Speeding Up the Bilateral Filter:
A Joint Acceleration Way

Longquan Dai, Mengke Yuan, and Xiaopeng Zhang, *Member, IEEE*

Abstract—Computational complexity of the brute-force implementation of the bilateral filter (BF) depends on its filter kernel size. To achieve the constant-time BF whose complexity is irrelevant to the kernel size, many techniques have been proposed, such as 2D box filtering, dimension promotion, and shiftability property. Although each of the above techniques suffers from accuracy and efficiency problems, previous algorithm designers were used to take only one of them to assemble fast implementations due to the hardness of combining them together. Hence, no joint exploitation of these techniques has been proposed to construct a new cutting edge implementation that solves these problems. Jointly employing five techniques: kernel truncation, best N-term approximation as well as previous 2D box filtering, dimension promotion, and shiftability property, we propose a unified framework to transform BF with arbitrary spatial and range kernels into a set of 3D box filters that can be computed in linear time. To the best of our knowledge, our algorithm is the first method that can integrate all these acceleration techniques and, therefore, can draw upon one another's strong point to overcome deficiencies. The strength of our method has been corroborated by several carefully designed experiments. In particular, the filtering accuracy is significantly improved without sacrificing the efficiency at running time.

Index Terms—Fast bilateral filter, best N-term approximation, haar functions, truncated trigonometric functions.

I. INTRODUCTION

THE bilateral filter (BF) is probably one of the most fundamental tools in computer vision and graphics applications [1]. The concept of BF was first introduced by Aurich and Weule [2] under the name "nonlinear Gaussian filter" in 1995, and then by Smith and Brady [3] within the so-called "SUSAN" approach in 1997. Later, it was rediscovered by Tomasi and Manduchi [4] with the current name "bilateral filter" in 1998. The basic idea underlying bilateral filtering is to do in the range domain of an image what traditional spatial filters [5] do in its spatial domain, because BF considers that two pixels which are close or visually similar to

one another have the same perceptual meaning. Unlike traditional spatial filters, the weights of BF take into account both spatial affinity and intensity similarity with respect to the central pixel. Therefore BF can be used to preserve edges while performing smoothing.

BF's output at pixel $\boldsymbol{x} = (x, y)$ is a weighted average of its neighbors $\mathcal{N}_{\boldsymbol{x}}$. The weights assigned to the pixels in $\mathcal{N}_{\boldsymbol{x}}$ are inversely proportional to both the distance in the spatial domain $\mathcal{S}$ and the dissimilarity in the range domain $\mathcal{R}$. Let I be a gray-level image, $K_r(x)$ and $K_s(x)$ be decreasing functions on the region $\mathbb{R}^+$ and symmetric functions on the entire definition domain $\mathbb{R}$, BF is specified as follows:

$$\hat{I}(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{y} \in \mathcal{N}_{\boldsymbol{x}}} K_s(\|\boldsymbol{x} - \boldsymbol{y}\|) K_r(I(\boldsymbol{x}) - I(\boldsymbol{y})) I(\boldsymbol{y})}{\sum_{\boldsymbol{y} \in \mathcal{N}_{\boldsymbol{x}}} K_s(\|\boldsymbol{x} - \boldsymbol{y}\|) K_r(I(\boldsymbol{x}) - I(\boldsymbol{y}))} \quad (1)$$

Although the Gaussian function $G_\sigma(x) = \exp(-x^2/2\sigma^2)$ is a common choice for the spatial and range kernels, the options are not unique. More candidates can be found in [6].

As a simple, non-iterative and edge-preserving filtering tool, BF has been found with a wide range of applications such as stereo matching [7], flash and no-flash images fusion [8] and contrast enhancement [9]. However, on the flip side of the power, the complexity of its brute-force implementation is $O(|\mathcal{N}_{\boldsymbol{x}}||I|)$, where $|I|$ is the number of pixels of the image I and $|\mathcal{N}_{\boldsymbol{x}}|$ is the size of the neighborhood $\mathcal{N}_{\boldsymbol{x}}$. Since $O(|\mathcal{N}_{\boldsymbol{x}}||I|)$ relies on the size of $\mathcal{N}_{\boldsymbol{x}}$, the run time increases with the size of $\mathcal{N}_{\boldsymbol{x}}$. We thus have to spend several minutes for final results when $\mathcal{N}_{\boldsymbol{x}}$ is large. It is unacceptable for time-critical applications, such as stereo matching [10] and video abstraction [11].

Considering the importance of BF in practice, we will study and retrofit the acceleration of BF to reduce its computational complexity from $O(|\mathcal{N}_{\boldsymbol{x}}||I|)$ down to $O(|I|)$. A typical acceleration approach is first to decompose $\sum_{\boldsymbol{y} \in \mathcal{N}_{\boldsymbol{x}}} K_s(\|\boldsymbol{x} - \boldsymbol{y}\|) K_r(I(\boldsymbol{x}) - I(\boldsymbol{y})) I(\boldsymbol{y})$ into a set of linear convolutions $\sum_{\boldsymbol{y} \in \mathcal{N}_{\boldsymbol{x}}} K_s(\|\boldsymbol{x} - \boldsymbol{y}\|) g(\boldsymbol{y})$ and then to speed up the linear convolution, where $g(\boldsymbol{x})$ is a scalar function. In the literature, a distinction was made between the two operations and therefore different strategies are adopted to accelerate them. Particularly, some fast implementations are limited to accelerate the specific Gaussian kernel. Unlike these approaches, we propose a unified framework to complete the two tasks. At first, we take advantage of the best N-term approximation of $K_r(x)$ on truncated trigonometric functions to transform $\sum_{\boldsymbol{y} \in \mathcal{N}_{\boldsymbol{x}}} K_s(\|\boldsymbol{x} - \boldsymbol{y}\|) K_r(I(\boldsymbol{x}) - I(\boldsymbol{y})) I(\boldsymbol{y})$ into a set of linear convolution $\sum_{\boldsymbol{y} \in \mathcal{N}_{\boldsymbol{x}}} K_s(\|\boldsymbol{x} - \boldsymbol{y}\|) g(\boldsymbol{y})$, then we exploit the best N-term approximation of $K_s(x)$ on 2D Haar functions

to decompose $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) g(y)$ into a set of 3D box filters. More importantly, we disclose that our implementation cannot only be fast computed by the summed area table [12], but also be used to speed up BF with arbitrary kernels.

Contributions of this paper are fourfold: **1)**, we propose the truncated kernels which are exploited to replace BF's original kernels; **2)**, we use the best N-term approximation on Haar functions and truncated trigonometric functions to approximate the truncated spatial and range kernels respectively; **3)**, we find that the product of the two best N-term approximations can be fast computed by 3D box filters; **4)**, compared with other methods, our filtering accuracy can be significantly improved without sacrificing efficiency. In order to clarify our contribution, the rest of this paper is structured as follows. First, existing fast bilateral filtering algorithms are reviewed in section II. After that, section III lists the background techniques. Sequentially, our proposed method is described in section IV and a full comparison is conducted with other acceleration techniques in section V.

II. RELATED WORK

Fast implementations of BF can be roughly divided into two categories: the first one is the high-dimensional implementations [13]–[15], the second one is the low-dimensional case. In this paper, we focus on the second one. For a clear description of our method, we briefly introduce the low dimensional acceleration algorithms in two aspects below.

A. Speeding Up the Linear Convolution of $K_s(x)$

Acceleration techniques of computing $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) g(y)$ have been well studied in the literature. Here, we roughly divided them into three classes: Fast Fourier Transform, Kernel separation and Box filtering.

1) Fast Fourier Transform (FFT): Durand and Dorsey [6] first employed FFT to fast compute $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) g(y)$ as the linear convolution of $K_s(x)$ can be greatly accelerated using FFT. In mathematics, an $O(|\mathcal{N}_x||I|)$ convolution with arbitrary $K_s(x)$ in the spatial domain becomes multiplication in the frequency domain with $O(|I|)$ complexity. Although FFT can be used to produce accurate filtering, FFT and its inverse have the cost $O(|I| \log(|I|))$. But, in practice, an algorithm with linear complexity $O(|I|)$ is preferred.

2) Kernel Separation: Kernel separation based methods decompose 2D filter kernel into two 1D kernels. Rows of an image are filtered at first. After that, the intermediate result is filtered along the columns [16]. Yang *et al.* [17] advocated using Deriche's recursive method [18] to approximate Gaussian filtering. More methods can be found in [19]. Compared with FFT, this kind of algorithms is much faster, but they do not perform well in texture regions.

3) Box Filtering: A 2D box filter $\ddot{B}(x)$ is a spatial domain linear filter in which each pixel x has a value equal to the average of its neighboring pixels $y \in \mathcal{N}_x$ of the input image. Due to the property of equal weights, box filters can be implemented using the summed area table [12] which is significantly faster than using a sliding window algorithm. Note that box filters can be used to approximate the Gaussian filter.

In order to decompose the Gaussian spatial kernel into several box functions, Zhang *et al.* [20] employed the de Moivre-Laplace theorem, which says that for k in the neighborhood of np, $\lim_{n \to \infty} \binom{n}{k} p^k (1 - p)^{n-k} = \frac{1}{\sqrt{2\pi np(1-p)}} \exp(-\frac{(k-np)^2}{2np(1-p)})$. The method however is not problem-free because it can only be applied to speed up the Gaussian filter. Gunturk [21] generalized the Gaussian spatial kernel to arbitrary kernels and employed the least squares optimization to find the optimal coefficients β_i^s that minimize the approximation error $(K_s(\|x\|) - \sum_{i=1}^{M_s} \beta_i^s \ddot{B}_{\mathcal{N}_x^i}(x))^2$. Unlike Gunturk, Pan *et al.* [22] formulated their objective function from the sparsity perspective and exploited the efficient Batch-OMP algorithm [23] to solve the optimal coefficients β_i^s as well as the window radius of $\mathcal{N}_x^i$.

B. Speeding Up the Nonlinear Convolution of $K_r(x)$

Unlike the spatial kernel $K_s(x)$, the range kernel $K_r(x)$ introduces nonlinearity to BF as $K_r(I(x) - I(y))$ is signal-dependent. A common idea shared by algorithm designers to speed up $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) K_r(I(x) - I(y)) I(y)$ is to transform the nonlinear convolution of $K_r(x)$ into a set of linear convolutions of $K_s(x)$. Roughly, there are three kinds of acceleration techniques which are dimension promotion, principle bilateral filtered image component and shiftability property, respectively.

1) Dimension Promotion: This acceleration technique is obtained through representing an image in 3D space by adding the intensity to the spatial domain of an image as the 2D nonlinear convolution of $K_r(x)$ becomes linear convolution in 3D space which is easy to speed up. This idea is similar to the well-known level set method [24] which considers that the breaking and merging operations are hard to perform in 2D space, but they can be easily handled in higher dimensional space.

Mathematically, let $\delta_z(x)$ be an impulse function at z and $F(y, z) = I(y) \delta_{I(y)}(z)$, BF in (1) can be transformed to

$$\hat{I}(x) = \frac{\sum_z K_r(I(x) - z) \sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) F(y, z)}{\sum_z K_r(I(x) - z) \sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)} \quad (2)$$

where z can be explained as the sample in the range domain. For an 8-bit image, $z \in [0, 255]$. In (2), BF is decomposed in the range domain, and the nonlinear relationship between $I(y)$ and $I(x)$ in the range kernel is eliminated. Hence the response of BF can be computed by first performing linear convolution on the auxiliary image $I(y) \delta_{I(y)}(z)$ for each fixed z and then calculating the sum weighted by $K_r(I(y) - z)$ along z.

