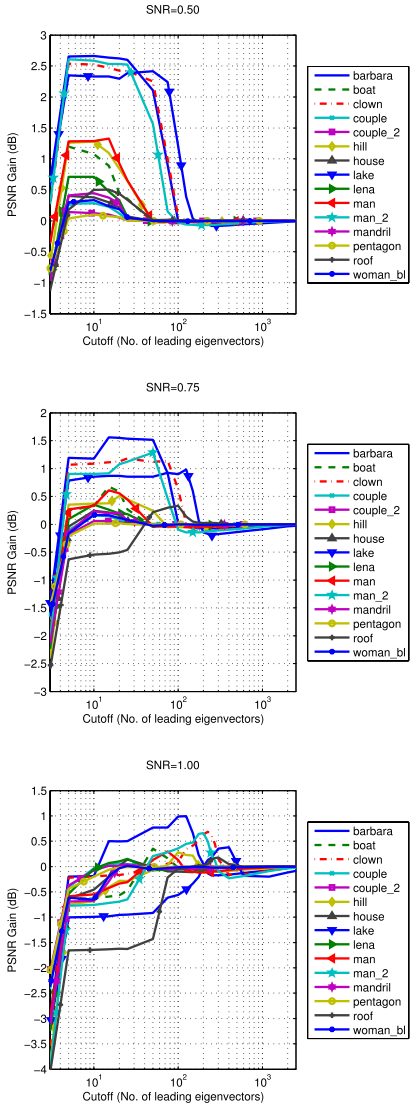


Fig. 5. The differences between the PSNR of the low-rank operator based on truncated eigenvalues and the original NLM operator (PSNR gain), as a function of the cutoff point (number of preserved leading eigenvalues).

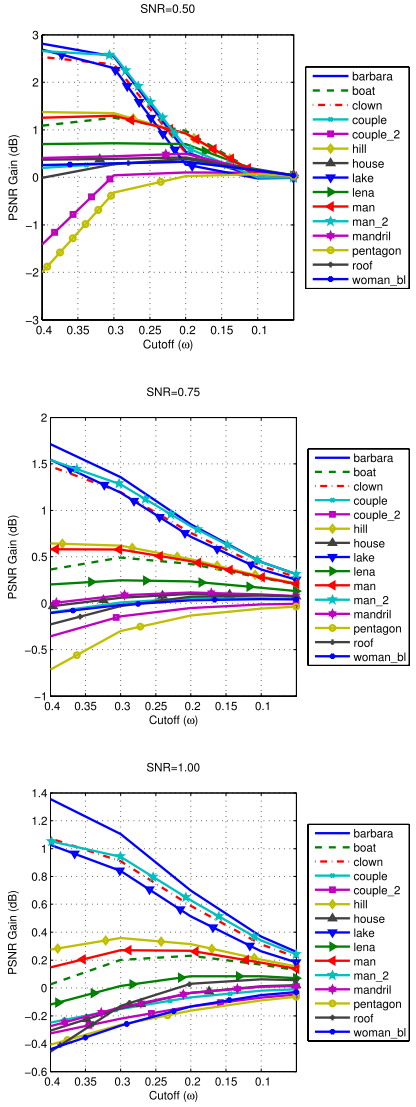


Fig. 6. The differences between the PSNR of the low-rank operator based on the SB function and the original NLM operator (PSNR gain), as a function of the cutoff point (the parameter ω).

the PSNR of the low-rank operator and that of the original NLM operator it has been created from, as a function of the cutoff parameter. The parameters of the original NLM operators are listed in Appendix C.

Figures 5 and 6 show the results of the experiment, per image and noise level, using eigenvalues truncation and our method, respectively. The full set of clean test images is depicted in Figure 2 of the supplementary material file. One can observe that for both methods, the best-performing cutoff, given the SNR, is far from being the same for all images. For cutoffs that are very low (resulting in extremely low-rank operators), the PSNR gain may be negative. For cutoffs that are very high, the PSNR of the low rank operator converges to the PSNR achieved by the original NLM operator, since the modified operator itself in both methods converges to the original NLM operator.

For a different perspective, we show in Figure 7 the mean improvement per noise level for the two methods. Indeed, there is a range of cutoffs that yields an improvement on the dataset for both methods. We can see that the improvement is greatly affected by the noise level. Namely, as the SNR decreases, the improvement increases.

