3-1-2021

Color-compressive bilateral filter and nonlocal means for high-dimensional images

Christina Karam
University of Dayton

Kenjiro Sugimoto
Waseda University

Keigo Hirakawa
University of Dayton

Color-compressive bilateral filter and nonlocal means for high-dimensional images

Christina Karam,[a,*] Kenjiro Sugimoto,[b] and Keigo Hirakawa[c]
[a]University of Dayton Research Institute, Sensors and Systems Division, Dayton, Ohio, United States
[b]Waseda University, Graduate School of Information, Production and Systems IPS, Fukuoka, Japan
[c]University of Dayton, Department of Electrical and Computer Engineering, Dayton, Ohio, United States

Abstract. We propose accelerated implementations of bilateral filter (BF) and nonlocal means (NLM) called color-compressive bilateral filter (CCBF) and color-compressive nonlocal means (CCNLM). CCBF and CCNLM are random filters, whose Monte-Carlo averaged output images are identical to the output images of conventional BF and NLM, respectively. However, CCBF and CCNLM are considerably faster because the spatial processing of multiple color channels are combined into a single random filtering process. This implies that the complexity of CCBF and CCNLM is less sensitive to color dimension (e.g., hyperspectral images) relatively to other BF and NLM methods. We experimentally verified that the execution time of CCBF and CCNLM are faster than the existing "fast" implementations of BF and NLM, respectively.
 [DOI: 10.1117/1.JEI.30.2.023001]

Keywords: bilateral filter; nonlocal means; random filters.

Paper 200778 received Nov. 28, 2020; accepted for publication Feb. 15, 2021; published online Mar. 4, 2021.

1 Introduction

Bilateral filter (BF) is an image smoothing filtering technique introduced by Tomasi and Maduchi.[1–3] Image features such as textures and edges are preserved by BF due to the adaptive weighting of the spatial and range kernels. The former assigns higher weights to the spatially nearby pixels, while the latter gives more importance to the pixels of similar appearance. Despite the proven usefulness of BF in many image processing and computer vision applications—as evidenced by prior examples in denoising,[4] demosaicking,[5] tone mapping,[6] stereo matching,[7] and segmentation[8]—the complexity of BF remains a limiting factor.

Similarly, nonlocal means (NLM) is a generalization of the BF that has shown advantages in denoising.[9] It replaces the notion of pixel-to-pixel similarity[9] in a range kernel with a block-to-block similarity. While edges and textures are better preserved by the block extension of the range kernel, the computational complexity increases significantly. Although there are arguably superior denoising techniques available,[10] NLM remains popular today owing to its intuitiveness, filtering quality, and suitability for working with color images.

Significant work has followed BF and NLM to accelerate them.[11–22] The key technique employed to reduce complexity is to replace the range kernel computation by an efficient, approximately equivalent convolutional filtering process. The speedup in the resultant "fast" implementations is achieved by ensuring that the required number of convolutions are invariant to window size, quantization step size, color dimension of the edge image, and NLM block size.

One main challenge that has not been resolved by any of the previous "fast" implementations is that the complexities of BF and NLM grow linearly or exponentially with respect to the color dimension N of the filter image.[11–20] That is, although previous methods successfully addressed the scenario that the edge image is a large dimension (including our previous method called

*Address all correspondence to Christina Karam, ckaram1@udayton.edu

1017-9909/2021/$28.00

SBF/SNLM[20]), these BF/NLM implementations are nevertheless slow when filtering RGB ($N = 3$) and hyperspectral images ($N \gg 3$). In this paper, we propose a technique to accelerate BF and NLM further by combining the convolutional filtering of multiple color channels into a single random filtering process. The resultant color-compressive bilateral filter (CCBF) and color-compressive nonlocal means (CCNLM) implementations require far fewer convolutions than the existing methods when the color dimension of the filtering image is large. That is, they allow the processing of high-dimensional images ($N \gg 3$ such as hyperspectral images) with only a modest complexity.

Another way to speed up BF and NLM is via GPU acceleration, involving massive parallization. These methods do not fundamentally reduce the computational complexity of BF and NLM (unlike CCBF and CCNLM), but execute them faster. Proposed CCBF and CCNLM are also compatible with GPU parallelization, in fact, meaning further speedup can be expected by implementing parallel versions of CCBF and CCNLM.

This paper is organized as follows. In Sec. 2, we briefly review of state-of-the-art fast algorithms, including our previous work SBF/SNLM.[20] In Sec. 3, we develop the proposed CCBF and CCNLM methods, whose complexity is analyzed in Sec. 4. We demonstrate the speedup in Sec. 5 before making concluding remarks in Sec. 6.

We note that the simplified versions of Lemmas 1 and 2, Theorem 1, and Corollary 1 have appeared in our preliminary work.[23] In this paper, more general versions are rederived to accommodate the NLM weights in Eq. (5) (subsequently used to derive the CCNLM results in Lemma 3 and Theorem 2). It is also worth emphasizing that the work presented in this paper is not meant to improve the filtering qualities of BF or NLM. Rather, CCBF and CCNLM concern the complexity issues of BF and NLM, particularly for high-dimensional data.

2 Background and Related Works

2.1 *Bilateral Filter and Nonlocal Means*

Let $\mathbf{u}:\mathbb{Z}^2 \to \mathbb{R}^N$ be the input filter image, where $\mathbf{u}(\mathbf{i}) \in \mathbb{R}^N$ is a color vector at the pixel position $\mathbf{i} \in \mathbb{Z}^2$. The BF yields the output image $\mathbf{x}:\mathbb{Z}^2 \to \mathbb{R}^N$ defined by the relation:

$$\mathbf{x}\{\mathbf{u}\}(\mathbf{i}) \coloneqq \frac{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))\cdot\mathbf{u}(\mathbf{j})}{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}, \tag{1}$$

where the spatial kernel $\rho:\mathbb{Z}^2 \to \mathbb{R}$ and range kernel $\delta:\mathbb{R}^N \to \mathbb{R}$ are defined as

$$\begin{aligned}\rho(\mathbf{i}-\mathbf{j}) &\coloneqq \exp\left(-\frac{\|\mathbf{i}-\mathbf{j}\|^2}{2\gamma^2}\right)\\ \delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j})) &\coloneqq \exp\left(-\frac{(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))^T\Phi^{-1}(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}{2}\right).\end{aligned} \tag{2}$$

Here, $\|\cdot\|$ denotes the ℓ^2 norm in $\mathbb{Z}^2$, and $\gamma^2 \in \mathbb{R}$ and $\Phi \in \mathbb{R}^{N\times N}$ are smoothing parameters. Intuitively, BF preserves edges by ensuring kernels ρ and δ are small when $\|\mathbf{i}-\mathbf{j}\|$ and/or $(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))^T\Phi^{-1}(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))$ are large. Although smoothing parameter Φ in most cases is a constant diagonal matrix of the form

$$\Phi = \begin{bmatrix}\phi^2 & & \\ & \ddots & \\ & & \phi^2\end{bmatrix}, \tag{3}$$

which gives equal importance to each color component, we use the general form of Φ later in this paper. Cross-color correlation can be considered by introducing off-diagonal elements in Φ.

Recall that the bilateral pixel range kernel determines the weight of the linear combination in Eq. (1) based on the pixel-to-pixel similarity of $\mathbf{i}$ and $\mathbf{j}$.[1–3] The NLM generalizes this by computing the linear weight based on the similarity of pixel blocks centered at $\mathbf{i}$ and $\mathbf{j}$. Define $\mathbf{y}:\mathbb{Z}^2 \to \mathbb{R}^2$ be the result of filtering the input image $\mathbf{u}$ using the relation:

$$\mathbf{y}\{\mathbf{u}\}(\mathbf{i}) := \frac{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\eta\{\mathbf{u}\}(\mathbf{i},\mathbf{j})\cdot\mathbf{u}(\mathbf{j})}{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\eta\{\mathbf{u}\}(\mathbf{i},\mathbf{j})}, \tag{4}$$

where NLM range kernel $\eta:\mathbb{Z}^2\times\mathbb{Z}^2\to\mathbb{R}$ is defined as

$$\eta\{\mathbf{u}\}(\mathbf{i},\mathbf{j}) := \exp\left(-\sum_{\boldsymbol{\omega}\in\Omega}\frac{(\mathbf{u}(\mathbf{i}-\boldsymbol{\omega})-\mathbf{u}(\mathbf{j}-\boldsymbol{\omega}))^T\Phi_{\boldsymbol{\omega}}^{-1}(\mathbf{u}(\mathbf{i}-\boldsymbol{\omega})-\mathbf{u}(\mathbf{j}-\boldsymbol{\omega}))}{2}\right), \tag{5}$$

and $\Omega\subset\mathbb{Z}^2$ indicates the pixel indexes in the $B\times B$ block. In other words, pixel similarity of $\mathbf{u}(\mathbf{i}-\boldsymbol{\omega})$ and $s(\mathbf{j}-\boldsymbol{\omega})$ for all $\boldsymbol{\omega}\in\Omega$ are considered jointly to determine the range weight $\eta\{\mathbf{u}\}(\mathbf{i},\mathbf{j})$. The usual choice for the smoothing parameter $\Phi_{\boldsymbol{\omega}}\in\mathbb{R}^{N\times N}$ is a constant diagonal matrix:

$$\Phi_{\boldsymbol{\omega}} = \begin{bmatrix} \phi_{\boldsymbol{\omega}}^2 & & \\ & \ddots & \\ & & \phi_{\boldsymbol{\omega}}^2 \end{bmatrix}, \tag{6}$$

but within-block weights $\phi_{\boldsymbol{\omega}}^2$ can be varied optionally to increase the influence of the pixels at the center of the block.

The complexities of the BF and NLM are reported in Tables 1 and 2, respectively. The bottleneck is highlighted in bold. Due to the fact that the spatial kernel $\rho(\mathbf{i}-\mathbf{j})$ decays quickly relative

Table 1 Complexity of bilateral filtering implementations. The bottlenecks are shown in bold. N =# color/spectrum of filtering image, W =window size, T =# quantization steps, K =# summations[12] or # clustering,[16] M =# Monte-Carlo draws. Expected per-pixel costs of convolution and clustering are of order $\mathcal{O}(1)$ and $\mathcal{O}(NKM)$, respectively. Table extended.[20]

	Per pixel					Per image	
	Multiply	Add	Divide	Exp sin/cos	Memory	Conv	Cluster
Naive BF[1–3]	$(\mathbf{2N}+\mathbf{2})\mathbf{W^2}$	$(2N-1)W^2+(W^2-1)(N+1)$	N	W^2	$1+2N$	0	0
Paris[11]	$(N+2)T^N$	$(N+1)T^N$	N	T^N	$\mathbf{NT^N}$	2	0
Chaudhury[12]	$(3N+1)K^N$	NK^N+2K^{N-1}	N	K^N	$1+2N$	$(\mathbf{N}+\mathbf{1})\mathbf{K^N}$	0
Sugimoto[13]							
Deng[14]	$(3N+1)K^N$	$NK^N+2K^{N-1}+N$	N	$2K^N$	$1+2N$	$\mathbf{K^N}$	0
Karam[20]	$(5N+2)M$	$2NM+(N+1)(M-1)$	N	$2M$	$1+2N$	$(\mathbf{2N}+\mathbf{2})\mathbf{M}$	0
Sugimoto[15]	$(N+1)K$	$KN+2(K-1)$	$N+1$	K	$1+2N$	$(\mathbf{N}+\mathbf{1})\mathbf{K}$	$\mathbf{K}$
Nair[16]	NK	NK	N	K	$1+2N$	$(\mathbf{N}+\mathbf{1})\mathbf{K}$	$\mathbf{K}$
CCBF (proposed)[23]	$(2N+4)M$	$(N+1)M+(N+1)(M-1)+N$	N	$2M$	$1+2N$	$\mathbf{2M}$	0

to increasing $\|\mathbf{i}-\mathbf{j}\|^2$, it is common to limit the summation in Eq. (4) to a spatial neighborhood of the window size $W \times W$. Hence, the overall per-pixel complexity is $\mathcal{O}(W^2N)$ for BF and $\mathcal{O}(W^2B^2N)$ for NLM.