Porikli [25] first employed this technique to speed up BF. Sequentially, Zhang *et al.* [20] applied it to the joint bilateral filtering. Incorporating with different spatial kernel acceleration methods, Gunturk [21] and Pan *et al.* [22] designed two different fast BF implementations. The biggest problem of them is that they need to perform 255 times linear filtering as well as 255 addition and multiplication operations along z for each fixed y. This is not efficient.

2) Principle Bilateral Filtered Image Component (PBFIC): This method was first proposed by Durand and Dorsey [6] in 2002. Seven years later, Yang *et al.* [17] generalized this

idea for fast bilateral filtering with arbitrary range kernels. At first, Yang transforms BF (1) into (3) by letting $I(x) = z \in [0, 255]$ for 8-bit images

$$\hat{I}(x) = \frac{\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) K_r(z - I(y)) I(y)}{\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) K_r(z - I(y))} \quad (3)$$

then he defines PBFIC $\hat{I}_z(x)$ as

$$\hat{I}_z(x) = \frac{\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) J_z(y)}{\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) W_z(y)} \quad (4)$$

where $W_z(y) = K_r(z - I(y))$ and $J_z(y) = W_z(y) I(y)$. According to (3) (4), BF is decomposed into a set of linear filter responses $\hat{I}_z(x)$. So that we have

$$\hat{I}(x) = \hat{I}_{I(x)}(x) \quad (5)$$

Further, assuming only N out of 256 PBFICs are used ($z \in [L_0, \cdots, L_{N-1}]$), and the intensity $I(x) \in [L_z, L_{z+1}]$, the value of BF can be linearly interpolated as follows:

$$\hat{I}(x) = (L_{z+1} - I(x)) \hat{I}_z(x) + (I(x) - L_z) \hat{I}_{z+1}(x) \quad (6)$$

PBFIC has also been used to design the bilateral grid data structure [26], [27] for fast BF computation. However, the approximation accuracy is usually very low because the linear interpolation is introduced to approximate filtering results. An *et al.* [28] provided a quantitative error analysis for it.

3) Shiftability Property: A range kernel $K_r(x)$ is shiftable if there exists N functions such that, for every translation τ, we have

$$K_r(x - \tau) = \sum_{i=1}^{N} c_i(x) K_i(\tau) \quad (7)$$

Based on this shiftability property, Chaudhury [29] pointed out that BF with a shiftable range kernel $K_r(x)$ can be computed in linear complexity. This is because we have

$$\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) K_r(I(x) - I(y)) I(y)$$

$$= \sum_{i=1}^{N} c_i(I(x)) \sum_{y \in \mathcal{N}_x} K_s(\|x - y\|) K_i(I(y)) I(y) \quad (8)$$

which transforms the nonlinear convolution of the range kernel $K_r(x)$ into a set of linear convolutions of the spatial kernel $K_s(x)$ on the auxiliary image $K_i(I(y)) I(y)$. Hence, employing different acceleration methods for $K_s(x)$, we can derive different fast implementations for BF under the shiftability based acceleration framework.

As for the non-shiftable range kernel, we can exploit a set of shiftable range kernels to approximate it. Following the idea, Chaudhury *et al.* [30] took trigonometric functions to approximate the Gaussian range kernel as illustrated in (9).

$$G_\sigma(I(x) - I(y)) \approx \sum_{n=0}^{N} 2^{-N} \binom{N}{n} \cos(\omega_n I(x)) \cos(\omega_n I(y))$$

$$+ \sum_{n=0}^{N} 2^{-N} \binom{N}{n} \sin(\omega_n I(x)) \sin(\omega_n I(y)) \quad (9)$$

where $\omega_n = \frac{2n - N}{\sqrt{N}\sigma}$. Let $H_n(x)$ be an Hermite polynomial with order n, Dai *et al.* [31] employed $H_n(x)$ to approximate the Gaussian range kernel as illustrated in (10).

$$G_\sigma(I(x) - I(y)) \approx e^{-\frac{I^2(x)}{\sigma}} \sum_{n=0}^{N} H_n(\frac{I(x)}{\sigma}) \frac{I^n(y)}{n! \sigma^{\frac{n}{2}}} \quad (10)$$

Compared with other range kernel acceleration methods, the two methods are limited to the Gaussian kernel. Another drawback shared by the two methods is reported by Chaudhury [32] who complained that it is difficult to approximate the Gaussian range kernel using above expansions when σ is small. In particular, a great deal of approximation terms are required to get a good approximation of a narrow Gaussian function. This will considerably increase the run time.

III. BACKGROUND TECHNIQUES

In this section, we will provide a brief introduction for the necessary background knowledge required by our method.

A. Best N-Term Approximation

Given a set of orthogonal basis functions φ_k of a L^2 space V with the 2 norm $\|\cdot\|_2$, a best N term approximation g_Λ to a function $f \in V$ minimizes

$$\min_{g_\Lambda} \|f - g_\Lambda\|_2$$

$$\text{s.t. } g_\Lambda = \sum_{k \in \Lambda} c_k \varphi_k \quad (11)$$

where Λ denotes the index set formed by N functions φ_k and the space of g_Λ, in which the approximation is sought, is the nonlinear manifold consisting of all linear combinations of the given bases with at most N terms.

Without solving optimization (11), the best N-term approximation of the function f can be obtained by selecting the first N largest coefficients, because the orthogonal projection g_Λ of f on the space V_Λ spanned by $\varphi_k, k \in \Lambda$ is

$$g_\Lambda = \sum_{k \in \Lambda} \langle f, \varphi_k \rangle \varphi_k \quad (12)$$

Hence, the approximation error can be rewritten as $\|f - g_\Lambda\|_2^2 = \sum_{k \notin \Lambda} |\langle f, \varphi_k \rangle|^2 = \|f\|_2^2 - \sum_{k \in \Lambda} |\langle f, \varphi_k \rangle|^2$ which indicates our conclusion.

Selecting the N largest coefficients provides us a simple way to find the best N-term approximation. In the following sections, we use this strategy to find the best N-term approximations of the range kernel $K_r(x)$ and the spatial kernel $K_s(x)$ on 1D truncated trigonometric functions and 2D Haar functions respectively.

B. Truncated Trigonometric Functions

Truncated trigonometric functions form the basis function of the L^2 space on the interval $[-T, T]$. For arbitrary functions $f \in L^2([-T, T])$, we have

$$f(x) = \dot{B}(x)(\sum_{j=0}^{\infty} a_j \cos(\frac{\pi j}{T} x) + \sum_{j=1}^{\infty} b_j \sin(\frac{\pi j}{T} x)) \quad (13)$$

where $\dot{B}(x)$ denotes a 1D box function with the support region $[-T, T]$ and

$$a_0 = \frac{1}{2T} \int_{-T}^{T} f(x)dx \tag{14}$$

$$a_j = \frac{1}{T} \int_{-T}^{T} f(x)\cos(\frac{\pi j}{T}x)dx \tag{15}$$

$$b_j = \frac{1}{T} \int_{-T}^{T} f(x)\sin(\frac{\pi j}{T}x)dx \tag{16}$$

Above equations are the Fourier series which is constrained on the interval $[-T, T]$. (13) holds for the reason that Fourier series can decompose a periodic function into the sum of a (possibly infinite) set of simple oscillating functions, namely sines and cosines. If we define $\hat{f}$ as the periodic extension of f and compute its Fourier periodic expansion, we always have $f(x) = \hat{f}(x) = \sum_{j=0}^{\infty} a_j \cos(\frac{\pi j}{T}x) + \sum_{j=1}^{\infty} b_j \sin(\frac{\pi j}{T}x)$ on the interval $[-T, T]$.

C. Haar Functions

Haar functions were proposed by Haar [33] in 1910 to give an example of the orthonormal system of square-integrable functions on the unit interval. In general, Haar functions are a sequence of rescaled "square-shaped" functions. Specifically, the Haar scaling function $\phi(x)$ is defined as

$$\phi(x) = \begin{cases} 1 & -T \le x \le T \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

and the Haar mother function $\psi_{0,0}(x)$ is described as

$$\psi_{0,0}(x) = \begin{cases} 1 & -T \le x < 0 \\ -1 & 0 \le x \le T \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Let $\mathcal{K}_j = \{-2^{j-1}, \cdots, -1, 1, \cdots, 2^{j-1}\}$ and j be nonnegative integers. For an integer $k \in \mathcal{K}_j$, the Haar function $\psi_{j,k}(x)$ is defined by the formula

$$\psi_{j,k}(x) = \psi_{0,0}(2^j x - \text{sign}(k)(2|k| - 1)T) \tag{19}$$

So, for example, Fig. 1 illustrates $\phi(x)$ as well as the first few values of $\psi_{j,k}(x)$, where

$$\psi_{1,-1}(x) = \psi_{0,0}(2x + T) \quad \psi_{1,1}(x) = \psi_{0,0}(2x - T)$$
$$\psi_{2,-1}(x) = \psi_{0,0}(4x + T) \quad \psi_{2,1}(x) = \psi_{0,0}(4x - T)$$
$$\psi_{2,-2}(x) = \psi_{0,0}(4x + 3T) \quad \psi_{2,2}(x) = \psi_{0,0}(4x - 3T)$$

All these functions are called Haar functions and form an orthogonal basis of $L^2([-T, T])$. Then arbitrary functions $f(x) \in L^2([-T, T])$ can be written as a series expansion by

$$f(x) = c_0\phi(x) + \sum_{j=0}^{\infty} \sum_{k \in \mathcal{K}_j} c_{jk}\psi_{j,k}(x) \tag{20}$$

where $c_0 = \frac{\int_{-T}^{T} f(x)\phi(x)dx}{2T}$ and $c_{jk} = \frac{2^j \int_{-T}^{T} f(x)\psi_{j,k}(x)dx}{2T}$.

2D Haar functions are a natural extension from the single dimension case. For any orthogonal basis $\varphi_k \in L^2([-T, T])$, one can associate a separable orthogonal
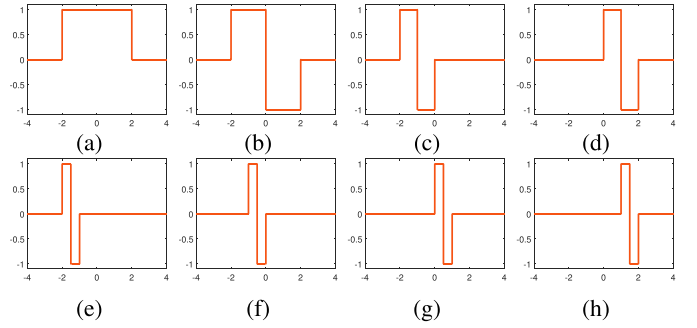

Fig. 1. A visual illustration of the Haar functions $\phi(x)$, $\psi_{0,0}(x)$, $\psi_{1,-1}(x)$, $\psi_{1,1}(x)$, $\psi_{2,-2}(x)$, $\psi_{2,-1}(x)$, $\psi_{2,1}(x)$ and $\psi_{2,2}(x)$, where $T = 2$. (a) $\phi(x)$. (b) $\psi_{0,0}(x)$. (c) $\psi_{1,-1}(x)$. (d) $\psi_{1,1}(x)$. (e) $\psi_{2,-2}(x)$. (f) $\psi_{2,-1}(x)$. (g) $\psi_{2,1}(x)$. (h) $\psi_{2,2}(x)$.