C. Comparison With the NLM Algorithm

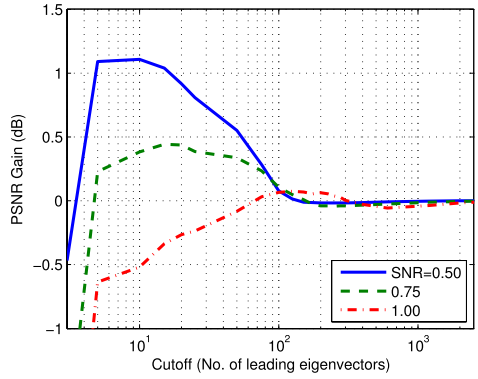
We next demonstrate the advantage of our method over the baseline NLM method. The parameters of all methods are described in Appendix C. The numerical experiments are performed on real images corrupted by synthetically added white Gaussian noise (as described in Section II) of SNR 0.5, 0.75 and 1. The set of clean test images consists of 23 standard images of size about 512×512 pixels each, which are presented in Figures 3–25 in the supplementary material. Henceforth, we refer to our method, given in Algorithm 3 in Appendix B, as "NLM-SB".

The results for the entire set of 23 images are given in Table I. As we can see from the table, as the level of noise increases (lower SNR values), the improvements over the

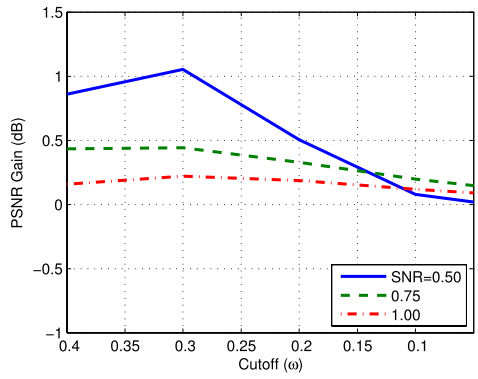
TABLE I

PSNR COMPARISON (IN dB) BETWEEN THE NLM AND NLM-SB ALGORITHMS OVER THREE LEVELS OF NOISE. THE COLUMN "DIFF" CORRESPONDS TO THE DIFFERENCE IN PSNR OF THE TWO METHODS

Noise level	SNR=0.5			SNR=0.75			SNR=1		
Image	NLM	NLM-SB	Diff	NLM	NLM-SB	Diff	NLM	NLM-SB	Diff
flower	17.17	20.8	3.62	21.25	23.74	2.49	23.41	24.94	1.53
girlface	17.71	21.13	3.42	20.6	23.05	2.45	23.38	24.91	1.53
clown	17.6	20.99	3.39	18.72	21.16	2.44	22.61	24.05	1.44
sailboat	17.7	21.02	3.32	21.15	23.57	2.42	20.64	22.01	1.37
pens	15.15	18.43	3.28	21.08	23.1	2.02	23.18	24.44	1.26
tulips	18.19	21.18	2.99	20.71	22.69	1.98	22.26	23.47	1.21
cameraman	16.84	19.67	2.84	20.13	22.05	1.92	23.39	24.55	1.16
goldhill	16.02	18.85	2.83	21.43	23.22	1.8	22.58	23.71	1.13
girl	17.59	20.39	2.8	21.7	23.39	1.69	23.67	24.7	1.03
cat	18.45	20.97	2.52	18.78	20.41	1.64	20.39	21.28	0.89
zelda	19.23	20.84	1.61	22.16	23.14	0.98	24.25	24.89	0.64
boats	17.82	19.37	1.55	21.02	21.85	0.83	23.53	24.02	0.49
crowd	18.84	20.36	1.52	21.75	22.57	0.81	22.84	23.31	0.48
peppers	18.23	19.7	1.47	20.65	21.41	0.76	24.3	24.75	0.45
yacht	15.88	17.28	1.41	21.99	22.58	0.59	22.49	22.94	0.45
houses	17.31	18.64	1.34	18.42	19.01	0.58	20.01	20.37	0.36
airfield	18.31	19.57	1.26	19.8	20.3	0.5	21.47	21.73	0.26
flowers	18.95	19.96	1.01	20.92	21.38	0.46	22.55	22.78	0.23
soccer	18.4	19.26	0.86	20.95	21.18	0.24	22.66	22.75	0.09
lighthouse	19	19.63	0.64	21.26	21.34	0.07	22.82	22.81	-0.01
monarch	21.01	21.46	0.45	22.84	22.77	-0.07	24.6	24.48	-0.13
Lichtenstein	20.51	20.65	0.15	23.13	22.99	-0.14	24.4	24.19	-0.2
couple	20.14	19.97	-0.17	21.95	21.73	-0.22	23.29	23.02	-0.27
Average	18.08	20.00	**1.91**	20.97	22.11	**1.14**	22.81	23.48	**0.66**



(a)



(b)

Fig. 7. The average PSNR improvement over the full set of test images for three levels of noise. (a) results of the method of truncated eigenvalues (Algorithm 1), (b) results of our method, based upon the SB function (Algorithm 3).

original NLM algorithm become more significant. Specifically, we improve the denoising (in average) by almost 2dB for the highest level of noise (SNR=0.5). Two visual examples are given in Figures 8 and 9, corresponding to noise levels of 0.75 and 0.5, respectively. As can be seen in the differences images, the NLM-SB results in superior denoised images. All denoised images of this experiment are given in Figures 3–71 in the supplementary material.