2.2 *Prior Work on Accelerated Filters*

To date, various techniques have been introduced to replace the range kernels $\delta:\mathbb{R}^N \to \mathbb{R}$ and $\eta:\mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{R}$ by efficient, approximately equivalent convolutional filtering processes. The techniques and their complexities are shown in Tables 1 and 2. Early efforts to accelerate BF and NLM have largely focused on developing implementations whose complexities are invariant to the window size W^2 and block size B^2. The dependence of BF and NLM complexity on W^2 and B^2 is unattractive because the resolutions of modern imaging devices are increasing rapidly—a larger window/block neighborhood around the pixel $\mathbf{i}$ is needed to represent the same underlying image feature near $\mathbf{i}$. Indeed, the complexity of "fast" BF implementations[11–16] shown in Table 1 are constant with respect to W; and "fast" NLM implementation of Goosens[19] as shown in Table 2 is invariant to block size B.

To process high-dimensional images in a reasonable amount of time, however, the implementation[12–14] complexity of $\mathcal{O}(NC^N)$ and $\mathcal{O}(C^N)$ for some constant C is unacceptably large unless color dimension N is 1 (i.e., $u:\mathbb{Z}^2 \to \mathbb{R}$ is a grayscale image). This fact is evident in Fig. 1, where the number of required convolutions increases rapidly with the color dimension. More recent treatments of BF (including SBF) further reduced the complexity (as determined by the number of convolution operators) to $\mathcal{O}(NK)$, where K is the number of basis summations

Table 2 Complexity analysis of NLM implementations. Complexity bottleneck is marked in bold. N =# color/spectrum of filtering image, W =window size, B =block size, Q = percentage of pixels kept in a window, M =# Monte-Carlo draws. Expected per-pixel cost of convolution is of order $\mathcal{O}(1)$. Table extended.[20]

	Per pixel						Per image	
	Multiply	Add	Divide	Exp sin/cos	Ineq	Sqrt	Conv/FFT	Sqrt
Naive NLM[9]	$(\mathbf{B^2N}+\mathbf{2}+\mathbf{N})\mathbf{W^2}$	$(2B^2N-1)W^2+(W^2-1)(N+1)$	N	W^2	0	0	0	0
BNLM	$(\mathbf{2B^2N}+\mathbf{2})\mathbf{W^2}+\mathbf{1}$	$(2B^2N-1)W^2+(W^2-1)(B^2N+1)+(B^2-1)N$	NB^2	W^2	0	0	0	0
Dauwe[17]	$(\mathbf{B^2N}+\mathbf{2}+\mathbf{N})\mathbf{W^2Q}+\mathbf{10}$	$(2B^2N-N)W^2Q+(N-1)+(W^2Q-1)(N+1)+3W^2+24$	$N+1$	W^2Q	$6W^2$	1	0	1
Chan[18]	$(\mathbf{B^2N}+\mathbf{2}+\mathbf{N})\mathbf{W^2Q}$	$(2B^2N-N)W^2Q+(N-1)+(W^2Q-1)(N+1)$	N	W^2Q	0	0	0	0
Goossens[19]	$\mathbf{NW^2}+(\mathbf{N}+\mathbf{1})\mathbf{W^2}$	$(2N-1)W^2+(N+1)(W^2-1)$	N	W^2	0	0	$\mathbf{W^2}$	0
Karam[20]	$(5N+2)M$	$2NM+(N+1)(M-1)$	N	$2M$	0	0	$(\mathbf{2N}+\mathbf{3})\mathbf{M}+\mathbf{N}$	0
CCNLM (proposed)	$(N+4)M$	$2M+(N+1)(M-1)+N$	N	$2M$	0	0	$\mathbf{3M}+\mathbf{N}$	0
CCBNLM (proposed)	$(B^2N+4)M$	$2M+(B^2N+1)(M-1)+B^2N$	NB^2	$2M$	0	0	$\mathbf{3M}+\mathbf{N}$	0

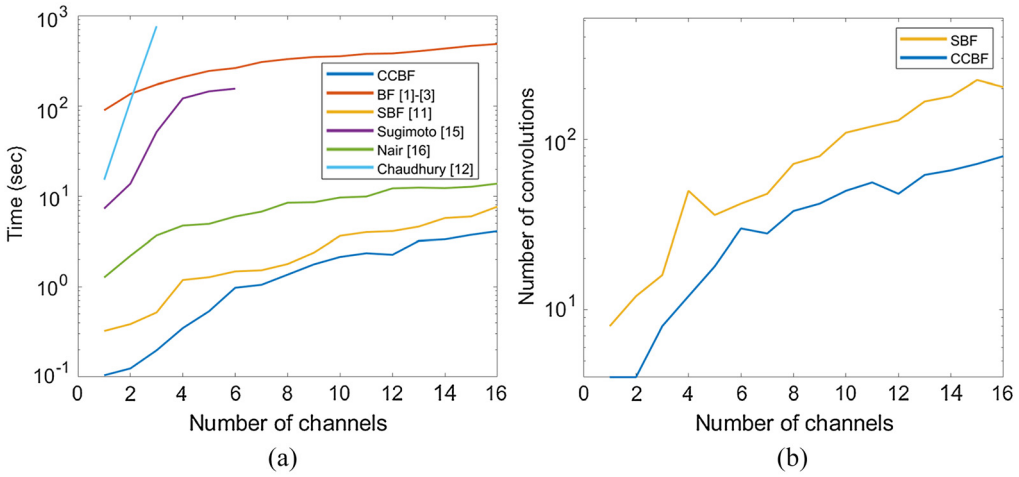


Fig. 1 (a) Graph showing the execution time (necessary to achieve MSE averaged over color channels of 5) of the BF, SBF, and proposed CCBF as well as other fast implementation,[12,15,16] as a function of color channels N. (b) Graph showing the number of convolutions for SBF and CCBF to obtain the results in (a). A hyperspectral image[24] ($1392 \times 1040 \times 31$) with 8-bits per pixel was used.

used in Sugimoto and Nair's implementation.[15,16] Similarly, earlier efforts to accelerate NLM resulted in only a slight improvement at $\mathcal{O}(W^2B^2NQ)$ where $0 < Q < 1$ is a random sampling rate.[17,18] More recent NLM implementations yielded complexities of $\mathcal{O}(W^2N)$[19] and $\mathcal{O}(NM)$[20] where M is a number of Monte-Carlo draws—that is, invariant to the block size B. Note that the complexity of these methods[15–19] scales with color dimension N because the convolution needs to be applied to each color channel separately.

By contrast, the CCBF and CCNLM implementations proposed in this paper achieve the per-pixel complexity of $\mathcal{O}(M)$ where M is the number of Monte-Carlo draw, i.e., independent of W, B, and N. We achieve this feat by combining the convolutional filtering of N color channels into a single random filtering process shared by all N color channels. Thus, CCBF and CCNLM are advantageous when processing high-dimensional images, such as color imaging ($N = 3$), color + depth imaging ($N = 4$), multispectral imaging ($N > 3$), and hyperspectral imaging ($N \gg 3$).

2.3 *Review: Stochastic Bilateral Filter and Stochastic Nonlocal Means*

We briefly review stochastic bilateral filter (SBF) and stochastic nonlocal means (SNLM) method we previously developed to accelerate BF and NLM, respectively.[20] SBF/SNLM filters $\tilde{\mathbf{x}}\{\mathbf{u}\}(\mathbf{i})$ and $\tilde{\mathbf{y}}\{\mathbf{u}\}(\mathbf{i})$ are random filters which agree on average with the BF $\mathbf{x}\{\mathbf{u}\}(\mathbf{i})$ and NLM $\mathbf{y}\{\mathbf{u}\}(\mathbf{i})$ results, respectively.

Proposition 1 (SBF). *Let* $\boldsymbol{\xi} \sim \mathcal{N}(0, \Phi^{-1})$, $\boldsymbol{\xi} \in \mathbb{R}^N$ *be a normal random vector. Define*

$$\tilde{\mathbf{x}}\{\mathbf{u}\}(\mathbf{i}) \coloneqq \frac{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}^T\left(\rho(\mathbf{i})\star\left\{\mathbf{u}(\mathbf{i})\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\end{bmatrix}\right\}\right)\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}^T\left(\rho(\mathbf{i})\star\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\end{bmatrix}\right)\right]}, \tag{7}$$

where $\mathbf{u}:\mathbb{Z}^2 \to \mathbb{R}^N$ *is the input image and* $\rho:\mathbb{Z}^2 \to \mathbb{R}$ *the spatial kernel. Then,* $\tilde{\mathbf{x}}\{\mathbf{u}\}(\mathbf{i})$ *is equivalent to the BF* $\mathbf{x}\{\mathbf{u}\}(\mathbf{i})$ *in* Eq. (1).

Proposition 2 (SNLM). *Let* $\boldsymbol{\xi}(\boldsymbol{\omega}) \sim \mathcal{N}(0, \Phi_{\boldsymbol{\omega}}^{-1})$, $\boldsymbol{\xi}(\boldsymbol{\omega}) \in \mathbb{R}^N$, *be independent random vectors defined over a block* $\omega \in \Omega$ *of size* $B \times B$. *Define convolution-sum operator* $*$ *as*

$$\{\boldsymbol{\xi} * \mathbf{u}\}(\mathbf{i}) \coloneqq \sum_{\boldsymbol{\omega}\in\Omega} \boldsymbol{\xi}^T(\boldsymbol{\omega})\mathbf{u}(\mathbf{i}-\boldsymbol{\omega}). \tag{8}$$

The SNLM $\tilde{\mathbf{y}}\{\mathbf{u}\}(\mathbf{i})$ *defined as*

$$\tilde{\mathbf{y}}\{\mathbf{u}\}(\mathbf{i}) = \frac{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\xi}*\mathbf{u})\\ \sin(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}^T\left(\rho(\mathbf{i})\star\left\{\mathbf{u}(\mathbf{i})\begin{bmatrix}\cos(\boldsymbol{\xi}*\mathbf{u})\\ \sin(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}\right\}\right)\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\xi}*\mathbf{u})\\ \sin(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}^T\left(\rho(\mathbf{i})\star\begin{bmatrix}\cos(\boldsymbol{\xi}*\mathbf{u})\\ \sin(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}\right)\right]}, \tag{9}$$

is also equivalent to NLM $\mathbf{y}\{\mathbf{u}\}(\mathbf{i})$ *in* Eq. (4), *where* $\mathbf{u}:\mathbb{Z}^2 \to \mathbb{R}^N$ *is the input image and* $\rho:\mathbb{Z}^2 \to \mathbb{R}$ *the spatial kernel.*

Proof of Propositions 1 and 2 are provided in Ref. 20. In practice, the range kernel is approximated by Monte-Carlo. As a Monte-Carlo implementations of Propositions 1 and 2, a random vector $\boldsymbol{\xi} \sim \mathcal{N}(0, \Phi^{-1})$ is generated M times, and then averaged to obtain an approximation of the expected value $\mathbb{E}[\cdot]$. Practically speaking, convolutions in the numerators of Eqs. (7) and (9) are actually repeated MN times in their implementations due to the fact that the input image $\mathbf{u}:\mathbb{Z}^2 \to \mathbb{R}^N$ is an N-dimensional signal. Thus, the execution times of SBF and SNLM are $\mathcal{O}(NM)$—considerably faster than original BF/NLM, but nevertheless scaling linearly with N. See Tables 1 and 2.