basis $\varphi_{k_1}(x)\varphi_{k_2}(y)$ of $L^2([-T, T] \times [-T, T])$. Following the strategy, we define 2D Haar functions as the set of $\{\phi(x)\phi(y), \phi(x)\psi_{j_2,k_2}(y), \psi_{j_1,k_1}(x)\phi(y), \psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)\}$. Then for arbitrary functions $f(\mathbf{x}) \in L^2([-T, T] \times [-T, T])$, we have

$$f(\mathbf{x}) = c_0\phi(x)\phi(y)$$
$$+ \sum_{j_2=0}^{\infty} \sum_{k_2 \in \mathcal{K}_{j_2}} c_{0,j_2,k_2}\phi(x)\psi_{j_2,k_2}(y)$$
$$+ \sum_{j_1=0}^{\infty} \sum_{k_1 \in \mathcal{K}_{j_1}} c_{j_1,k_1,0}\psi_{j_1,k_1}(x)\phi(y)$$
$$+ \sum_{j_1=0}^{\infty} \sum_{k_1 \in \mathcal{K}_{j_1}} \sum_{j_2=0}^{\infty} \sum_{k_2 \in \mathcal{K}_{j_2}} c_{j_1,k_1,j_2,k_2}\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y) \tag{21}$$

where $\mathbf{x} = (x, y)$ and

$$c_0 = \frac{1}{4T^2} \int_{-T}^{T}\int_{-T}^{T} f(x, y)\phi(x)\phi(y)dxdy$$
$$c_{0,j_2,k_2} = \frac{2^{j_2}}{4T^2} \int_{-T}^{T}\int_{-T}^{T} f(x, y)\phi(x)\psi_{j_2,k_2}(y)dxdy$$
$$c_{j_1,k_1,0} = \frac{2^{j_1}}{4T^2} \int_{-T}^{T}\int_{-T}^{T} f(x, y)\psi_{j_1,k_1}(x)\phi(y)dxdy$$
$$c_{j_1,k_1,j_2,k_2} = \frac{2^{j_1}2^{j_2}}{4T^2} \int_{-T}^{T}\int_{-T}^{T} f(x, y)\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)dxdy. \tag{22}$$

D. The Summed Area Table (SAT)

As a data structure for quickly and efficiently generating the sum of values in a rectangular subset of a grid, the 2D SAT was first introduced to computer graphics society in 1984 by Crow [12] for texture mapping, but was not properly introduced to the world of computer vision till 2004 by Viola and Jones [34] with their landmark face detection algorithm to fast compute the sum of image values $I(x', y')$ on a given rectangle $\mathfrak{R} = (x_0, x_1] \times (y_0, y_1]$.

$$S(\mathfrak{R}) = \sum_{x_0 < x' \le x_1} \sum_{y_0 < y' \le y_1} I(x', y') \tag{23}$$

The complexity of (23) is proportional to the size of the rectangle $\mathfrak{R}$. Viola and Jones employ the SAT $\mathbb{S}(x, y)$ in (24) to compute arbitrary S in constant time.

$$\mathbb{S}(x, y) = \sum_{0 \leq x' \leq x} \sum_{0 \leq y' \leq y} I(x', y') \quad (24)$$

First, SAT can be calculated in one pass over the image I by putting $c(x, -1) = \mathbb{S}(-1, y) = 0$ and using the recurrence (25)

$$c(x, y) = c(x, y - 1) + I(x, y)$$
$$\mathbb{S}(x, y) = \mathbb{S}(x - 1, y) + c(x, y) \quad (25)$$

After that, $S(\mathfrak{R})$ can be computed in linear complexity over the image I according to

$$S(\mathfrak{R}) = \mathbb{S}(x_1, y_1) - \mathbb{S}(x_1, y_0) - \mathbb{S}(x_0, y_1) + \mathbb{S}(x_0, y_0) \quad (26)$$

which only contains four references to $\mathbb{S}(x, y)$.

A special case of the 2D SAT is the 1D SAT defined by $\mathbb{S}(x) = \sum_{0 \leq x' \leq x} I(x')$. We can computed it in one pass according to $\mathbb{S}(x) = \mathbb{S}(x - 1) + I(x)$. The sum $S((x_0, x_1])$ on the interval $(x_0, x_1]$ thus equals to

$$S((x_0, x_1]) = \mathbb{S}(x_1) - \mathbb{S}(x_0) \quad (27)$$

The 2D SAT can also be extended to the high-dimensional case to compute the sum of values in a N-D cube. In the literature, Ke *et al.* [35] considered the image sequences as 3D images and defined the integral video (*i.e.* the 3D SAT) to compute volumetric features from the optical flow of videos for the motion and activity detection. Six years later, Tapia [36] provided a generalized procedure to compute the sum of values in a N dimensional cube using the N-D SAT. Here, instead of directly computing the 3D SAT, we jointly employ the 1D SAT and the 2D SAT to compute the sum $S(\mathfrak{C}) = \sum_{x_0 < x' \leq x_1} \sum_{y_0 < y' \leq y_1} \sum_{z_0 < z' \leq z_1} I(x', y', z')$ of a 3D image $I(x', y', z')$ on a given cube $\mathfrak{C} = (x_0, x_1] \times (y_0, y_1] \times (z_0, z_1]$. This is because we have

$$\bar{I}(x, y, z) = \sum_{x_0 < x' \leq x_1} \sum_{y_0 < y' \leq y_1} I(x', y', z) \quad (28)$$

$$S(\mathfrak{C}) = \sum_{z_0 < z' \leq z_1} \bar{I}(x, y, z') \quad (29)$$

and for each given z, we can employ the 2D SAT in (26) to fast compute $\bar{I}(x, y, z)$. Similarly, (29) can be fast calculated by the 1D SAT in (27).

IV. PROPOSED METHOD

In order to decompose BF into a set of 3D box filters while keeping high accuracy, our method employs Haar functions and truncated trigonometric functions to compute the best N-term approximation of the truncated spatial and range kernels. Further analysis discloses that our approximation can be computed by the 3D SAT with complexity $O(|I|)$.

A. Truncated Spatial Kernel $K_s^T(x)$ and Range Kernel $K_r^T(x)$

Since $K_s(x)$ and $K_r(x)$ are symmetric on the region $\mathbb{R}$ and decrease their values on the region $\mathbb{R}^+$, a point x with large values $K_s(x)$ or $K_r(x)$ usually locates at a small region around the original point. For instance, the Gaussian kernel $G_\sigma(x)$ falls off very fast, and almost vanishes outside the interval $[-3\sigma, 3\sigma]$. Hence, we can simply discard the points outside $[-3\sigma, 3\sigma]$ without introducing significant errors. It is thus reasonable to substitute $K_s(x)$ and $K_r(x)$ with the truncated kernels $K_s^T(x)$ and $K_r^T(x)$ (refer to sections V-A and V-B). Here $K_s^T(x)$ and $K_r^T(x)$ equal to $K_s(x)$ and $K_r(x)$ on the intervals $[-T_s, T_s]$ and $[-T_r, T_r]$ respectively, otherwise $K_s^T(x) = K_r^T(x) = 0$, where $T_s = K_s^{-1}(\epsilon)$, $T_r = K_r^{-1}(\epsilon)$, and ϵ is a predefined value. In practice, 0.01 is a reasonable value.

Note that the truncation operation for $K_s^T(x), K_r^T(x)$ does not increase the run time of BF because we can precompute the truncated regions $[-T_s, T_s]$ or $[-T_r, T_r]$. In the sequent sections, we will describe the methods to approximate the truncated kernels.

B. Approximation for the Linear Convolution of $K_s(x)$

In computer vision and computer graphics [12], [37], [38], the box filter has been used to accelerate many computation-intensive applications as it has the advantage of being fast to compute, but its adoption has been hampered by the fact that it presents serious restrictions to filter construction. To solve the problem, we employ 2D Haar functions to transform arbitrary spatial kernels to a set of box functions.

As the truncated substitution $K_s^T(\|x\|)$ of $K_s(\|x\|)$ is a 2D function, we can employ (21) to decompose it into a set of 2D Haar functions which are defined on the region $[-T_s, T_s] \times [-T_s, T_s]$. After that, we select the first N largest coefficients from the set $\{c_0\} \bigcup \{c_{0,j_2,k_2}\} \bigcup \{c_{j_1,k_1,0}\} \bigcup \{c_{j_1,k_1,j_2,k_2}\}$. Let Λ_1^s, Λ_2^s, Λ_3^s and Λ_4^s denote the selected coefficients from sets $\{c_0\}$, $\{c_{0,j_2,k_2}\}$, $\{c_{j_1,k_1,0}\}$, $\{c_{j_1,k_1,j_2,k_2}\}$, so we have

$$K_s(\|x\|) \underset{①}{\approx} K_s^T(\|x\|) \underset{②}{\approx} \sum_{c_0 \in \Lambda_1^s} c_0 \phi(x)\phi(y)$$
$$+ \sum_{c_{0,j_2,k_2} \in \Lambda_2^s} c_{0,j_2,k_2} \phi(x)\psi_{j_2,k_2}(y)$$
$$+ \sum_{c_{j_1,k_1,0} \in \Lambda_3^s} c_{j_1,k_1,0} \psi_{j_1,k_1}(x)\phi(y)$$
$$+ \sum_{c_{j_1,k_1,j_2,k_2} \in \Lambda_4^s} c_{j_1,k_1,j_2,k_2} \psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y) \quad (30)$$

The equation can be simplified further as the basis functions $\{\phi(x)\phi(y), \phi(x)\psi_{j_2,k_2}(y), \psi_{j_1,k_1}(x)\phi(y), \psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)\}$ can be effectively represented by 2D box functions. Fig. 2 provides us a visual illustration of four 2D Haar functions $\phi(x)\phi(y)$, $\phi(x)\psi_{0,0}(y)$, $\psi_{0,0}(x)\phi(y)$, $\psi_{0,0}(x)\psi_{0,0}(y)$ which stand for the four sets $\{\phi(x)\phi(y)\}$, $\{\phi(x)\psi_{j_2,k_2}(y)\}$, $\{\psi_{j_1,k_1}(x)\phi(y)\}$, $\{\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)\}$ respectively. From the figure, we can verify that $\phi(x)\phi(y)$ is a 2D box filter with the support region $[-T_s, T_s] \times [-T_s, T_s]$, and
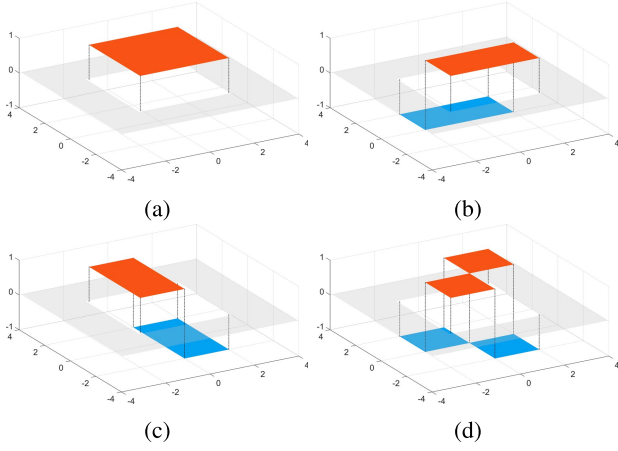
Fig. 2. A visual illustration of 2D Haar functions $\phi(x)\phi(y)$, $\phi(x)\psi_{0,0}(y)$, $\psi_{0,0}(x)\phi(y)$ and $\psi_{0,0}(x)\psi_{0,0}(y)$, where $T = 2$, the red, gray and blue planes denote the values of 1, 0 and -1, respectively. (a) $\phi(x)\phi(y)$. (b) $\phi(x)\psi_{0,0}(y)$. (c) $\psi_{0,0}(x)\phi(y)$. (d) $\psi_{0,0}(x)\psi_{0,0}(y)$.