D. Two Stages Denoising Schemes

Iterating the NLM operator, as introduced in [1] and [2], applies denoising gradually over several iterations. Results of this algorithm over our dataset are presented in the supplementary material. A similar approach is presented in [3] where the iterative NLM is based upon gradient descent iterations and the NLM denoising operator is repeatedly reconstructed from the progressively denoised image. Along these lines, Meyer and Shen [13] proposed to compute a second NLM operator from the denoised image and apply it again.

In this section, we adapt Meyer's two stages scheme [13] to use our SB based NLM operator, denoting the resulting algorithm by SB-Meyer. However, care must be taken with this adaptation, as there are a few differences between the algorithm in [13] and the NLM algorithm, as given in Definition 1. First, the implementation of [13] is based on a k-nearest neighbours search for constructing the denoising operators. Instead of constructing the operator $A = D^{-1}W$, where the elements of W are given as in Definition 1, they construct W as $W = 0.5(Z + Z^*)$, where each row i of the matrix Z contains k nonzero elements, which are the k largest values of the set

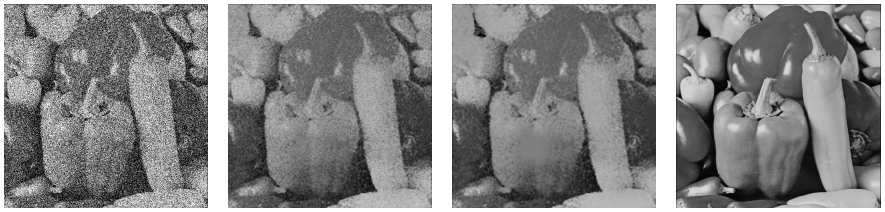
$$\left\{K_h(v(N_i^Y) - v(N_j^Y)) \mid 1 \leq j \leq n\right\}.$$

Second, the way that the denoising operator in [13] is applied is different than in the NLM algorithm. Instead of computing the denoised image as $\hat{x} = Ay$, where each pixel is denoised using only the patch of which it is the center, they average the values of the pixel from all patches in which it is included.

In order to use $f_{\omega,d}^{sb}$ in the scheme [13], we had to modify the way the denoising operator in Meyer's scheme [13] is constructed. In contrast to the NLM operator, the operator in

(a) (b) (c) (d) (e) (f)

Fig. 8. Comparison of denoising performance for the *clown* image. (a) noisy image (SNR = 0.75), (b) image denoised by NLM, (c) image denoised by NLM-SB, (d) clean image, (e) the difference between the clean image and the image denoised by NLM, (f) the difference between the clean image and the image denoised by NLM-SB. The difference images are heat maps with a dynamic range of 0 to 0.2, corresponding to blue (dark) to red (bright) colors. The original images have values in the range of 0 to 1.



(a) (b) (c) (d)

Fig. 9. Comparison of denoising performance for the *peppers* image. (a) noisy image (SNR = 0.5), (b) image denoised by NLM, (c) image denoised by NLM-SB, (d) clean image.

the scheme [13] is not positive definite (and not even non-negative) due to the k-nearest neighbours step in its construction. Since our framework, as given in Section III, requires a non-negative spectrum, we have approximated the denoising operator in Meyer's scheme with a new semi-positive definite matrix, by shifting its spectrum such that the largest negative eigenvalue becomes zero. Then, we normalized its rows such that they will sum to unity. This normalization transforms the largest positive eigenvalue back to unity (see also the proof of Lemma 2 in Appendix A).

The original algorithm of [13] uses parameters given by the authors in their source code, which were tuned only for SNR level of 1. Thus, we report in Table II the results of the algorithm [13] (in the column "Meyer") as well as those of SB-Meyer only for that noise level. The comparison is over 15 test images of size 120×120, shown in Figures 1–2 in the supplementary material.

In addition to SB-Meyer, we have implemented in Algorithm 4 in Appendix B a simpler two-stage scheme employing our operator based on the SB function, which is referred to as NLM-SB2. The parameters used by this algorithm are given in Appendix C. From Table II we see that our NLM-SB outperforms the baseline NLM in terms of PSNR, and that the two stages schemes further improve the resulting PSNR.