3 Proposed Stochastic Filters

3.1 *Color-Compressive Bilateral Filter*

In this section, we aim to reduce the number of convolution operators by leveraging prior work known as fast compressive bilateral filtering, or FCBF.[14] The computational complexity of FCBF is $\mathcal{O}(C^N)$ for some constant $C > 1$, which is unacceptably slow unless color dimension N is 1 (i.e., $\mathbf{u}:\mathbb{Z}^2 \to \mathbb{R}^N$ is a grayscale image). Nevertheless, we begin the development of CCBF by first extending a key result of FCBF in Ref. 14 to the color BF version (i.e., $N > 1$), as follows.

Lemma 1. *Define the gradient of a range kernel* $\delta(\mathbf{q})$ *for Nth dimensional vector* $\mathbf{q} = (q_1, \ldots, q_N)^T \in \mathbb{R}^N$ *as*

$$\nabla\delta(\mathbf{q}) = \begin{bmatrix}\frac{\partial}{\partial q_1}\delta(\mathbf{q})\\ \vdots \\ \frac{\partial}{\partial q_N}\delta(\mathbf{q})\end{bmatrix}. \tag{10}$$

Then, the BF $\mathbf{x}\{\mathbf{u}\}(\mathbf{i})$ *applied to an image* $\mathbf{u}:\mathbb{Z}^2 \to \mathbb{R}^N$ *in* Eq. (1) *can be rewritten as*

$$\mathbf{x}\{\mathbf{u}\}(\mathbf{i}) = \mathbf{u}(\mathbf{i}) + \Phi \cdot \frac{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\nabla\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}, \tag{11}$$

where ρ *is a spatial kernel.*

Proof. Consider the difference image of the form:

$$\mathbf{u}(\mathbf{i}) - \mathbf{x}\{\mathbf{u}\}(\mathbf{i}) = \frac{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))\cdot(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}.$$

Following recursion is a well-known property of Gaussian functions:

$$\nabla\delta(\mathbf{q}) = -\Phi^{-1}\mathbf{q}\exp\left(-\frac{\mathbf{q}^T\Phi^{-1}\mathbf{q}}{2}\right) = -\Phi^{-1}\mathbf{q}\delta(\mathbf{q}). \tag{12}$$

Substituting $\mathbf{q}\delta(\mathbf{q})$ by $-\Phi\nabla\delta(\mathbf{q})$ in Eq. (12) with $\mathbf{q}=\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j})$, we have

$$\mathbf{u}(\mathbf{i})-\mathbf{x}\{\mathbf{u}\}(\mathbf{i})=-\Phi\cdot\frac{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\nabla\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}{\sum\limits_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}-\mathbf{j})\cdot\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))}.$$

Reorganizing the above proves the Lemma.

The literal implementation of Eq. (11) incurs per-pixel complexity of $\mathcal{O}(NW^2)$, where N is the color dimension and W is the window size, which is same as the original BF. Readers are referred to Ref. 14 for the remainder of steps taken by FCBF to achieve the complexity order of $\mathcal{O}(C^N)$ (but this is not necessary for our paper). To overcome this prohibitively large computational burden, we instead consider applying Lemma 1 by appealing to the random filtering approach in SBF.[20] Consider the following relation.

Lemma 2. *Let $\boldsymbol{\xi}\sim\mathcal{N}(0,\Phi^{-1})$, $\boldsymbol{\xi}\in\mathbb{R}^N$ be a normal random vector, and $\mathbf{q}\in\mathbb{R}^N$. Then, the range kernel $\delta:\mathbb{R}^N\to\mathbb{R}$ and its gradient can be written as expected values:*

$$\delta(\mathbf{q})=\mathbb{E}[\cos(\boldsymbol{\xi}^T\mathbf{q})], \tag{13}$$

$$\nabla\delta(\mathbf{q})=-\mathbb{E}[\boldsymbol{\xi}\,\sin(\boldsymbol{\xi}^T\mathbf{q})]. \tag{14}$$

Proof. The proof of Eq. (13) is a classical result of characteristic function of normal random vector.[20] For Eq. (14), we apply derivatives in Eq. (10) to the cosine functions in Eq. (13) in the following manner:

$$\nabla\delta(\mathbf{q})=\mathbb{E}\begin{bmatrix}\frac{\partial}{\partial q_1}\cos(\xi^T\mathbf{q})\\ \vdots\\ \frac{\partial}{\partial q_N}\cos(\xi^T\mathbf{q})\end{bmatrix}=\mathbb{E}\begin{bmatrix}-\xi_1\sin(\xi^T\mathbf{q})\\ \vdots\\ -\xi_N\sin(\xi^T\mathbf{q})\end{bmatrix}=-\mathbb{E}[\boldsymbol{\xi}\,\sin(\xi^T\mathbf{q})]. \tag{15}$$

Lemma 2 allows us to rewrite the range kernel $\delta:\mathbb{R}^N\to\mathbb{R}$ as averaged random phenomena. Combining Lemmas 1 and 2, we arrive at the proposed CCBF below.

Theorem 1 (CCBF). *Let $\boldsymbol{\xi}\sim\mathcal{N}(0,\Phi)$, $\boldsymbol{\xi}\in\mathbb{R}^N$ be a normal random vector. Define*

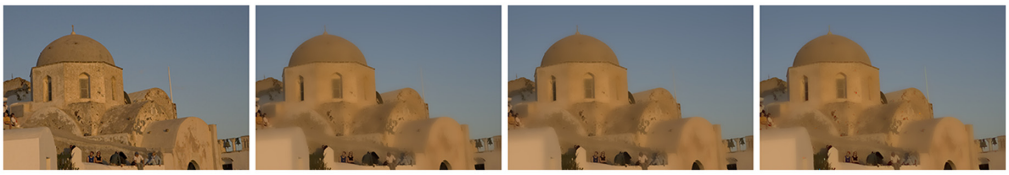
$$\hat{\mathbf{x}}\{\mathbf{u}\}(\mathbf{i})\coloneqq\mathbf{u}(\mathbf{i})+\Phi\frac{\mathbb{E}\left[\boldsymbol{\xi}\begin{bmatrix}-\sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\\ \cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}^T\left(\rho(\mathbf{i})\star\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}\right)\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}^T\left(\rho(\mathbf{i})\star\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}\right)\right]}. \tag{16}$$

Then, $\hat{\mathbf{x}}\{\mathbf{u}\}(\mathbf{i})$ is equivalent to $\mathbf{x}\{\mathbf{u}\}(\mathbf{i})$ in Eq. (1), *where $\mathbf{u}:\mathbb{Z}^2\to\mathbb{R}^N$ is the input image and ρ is the spatial kernel.*

Proof. Lemma 2 inspires separable representations of range kernal $\delta(\cdot)$ and its gradient $\nabla\delta(\cdot)$, respectively, as follows:

$$\begin{aligned}\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))&=\mathbb{E}[\cos(\boldsymbol{\xi}^T(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j})))],\\&=\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))&\sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\end{bmatrix}\right]\end{aligned} \tag{17}$$

$$\begin{aligned}\nabla\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))&=\mathbb{E}[-\boldsymbol{\xi}\,\sin(\boldsymbol{\xi}^T(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j})))]\\&=\mathbb{E}\left[\boldsymbol{\xi}\begin{bmatrix}-\sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))&\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{i}))\end{bmatrix}\begin{bmatrix}\cos(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\\ \sin(\boldsymbol{\xi}^T\mathbf{u}(\mathbf{j}))\end{bmatrix}\right],\end{aligned} \tag{18}$$



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

Fig. 2 Example bilateral filtering results. (a)–(d) Parameters were $\gamma = 10$, $\phi = 51$, $W = 61$. (a) Input image, (b) BF (61.98 s), (c) BF implementation[16] (2.96 s), and (d) proposed CCBF (0.93 s, with $M = 31$, #conv = 62). Iteration numbers were chosen to yield PSNR ≥ 41.14 dB relative to the BF output. Image[11] ($876 \times 584 \times 3$) with 8-bits per pixel.

where trig identities were used to expand $\cos(\cdot)$ and $\sin(\cdot)$. Thus, Eq. (16) follows from substituting Eqs. (17) and (18) into Lemma 1.

In practice, CCBF proposed in Eq. (16) is carried out by Monte-Carlo. We approximate the expectation by drawing M random vectors $\boldsymbol{\xi} \sim \mathcal{N}(0, \Phi)$, executing the computations inside the expectation operator, and averaging. To emphasize the significance of CCBF in Theorem 1, we now draw contrast to SBF[20] and FCBF.[14] First, the convolutions in the numerator and the denominator of CCFB in Eq. (16) are identical [unlike SBF in Eq. (7) but like FCBF], meaning their computational burden can be shared. Second, the convolution in the numerator of CCBF in Eq. (16) combines all N color channels into two "color-compressive" convolutions—$\rho \star \cos(\boldsymbol{\xi}^T \mathbf{u}) \in \mathbb{R}$ and $\rho \star \sin(\boldsymbol{\xi}^T \mathbf{u}) \in \mathbb{R}$ (i.e., not $\mathbb{R}^N$) where ρ is the spatial kernel. This is in contrast to the $2N$ convolutions required to compute the numerator of SBF in Eq. (7) because of the presence of the N-dimensional color image $\mathbf{u}(\mathbf{i}) \in \mathbb{R}^N$. Similarly, C^N convolutions are carried out in FCBF because decoupling between the random vector $\boldsymbol{\xi}$ and $\sin(\boldsymbol{\xi}^T \mathbf{q})$ in Eq. (14) is not possible without randomization. As a result, the proposed CCBF achieves a far superior overall complexity (as determined by the number of convolutions) of $\mathcal{O}(M)$ where M is the number of Monte-Carlo instantiations—a significant improvement over SBF and FCBF of complexity order $\mathcal{O}(NM)$ and $\mathcal{O}(C^N)$, respectively. See Table 1 and an example results in Fig. 2. The rate of Monte-Carlo convergence determining M is described in Sec. 4.1.

We also emphasize that the "color-compressive" convolutions $\rho \star \cos(\boldsymbol{\xi}^T \mathbf{u}) \in \mathbb{R}$ and $\rho \star \sin(\boldsymbol{\xi}^T \mathbf{u}) \in \mathbb{R}$ by spatial kernel ρ are mathematically equivalent ways to implement the multicolor gradient range kernel in Eq. (10). Thus, the change from the random filtering of multiple color channels (SBF) into a single random filtering process (CCBF) will behave correctly, regardless of the differences of intensity patterns in N-dimensional color image $\mathbf{u}: \mathbb{Z}^2 \rightarrow \mathbb{R}^N$.

3.2 *Color-Compressive Nonlocal Means*

We propose CCNLM as a block generalization of CCBF we proposed in Theorem 1. The relationship between the conventional NLM and the conventional BF is made explicit in the Lemma 3 below.