$\phi(x)\psi_{0,0}(y)$ can be represented by two 2D box filters located at $[-T_s, T_s] \times [-T_s, 0]$ and $[-T_s, T_s] \times [0, T_s]$. Similarly, $\psi_{0,0}(x)\phi(y)$ can be reformulated as two 2D box filters and $\psi_{0,0}(x)\psi_{0,0}(y)$ is equal to four 2D box filters. In appendix A, we prove that the conclusion can be generalized to the set $\{\phi(x)\phi(y)\}$, $\{\phi(x)\psi_{j_2,k_2}(y)\}$, $\{\psi_{j_1,k_1}(x)\phi(y)\}$, $\{\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)\}$. Hence $K_s(\|x\|)$ can be represented by a set of 2D box filters $\ddot{B}_j(x)$:

$$K_s(\|x\|) \underset{③}{\approx} \sum_{c_j \in \Lambda^s} c_j \ddot{B}_j(x) \qquad (31)$$

where c_j denotes coefficient of $\ddot{B}_j(x)$, and Λ^s stands for the collection of c_j. Further, putting N_x^j as the support region of $\ddot{B}_j(x)$, we can reformulate the convolution of $K_s(x)$ as

$$\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)g(y) \approx \sum_{c_j \in \Lambda^s} c_j \ddot{B}_j(x - y)g(y)$$
$$\underset{④}{=} \sum_{c_j \in \Lambda^s} c_j \sum_{y \in N_x^j} g(y) \qquad (32)$$

As is known to us, the 2D SAT can be used to compute the box filters in (32) in constant time by using (26).

C. Approximation for the Nonlinear Convolution of $K_r(x)$

In this section, we employ the best N-term approximation on 1D truncated trigonometric functions to approximate the truncated range kernel $K_r^T(x)$. Let Λ^r represent the selected coefficients from the sets $\{a_k\}$ with $k > 0$, we have

$$K_r(x) \underset{⑤}{\approx} K_r^T(x) \underset{⑥}{\approx} \sum_{a_k \in \Lambda^r} a_k \cos(\frac{\pi k}{T_r}x)\dot{B}(x) \qquad (33)$$

Note that $b_k = 0$ for $K_r^T(x)$ due to the symmetry of $K_r^T(x)$. Employing cosine functions's shiftability property,

we have

$$\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)K_r(I(x) - I(y))I(y)$$
$$\underset{⑦}{\approx} g_k^c(x) \sum_{a_k \in \Lambda^r} \sum_{y \in \mathcal{N}_x} a_k K_s(\|x - y\|)\dot{B}(I(x) - I(y))G_k^c(y)$$
$$+ g_k^s(x) \sum_{a_k \in \Lambda^r} \sum_{y \in \mathcal{N}_x} a_k K_s(\|x - y\|)\dot{B}(I(x) - I(y))G_k^s(y)$$
$$(34)$$

where $G_k^c(y) = g_k^c(y)I(y)$, $G_k^s(y) = g_k^s(y)I(y)$, $g_k^c(x) = \cos(\frac{\pi k}{T_r}I(x))$ and $g_k^s(x) = \sin(\frac{\pi k}{T_r}I(x))$. The computational complexity of (34) depends on the size of $\mathcal{N}_x$. This dependency can be eliminated by the dimension promotion technique. Let $\mathcal{N}_{I(x)}$ denote the interval $[I(x) - T_r, I(x) + T_r]$, $F_c(y, z) = G_k^c(y)\delta_{I(y)}(z)$, $F_s(y, z) = G_k^s(y)\delta_{I(y)}(z)$, we reformulate (34) as

$$\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)K_r(I(x) - I(y))I(y)$$
$$\underset{⑧}{\approx} g_k^c(x) \sum_{a_k \in \Lambda^r} a_k \sum_{z \in \mathcal{N}_{I(x)}} \sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)F_c(y, z)$$
$$+ g_k^s(x) \sum_{a_k \in \Lambda^r} a_k \sum_{z \in \mathcal{N}_{I(x)}} \sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)F_s(y, z) \quad (35)$$

which only involves the linear convolution of $K_s(x)$.

D. 3D Box Filter Based Approximation for BF

Now we are able to decompose the nonlinear convolution of $K_r(x)$ into a set of linear convolutions of $K_s(x)$ (35) as well as to fast compute the linear convolution of $K_s(x)$ (32). Putting (32) into (35), we can further transform the convolution $f(x) = \sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)K_r(I(x) - I(y))I(y)$ the numerator of BF to

$$f(x) \underset{⑨}{\approx} g_k^c(x) \sum_{a_k \in \Lambda^r} \sum_{c_j \in \Lambda^s} a_k c_j \sum_{z \in \mathcal{N}_{I(x)}} \sum_{y \in N_x^j} F_c(y, z)$$
$$+ g_k^s(x) \sum_{a_k \in \Lambda^r} \sum_{c_j \in \Lambda^s} a_k c_j \sum_{z \in \mathcal{N}_{I(x)}} \sum_{y \in N_x^j} F_s(y, z) \quad (36)$$

$\sum_{z \in \mathcal{N}_{I(x)}} \sum_{y \in N_x^j} F_c(y, z)$ and $\sum_{z \in \mathcal{N}_{I(x)}} \sum_{y \in N_x^j} F_s(y, z)$ can be interpreted as the sum of $F_c(y, z)$ and $F_s(y, z)$ in the cube $N_x^j \times \mathcal{N}_{I(x)}$. Hence, (36) denotes a linear combination of 3D box filters that performed on the auxiliary images $F_c(y, z)$, $F_s(y, z)$. So, using (28) (29) to compute 3D box filtering results, we can reduce the complexity of $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)K_r(I(x) - I(y))I(y)$ down to $O(|I|)$. In addition, similar discussion can be applied to BF's denominator $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)K_r(I(x) - I(y))$. Therefore, the filtering result $\hat{I}(x)$ of BF in (1) can be figured out with linear complexity $O(|I|)$.

Finally, we plot Fig 3 to outline the flowchart of our algorithm as well as the acceleration techniques exploited in each step as math symbols stated above may have overshadowed the underlying ideas. Generally speaking, our acceleration algorithm can be divided into two parts (*i.e.* the linear/nonlinear convolution acceleration steps) which employ following speeding up techniques:
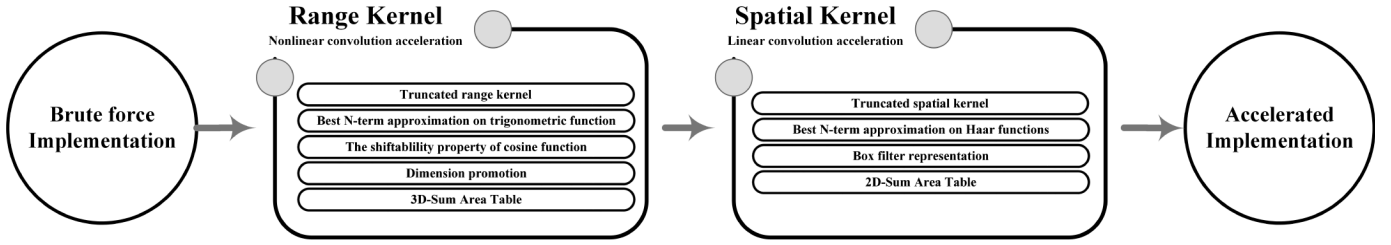
Fig. 3. A compressive illustration for the flowchart of our two steps acceleration algorithm as well as the acceleration techniques used in each step. Through the nonlinear convolution acceleration step containing five techniques and the linear convolution acceleration step assembled by four techniques, our algorithm successfully reduces the computational complexity from $O(|\mathcal{N}_x||I|)$ to $O(|I|)$.

① Truncated spatial kernel $K_s^T(x)$.

② Best N-term approximation for $K_s^T(x)$ on Haar functions.

③ Box filter representation for Haar functions.

④ 2D SAT (2D box filtering).

⑤ Truncated range kernel $K_r^T(x)$.

⑥ Best N-term approximation for $K_r^T(x)$ on truncated trigonometric functions

⑦ The shiftability property of cosine functions.

⑧ Dimension promotion.

⑨ 3D SAT (3D box filtering).

Specifically, in the linear convolution acceleration step for the spatial kernel, Eq (30) takes the truncated spatial kernel $K_s^T(x)$ and best N-term approximation for $K_s^T(x)$ on Haar functions to approximate original spatial kernel $K_s(x)$ in steps ① ②, respectively. Moreover, step ③ in (31) holds due to the reason that Haar functions can be represented by the linear combination of box filters as Fig 2 indicated. Finally, applying 2D SAT to step ④, we can figure out (32) in linear computational complexity. Next, we accelerate the nonlinear convolution of the range kernel. Similar to the linear convolution acceleration step, Eq (33) of the nonlinear convolution acceleration step adopts the truncated range kernel $K_r^T(x)$ and best N-term approximation for $K_r^T(x)$ on trigonometric functions to approximate original range kernel $K_r(x)$ in steps ⑤ ⑥, respectively. Sequentially, the shiftability property of cosine functions is exploited by step ⑦ to eliminate the nonlinearity in the trigonometric best N-term approximation for $K_r^T(x)$. In addition, the nonlinearity in the box $\dot{B}(I(x) - I(y))$ is removed by the dimension promotion technique used by step ⑧ in (35). At last, putting (32) into (35) and employing 3D SAT, we can compute step ⑨ in (36) in linear time.

V. COMPARISON WITH PREVIOUS METHODS

Based on best N-term approximation, we employ 2D Haar functions and 1D truncated trigonometric functions to speed up the convolution of $K_s(x)$ and $K_r(x)$, and we put the two acceleration techniques together to compose our 3D box filter based acceleration method. In this section, we provide some results on synthetic and natural data as well as a detailed analysis to understand the improvements of our proposal and the differences between our approach and existing methods.

A. Comparison With the Acceleration Techniques of $K_s(x)$

Although FFT [6] can fast compute $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)g(y)$, its complexity is not linear. Kernel separation based

TABLE I

THE TIME OF COMPUTING THE COEFFICIENTS AND SUPPORT REGIONS OF BOX FILTERS, WHERE THE NUMBER OF USED BOX FILTERS IS 3. NOTE THAT ALTHOUGH THE METHOD OF ZHANG IS COMPARABLE WITH OURS, THIS METHOD CAN ONLY ACCELERATE THE GAUSSIAN FUNCTION

	Zhang [20]	Gunturk [21]	Pan [22]	Ours
Time	0.1s	2s	3s	0.05s

methods [19] can complete the task in linear complexity. However, they are limited to the Gaussian function. Even worse, the approximation error will significantly degrade the smoothing quality of texture regions. To the best of our knowledge, the 2D box filtering based algorithms are the first kind of techniques that can reduce the complexity $O(|\mathcal{N}_x||I|)$ of the linear convolution of arbitrary $K_s(x)$ down to $O(|I|)$. Our Haar based fast computation method (32) also belongs to this kind of techniques. In this section, we will highlight the improvements and advantages of our approach, compared with other methods.

1) Coefficients and Support Regions: Employing the linear combination $\sum_{i=1}^{M_s} \sum_y \beta_i^s \ddot{B}_i(x - y)g(y)$ of box filters to approximate $\sum_{y \in \mathcal{N}_x} K_s(\|x - y\|)g(y)$ is the key idea of box filtering based algorithms. In the acceleration literature [20]–[22], the optimization approach is utilized to determine the coefficients and support regions of box functions $\ddot{B}_i(x)$. Compared with previous algorithms, the procedure of our method to determine the two parameters is much simpler. This is because we choose the first N largest coefficients of Haar functions (*i.e.* the best N-term approximation on Haar functions). Meanwhile, the support regions of box functions are predefined by Haar functions $\{\phi(x), \psi_{j,k}(x)\}$ with the size of $\frac{T_s}{2^{j-1}}$, where $j \in \{0, \ldots, \infty\}$. Table I plots the run time of four methods used to determine the coefficients and support regions of box filters. It can be seen that our method spends the least time for acquiring these parameters.