For a visual assessment of the above comparison, we provide Figure 10, where a sub-image of the *Mandrill* image is denoised by the various tested algorithms, including the state-of-the-art BM3D [7] (for full comparison with BM3D see the supplementary material). The clean sub-image of the Mandrill is shown in Figure 10a. Its denoised versions are presented in Figures 10c–10f. One can see that SB-Meyer retained the texture much better than BM3D, and it also contains less artifacts than Meyer's original scheme (the finding regarding texture preservation is consistent with the conclusions in [13]).

TABLE II

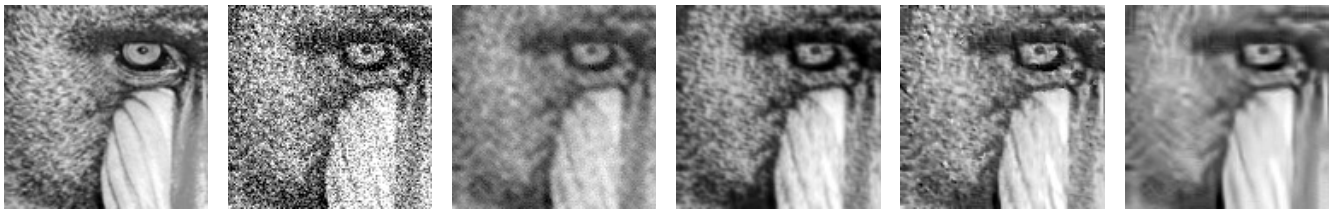
COMPARING THE PSNR OF THE NLM ALGORITHMS FOR IMAGES WITH SNR = 1

Image	NLM	NLM-SB	NLM-SB2	SB-Meyer	Meyer
barbara	20.84	21.94	22.11	23.18	20.30
boat	21.24	21.44	22.01	22.72	21.33
clown	20.51	21.42	21.08	22.23	20.14
couple	21.66	21.51	22.54	23.50	23.06
couple 2	22.03	21.81	23.17	24.46	24.61
hill	21.27	21.63	22.20	22.57	21.21
house	22.83	22.68	23.89	24.82	26.19
lake	19.73	20.57	19.24	21.42	20.78
lena	21.73	21.75	22.39	23.12	22.44
man	21.18	21.45	21.70	22.04	21.16
man 2	20.94	21.88	21.47	22.08	20.01
mandrill	21.08	20.94	21.45	22.37	21.50
pentagon	20.84	20.57	21.61	23.50	24.22
roof	21.97	21.84	22.66	22.57	24.42
woman blonde	21.87	21.60	21.89	23.18	23.64
Average	21.31	21.53	21.96	22.92	22.33

E. Features Preservation

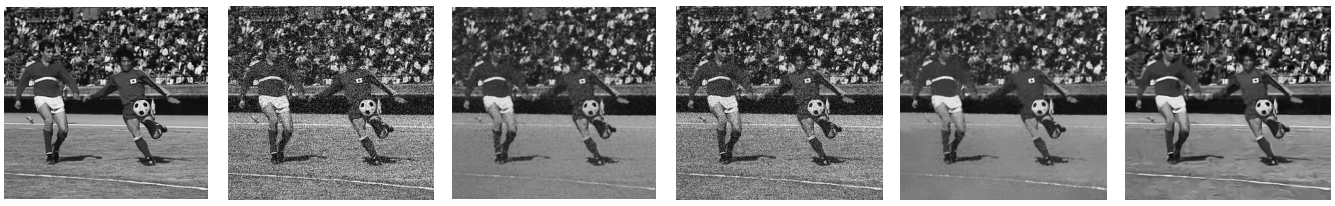
We conclude the experiments with a few comparisons of NLM, NLM-SB, iterated-NLM [1], and BM3D [7]. The latter algorithm is considered state-of-the-art and often exhibits superior performance in terms of both runtime and PSNR. In particular, on a standard 3.4 GHz Intel Core i7 processor, BM3D has an average runtime of 6 seconds for denoising images of size 512×512 where the current implementation of NLM-SB requires an average runtime of 120 seconds. The average PSNR also leans in favour of BM3D, with 23.22dB for BM3D and 20.21dB for NLM-SB, for images with noise level of SNR = 1 (the full table of PSNR results is available in the supplementary material). Nevertheless, a visual examination of images denoised by both algorithms shows that in terms of features preservation, NLM-SB is a fair competitor.