Lemma 3. *Define* $\mathbf{e}: \mathbb{Z}^2 \rightarrow \mathbb{R}^{B^2 N}$ *as*

$$\mathbf{e}(\mathbf{i}) = \begin{bmatrix} \mathbf{u}(\mathbf{i} - \boldsymbol{\omega}_1) \\ \vdots \\ \mathbf{u}(\mathbf{i} - \boldsymbol{\omega}_{B^2}) \end{bmatrix}, \tag{19}$$

where $\mathbf{u}: \mathbb{Z}^2 \rightarrow \mathbb{R}^N$ *is a N-dimensional color image and* $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{B^2}\} \in \Omega$ *refers to the indexes of the* $B \times B$ *block. Set BF parameter* $\Phi \in \mathbb{R}^{B^2 N \times B^2 N}$ *as a block diagonal matrix of the form:*

$$\Phi = \begin{bmatrix} \Phi_{\boldsymbol{\omega}_1} & & \\ & \ddots & \\ & & \Phi_{\boldsymbol{\omega}_{B^2}} \end{bmatrix}. \tag{20}$$

Suppose we apply the BF $\mathbf{x}\{\mathbf{e}\}(\mathbf{i}) \in \mathbb{R}^{B^2N}$ *to an* $B^2 \times N$ *dimensional color image* $\mathbf{e}: \mathbb{Z}^2 \to \mathbb{R}^{B^2 \times N}$. *Partition the output of* $\mathbf{x}\{\mathbf{e}\}(\mathbf{i}) \in \mathbb{R}^{B^2N}$ *as*

$$\mathbf{x}\{\mathbf{e}\}(\mathbf{i}) = \begin{bmatrix} \mathbf{x}_{\boldsymbol{\omega}_1}\{\mathbf{e}\}(\mathbf{i}) \\ \vdots \\ \mathbf{x}_{\boldsymbol{\omega}_{B^2}}\{\mathbf{e}\}(\mathbf{i}) \end{bmatrix}. \tag{21}$$

Then, the NLM output $\mathbf{y}\{\mathbf{u}\}(\mathbf{i})$ *can be rewritten as*

$$\mathbf{y}\{\mathbf{u}\}(\mathbf{i}) = \mathbf{x}_{\boldsymbol{\omega}_m}\{\mathbf{e}\}(\mathbf{i}), \tag{22}$$

where the pixel index $m = (B^2+1)/2$ *corresponds to the center position within the* $B \times B$ *block* (*i.e.*, "$\mathbf{x}_{\boldsymbol{\omega}_m}$" *is the middle pixel in the* $B \times B$ *block, as shown in* Fig. 3).

Proof. Using the BF equation in Eq. (1), and the definition of $\mathbf{e}$ as Eq. (19), the bilateral filtered image $\mathbf{x}_{\boldsymbol{\omega}}\{\mathbf{e}\}(\mathbf{i})$ applied to an N-dimensional color image $\mathbf{u}: \mathbb{Z}^2 \to \mathbb{R}^N$ takes the form:

$$\mathbf{x}_{\boldsymbol{\omega}}\{\mathbf{e}\}(\mathbf{i}) = \frac{\sum_{\mathbf{j} \in \mathbb{Z}^2} \rho(\mathbf{i}-\mathbf{j}) \cdot \delta(\mathbf{e}(\mathbf{i}) - \mathbf{e}(\mathbf{j})) \cdot \mathbf{u}(\mathbf{j}-\boldsymbol{\omega})}{\sum_{\mathbf{j} \in \mathbb{Z}^2} \rho(\mathbf{i}-\mathbf{j}) \cdot \delta(\mathbf{e}(\mathbf{i}) - \mathbf{e}(\mathbf{j}))}, \tag{23}$$

where ρ is the spatial kernel and δ is the range kernel. Plugging $\mathbf{e}(\cdot)$ in $\delta(\cdot)$, we get

$$\begin{aligned} \delta(\mathbf{e}(\mathbf{i}) - \mathbf{e}(\mathbf{j})) &= \exp\left(-\frac{(\mathbf{e}(\mathbf{i}) - \mathbf{e}(\mathbf{j}))^T \Phi^{-1} (\mathbf{e}(\mathbf{i}) - \mathbf{e}(\mathbf{j}))}{2} \right) \\ &= \exp\left(-\sum_{\boldsymbol{\omega} \in \Omega} \frac{(\mathbf{e}(\mathbf{i}-\boldsymbol{\omega}) - \mathbf{e}(\mathbf{j}-\boldsymbol{\omega}))^T \Phi_{\boldsymbol{\omega}}^{-1} (\mathbf{e}(\mathbf{i}-\boldsymbol{\omega}) - \mathbf{e}(\mathbf{j}-\boldsymbol{\omega}))}{2} \right) \\ &= \eta\{\mathbf{e}\}(\mathbf{i}, \mathbf{j}). \end{aligned} \tag{24}$$

Plugging Eq. (24) into Eq. (23) with $\boldsymbol{\omega} = \boldsymbol{\omega}_m$ proves the Lemma.

The significance of Lemma 3 is that we may regard NLM as a special case of BF. Thus, we substitute CCBF (Theorem 1) into Lemma 3 to arrive at the proposed CCNLM implementation.

Theorem 2 (CCNLM). *Let* $\boldsymbol{\xi}(\boldsymbol{\omega}) \sim \mathcal{N}(0, \Phi_{\boldsymbol{\omega}}^{-1})$, $\boldsymbol{\xi}(\boldsymbol{\omega}) \in \mathbb{R}^N$, *be independent random vectors defined over* $\boldsymbol{\omega} \in \Omega$. *Let* $\mathbf{u}: \mathbb{Z}^2 \to \mathbb{R}^N$ *be N-dimensional color image and* ρ *is a spatial kernel. Define* $\hat{\mathbf{y}}: \mathbb{Z}^2 \to \mathbb{R}^N$ *as*

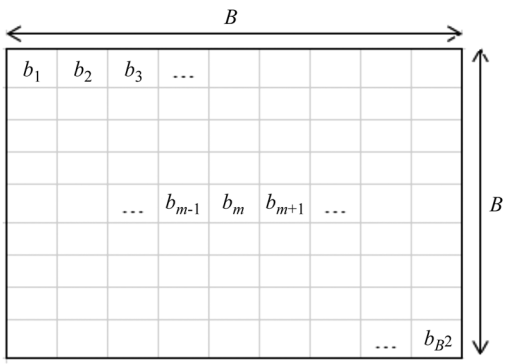


Fig. 3 Pixel locations within a $B \times B$ block Ω.

$$\hat{\mathbf{y}}\{\mathbf{u}\}(\mathbf{i}) \coloneqq \mathbf{u}(\mathbf{i}) + \mu_{\boldsymbol{\omega}_m}(\mathbf{i}), \tag{25}$$

where

$$\mu_{\boldsymbol{\omega}}(\mathbf{i}) = \Phi_{\boldsymbol{\omega}} \frac{\mathbb{E}\left[\boldsymbol{\xi}(\boldsymbol{\omega})\begin{bmatrix}-\sin(\{\boldsymbol{\xi}*\mathbf{u}\})\\ \cos(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}^T\left\{\rho\star\begin{bmatrix}\cos(\boldsymbol{\xi}*\mathbf{u})\\ \sin(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}\right\}\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\xi}*\mathbf{u})\\ \sin(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}^T\left\{\rho\star\begin{bmatrix}\cos(\boldsymbol{\xi}*\mathbf{u})\\ \sin(\boldsymbol{\xi}*\mathbf{u})\end{bmatrix}\right\}\right]}. \tag{26}$$

Then $\hat{\mathbf{y}}\{\mathbf{u}\}(\mathbf{i})$ *is equivalent to NLM* $\mathbf{y}\{\mathbf{u}\}(\mathbf{i})$ *in* Eq. (4).

Proof. Let $\mathbf{e}:\mathbb{Z}^2 \to \mathbb{R}^{B^2N}$ be a $B^2 \times N$-dimensional color image defined as Eq. (19). Define a new random vector

$$\Xi \coloneqq \begin{bmatrix}\boldsymbol{\xi}(\boldsymbol{\omega}_1)\\ \vdots\\ \boldsymbol{\xi}(\boldsymbol{\omega}_{B^2})\end{bmatrix}. \tag{27}$$

Or equivalently, $\Xi \sim \mathcal{N}(0, \Phi^{-1})$, where $\Phi \in \mathbb{R}^{B^2N \times B^2N}$ is as defined in Eq. (20). Substituting Theorem 1 into Lemma 3 and recalling that $\mathbf{u}(\mathbf{i}-\boldsymbol{\omega}_m) = \mathbf{u}(\mathbf{i})$ (where $\boldsymbol{\omega}_m$ is the middle pixel within the $B \times B$ block, as described in Lemma 3), we have

$$\mathbf{y}\{\mathbf{u}\}(\mathbf{i}) = \hat{\mathbf{x}}_{\boldsymbol{\omega}_m}\{\mathbf{e}\}(\mathbf{i}) \quad = \mathbf{u}(\mathbf{i}) + \Phi_{\boldsymbol{\omega}_m} \frac{\mathbb{E}\left[\boldsymbol{\xi}(\boldsymbol{\omega}_m)\begin{bmatrix}-\sin(\Xi^T\mathbf{e}(\mathbf{i}))\\ \cos(\Xi^T\mathbf{e}(\mathbf{i}))\end{bmatrix}^T\left\{\rho(\mathbf{i})\star\begin{bmatrix}\cos(\Xi^T\mathbf{e}(\mathbf{i}))\\ \sin(\Xi^T\mathbf{e}(\mathbf{i}))\end{bmatrix}\right\}\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\Xi^T\boldsymbol{e}(\boldsymbol{i}))\\ \sin(\Xi^T\boldsymbol{e}(\boldsymbol{i}))\end{bmatrix}^T\left\{\rho(\mathbf{i})\star\begin{bmatrix}\cos(\Xi^T\mathbf{e}(\mathbf{i}))\\ \sin(\Xi^T\mathbf{e}(\mathbf{i}))\end{bmatrix}\right\}\right]}. \tag{28}$$

Furthermore, the inner product $\Xi^T\mathbf{e}(\mathbf{i})$ can be rewritten as a convolution-sum in Eq. (8), as follows:

$$\Xi^T\mathbf{e}(\mathbf{i}) = \sum_{\boldsymbol{\omega}\in\Omega}\boldsymbol{\xi}^T(\boldsymbol{\omega})\mathbf{u}(\mathbf{i}-\boldsymbol{\omega}) = \{\boldsymbol{\xi}*\mathbf{u}\}(\mathbf{i}). \tag{29}$$

Substituting Eq. (29) into Eq. (28) proves the Theorem.

Steps required to carry out the proposed CCNLM in Theorem 2 are summarized in Algorithm 2. Similar to CCBF, the expectation operator in CCNLM is carried out by the Monte-Carlo averaging over M random vectors. The convolution operator is shared between the denominator and all of the color channels in the numerator, reducing the number of convolutions (denoted by $\star$) to $\mathcal{O}(M)$. [Compare this to SNLM in Eq. (9) with $\mathcal{O}(NM)$ convolutions.] As described earlier, the complexity of convolution-sum defined in Eq. (8) is invariant to block size B and color dimension N, and comparable to a conventional convolution if implemented with fast Fourier transform (FFT).

4 Complexity Analysis and Further Acceleration Techniques

4.1 *Convergence Rate*

The complexities of CCBF in Algorithm 1 and CCNLM in Algorithm 2 are dominated by the Gaussian filters. Recent advancements in filter designs established that Gaussian filtering can be approximated by $\mathcal{O}(1)$ per-pixel complexity processes using feedback[25–28] or short-time discrete cosine transform.[29–32] Hence, the per-pixel complexity of CCBF/CCNLM implemented with such filters would be $\mathcal{O}(M)$, i.e., linearly dependent on Monte-Carlo draws M but invariant to the filter window size and image size.