2) The Computational Complexity Equivalence Between the Linear Combinations of Haar Functions and Box Filters: To approximate the spatial kernel $K_s(\|x - y\|)$ centered at pixel x, all previous methods assume $\ddot{B}_i(x - y)$ must be centered at the given pixel x too. Our Haar functions based method breaks this assumption and jointly uses a set of box functions deviated from the pixel x as well as the box functions centered at x to approximate the kernel $K_s(\|x - y\|)$. The major advantage of our approach is that it reuses box filtering
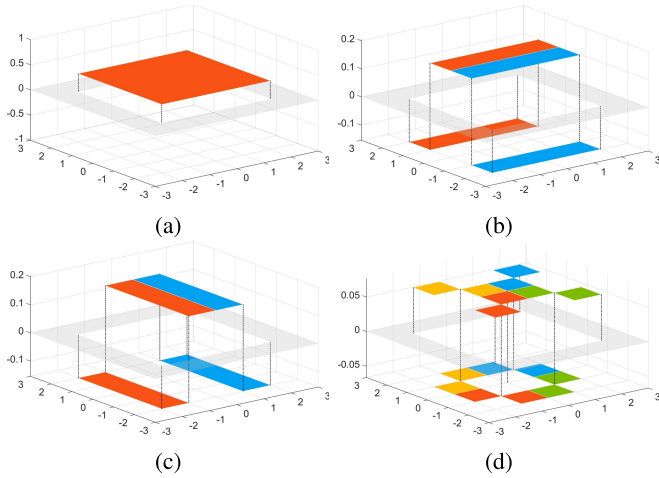
Fig. 4. Visual illustration for the linear combinations on the 2D basis functions $\phi(x)\phi(y)$, $\phi(x)\psi_{1,k_2}(y)$, $\psi_{1,k_1}(x)\phi(y)$, $\psi_{1,k_1}(x)\psi_{1,k_2}(y)$. Here, we denote the four linear combinations as $f_1(x, y)$, $f_2(x, y)$, $f_3(x, y)$, $f_4(x, y)$, respectively, and different color implies different basis functions exploited in the linear combinations. (a) $f_1(x, y)$. (b) $f_2(x, y)$. (c) $f_3(x, y)$. (d) $f_4(x, y)$.

TABLE II
ACCURACY COMPARISON FOR BOX FILTER BASED ACCELERATION METHODS, WHERE THE APPROXIMATION ERROR (OR ACCURACY ERROR) IS MEASURED BY SSIM AND PSNR AND N DENOTES THE NUMBER OF BOX FILTERS

		Zhang [20]	Gunturk [21]	Pan [22]	Ours
SSIM	$N=1$	0.80	0.83	0.84	**0.85**
	$N=2$	0.81	0.85	0.87	**0.89**
	$N=3$	0.83	0.87	0.88	**0.93**
	$N=4$	0.86	0.92	0.94	**0.97**
PSNR	$N=1$	23.16	24.69	25.01	**25.33**
	$N=2$	23.31	25.32	27.01	**27.93**
	$N=3$	24.68	27.03	27.23	**35.22**
	$N=4$	25.91	33.12	36.76	**42.16**

results and therefore can employ fewer box filters to obtain more accurate approximation results. For instance, although $\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)$ is consisted of four box functions (or four 2D box filters), the size of the support regions of these box functions is same to $2^{-j_1+1}T_s \times 2^{-j_2+1}T_s$. This can be demonstrated in Fig 2 which implies that for the convolution kernel $\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)$, the filtering result of x is a linear composition of the box filtering result of four pixels around pixel x. Hence, the computational complexity of the convolution with the kernel $\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)$ is same to the computational complexity of the box filter with the support region $2^{-j_1+1}T_s \times 2^{-j_2+1}T_s$. Similar discussion can also be applied to the basis functions $\phi(x)\phi(y)$, $\phi(x)\psi_{j_2,k_2}(y)$, $\psi_{j_1,k_1}(x)\phi(y)$ according to the discussion in section IV-B.

Employing box filters, we can synthesis the linear combinations of 2D Haar functions. For instance, Fig 4 demonstrates four 2D Haar functions' linear combinations $f_1(x, y) = c_0\phi(x)\phi(y)$, $f_2(x, y) = \sum_{k_2 \in \mathcal{K}} c_{0,1,k_2}\phi(x)\psi_{1,k_2}(y)$, $f_3(x, y) = \sum_{k_1 \in \mathcal{K}} c_{1,k_1,0}\psi_{1,k_1}(x)\phi(y)$ and $f_4(x, y) = \sum_{k_1,k_2 \in \mathcal{K}} c_{1,k_1,1,k_2}\psi_{1,k_1}(x)\psi_{1,k_2}(y)$, where $\mathcal{K} = \{-1, 1\}$. The coefficients c_0, $c_{0,1,k_2}$, $c_{1,k_1,0}$, $c_{1,k_1,1,k_2}$ are computed from (22) based on the Gaussian function $G_1(x)$ as illustrated in Fig 5. From Fig 4, we can find that each linear combination can be decomposed into several box functions with the same support regions. This conclusion can be generalized further. According to appendix A, we could reasonably conclude that given j_1, j_2, the linear combinations of $\phi(x)\phi(y)$, $\phi(x)\psi_{j_2,k_2}(y)$, $\psi_{j_1,k_1}(x)\phi(y)$, $\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)$ can be synthesized from four box filters with support regions $2T_s \times 2T_s$, $2T_s \times 2^{-j_2+1}T_s$, $2^{-j_1+1}T_s \times 2T_s$, $2^{-j_1+1}T_s \times 2^{-j_2+1}T_s$, respectively. So it is rational to equally treat the four linear combinations of Haar functions and the four box filters from the computational complexity perspective. Moreover, the computational complexity of the linear combinations $f_1(x, y) + f_2(x, y) + f_3(x, y)$ equals to three box filters. In the next section, we will take it to approximate the spatial Gaussian kernel function.

3) Approximation Comparison: Fig 5 shows the Gaussian approximation result of our method as well as box filters based approximations such as Zhang *et al.* [20], Gunturk [21], Pan *et al.* [22]. To approximate the Gaussian function, we use five Haar functions illustrated in Figs 4a-4c whereas Zhang *et al.* [20], Gunturk [21], Pan *et al.* [22] take three box filters. Unlike the box filters based methods [20]–[22] employing three box filters to approximate the Gaussian function, our method adopts the linear combination $f_1(x, y) + f_2(x, y) + f_3(x, y)$ which contains five Haar functions to fulfill the same task. Note that the linear combination of five Haar functions can be synthesized by three box filters and therefore the complexity of them are the same. From Fig 5a-5c, we can observe that Zhang, Gunturk and Pan simply take three cubes and cascade them together to approximate the Gaussian function. In contrast, our approximation in Fig 5d is different from theirs. The approximation surface is more complicated than them and the approximation error is the smallest. Table II illustrates the relationship between the approximation quality and the computational complexity. We choose the structural similarity (SSIM) and Peak signal-to-noise ratio (PSNR) to evaluate the approximation error. In addition, the number N of box filters denotes the computational complexity. Note that it is rational to use the number of box filters to indicate the computational complexity because for fixed j_1, j_2, the filtering result of the linear combination of Haar functions is a linear combination of the values of box filtering result at different points. Table II clearly indicates that our approximation error is the smallest among the four methods and the decay rate of our approximation is the fastest with respect to the number of approximation terms.

Other than box filter based acceleration methods, we also compare our approach with other spatial kernel acceleration algorithms such as Deriche [18] which belongs to the kernel separation method. However, this kind of methods cannot perform satisfactorily on texture regions. We can observe these artifacts produced by the kernel separation method in Fig 6.

B. Comparison With the Acceleration Techniques of $K_r(x)$

Decomposing the nonlinear convolution of $K_r(x)$ into a set of linear convolutions is the key step to speed up BF. In this section, we compare our acceleration approach with previous techniques. Without loss of generality, we assume $K_s(\|x\|)$ is a 2D box function in the following paragraphs.
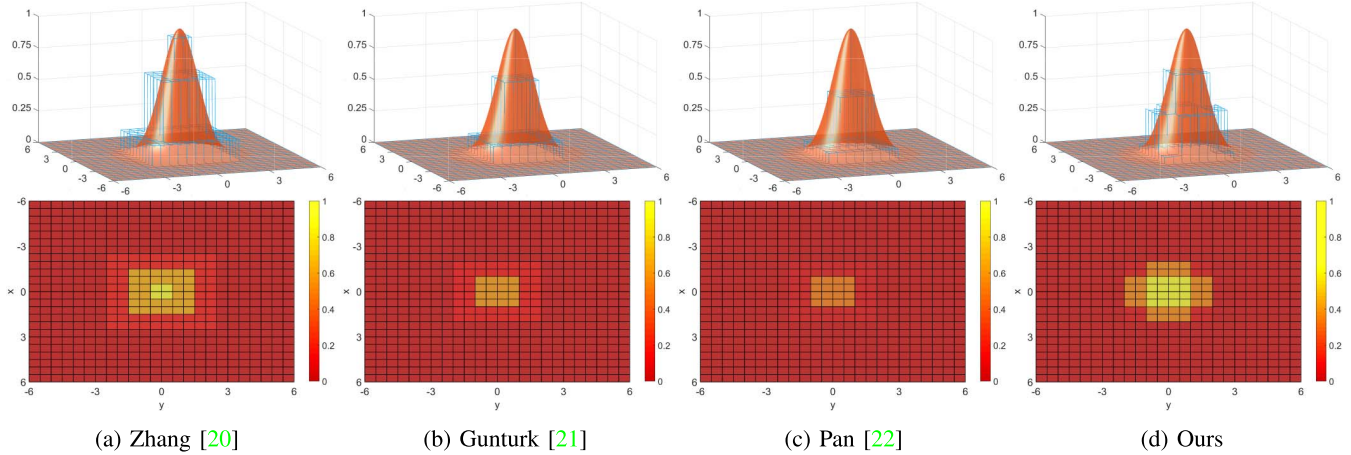
Fig. 5. Visual comparison of four different box filter approximations for the Gaussian function $G_1(x)$, where the first row shows the Gaussian function $G_1(x)$ and the four methods' approximations. In order to disclose the difference of the four approximations more clearly, the second row illustrates the density map plots of the four approximations. Note that Zhang *et al.* [20], Gunturk [21] and Pan *et al.* [22] cascade three box filters together, our method compute the linear combination $f_1(x, y) + f_2(x, y) + f_3(x, y)$ which can be synthesized by three box filters. (a) Zhang *et al.* [20]. (b) Gunturk [21]. (c) Pan *et al.* [22]. (d) Ours.
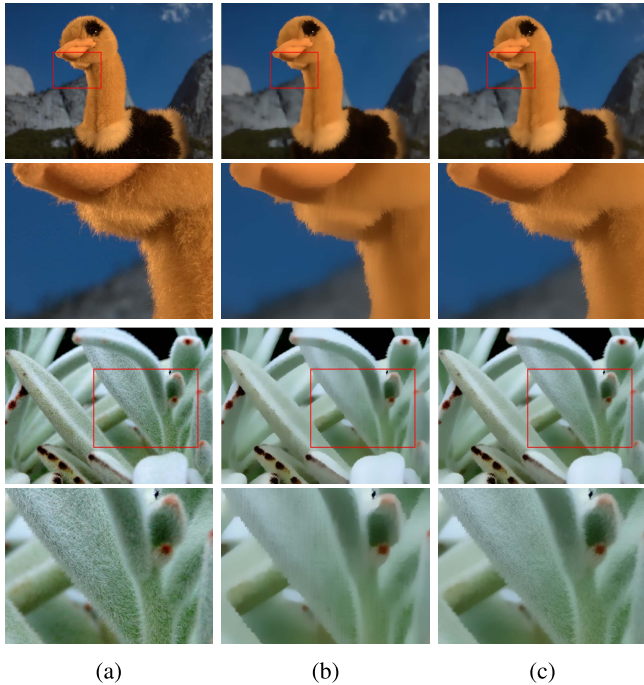

Fig. 6. Comparisons of smoothing result in texture regions. (a) is the input image. (b) is the smoothing result produced by the kernel separation method [18]. (c) is the smoothing result of ours. We can observe that the kernel separation method tends to over smooth the image along the vertical and horizontal edges due to 1D handling of spatial domain.