In Figures 11 and 12 we present two visual comparisons of NLM, NLM-SB, iterated-NLM [1], and BM3D [7]. In each figure we present the clean image, its noisy version, and the output of the various denoising algorithms. In the "soccer"



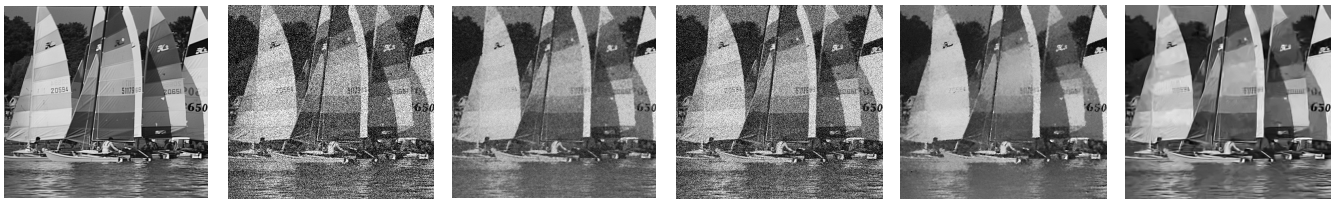
(a) (b) (c) (d) (e) (f)

Fig. 10. Denoised examples of the Mandril image for SNR=1. (a) Clean. (b) Noisy. (c) NLM. (d) SB-Meyer. (e) Meyer. (f) BM3D.



(a) (b) (c) (d) (e) (f)

Fig. 11. A comparison of denoising the *soccer* image with SNR=1. (a) Clean. (b) Noisy. (c) NLM. (d) Iterated-NLM. (e) NLM-SB. (f) BM3D.



(a) (b) (c) (d) (e) (f)

Fig. 12. A comparison of denoising the *yacht* image with SNR=1. (a) Clean. (b) Noisy. (c) NLM. (d) Iterated-NLM. (e) NLM-SB. (f) BM3D.

(a) (b) (c) (d)

Fig. 13. Zoom-in of the denoising of *zelda* with SNR=0.75 by NLM-SB and BM3D. (a) Clean. (b) Noisy. (c) NLM-SB. (d) BM3D.

image in Figure 11, we can see that some fine details (such as the hands of the players) appear closer to their original shapes in the NLM-SB denoised image compared to the other denoised images. In the "yacht" image in Figure 12, one can notice a similar phenomenon, as for example, some of the numbers on the boats sails have been preserved in the NLM-SB denoised image. To further assess the visual differences between NLM-SB and BM3D we present in Figure 13 a zoom-in of the denoised images of the noisy "zelda" image at SNR $= 0.75$. The full set of comparisons, both visual and numerical, is presented in the supplementary material.

VI. Summary

In this paper, we have investigated the idea of improving a Non-Local Means operator by manipulating its spectrum. We have shown a method to do so without computing explicitly neither the eigenvectors of the original operator nor the matrix of the modified operator.

Our method operates by applying a filtering function to the original Non-Local Means operator. To that end, we derive sufficient conditions for such filtering functions and an efficient procedure for their application. We also show the connection between spectral shaping of the operator and the application of a matrix function on the operator. In the implementation of our approach, we demonstrate the well-known efficiency of Chebyshev polynomials for matrix functions. Moreover, a bound on the approximation error of the truncated Chebyshev expansion for the class of Non-Local Means matrices is proved.

The numerical experiments demonstrate the properties and advantages of constructing a low-rank approximation of a Non-Local Means operator. It is also demonstrated that

Algorithm 1 NLM-Eig Denoising Scheme. Here, in the Notation of Lemma 2, $\mathrm{EIG}(A,k) = Q^{-1}\Lambda_k Q$, Where $\Lambda_k = \mathrm{diag}\,(\lambda_1,\ldots,\lambda_k,0,\ldots,0)$

Input: Noisy image Y, patch size p, kernel width h, matrix approximation rank k.
Output: Denoised image $\hat{X}$.
procedure NLM-EIG(Y,p,h,k)
  $A \leftarrow \mathrm{NLM}_{p,h}(Y)$ ▷ Construct a NLM operator for the image.
  $B \leftarrow \mathrm{EIG}(A,k)$ ▷ Compute a low-rank approximation of the NLM operator.
  $y \leftarrow \mathrm{COL}(Y)$ ▷ Convert the image Y to a vector.
  $\hat{x} \leftarrow By$ ▷ Compute the output image using the low-rank approximation of A.
  return $\mathrm{IMAGE}(\hat{x})$ ▷ Reshape $\hat{x}$ as an image.
end procedure

a further PSNR improvement can be achieved by incorporating our approach in a two-stage scheme, as suggested in [13].

APPENDIX A
PROOF OF LEMMA 2

Lemma 2 summarizes known properties of NLM operators. We provide its proof for the self-containedness of the paper.

Proof: The NLM operator of Definition 1 is in general not symmetric, but it is conjugated via $D^{\frac{1}{2}}$ to the symmetric matrix $S = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, namely $A = D^{-1/2}SD^{1/2}$. Therefore, the NLM operator is diagonalizable and has the same eigenvalues as S. On other hand, K_h is the Gaussian kernel function, which implies that W is a symmetric positive definite matrix. Since S is obtained from W by multiplication by $D^{-1/2}$ on both sides, by Sylvester's law of inertia, the eigenvalues of S are positive as well. Thus, since S and A are conjugated, all eigenvalues of A are also positive.