Algorithm 1 Color-compressive bilateral filter

input: $\mathbf{u}: \mathbb{Z}^2 \rightarrow \mathbb{R}^N$

output: $\hat{\mathbf{x}}: \mathbb{Z}^2 \rightarrow \mathbb{R}^N$

parameters: γ, Φ

initialize numerator $\mathbf{n}(\mathbf{i}) \Leftarrow 0$ and denominator $d(\mathbf{i}) \Leftarrow 0$

for M times **do**

generate $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \Phi^{-1})$

compute $\rho(\mathbf{i}) \Leftarrow \boldsymbol{\xi}^T \mathbf{u}(\mathbf{i})$

compute $\epsilon(\mathbf{i}) \Leftarrow \cos(\rho(\mathbf{i}))$ and $\lambda(\mathbf{i}) = \sin(\rho(\mathbf{i}))$

compute $\tau(\mathbf{i}) \Leftarrow \rho(\mathbf{i}) \star \epsilon(\mathbf{i})$

compute $\beta(\mathbf{i}) \Leftarrow \rho(\mathbf{i}) \star \lambda(\mathbf{i})$

update $\mathbf{n}(\mathbf{i}) \Leftarrow \mathbf{n}(\mathbf{i}) + \boldsymbol{\xi}(\beta(\mathbf{i})\epsilon(\mathbf{i}) - \tau(\mathbf{i})s(\mathbf{i}))$

update $d(\mathbf{i}) \Leftarrow d(\mathbf{i}) + \epsilon(\mathbf{i})\tau(\mathbf{i}) + \lambda(\mathbf{i})\beta(\mathbf{i})$

end for

set $\hat{\mathbf{x}}(\mathbf{i}) \Leftarrow \mathbf{u}(\mathbf{i}) + \Phi \mathbf{n}(\mathbf{i}) / d(\mathbf{i})$

Algorithm 2 Color-compressive nonlocal means.

input: $\mathbf{u}: \mathbb{Z}^2 \rightarrow \mathbb{R}^N$

output: $\hat{\mathbf{y}}: \mathbb{Z}^2 \rightarrow \mathbb{R}^N$

parameter: γ, Φ

initialize numerator $\mathbf{n}(\mathbf{i}) \Leftarrow 0$

initialize denominator $d(\mathbf{i}) \Leftarrow 0$

for M times **do**

generate $\boldsymbol{\xi} \in \mathbb{R}^{B \times B \times N}$: $\boldsymbol{\xi}(\boldsymbol{\omega}) \sim \mathcal{N}(0, \Phi_b^{-1})$, $\forall\ \omega \in \Omega$

compute $\rho(\mathbf{i}) \Leftarrow \boldsymbol{\xi}(\mathbf{i}) * \mathbf{u}(\mathbf{i})$

compute $\epsilon(\mathbf{i}) \Leftarrow \cos(\rho(\mathbf{i}))$ and $\lambda(\mathbf{i}) = \sin(\rho(\mathbf{i}))$

compute $\tau(\mathbf{i}) \Leftarrow \rho(\mathbf{i}) \star \epsilon(\mathbf{i})$

compute $\beta(\mathbf{i}) \Leftarrow \rho(\mathbf{i}) \star \lambda(\mathbf{i})$

update $\mathbf{n}(\mathbf{i}) \Leftarrow \mathbf{n}(\mathbf{i}) + \boldsymbol{\xi}(\boldsymbol{\omega}_m)(\beta(\mathbf{i})\epsilon(\mathbf{i}) - \tau(\mathbf{i})\lambda(\mathbf{i}))$

update $d(\mathbf{i}) \Leftarrow d(\mathbf{i}) + \epsilon(\mathbf{i})\tau(\mathbf{i}) + \lambda(\mathbf{i})\beta(\mathbf{i})$

end for

set $\hat{\mathbf{y}}(\mathbf{i}) \Leftarrow \mathbf{u}(\mathbf{i}) + \Phi_{\boldsymbol{\omega}_m} \mathbf{n}(\mathbf{i}) / d(\mathbf{i})$

Furthermore, Corollary 1 below suggests that the iteration number M grows very slowly with the increased color dimension. Recalling Theorem 1, the convergence rate of Monte-Carlo averaging in CCBF and CCNLM is proportional to the variances of $\cos(\boldsymbol{\xi}^T \mathbf{q})$ and $\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T \mathbf{q})$ (which are fixed values) in Lemma (2).

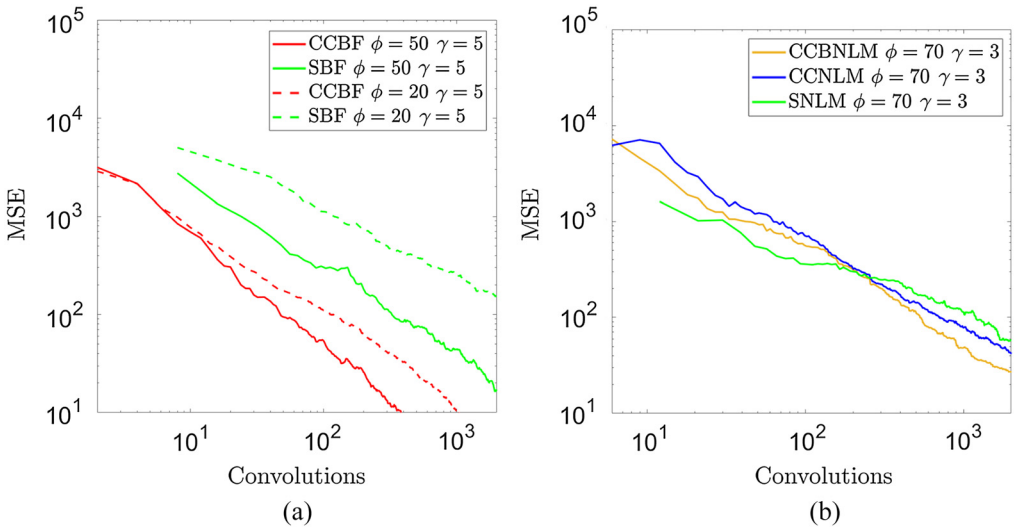


Fig. 4 Graph showing the MSE (averaged over color channels) (a) for SBF and proposed CCNLM with respect to BF; (b) for SNLM and proposed CCNLM with respect to NLM, and proposed CCBNLM with respect to BNLM. The results are obtained averaging over 100 images with 8-bits per pixel taken from the McGill Color Image Database.[33]

Corollary 1. *Let* $\boldsymbol{\xi} \sim \mathcal{N}(0, \Phi^{-1})$, $\boldsymbol{\xi} \in \mathbb{R}^N$ *be a normal random vector, and* $\mathbf{q} \in \mathbb{R}^N$. *Then, the variance of* $\cos(\boldsymbol{\xi}^T\mathbf{q})$ *is no greater than* $\frac{1}{2}$. *The covariance matrix of* $\boldsymbol{\xi}\sin(\boldsymbol{\xi}^T\mathbf{q})$ *is*

$$\Phi^{-1}\frac{1-\alpha^2}{2} + \Phi^{-1}\mathbf{q}\mathbf{q}^T\Phi^{-1}(2\alpha^2 - \alpha), \tag{30}$$

where

$$\alpha = \exp(-\mathbf{q}^T\Phi^{-1}\mathbf{q}). \tag{31}$$

The proof is found in the Appendix. The significance of Corollary 1 is that the Monte-Carlo convergence rate of range kernel $\delta(\mathbf{q}) = \mathbb{E}[\cos(\boldsymbol{\xi}^T\mathbf{q})]$ in the denominator of Eq. (17) is bounded by a constant—i.e., invariant with respect to color dimension N (in the worst case; faster in the usual case). The numerator of Eq. (16) is similarly governed by $\Phi\nabla\delta(\mathbf{q}) = \mathbb{E}[\Phi\boldsymbol{\xi}\sin(\boldsymbol{\xi}^T\mathbf{q})]$ in Eq. (18), whose Monte-Carlo convergence rate is proportional to the diagonal entries of its covariance matrix

$$\Phi\frac{1-\alpha^2}{2} + \mathbf{q}\mathbf{q}^T(2\alpha^2 - \alpha). \tag{32}$$

We therefore conclude that the overall convergence rate of the proposed CCBF is invariant to the color dimension N, and so MSE scales only with the number of convolutions (as determined by M only). See Fig. 4. It ensures that the accuracy requirement for higher color dimensional images can be met without increasing the number of iterations significantly.

4.2 *Color-Compressive Block Nonlocal Means*

Leveraging block/patch-based denoising idea that pixels within the $B \times B$ block share filtering weights, suppose we approximate the conventional NLM in Eq. (4) by a block nonlocal means (BNLM) of the form:

$$z\{\mathbf{u}\}(\mathbf{i}) \coloneqq \sum_{\boldsymbol{\omega}\in\Omega} \frac{\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}+\boldsymbol{\omega}-\mathbf{j})\cdot\eta\{\mathbf{u}\}(\mathbf{i}+\boldsymbol{\omega},\mathbf{j})\cdot\mathbf{u}(\mathbf{j}-\boldsymbol{\omega})}{B^2\sum_{\mathbf{j}\in\mathbb{Z}^2}\rho(\mathbf{i}+\boldsymbol{\omega}-\mathbf{j})\cdot\eta\{\mathbf{u}\}(\mathbf{i}+\boldsymbol{\omega},\mathbf{j})}, \tag{33}$$

Algorithm 3 Color-compressive block nonlocal means.

input: $\mathbf{u}:\mathbb{Z}^2 \to \mathbb{R}^N$

output: $\hat{\mathbf{z}}:\mathbb{Z}^2 \to \mathbb{R}^N$

parameter: γ, Φ

initialize numerator $\mathbf{n}_\omega(\mathbf{i}) \Leftarrow 0$

initialize denominator $d(\mathbf{i}) \Leftarrow 0$

for M times **do**

generate $\boldsymbol{\xi} \in \mathbb{R}^{B\times B\times N}$: $\boldsymbol{\xi}(\omega) \sim \mathcal{N}(0, \Phi_b^{-1})$, $\forall\ \omega \in \Omega$

compute $\rho(\mathbf{i}) \Leftarrow \boldsymbol{\xi}(\mathbf{i}) * \mathbf{u}(\mathbf{i})$

compute $\epsilon(\mathbf{i}) \Leftarrow \cos(\rho(\mathbf{i}))$ and $\lambda(\mathbf{i}) = \sin(\rho(\mathbf{i}))$

compute $\gamma(\mathbf{i}) \Leftarrow \rho(\mathbf{i}) \star \epsilon(\mathbf{i})$

compute $\beta(\boldsymbol{i}) \Leftarrow \rho(\boldsymbol{i}) \star \lambda(\boldsymbol{i})$

update $\mathbf{n}_{\boldsymbol{\omega}}(\mathbf{i}) \Leftarrow \mathbf{n}_{\boldsymbol{\omega}}(\mathbf{i}) + \boldsymbol{\xi}(\boldsymbol{\omega})(\beta(\mathbf{i})\epsilon(\mathbf{i}) - \tau(\mathbf{i})\lambda(\mathbf{i}))$

update $d(\mathbf{i}) \Leftarrow d(\mathbf{i}) + \epsilon(\mathbf{i})\tau(\mathbf{i}) + \lambda(\mathbf{i})\beta(\mathbf{i})$