1) Dimension Promotion: Both our method and the dimension promotion based algorithm employ the dimension promotion technique to eliminate the dependency between $I(y)$ and $I(x)$ in the kernel $K_r(I(x) - I(y))$ and $\dot{B}(I(x) - I(y))$ according to (2) (28) (29) (36). Here $\dot{B}(x)$ is a box function with the support region $[-T_r, T_r]$. The response of BF can be computed by first performing box filtering on the auxiliary images $F(y, z)$ (or $F_c(y, z)$, $F_s(y, z)$) for each fixed z and then calculating the sum weighted by $K_r(I(y) - z)$ (or $\dot{B}(I(y) - z)$) along z.

TABLE III
COMPUTATIONAL COMPLEXITY AND ACCURACY FOR COMPARISON
BETWEEN THE DIMENSION PROMOTION BASED METHOD
(PORIKLI) IN [25] AND OUR METHOD

	Porikli	Ours		
		3	4	5
Add	$256\|I\|$	$12T_r\|I\|$	$16T_r\|I\|$	$20T_r\|I\|$
Mul	$256\|I\|$	$6\|I\|$	$8\|I\|$	$10\|I\|$
SSIM	0.96	0.95	0.97	0.99
PSNR	40.21	38.97	40.32	44.53
Time	1.63s	1.35s	1.36s	1.37s

The major difference between the dimension promotion based algorithms and ours is the computational complexity of the second step as all these methods perform box filtering at the first step. Specifically, let $g(z)$ be a scalar function. For an 8-bit image, $\sum_{z=0}^{255} K_r(I(x) - z)g(z)$ needs 256 multiplications and 255 additions. In contrast, employing the 1D SAT along the z axis, we only require 255 additions and 1 subtraction to compute $\sum_{z=0}^{255} \dot{B}(I(x) - z)g(z) = \sum_{z \in \mathcal{N}_{I(x)}} g(z)$. Note that the floating point addition or subtraction requires 6 clock cycles, multiplication or division require 30-44 clock cycles on the Intel x86 processor. Due to the high running cost for multiplication operation, our method can significantly decrease the run time. Indeed, we can cut down the cost further because for each fixed x, we only need to compute the box filtering result at $I(x)$ along the z axis. Hence in this situation the 1D SAT degrade to the sliding window summation and therefore we can reduce the running cost to $2T_r$ additions.

Our approach is not problem-free. The biggest shortcoming is that our method can only give an approximation filtering result. Fortunately, the approximation error is very small and the accuracy will be improved when we take more terms to approximate the range kernel $K_r(x)$. In Table III, we list the overall computational cost of filtering an image I as well as the SSIM and PSNR indices that measure the similarity between the filtering image and the ground truth, where $|I|$ represents

(a) SSIM evaluation   (b) Dai [31]   (c) Chaudhury [30]   (d) Chaudhury [32]   (e) Ours
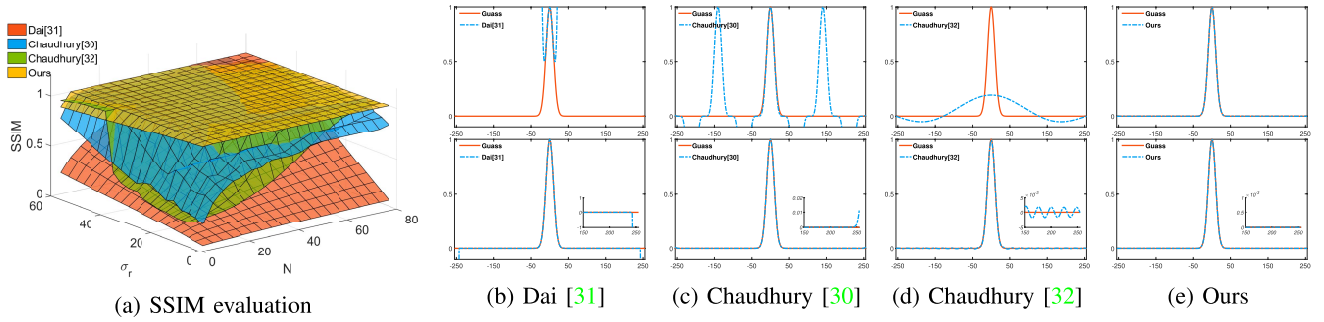
Fig. 7. Visual and quantitative approximation comparison for the Gaussian range kernel $G_{10}(x)$. (a) illustrates the approximation error measured by SSIM, where x-axis denotes the bandwidth parameter σ_r of the Gaussian range kernel, y-axis stands for the number N of approximation terms and z-axis presents the value of SSIM. From (b) to (c), the first row shows the approximation curves of Dai *et al.* [31], Chaudhury *et al.* [30], Chaudhury [32] and ours with 5 approximation terms for the Gaussian kernel. In the second rows, we keep the SSIM indices of the four methods as 0.99 to find the smallest approximation number N of each method. The number N of Dai *et al.* [31], Chaudhury *et al.* [30], Chaudhury [32] and ours are 1700, 81, 42 and 5 respectively.

TABLE IV
TECHNIQUE SUMMARY FOR BF ACCELERATING METHODS

	Kernel separation	Box filtering	Dimension promotion	PBFIC	Shiftability property	Best N-term	Truncated kernels
Porikli [25]	✗	✔	✔	✗	✗	✗	✗
Yang [17]	✔	✗	✗	✔	✗	✗	✗
Gunturk [21]	✗	✔	✔	✗	✗	✗	✗
Pan [22]	✗	✔	✔	✗	✗	✗	✗
Chaudhury [32]	✗	✗	✗	✗	✔	✗	✗
Ours	✗	✔	✔	✗	✔	✔	✔

the pixel number of the image I. Note that the multiplications in our method are caused by the multiplication of a_k in (35) and Table III does not take into account the complexity of linear convolution of $K_s(x)$ because both our approach and the dimension promotion based algorithm perform 255 linear convolutions. Compared with the dimension promotion based algorithm, the number of multiplications is very small and thus can be neglected. The approximation error almost vanishes when we use three approximation terms. We can verify this from the SSIM and PSNR indices and the approximation error map shown in Fig 7. More importantly, the run time is still smaller than the dimension promotion based algorithm.

2) Principle Bilateral Filtered Image Component (PBFIC): This kind of methods is equivalent to the dimension promotion based methods if we compute 256 PBFICs as they do not need to interpolate missing values in this situation. Hence it is reasonable to say that these methods speed up BF by employing downsampling and interpolation operation to reduce the computational burden. The major shortcoming is the large approximation error and it is the origin of stepwise artifacts which degrade the quality of filtering results for HDR image intensively as illustrated in Fig 8. In contrast, our method does not suffer from the problem.

3) Shiftability Property: Chaudhury *et al.* [30] first employed the shiftability property of trigonometric functions to accelerate BF. Using Hermite polynomials, Dai *et al.* [31] gave another application instance of the shiftability property. Unfortunately, the two methods can only deal with the Gaussian kernel. Even worse, they require a large number of terms to closely approximate a narrow Gaussian on the interval $[-255, 255]$ as illustrated in Fig 7. In [32], Chaudhury adopted two measures to solve these problems: 1, shrinking

the approximation interval; 2, dropping off the terms with small coefficients in the approximation series. But, his method inevitably increases the running time because shrinking operation happens at filtering. Dropping off small terms is because some c_i in (8) are extremely small. Hence it is safe to get rid of them to decrease the computational cost without introducing significant errors. The advantage does not come at no cost as the nonnegative assumption of the approximation is broken as Fig 7d shown.

We inherit the idea of Chaudhury [32], but take a different implementation. To decrease the size of approximation interval, we find that the larger values of $K_r(x)$ almost concentrate on a small interval $[-T_r, T_r]$. We introduce the truncated trigonometric function to approximate $K_r(x)$ on the precomputed interval $[-T_r, T_r]$. As for other values, we can simply set them as zeros. To verify the performance of our modifications, we plot Fig 7a to show the approximation error measured by the SSIM index with respect to the number N of approximation terms and the parameter σ_r of the Gaussian range kernel. From the figure, we can observe that the approximation error of our approach is almost the smallest on a wide range. Compared with other shiftability property based methods [30]–[32], the number N of approximation terms is nearly irrelevant to σ_r. In contrast, the number of the approximation terms of [30]–[32] are inversely proportional to σ_r. This is because they need many terms to approximate the long tail of the narrow Gaussian function as shown in Figs. 7b-7d. (The first row shows the approximation results with 5 terms. In the second row, Dai *et al.* [31], Chaudhury *et al.* [30], Chaudhury [32] and our method take 1700, 81, 42 and 5 terms respectively to obtain approximation curves with the value 0.99 of the SSIM index.) So, the run times of these methods depend on σ_r. It is

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

Fig. 8. Quantization artifacts demonstration. (a) Input HDR image (32 bit float, displayed by linear scaling). (b) Compressed image using 32 PBFIC (32 bins). (c) Compressed image using our method. (d) Zoom-in of the square in (b) as the upper image and that in (c) as the lower one.

not a good behavior for acceleration methods. To reduce the number of approximation terms, we employ the best N-term approximation on truncated trigonometric functions to find the best approximation for arbitrary range kernels. Moreover, the best N-term approximation strategy also helps us solve the shortcoming of previous methods which can only approximate the Gaussian function. It is also worth noting that the dropping off small coefficients strategy adopted by Chaudhury [32] to reduce the approximation terms can be interpreted as a special case of best N-term approximation. Other than these, our acceleration strategy also has two advantages: First, our approximation is nonnegative as illustrated in Fig 7e. In contrast, some values of the approximation in Figs 7c 7d are negative; Second, different from Figs 7b 7c, our approximation does not suffer from blowing up small coefficients.

C. BF Comparison

In this section, we compare our acceleration method with five state-of-the-art methods (*i.e.* Porikli [25], Yang *et al.* [17], Gunturk [21], Pan *et al.* [22] and Chaudhury [32]) in terms of qualitative and quantitative aspects. The techniques adopted by the six methods are summarized in Table IV. For a fair comparison, we implement all acceleration methods in C++ without SIMD instructions on a laptop with a 2.0 GHz CPU. Here only experiments on approximating the bilateral filter with Gaussian spatial and range kernels are provided to illustrate the validity and the effectiveness of the proposed method because the shiftability property based acceleration technique [32] can only speed up the Gaussian range kernel. Moreover, we set the number of bins equal to 32 in our evaluation for the PBFIC based method [17] and the box filtering based methods [21], [22], [25].