Since A is element-wise positive, it follows from the Perron-Frobenius theorem that

$$\min_{1\le i\le n}\sum_{j=1}^{n} A_{ij} \le \left|\max_{1\le i\le n}\lambda_i\right| \le \max_{1\le i\le n}\sum_{j=1}^{n} A_{ij}.$$

Moreover, since for any $1 \le i \le n$, $\sum_j A_{ij} = 1$, we get that $|\max_{1\le i\le n}\lambda_i| = 1$. Thus, from the positivity of the eigenvalues we conclude that $\lambda_1 = 1$ and $\lambda_i \in (0,1]$, $1 \le i \le n$. □

APPENDIX B
ALGORITHMS

This appendix contains the pseudocode for the algorithms described in the paper. In these algorithms we denote by CHEB-COEF(f, N) the set of the first $N+1$ Chebyshev coefficients calculated via (8) and (11).

Algorithm 2 Evaluate the Product of a Truncated Matrix Chebyshev Expansion (10) by a Vector

Input: Matrix A, vector y, vector of coefficients c of length N.
Output: The vector $\sum_{k=1}^{N} c_k T_k(A)\,y$.
procedure CLENSHAWMATVEC(A, y, c)
  $N \leftarrow \#c$ ▷ # stands for the number of elements.
  $T \leftarrow 2\cdot A - I_n$
  $d \leftarrow 0$
  $dd \leftarrow 0$
  for $i \leftarrow N\ downto\ 2$ **do**
    $temp \leftarrow d$
    $d \leftarrow 2\cdot T\cdot d - dd + c_i\cdot y$
    $dd \leftarrow temp$
  end for
  return $T\cdot d - dd + 0.5\cdot c_1\cdot y$
end procedure

Algorithm 3 NLM-SB Denoising Scheme

Input: Noisy image Y, patch size p, kernel width h, cutoff and order parameters ω and d of $f^{sb}_{\omega,d}$, number of Chebychev coefficients N.
Output: Denoised image $\hat{x}$.
procedure NLM-SB(Y,p,h,ω,d,N)
  $\{\alpha_j\} \leftarrow \mathrm{CHEB\text{-}COEF}(f^{sb}_{\omega,d}, N)$ ▷ Compute the Chebychev coefficients for $f^{sb}_{\omega,d}$.
  $A \leftarrow \mathrm{NLM}_{p,h}(Y)$ ▷ Construct a NLM operator for the image.
  $x \leftarrow \mathrm{COL}(Y)$ ▷ Convert the image Y to a vector.
  $\hat{x} \leftarrow \mathrm{ClenshawMatVec}(A,\{\alpha_j\},x)$ ▷ Algorithm 2.
  return $\mathrm{IMAGE}(\hat{x})$ ▷ Reshape $\hat{x}$ as an image.
end procedure

Algorithm 4 NLM-SB2 Two-Stage Denoising Scheme

Input: Noisy image Y, patch size p, kernel widths h_1 and h_2, mixing weight $\gamma \in [0,1]$, cutoff and order parameters ω_1,ω_2 and d_1,d_2 of $f^{sb}_{\omega,d}$, number of Chebychev coefficients N.
Output: Denoised image $X^{(3)}$.
procedure NLM-SB2($Y,p,\gamma,h_1,h_2,\omega_1,\omega_2,d_1,d_2$)
  $X^{(1)} \leftarrow \text{NLM-SB}(Y,p,h_1,\omega_1,d_1,N)$ ▷ Algorithm 3.
  $X^{(2)} \leftarrow (1-\gamma)X^{(1)} + \gamma Y$ ▷ Mix the esimated image with the original one.
  $X^{(3)} \leftarrow \text{NLM-SB}(X^{(2)},p,h_2,\omega_2,d_2,N)$ ▷ Denoise again.
  return $X^{(3)}$
end procedure

APPENDIX C
PARAMETERS OF THE ALGORITHMS

In this appendix we provide a full list of parameters for the algorithms used in Section V.

The original NLM operators are constructed with patch size $p = 5$ and kernel widths h of 1.5, 1.0 and 0.5 for the different SNR values 0.5, 0.75 and 1, respectively. These values were

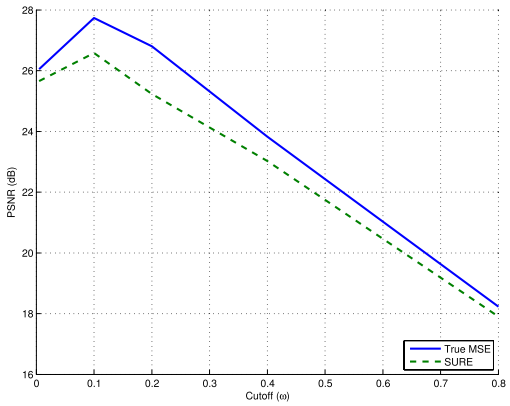


Fig. 14. Performance of NLM-SB algorithm as evaluated by SURE, for different values of the bandwidth parameter ω, compared to the actual MSE.