end for

set $\hat{\mathbf{z}}(\mathbf{i}) \Leftarrow \mathbf{u}(\mathbf{i}) + B^{-2}\sum_{\boldsymbol{\omega}\in\Omega} \Phi_{\boldsymbol{\omega}} \mathbf{n}_{\boldsymbol{\omega}}(\mathbf{i}+\boldsymbol{\omega})/d(\mathbf{i})$

where $\mathbf{u}$ is the input image, ρ is the spatial kernel, and η is the range kernel. Intuitively, we have allowed pixels within the block $\mathbf{j}-\boldsymbol{\omega}$ to be averaged together using the shifted weights $\eta\{\mathbf{u}\}(\mathbf{i},\mathbf{j})$—to compensate for the shift, we shift the output pixel in the opposite direction (hence $\mathbf{i}+\boldsymbol{\omega}$ term). This can be written equivalently as

$$z\{\mathbf{u}\}(\mathbf{i}) = \frac{1}{B^2}\sum_{\boldsymbol{\omega}\in\Omega} \boldsymbol{x}_{\boldsymbol{\omega}}\{\mathbf{e}\}(\mathbf{i}+\boldsymbol{\omega}), \tag{34}$$

where $\mathbf{x}_{\boldsymbol{\omega}}\{\mathbf{u}\}(\mathbf{i}+\boldsymbol{\omega})$ is the shifted BF output in Eq. (23) for each pixel within the block. The BNLM complexity found in Table 2 is similar to that of the NLM, with some extraadditions to carry out Eq. (34) and a small overhead for carrying out Eq. (23) B^2 times. In practice, BNLM approximates NLM well because they share the same weights $\eta\{\mathbf{u}\}(\mathbf{i},\mathbf{j})$. Averaged over 90 images, the mean squared error $\frac{1}{N}\mathbb{E}\|\mathbf{y}-\mathbf{z}\|^2$ between NLM and BNLM was only 11 when $B = 3$.

Due to the averaging in Eq. (34), BNLM has a great potential to reduce the required number of Monte-Carlo instantiations by a factor of B^2. The color-compressive BNLM (CCBNLM) follows directly from Theorem 2:

Corollary 2 (CCBNLM). *Let* $\boldsymbol{\xi}(\boldsymbol{\omega}) \sim \mathcal{N}(0, \Phi_{\boldsymbol{\omega}}^{-1})$, $\boldsymbol{\xi}(\boldsymbol{\omega}) \in \mathbb{R}^N$, *be independent random vectors defined over* $\boldsymbol{\omega} \in \boldsymbol{\Omega}$. *Define* $\hat{z}:\mathbb{Z}^2 \to \mathbb{R}^N$ *as*

$$\hat{\mathbf{z}}\{\mathbf{u}\} = \frac{1}{B^2}\sum_{\boldsymbol{\omega}\in\Omega} \hat{\mathbf{x}}_{\boldsymbol{\omega}}\{\mathbf{e}\}(\mathbf{i}+\boldsymbol{\omega}) = \mathbf{u}(\mathbf{i}) + \frac{1}{B^2}\sum_{\boldsymbol{\omega}\in\Omega} \mu_{\boldsymbol{\omega}}(\mathbf{i}+\boldsymbol{\omega}), \tag{35}$$

where $\mu_{\boldsymbol{\omega}}(\mathbf{i})$ *is as defined in* Eq. (26) *with* $B\times B$ *being the block size of NLM. Then,* $\hat{\mathbf{z}}\{\mathbf{u}\}(\mathbf{i})$ *is equivalent to* $\mathbf{z}\{\mathbf{u}\}(\mathbf{i})$ *in* Eq. (33).

Proof is omitted because it is very similar to Theorem 2. In terms of complexity, Eq. (35) is comparable to Eq. (25)—the additional cost of computing $\mu_{\boldsymbol{\omega}}$ for all $\boldsymbol{\omega} \neq \boldsymbol{\omega}_m$ is almost negligible because $\mu_{\boldsymbol{\omega}}$ and $\mu_{\boldsymbol{\omega}_m}$ share the convolution operations. Yet, CCBNLM converges significantly

faster than CCNLM. To see why this is the case, recall that the Monte-Carlo convergence rate of CCNLM is proportional to Eq. (32). The averaging in Eq. (35) reduces this covariance matrix to

$$[\mathbf{I}_N/B^2,\ldots,\mathbf{I}_N/B^2]\left(\Phi\frac{1-\alpha^2}{2}+\mathbf{q}\mathbf{q}^T(2\alpha^2-\alpha)\right)\begin{bmatrix}\mathbf{I}_N/B^2\\ \vdots\\ \mathbf{I}_N/B^2\end{bmatrix}$$
$$=\frac{1-\alpha^2}{2B^4}\sum_{\boldsymbol{\omega}\in\Omega}\Phi_{\boldsymbol{\omega}}+\frac{2\alpha^2-\alpha}{B^4}\sum_{\boldsymbol{\omega}\in\Omega}\mathbf{q}_{\boldsymbol{\omega}}\mathbf{q}_{\boldsymbol{\omega}}^T,$$

where $\mathbf{I}_N\in\mathbb{R}^{N\times N}$ is an identity matrix and $\Phi_{\boldsymbol{\omega}}\in\mathbb{R}^{N\times N}$ is as defined in Eq. (20), and $\mathbf{q}_b\in\mathbb{R}^N$ is the $\boldsymbol{\omega}$'th subvector of $\mathbf{q}\in\mathbb{R}^{B^2N}$. This is effectively a reduction of covariance matrix by factor of B^2 (since the summation over $\boldsymbol{\omega}\in\Omega$ takes B^2 elements). Hence, we conclude that the Monte-Carlo convergence rate of CCBNLM is faster than CCNLM. Steps required to carry out CCBNLM are provided in Algorithm 3.

We experimentally verify the speedup in Fig. 4(b). Here, the MSE is computed using the original NLM as a reference. The CCNLM converges to NLM faster than SNLM—while SNLM is slightly more computationally complex than CCNLM per iteration, the variance of Eq. (9) is more favorable than Eq. (32). By contrast, there is an implied MSE penalty associated with CCBNLM due to the fact that it converges to BNLM instead of NLM:

$$\mathbb{E}\|\mathbf{y}-\hat{\mathbf{z}}\|^2\geq\mathbb{E}\|\mathbf{y}-\mathbf{z}\|^2. \tag{36}$$

However, since CCBNLM converges B^2 times faster than CCNLM, the overall performance of CCBNLM is far more favorable.

5 Experimental Verification

Recall that in literature, NLM and BF serve different purposes—NLM is used as a denoising filter, while BF is used primarily for smoothing out textures while retaining the edges (often in computer vision). As such, the input data $\mathbf{u}$ in our experiments use non-noisy data for BF, while NLM experiments used noisy data (additive white Gaussian noise with variance $\sigma^2=25$).

Figure 4 shows the mean squared error convergence rate as a function of number of convolutions, averaged over 90 images, between the original and proposed implementations for the BF and the NLM ($\frac{1}{N}\mathbb{E}\|\mathbf{x}-\hat{\mathbf{x}}\|^2$, and $\frac{1}{N}\mathbb{E}\|\mathbf{y}-\hat{\mathbf{y}}\|^2$ or $\frac{1}{N}\mathbb{E}\|\mathbf{y}-\hat{\mathbf{z}}\|^2$), respectively. As expected CCBF requires far fewer convolutions to converge to desired outputs of BF as opposed to the SBF [Fig. 4(a)]. Similarly, as seen in Fig. 4(b), CCNLM converges to NLM with fewer convolutions than the SNLM. It is also clear that the CCBNLM converges to the desired result with even fewer convolutions than both the SNLM and CCNLM implementations, with the MSE between BNLM and the original NLM being $\sim$11 (averaged over 90 images).

The complexity analysis of Table 1 suggests the proposed BF implementation is four times faster than the SBF for color images (i.e., when $N=3$) as seen in the bottleneck of both algorithms. Similarly, CCNLM is two times faster than the SNLM implementation. For the BF experiments, range kernel parameters $\phi=50$ converges faster than $\phi=20$, which is expected because of Corollary 1. NLM is typically used for denoising, and so the experiment was run for one set of parameters that yield the best visual results (with $\phi=70$, a block size of 3×3 and a window of 19×19).

Figure 1 compares the execution time of the conventional BF and several fast implementations[12,15,16,20] and the proposed CCBF, for varying color dimension N. In this experiment, the range parameter ϕ is varied based on the number of channels N, as $\phi=30\sqrt{N}$. This ensures that the bilateral weights $\delta(\mathbf{u}(\mathbf{i})-\mathbf{u}(\mathbf{j}))$ remain relatively constant despite the increasing N. These times are based on Matlab R2019a running on a 2016 ThinkStation P300, with Intel Xeon E3-1241 v3, 32 GB RAM, and 1.5TB HDD, NVidia Quadro K620.

The reported execution times in Fig. 1(a) confirm that the proposed CCBF filter's complexity grows very slowly with respect to N. One implementation[15] runs out of memory at $N=6$, with large number of clusters $K>250$. We report the times it takes for the other methods,[12,15,16,20] to

yield an MSE (averaged over color channels) of 5 or less. The fast implementations,[12,15,16] albeit being fast for grayscale and color, tend to be slower than the proposed method—and sometimes even the conventional BF—with the increase of color dimension N. Their execution times grow at a faster-than-linear rate. In fact, the execution time required by the proposed CCBF to process 16 color channels is comparable to the execution time to process 10 color channels of SBF[20] and 3 color channels of clustering BF method.[16] We omit a similar test for NLM because its original implementation of NLM cannot process the high-dimensional images in a reasonable time.

Fig. 5 Example bilateral filtering results. (a)–(d) Parameters were $\gamma = 10$, $\phi = 130$, $W = 61$. (a) Input image, (b) BF (1209.29 s), (c) BF implementation[16] (76.29 s), and (d) proposed CCBF (62.43 s, with $M = 425$, #conv = 850). Iteration numbers were chosen to yield PSNR $\geq$ 41.14 dB relative to the BF output. Image[24] ($1392 \times 1040 \times 31$) with 8-bits per pixel.

Fig. 6 Example bilateral filtering results: (a), (e), and (i) the inputs. Parameters used were $\gamma = 10$, $\phi = 51$, and $W = 61$. (b), (f), and (j) the BF implementations with 54.825, 54.354, and 53.932 s respectively. (c), (g), and (k) The SBF results with 34.674 s ($M = 350$, #conv = 2800), 36.524 s ($M = 376$, #conv = 3008) and 3.86 s ($M = 38$, #conv = 304), respectively. (d), (h), and (l) The CCBF results with 10.871 s ($M = 374$, #conv = 748), 16.554 s ($M = 537$, #conv = 1074), and 2.29 s ($M = 29$, #conv = 58), respectively. Iteration numbers were chosen to yield PSNR $\geq$ 41.14 dB relative to the BF output. Image from McGill Color Image Database[22] ($876 \times 584 \times 3$) with 8-bits per pixel.

Figure 1(b) shows the number of convolutions needed for the SBF and proposed CCBF implementations to achieve an MSE of 5 with respect to color dimension N for reference. Unlike Fig. 1(a) where the gap between execution times of SBF and CCBF become narrower as color channel N increases, the number of required convolutions in SBF and CCBF diverge as the color channel N grows. From this opposite behaviors in Figs. 1(a) and 1(b), we may conclude that although CCBF requires fewer convolutions than SBF, terms such as additions and multiplications in Table 1 become non-negligible when N is sufficiently large.

Figures 2, 5, and 6 show the results of BF, the state-of-the-art accelerated BF implementation,[16] and CCBF for a color input image ($N = 3$) and a hyperspectral image ($N = 31$). In terms of the execution times of filtering color images, the implementation[16] has a 21-times speedup

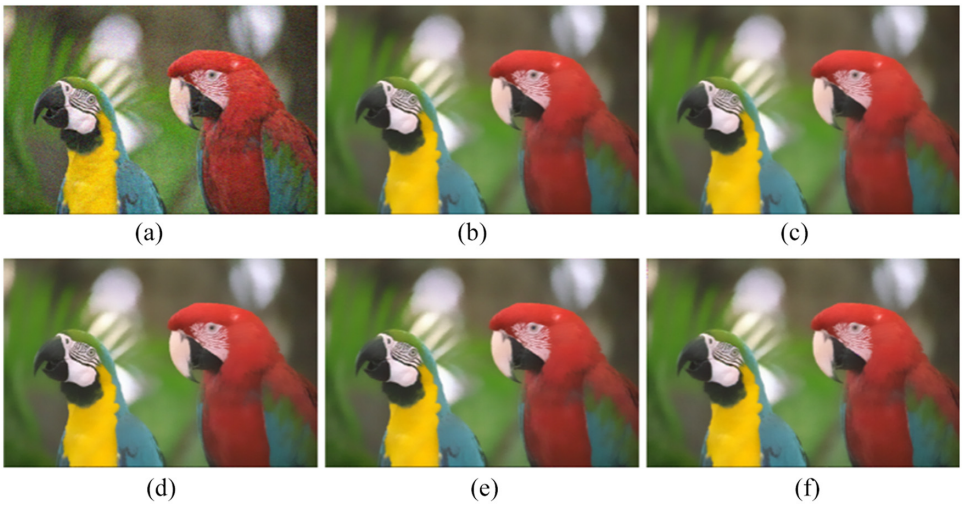


Fig. 7 Example NLM results. Parameters used: $\gamma = 10$, $\phi = 85$, $W = 61$, $B = 3$. Execution times (b) NLM (635.902 s, PSNR = 31.24 dB), (c) BNLM (804.997 s, PSNR = 30.50), (d) SNLM (62.07 s, PSNR = 30.08 dB, with $M = 795$, #conv = 7158), (e) proposed CCNLM (21.85 s, PSNR = 30.02 dB, with $M = 902$, #conv = 2709), and (f) proposed CCBNLM (134.49 s, PSNR = 30.07 dB, with $M = 1109$, #conv = 3330). Iteration numbers were chosen to yield a PSNR $\geq$30 relative to the clean image. Parrot image from Kodak color image set ($768 \times 512 \times 3$) with 8-bits per pixel.

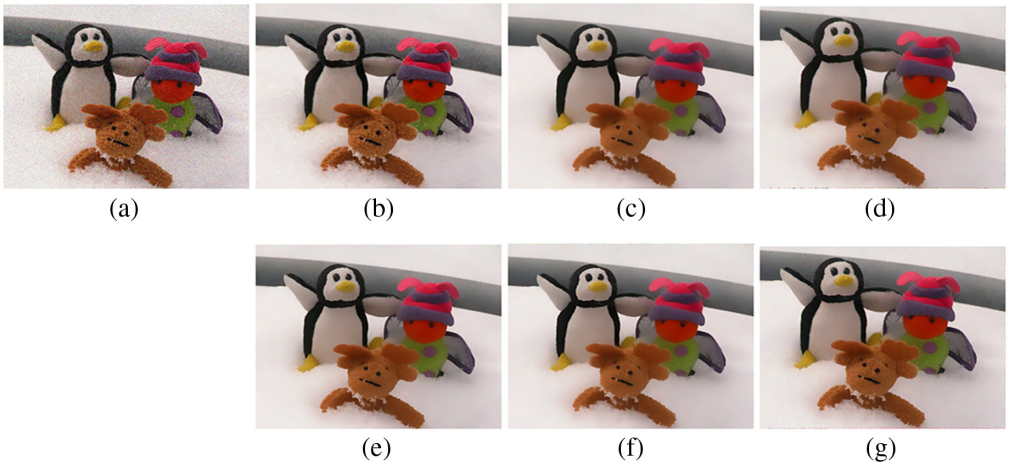


Fig. 8 Example NLM results. (a) Noisy input image. (b) BM3D (7.14 s) (c) BNLM (912.204 s). (d) CCBNLM (39.659 s, with $M = 250$, #conv = 753). (e) NLM (720.281 s). (f) SNLM (56.267 s, with $M = 600$, #conv = 5403). (g) CCNLM (20.059 s, with $M = 600$, #conv = 1803). Parameters used: $\gamma = 10$, $\phi = 85$, $W = 61$, and $B = 3$. Image from McGill Color Image Database[22] ($876 \times 584 \times 3$) with 8-bits per pixel. Note that BM3D is implemented in mexfile, and so the speed is not comparable to Matlab implementations of the CCNLM.

over the conventional BF, whereas the CCBF has a 66-fold speedup. In the case of the hyperspectral image, the speedup of implementation[16] over the naive BF implementation is ∼16 times, and the CCBF speeds it up by 19 times. Figures 7 and 8 show the result of the NLM and block NLM implementations for denoising. The SNLM execution is 10 times faster than the original NLM, while CCNLM and CCBNLM speed ups are 29 and 6 times, respectively. We also point

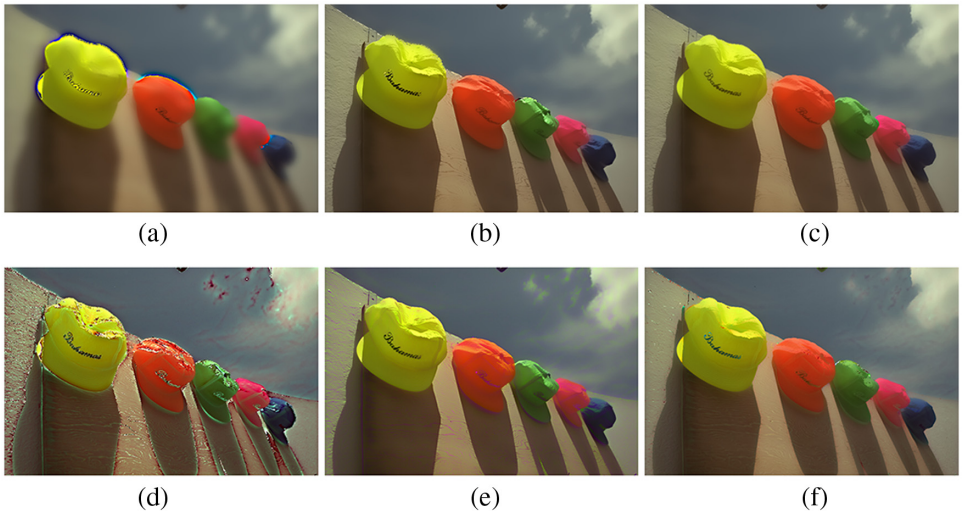


Fig. 9 Example results when number of iterations M progressively increases. (a)–(c) Results from SBF, with $M = 1$ (#conv = 8), $M = 10$ (#conv = 80), and $M = 50$ (#conv = 400), respectively. (d)–(f) Results from CCBF (proposed) with $M = 1$ (#conv = 2), $M = 10$ (#conv = 20), and $M = 50$ (#conv = 100), respectively. Parameters were $\gamma = 10$, $\phi = 51$, and $W = 61$. Image chosen from the Kodak color image set ($768 \times 512 \times 3$) with 8-bits per pixel.

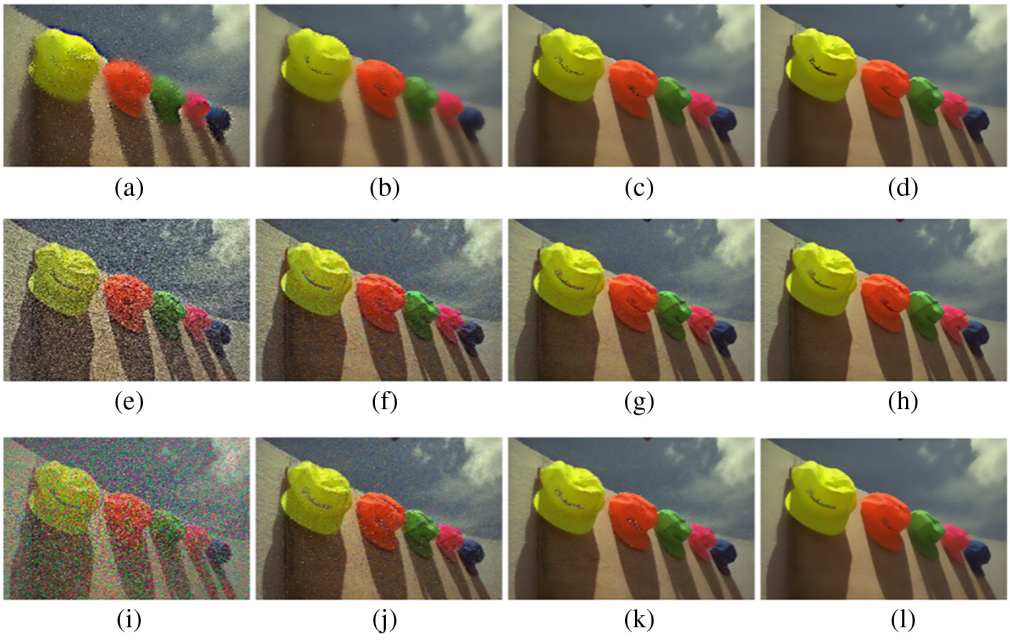


Fig. 10 Example results when number of iterations M progressively increases. (a)–(d) Results for SNLM with $M = 1$ (#conv = 12), $M = 10$ (#conv = 93), $M = 50$ (#conv = 453), and $M = 200$ (#conv = 1803), respectively. (e)–(h) Results for CCNLM with $M = 1$ (#conv = 6), $M = 10$ (#conv = 33), $M = 50$ (#conv = 153), and $M = 200$ (603), respectively. (i)–(l) Results for CCBNLM with $M = 1$ (#conv = 6), $M = 10$ (#conv = 93), $M = 50$ (#conv = 153), and $M = 200$ (#conv = 603), respectively. Parameters were $\gamma = 10$, $\phi = 85$, $W = 61$, and $B = 3$. Image chosen from the Kodak color image set ($768 \times 512 \times 3$) with 8-bits per pixel.

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

Fig. 11 Example results for CCBNLM and CCNLM denoising on a hyperspectral image[24] ($1392 \times 1040 \times 31$). Parameters were $\gamma = 10$, $\phi = 14$, $W = 61$, and $B = 3$. (a) Noisy input image, (b) CCBNLM results ($M = 2500$, #conv = 7503), (c) CCNLM result ($M = 2500$, #conv = 7503), and (d) CCNLM results ($M = 4000$, #conv = 12,003).

out one limitation to this work, which is that CCBF cannot use an edge image separate from the filtering image (i.e., no cross-bilateral filtering).

Figures 9 and 10 show the progression of CCBF and CCNLB for different number of iterations M, respectively. We can see that choosing too small of an M yields unsatisfactory results, as seen for $M = 1$ and $M = 10$. When $M = 50$, the BF implementations output a decent result, whereas the same can not be said for the NLM implementations. CCNLM results at $M = 200$ in Fig. 10 is comparable to CCBNLM at $M = 50$. This is in agreement with the graph in Fig. 4—CCBNLM overtakes CCNLM and converges faster after a certain M value.