TABLE III
THE PARAMETERS FOR THE NLM-SB METHOD (ALGORITHM 3)

	SNR=0.5	SNR=0.75	SNR=1
p	5	5	5
h	1.5	1.05	0.5
ω	0.3	0.3	0.3
d	15	4	4
N	150	150	150

TABLE IV
THE PARAMETERS OF THE SB-MEYER SCHEME FOR SNR = 1

v_1	v_2	p_1	p_2	h_1	h_2	ω_1	ω_2	d_1	d_2	γ	N
200	200	7	3	1	1	0.6	0.4	50	16	0.2	150

TABLE V
THE PARAMETERS OF THE NLM-SB2 SCHEME, GIVEN IN ALGORITHM 4

SNR	p	h_1	h_2	ω_1	ω_2	d_1	d_2	γ	N
0.5	5	1.5	1	0.3	0.3	50	50	0.5	150
0.75	5	1.05	0.35	0.3	0.3	15	15	0.15	150
1	5	0.5	0.3	0.3	0.3	4	4	0.15	150

chosen based on the experiments in Section V-A, as the kernel widths which result in the highest PSNR, on average. These are also the parameters for the truncated eigenvalues methods (Algorithm 1), used for comparison in Section V.

The parameters for our method (Algorithm 3) are given in Table III. These parameters remain fixed (in each noise level) for the experiments of Section V (excluding ω which varies on Subsections V-B). Note that the parameters d (degree of the filter) and N (length of the expansion) are related since the derivative of the filter grows with d. As a result, more expansion coefficients (larger N) are required for a given truncation error. Nevertheless, for small enough d (like in the case of Section V), a fixed N of 150 is sufficient. In general, it is also possible to estimate a priori the required N, see e.g., [21]. We set the parameter d by trying several values on a few test images.

As seen in Subsection V-B, the optimal bandwidth parameter ω is image-dependent. In the experiments of Subsections V-C–V-D we kept this value fixed and found that the performance of Algorithm 3 is satisfactory. Nevertheless, this parameter, as well as the others, can be tuned to further improve the results. This can be done, for example, by estimating the minimal squares error (MSE), see e.g. [26], or by tuning the parameters of the algorithms to minimize the entropy which measures the similarity between patches, as done in [1] and [2]. Another promising method for estimating the parameters is by using the SURE parameters estimation [19]. We numerically tested the SURE estimation, on a 100×100 pixels sub-image of the cameraman image. In this test we compare the SURE estimated PSNR to the real PSNR, for different values of the bandwidth parameter ω. The results, as appear in Figure 14, show high correlation between the real and estimated PSNR, which makes the SURE estimation reliable in this case.

The parameters of the SB-Meyer method are given in Table IV. Our two stage method, referred to as NLM-SB2, is presented in Algorithm 4. Its parameters are given in Table V.

REFERENCES

[1] S. P. Awate and R. T. Whitaker, "Higher-order image statistics for unsupervised, information-theoretic, adaptive, image filtering," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Jun. 2005, pp. 44–51.

[2] S. P. Awate and R. T. Whitaker, "Unsupervised, information-theoretic, adaptive image filtering for image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 364–376, Mar. 2006.

[3] T. Brox, O. Kleinschmidt, and D. Cremers, "Efficient nonlocal means for denoising of textural patterns," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1083–1092, Jul. 2008.

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

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

[6] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," *Wavelets and Statistics*. New York, NY, USA: Springer, 1995.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "BM3D image denoising with shape-adaptive principal component analysis," in *Proc. SPARS-Signal Process. Adapt. Sparse Struct. Represent.*, 2009.

[8] W. H. Press, S. A. Teukolsky, W. Vetterling, and B. P. Flannery, *Numerical Recipes in C*. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 3. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2012.

[10] N. J. Higham, "Stable iterations for the matrix square root," *Numer. Algorithms*, vol. 15, no. 2, pp. 227–242, 1997.

[11] N. J. Higham, *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: SIAM, 2008.

[12] A. Hyvärinen, J. Hurri, and P. O. Hoyer, *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, vol. 39. London, U.K.: Springer, 2009.

[13] F. G. Meyer and X. Shen, "Perturbation of the eigenvectors of the graph Laplacian: Application to image denoising," *Appl. Comput. Harmon. Anal.*, vol. 36, no. 2, pp. 326–334, 2014.

[14] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 106–128, Jan. 2013.

[15] G. Motta, E. Ordentlich, I. Ramirez, G. Seroussi, and M. J. Weinberger, "The iDUDE framework for grayscale image denoising," *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 1–21, Jan. 2011.

[16] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, vol. 5. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

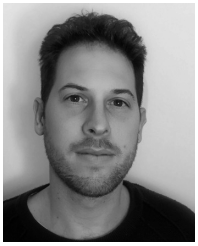
[17] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, Jul. 1990.

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

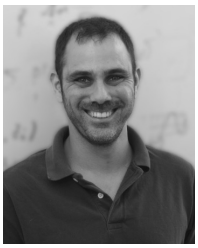
[19] S. Ramani, T. Blu, and M. Unser, "Monte-Carlo sure: A black-box optimization of regularization parameters for general denoising algorithms," *IEEE Trans. Image Process.*, vol. 17, no. 9, pp. 1540–1554, Sep. 2008.
[20] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D, Nonlinear Phenomena*, vol. 60, nos. 1–4, pp. 259–268, 1992.
[21] N. Sharon and Y. Shkolnisky. (2015). "Matrix functions based on Chebychev expansion." [Online]. Available: http://arxiv.org/abs/1507.03917
[22] A. Singer, Y. Shkolnisky, and B. Nadler, "Diffusion interpretation of nonlocal neighborhood filters for signal denoising," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 118–139, 2009.
[23] R. Szeliski *et al.*, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1068–1080, Jun. 2008.
[24] A. D. Szlam, M. Maggioni, and R. R. Coifman, "Regularization on graphs with function-adapted diffusion processes," *J. Mach. Learn. Res.*, vol. 9, pp. 1711–1739, Jun. 2008.
[25] H. Tal-Ezer, "Polynomial approximation of functions of matrices and applications," *J. Sci. Comput.*, vol. 4, no. 1, pp. 25–60, 1989.
[26] H. Talebi and P. Milanfar, "Global image denoising," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 755–768, Feb. 2014.
[27] K. M. Taylor and F. G. Meyer, "A random walk on image patches," *SIAM J. Imag. Sci.*, vol. 5, no. 2, pp. 688–725, 2012.
[28] L. N. Trefethen, *Spectral Methods in MATLAB*, vol. 10. Philadelphia, PA, USA: SIAM, 2000.
[29] L. N. Trefethen, "Is Gauss quadrature better than Clenshaw–Curtis?" *SIAM Rev.*, vol. 50, no. 1, pp. 67–87, 2008.
[30] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 977–984.

Victor May was born in Moscow in 1983. He received the B.Sc. degree in computer science and mathematics from Bar-Ilan University, Ramat Gan, Israel, in 2005, and the M.Sc. degree in applied mathematics from Tel Aviv University, Tel Aviv, Israel, in 2015. From 2005 to 2011, he was developing photogrammetry and computer vision systems for the Israel Ministry of Defense. From 2012 to 2015, he had been with start-up company Extreme Reality, where he worked on video motion capture technology. He is currently an Algorithm Engineer with Taboola, Tel Aviv, where he is working on online content recommendation systems.

Yosi Keller received the B.Sc. degree from the Technion–Israel Institute of Technology, Haifa, in 1994, and the M.Sc. and Ph.D. degrees from Tel Aviv University, Tel Aviv, in 1998 and 2003, respectively, all electrical engineering. From 2003 to 2006, he was a Gibbs Assistant Professor with the Department of Mathematics, Yale University. He is currently an Associate Professor with the Faculty of Engineering, Bar-Ilan University, Israel. His research interests relate to manifold learning, graph-based algorithms, signal processing, and deep learning.



Nir Sharon received the B.A. degree in computer science from the Open University of Israel, in 2008, and the M.Sc. and Ph.D. degrees in applied mathematics from Tel Aviv University, Tel Aviv, in 2010 and 2015, respectively. Since 2015, he has been a Post-Doctoral Research Associate with the Program of Applied and Computational Mathematics, Princeton University. His research interests include approximation theory, computational harmonic analysis, and data analysis.



Yoel Shkolnisky received the B.Sc. degree in mathematics and computer science and the M.Sc. and Ph.D. degrees in computer science from Tel Aviv University, Tel Aviv, Israel, in 1996, 2001, and 2005, respectively. From 2005 to 2008, he was a Gibbs Assistant Professor of Applied Mathematics with the Department of Mathematics, Yale University, New Haven, CT. Since 2009, he has been with the Department of Applied Mathematics, School of Mathematical Sciences, Tel Aviv University. His research interests include computational harmonic analysis, scientific computing, and data analysis.