Figure 11 shows the result of denoising a hyperspectral image using the proposed CCNLM and CCBNLM. It is clear that CCBNLM yields smoother results for a fewer number of iterations, but as CCNLM runs longer, it approaches the same results. It is, however, impossible to compare to the naive implementations of NLM and BNLM since these algorithms do not run within a reasonable time.

6 Conclusion

We proposed CCBF and CCNLM, new methods aimed at reducing the complexity of the conventional BF and NLM filter, respectively. CCBF and CCNLM combine the random filtering of multiple color channels into a single random convolutional filtering process, achieving the per-pixel complexity of $\mathcal{O}(M)$, where M is the number of random vectors drawn in Monte-Carlo. We proved theoretically that the Monte-Carlo convergence rate is invariant to the window size, the block size, and very slowly increasing with the increasing color dimension of the image. Our experiments confirm of the favorable results in terms of execution times when the color dimension N is large. We further improved the convergence speed by approximating NLM by BNLM. Taken as a whole, the complexity reductions have allowed for speedups up to 66 times faster than the naive implementation for the BF, and almost 30 times for the NLM implementation. To the best of the authors' knowledge, CCBF and CCNLM are the only implementations of BF/NLM to have combined all color channels to a single convolutional process. A future research direction includes GPU-based parallel implementations of CCBF and CCNLM to further accelerate BF and NLM.

7 Appendix

Proof of Corollary 1.

Proof. Regarding the variance of $\cos(\boldsymbol{\xi}^T\mathbf{Z})$, the proof is found in Ref. 20. Let $\mathbf{q} \in \mathbb{R}^N$ be a vector. By basic trigonometry, we have

$$\mathbb{E}[(-\boldsymbol{\xi}\,\sin(\boldsymbol{\xi}^T\mathbf{q}))(-\boldsymbol{\xi}\,\sin(\boldsymbol{\xi}^T\mathbf{q}))^T] = \mathbb{E}\left[\frac{1-\cos(2\boldsymbol{\xi}^T\mathbf{q})}{2}\cdot\boldsymbol{\xi}\boldsymbol{\xi}^T\right] = \frac{\boldsymbol{\Phi}^{-1}}{2} + \frac{1}{8}\nabla^2\mathbb{E}[\cos(2\boldsymbol{\xi}^T\mathbf{q})], \tag{37}$$

where ∇^2 is the Laplace operator defined over the vector $\mathbf{q} \in \mathbb{R}^C$, and the last equality stems from the relation

$$\nabla^2 \cos(2\boldsymbol{\xi}^T\mathbf{q}) = -\nabla 2\boldsymbol{\xi} \sin(2\boldsymbol{\xi}^T\mathbf{q}) = -4\boldsymbol{\xi}\boldsymbol{\xi}^T \cos(2\boldsymbol{\xi}^T\mathbf{q}). \tag{38}$$

Invoking Lemma 2 yields the following:

$$\begin{aligned}\mathbb{E}[(-\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T\mathbf{q}))(-\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T\mathbf{q}))^T] &= \frac{\Phi^{-1}}{2} + \frac{1}{8}\nabla^2 \exp\left(-\frac{4\mathbf{q}^T\Phi^{-1}\mathbf{q}}{2}\right) \\ &= \Phi^{-1}\frac{1-\exp(-2\mathbf{q}^T\Phi^{-1}\mathbf{q})}{2} \\ &\quad + 2\Phi^{-1}\mathbf{q}\mathbf{q}^T\Phi^{-1} \exp(-2\mathbf{q}^T\Phi^{-1}\mathbf{q}).\end{aligned} \tag{39}$$

Similarly, we obtain from Eq. (38) and Lemma 2 the relation

$$\begin{aligned}\mathbb{E}[-\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T\mathbf{q})] &= \nabla\mathbb{E}[\cos(\xi^T\mathbf{q})] \\ &= -\Phi^{-1}\mathbf{q} \exp\left(-\frac{\mathbf{q}^T\Phi\mathbf{q}}{2}\right).\end{aligned} \tag{40}$$

Substituting Eq. (31), we obtain the covariance matrix

$$\begin{aligned}&\mathbb{E}[(-\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T\mathbf{q}))(-\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T\mathbf{q}))^T] - (\mathbb{E}[-\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T\mathbf{q})])(\mathbb{E}[-\boldsymbol{\xi} \sin(\boldsymbol{\xi}^T\mathbf{q})])^T \\ &= \Phi^{-1}\frac{1-\alpha^2}{2} + \Phi^{-1}\mathbf{q}\mathbf{q}^T\Theta^{-1}(2\alpha^2 - \alpha).\end{aligned} \tag{41}$$

References

1. C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth Int. Conf. Comput. Vision*, IEEE, pp. 839–846 (1998).
2. V. Aurich and J. Weule, "Non-linear Gaussian filters performing edge preserving diffusion," in *Mustererkennung 1995*, G. Sagerer, S. Posch, and F. Kummert, Eds., pp. 538–545, Springer, Berlin, Heidelberg (1995).
3. S. M. Smith and J. M. Brady, "SUSAN—a new approach to low level image processing," *Int. J. Comput. Vision* **23**(1), 45–78 (1997).
4. M. Zhang and B. K. Gunturk, "Multiresolution bilateral filtering for image denoising," *IEEE Trans. Image Process.* **17**(12), 2324–2333 (2008).
5. R. Keshet et al., "Bilateral filtering in a demosaicing process," US6816197B2 (2004).
6. S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," *ACM Trans. Graphics* **25**(3), 637–645 (2006).
7. C. Richardt et al., "Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid," *Lect. Notes Comput. Sci.* **6313**, 510–523 (2010).
8. R. Crabb et al., "Real-time foreground segmentation via range and color imaging," in *IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognit. Workshops* (2008).
9. A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognit.*, IEEE, Vol. 2, pp. 60–65 (2005).
10. W. Cheng and K. Hirakawa, "Minimum risk wavelet shrinkage operator for Poisson image denoising," *IEEE Trans. Image Process.* **24**(5), 1660–1671 (2015).
11. S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Lect. Notes Comput. Sci.* **3954**, 568–580 (2006).
12. K. N. Chaudhury, D. Sage, and M. Unser, "Fast O(1) bilateral filtering using trigonometric range kernels," *IEEE Trans. Image Process.* **20**, 3376–3382 (2011).
13. K. Sugimoto and S. Kamata, "Compressive bilateral filtering," *IEEE Trans. Image Process.* **24**, 3357–3369 (2015).
14. G. Deng, "Fast compressive bilateral filter," *Electron. Lett.* **53**, 150–152 (2017).

15. K. Sugimoto, N. Fukushima, and S.-I. Kamata, "Fast bilateral filter for multichannel images via soft-assignment coding," in *Asia-Pacific Signal and Inf. Process. Assoc. Annu. Summit and Conf.*, IEEE, pp. 1–4 (2016).
16. P. Nair and K. N. Chaudhury, "Fast high-dimensional filtering using clustering," in *Proc. IEEE Int. Conf. Image Process*. (2017).
17. A. Dauwe et al., "A fast non-local image denoising algorithm," *Proc. SPIE* **6812**, 681210 (2008).
18. S. H. Chan, T. Zickler, and Y. M. Lu, "Monte Carlo non-local means: random sampling for large-scale image filtering," *IEEE Trans. Image Process.* **23**(8), 3711–3725 (2014).
19. B. Goossens et al., "An improved non-local denoising algorithm," in *Local and Non-Local Approximation Image Process., Int. Workshop, Proc.*, p. 143 (2008).
20. C. Karam and K. Hirakawa, "Monte-Carlo acceleration of bilateral filter and non-local means," *IEEE Trans. Image Process.* **27**, 1462–1474 (2018).
21. X. Zhang and L. Dai, "Fast bilateral filtering," *Electron. Lett.* **55**(5), 258–260 (2019).
22. J. Guo et al., "A fast bilateral filtering algorithm based on rising cosine function," *Neural Comput. Appl.* **31**(9), 5097–5108 (2019).
23. C. Karam, K. Sugimoto, and K. Hirakawa, "Near-constant time bilateral filter for high-dimensional images," in *Proc. IEEE Int. Conf. Image Process.* (2018).
24. A. Chakrabarti and T. Zickler, "Statistics of real-world hyperspectral images," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 193–200 (2011).
25. R. Deriche, "Fast algorithms for low-level vision," *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(1), 78–87 (1990).
26. I. T. Young and L. J. van Vliet, "Recursive implementation of the Gaussian filter," *Signal Process.* **44**, 139–151 (1995).
27. L. J. van Vliet, I. T. Young, and P. W. Verbeek, "Recursive Gaussian derivative filters," in *Proc. Int. Conf. Pattern Recognit.*, Vol. 1, pp. 509–514 (1998).
28. G. Farnebäck and C. Westin, "Improving Deriche-style recursive Gaussian filters," *J. Math. Imaging Vision* **26**, 293–299 (2006).
29. K. Sugimoto and S. Kamata, "Fast Gaussian filter with second-order shift property of DCT-5," in *Proc. IEEE Int. Conf. Image Process.*, pp. 514–518 (2013).
30. K. Sugimoto and S. Kamata, "Efficient constant-time Gaussian filtering with sliding DCT/DST-5 and dual-domain error minimization," *ITE Trans. Media Technol. Appl.* **3**(1), 12–21 (2015).
31. D. Charalampidis, "Recursive implementation of the Gaussian filter using truncated cosine functions," *IEEE Trans. Signal Process.* **64**(14), 3554–3565 (2016).
32. K. Sugimoto, S. Kyochi, and S. Kamata, "Universal approach for DCT-based constant-time Gaussian filter with moment preservation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.* (2018).
33. A. Olmos and F. A. Kingdom, "A biologically inspired algorithm for the recovery of shading and reflectance images," *Perception* **33**(12), 1463–1473 (2004).

Christina Karam is an associate machine learning engineer at the University of Dayton Research Institute, Dayton, Ohio. She received her BS degree in electrical engineering from the Notre-Dame University, Louaize, in Lebanon, in 2013, and her master's and PhD degrees in electrical engineering from the University of Dayton in 2015 and 2019, respectively.

Kenjiro Sugimoto currently serves as an assistant professor at the Graduate School of Information, Production and Systems, Waseda University, in Japan, since 2017. His research mainly focuses on signal/image processing, computer vision and machine learning, specializing in particular on acceleration of filters. He holds engineering degrees from Kurume National College of Technology (BE degree in 2007) and Waseda University (ME degree in 2009 and PhD in 2015). Before, he was a research fellow of the Japan Society for the Promotion of Science during 2010 to 2012, a visiting research scientist at Durham University in 2015, and a research associate at Waseda University from 2014 to 2017. He is a member of IEEE and IEICE.

Keigo Hirakawa received his BS degree in electrical engineering from Princeton University, Princeton, New Jersey, in 2000, with high honors, his MS and PhD degrees in electrical and

computer engineering from Cornell University, Ithaca, New York, in 2003 and 2005, respectively, and his MM degree in jazz performance studies from the New England Conservatory of Music, Boston, Massachusetts, in 2006 with high honors. He was a research associate at Harvard University, Cambridge, Massachusetts, from 2006 until 2009. He is currently an associate professor at the University of Dayton, Dayton, Ohio. He heads Intelligent Signal Systems Laboratory at the University of Dayton, where the group focuses on statistical signal processing, color image processing, and computer vision.