1) Accuracy: We start our experiments from three famous images (*i.e.* Lena, Barbara and Boat). The statistical data in the following paragraphs is an average of the three images. First, we perform the visual comparison of the fast bilateral filtering schemes. The filtering results of Lena are illustrated in Fig 9. The color-coded images represent the absolute error between the filtering image and ground truth. The variances of the spatial kernel and the range kernel are $\sigma_s \in \{1, 4, 7, 10\}$, $\sigma_r \in \{10, 23, 37, 50\}$. The number of boxes used is 3 as we find that the total run time of three box filters is nearly same to the kernel separation method used by Yang *et al.* [17], and the number of approximation terms for range kernel is 5.

TABLE V
RUN TIME COMPARISON. THE SIZE OF TESTED
IMAGE IS CHOSEN AS 1024 × 1024

	σ_r, where $\sigma_s = 6$						
	15	20	25	30	35	40	45
Porikli [25]	1.69s	1.69s	1.65s	1.64s	1.66s	1.63s	1.65s
Yang [17]	1.59s	1.62s	1.60s	1.61s	1.59s	1.63s	1.62s
Gunturk [21]	1.63s	1.61s	1.63s	1.60s	1.62s	1.60s	1.64s
Pan [22]	1.61s	1.63s	1.60s	1.62s	1.61s	1.63s	1.60s
Chaudhury [32]	1.81s	1.67s	1.59s	1.52s	1.48s	1.41s	1.35s
Ours	**1.39**s	**1.37**s	**1.39**s	**1.38**s	**1.34**s	**1.36**s	**1.37**s

	σ_s, where $\sigma_r = 30$						
	3	4	5	6	7	8	9
Porikli [25]	1.63s	1.64s	1.63s	1.64s	1.65s	1.64s	1.65s
Yang [17]	1.61s	1.63s	1.60s	1.61s	1.62s	1.63s	1.62s
Gunturk [21]	1.60s	1.59s	1.61s	1.60s	1.61s	1.63s	1.62s
Pan [22]	1.61s	1.61s	1.60s	1.62s	1.63s	1.62s	1.64s
Chaudhury [32]	1.51s	1.50s	1.52s	1.52s	1.51s	1.52s	1.53s
Ours	**1.38**s	**1.37**s	**1.37**s	**1.38**s	**1.39**s	**1.37**s	**1.39**s

Absolute error images in Fig 9 show that our method produces smaller error than the other methods. Hence, better accuracy is achieved in approximating BF. Besides the visual comparison, we also offer a convincing and quantitative illustration for the advantages of our method over previous methods. The SSIM index and the peak signal-to-noise ratio (PSNR) are exploited to evaluate the approximation errors of different methods. The statistical data with respect to different bandwidth parameters are plotted in Fig 10. We can easily verify that our method achieves the lowest approximation error for rather wide parameter variation interval.

2) Efficiency: A simple comparison of the run time of BF acceleration schemes is not reasonable because all acceleration methods make a tradeoff between speed and accuracy, and the speed is usually inversely proportional to the accuracy. To conduct a fair comparison, we keep the PSNR indices of filtering results unchanged and measure the speed of each method. Strictly, the PSNR indices cannot be kept the same since they are affected by various reasons. Here, we tweak the parameters of each acceleration method and make the PSNR indices of the filtering results around $32dB$ because these results are visually distinguished from the ground truth. Table V plots run times. We can observe that our approach consumes the smallest run time among different σ_s and σ_r. More importantly, unlike the shiftability property based methods, our run time does not depend on the parameter σ_r. This is because many terms are used to approximate the long tails of the Gaussian function by other methods. In contrast, we only approximate the value on the truncated interval. Note that for large σ_r, the shiftability properties based methods (*i.e.* Chaudhury [32] and ours) achieve the best performance. Unlike Chaudhury, our method needs to perform 3D box filtering caused by the dimension promotion technique. Although both our method and the dimension promotion based methods [21], [22] exploit the dimension promotion technique, our method is faster than these methods. This is because previous methods need the multiplication operation along z axis and our approach only requires the addition operation. This also proves that the filtering accuracy can be significantly improved without sacrificing efficiency.
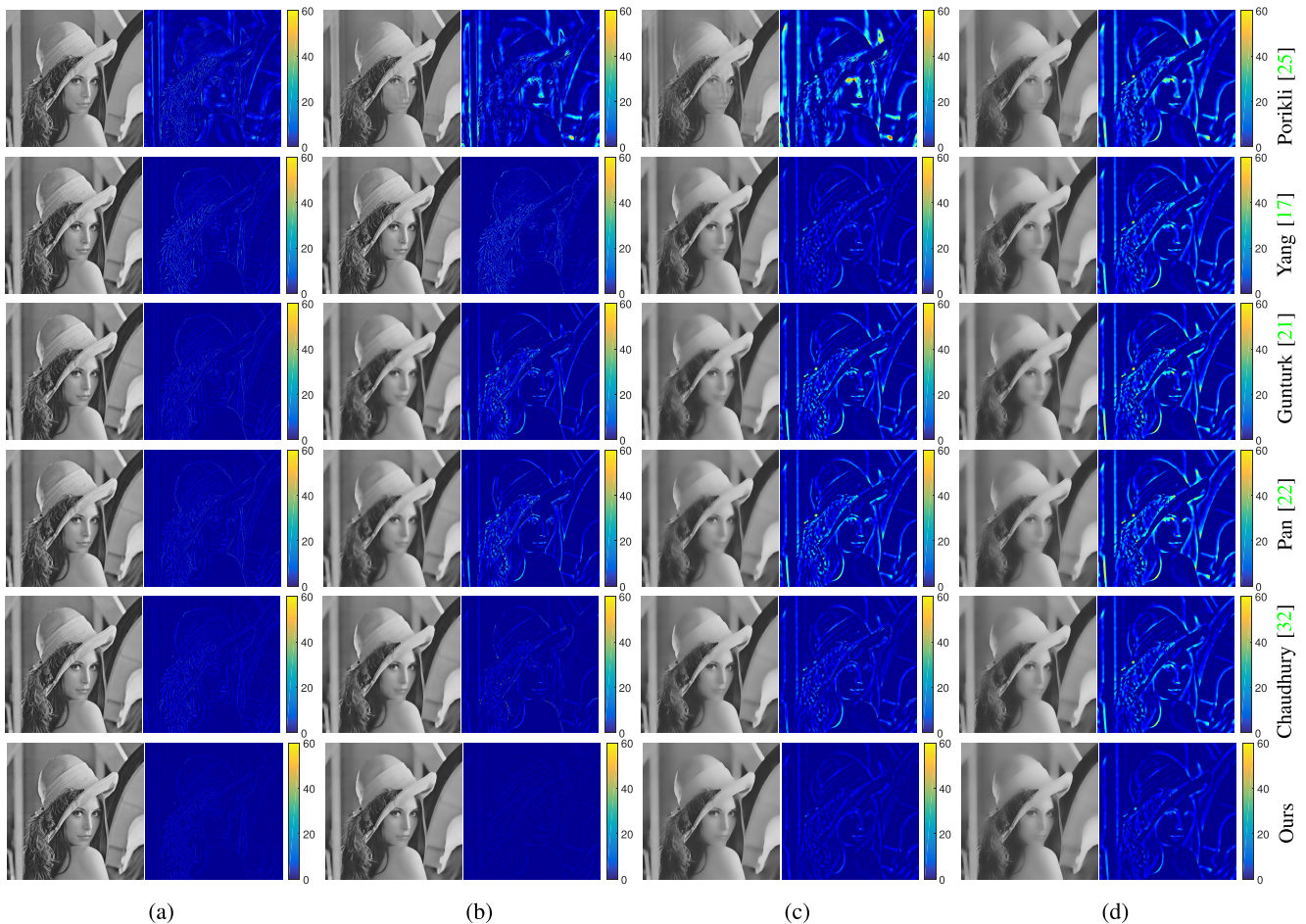
Fig. 9. Lena: a demonstration of the visual quality of the approximation results. From the first row to the sixth row, the results are Porikli [25], Yang *et al.* [17], Gunturk [21], Pan *et al.* [22], Chaudhury [32] and ours, each column shows the filtering results and approximation errors under different parameter settings, where the gray images are the filtering results procured by these BF acceleration methods and the color-coded absolute error maps denote the filtering error compared with the brute-force implementation of BF. (a) $\sigma_s = 1$, $\sigma_r = 10$. (b) $\sigma_s = 4$, $\sigma_r = 23$. (c) $\sigma_s = 7$, $\sigma_r = 37$. (d) $\sigma_s = 10$, $\sigma_r = 50$.

3) Acceleration by Pre-Computation: Our method can be accelerated further by pre-computation. In this section, we will investigate how to integrate pre-computation with our algorithm with respect to two cases.

Case 1 [The Input Image I Is Given, But the Parameters Setting (σ_r, σ_s) of BF Is Not Determined]: This is a frequently encountered situation in practice. For example, when an image already has been loaded into an image manipulation software such as Adobe Photoshop but the filtering parameters are still not determined by users, we are in this situation. If the image manipulation software can pre-compute something during the time waiting for the filtering parameters, the latency time for final results will be undoubtedly decreased. Badly, authors of existing acceleration methods [17], [21], [22], [25], [32] do not propose any method to complete the task because all their acceleration schemes depend on the exact values of σ_r, σ_s. That is to say, existing methods cannot perform any calculation without knowing (σ_r, σ_s). Unlike them, our method is able to perform pre-computation with little modification. Strictly speaking, our algorithm also cannot determine the exact value of T_r before knowing σ_r, but we can make a tiny modification for the original algorithm to pre-compute the value of $F_c(\mathbf{y}, z)$

and $F_s(\mathbf{y}, z)$ with respect to some predefined values $\{T_{r_i}\}$. The specific reason is that $F_c(\mathbf{y}, z)$ and $F_s(\mathbf{y}, z)$ in (36) only involves an unknown parameter T_r indicating the truncated region of the range kernel $K_r(x)$. Once the exact value of T_r is known, we can always take the minimum T_{r_i} such that $T_{r_i} \geq T_r$ from the predefined set $\{T_{r_i}\}$ to replace T_r. In plain English, this is equivalent to extend the truncated region from $[-T_r, T_r]$ to $[-T_{r_i}, T_{r_i}]$ and employ $\cos(\frac{\pi k}{T_{r_i}} x)$ to approximate the values of $K_r(x)$ on the region $[-T_{r_i}, T_{r_i}]$. Since $F_c(\mathbf{y}, z)$ and $F_s(\mathbf{y}, z)$ have been figured out during the time loading images, our algorithm only need to perform box filtering which is a fast computation. Hence, without the computational burden for $F_c(\mathbf{y}, z)$ and $F_s(\mathbf{y}, z)$, the actually filtering time can be reduced from about $1.37s$ to about $0.69s$ according to our experiment. At last, to describe the pre-computation algorithm more formally, we outline the major computation steps of our pre-computation algorithm in the following:

- Pre-computation while loading an image into the manipulation software
   1) Input image I and predefined values $\{T_{r_i}\}$.
   2) Calculate a set of $F_c(\mathbf{y}, z)$ and $F_s(\mathbf{y}, z)$ according to I and the values in $\{T_{r_i}\}$.
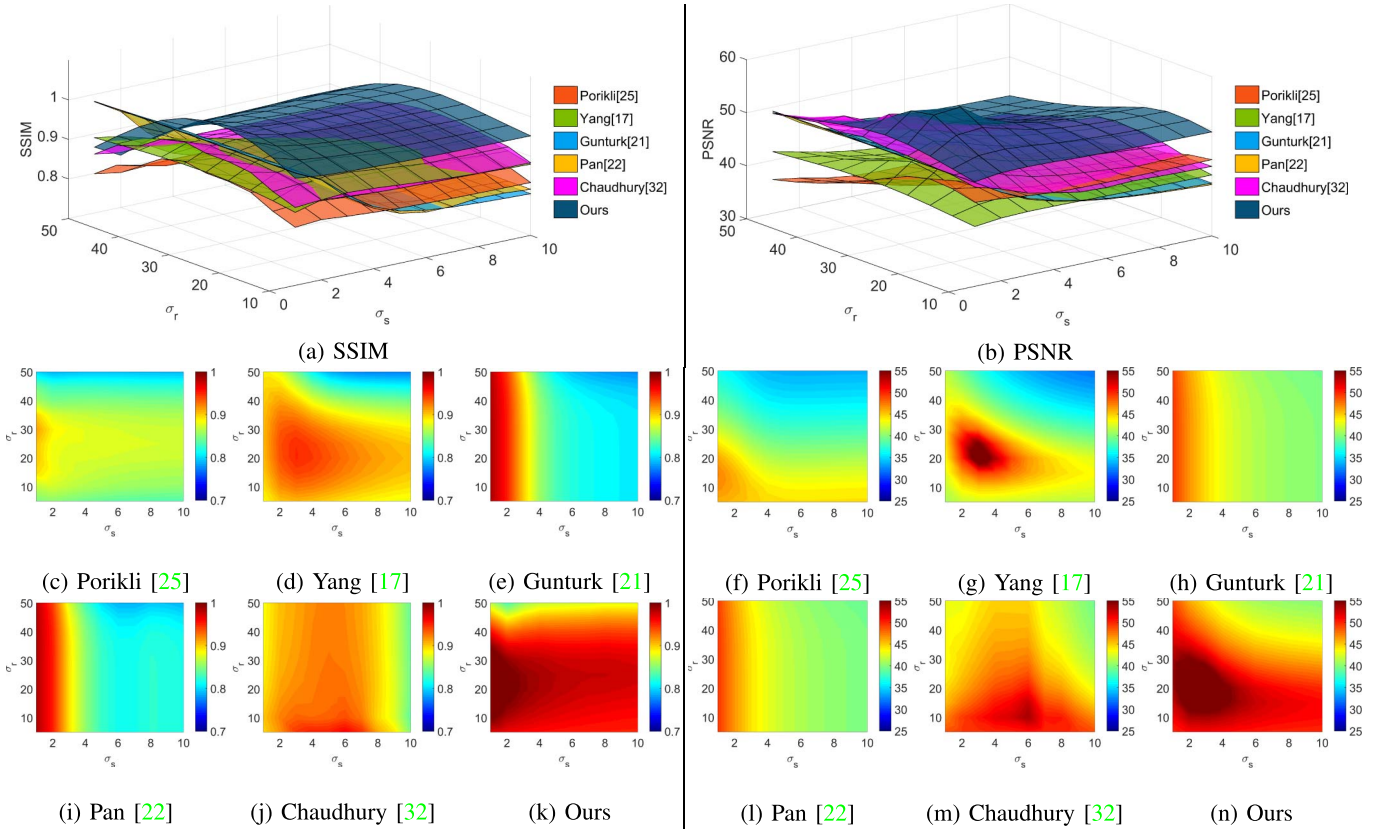
Fig. 10. Qualitative evaluation for the filtering results of six methods, where SSIM and PSNR are chosen as the quantitative index to measure the approximation error. (a) (b) illustrate the SSIM and PSNR approximation error surfaces. (c)-(n) plot the the SSIM and PSNR indices of each method, where the left part shows SSIM, and the right part shows PSNR.

- Filtering procedure after specifying the parameter (σ_r, σ_s)
    1) Find the minimum T_{r_i} such that $T_{r_i} \geq 3\sigma_r$ from the predefined set $\{T_{r_i}\}$.
    2) Compute the coefficients a_k of the best N-term approximation for the range kernel on the range $[-T_{r_i}, T_{r_i}]$.
    3) Put the pre-computed $F_c(\mathbf{y}, z)$ and $F_s(\mathbf{y}, z)$ corresponding to T_{r_i} as well as a_k into our fast computation formulas listed in Section IV-D to yield final filtering results.

Case 2 [Both Input Image I and Parameters Setting (σ_r, σ_s) Are Given]: This situation implies that we need to process input image immediately. Taking a close look at (2) (4) (8), we can find that all acceleration methods involve exponential function $\exp(x)$ or trigonometric functions $\sin(x), \cos(x)$. As we known, both exponential function and trigonometric function are time-consuming operations. Since the value of input image is discrete, we can pre-compute a look-up table for the value of exponential/trigonometric functions on these discrete points to speed up the computation further. It is also worth noting that the major reason that slows down the brute-force implementation of BF is also the exponential function used in (1). Taking the look-up table technique, we can accelerate the brute-force implementation too. However, we have to point out that the look-up table technique cannot reduce the computational complexity of the brute-force implementation and thus its run
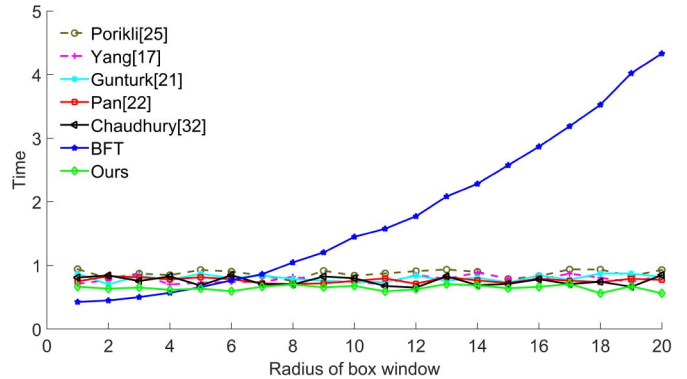

Fig. 11. Run time illustration, where the abscissa axis denotes the radius of box window and the ordinate axis presents the run time of each method. BFT achieves the smallest run time when the radius is small and increases its run time with the radius of box window. In contrast, run times of other acceleration algorithms are constants and therefore are not dependent on window's radius.

time still depends on the size of box window $\mathcal{N}_x$ of BF. Fig 11 illustrates the run time of seven methods with respect to different window radii. We can plot this figure because Table V implies that the speed of different acceleration methods is rather robust for different parameter settings, thus the run time at arbitrary parameter settings such as $(\sigma_r = 40, \sigma_s = 5)$ can stand for the performance of one kind of acceleration method. Here the seven methods include Porikli [25], Yang *et al.* [17],

TABLE VI

SUPPORT REGIONS AND THE SIZES OF THE BOX FILTERS
USED TO REPRESENT 2D HAAR FUNCTIONS

	Support Region	Support Region's Size
$\ddot{B}(\boldsymbol{x})$	$A \times A$	$2T_s \times 2T_s$
$\ddot{B}^1_{0,j_2,k_2}(\boldsymbol{x})$	$A \times A^{j_2,k_2}_1$	$2T_s \times 2^{-j_2+1}T_s$
$\ddot{B}^2_{0,j_2,k_2}(\boldsymbol{x})$	$A \times A^{j_2,k_2}_2$	$2T_s \times 2^{-j_2+1}T_s$
$\ddot{B}^1_{j_1,k_1,0}(\boldsymbol{x})$	$A^{j_1,k_1}_1 \times A$	$2^{-j_1+1}T_s \times 2T_s$
$\ddot{B}^2_{j_1,k_1,0}(\boldsymbol{x})$	$A^{j_1,k_1}_2 \times A$	$2^{-j_1+1}T_s \times 2T_s$
$\ddot{B}^1_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$	$A^{j_1,k_1}_1 \times A^{j_2,k_2}_1$	$2^{-j_1+1}T_s \times 2^{-j_2+1}T_s$
$\ddot{B}^2_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$	$A^{j_1,k_1}_1 \times A^{j_2,k_2}_2$	$2^{-j_1+1}T_s \times 2^{-j_2+1}T_s$
$\ddot{B}^3_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$	$A^{j_1,k_1}_2 \times A^{j_2,k_2}_2$	$2^{-j_1+1}T_s \times 2^{-j_2+1}T_s$
$\ddot{B}^4_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$	$A^{j_1,k_1}_2 \times A^{j_2,k_2}_1$	$2^{-j_1+1}T_s \times 2^{-j_2+1}T_s$

Gunturk [21], Pan *et al.* [22], Chaudhury [32], ours as well as the brute-force implementation accelerated by the look-up table (BFT). From Fig 11, we can verify that the run time of BFT increases with the radius of the box window $\mathcal{N}_x$. In addition, when the size of window is small, the fastest method is BFT. The reason is that the overall computational cost of BFT is dominated by the size of the window $\mathcal{N}_x$, so smaller window size, lower computational cost and the run time will increase with the window size. Unlike BFT, the run time of all acceleration methods do not change with the window size because their computational complexity is $O(|I|)$. Note that disregarding the small window case, our acceleration algorithm overwhelms other acceleration methods.

VI. CONCLUSION

In this paper, we propose a unified framework to accelerate BF with arbitrary spatial and range kernels. Unlike previous approaches, our method jointly employs five techniques: truncated spatial and range kernels, the best N-term approximation of these kernels as well as three existing acceleration techniques to speed up BF. It thus can inherit the advantage of previous acceleration algorithms while avoiding the problems of previous approaches. Moreover, we first transform 2D BF into a set of 3D box filters and disclose that BF can be fast computed by the 3D SAT. Taking advantage of the expressive orthogonal functions used in the best N-term approximation scheme, our approach employs fewer terms which means faster computing speed, while obtaining more gratifying filtering results among a great wide of parameter settings. More importantly, the strength of our method has been verified by several carefully designed experiments. All experiments indicate that our method outperforms other methods. Especially, the filtering accuracy is significantly improved without sacrificing efficiency.

APPENDIX A
BOX FILTER REPRESENTATION

All functions in the four sets $\{\phi(x)\phi(y)\}$, $\{\phi(x)\psi_{j_2,k_2}(y)\}$, $\{\psi_{j_1,k_1}(x)\phi(y)\}$ $\{\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)\}$ can be represented by a linear combination of box filters. First, $\phi(x)$ achieves 1 in the support region $A = [-T_s, T_s]$. Similarly, let $Z_k = \text{sign}(k)(2|k| - 1)T_s$, $A^{j,k}_1 = [2^{-j}(Z_k - T_s), 2^{-j}Z_k]$ and
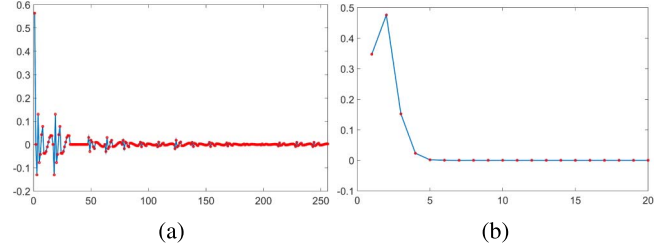

Fig. 12. Visual illustration for the amplitude of the decomposition coefficients of the spatial kernel $G_5(x)$ and range kernel $G_{40}(x)$ on the Haar and trigonometric basis. (a) Haar decomposition. (b) Cosine decomposition.

$A^{j,k}_2 = [2^{-j}Z_k, 2^{-j}(Z_k + T_s)]$, the support region of $\psi_{j,k}(x)$ can be presented as $A^{j,k}_1 \bigcup A^{j,k}_2$ because $\psi_{j,k}(x) = 1$ for $x \in A^{j,k}_1$ and $\psi_{j,k}(x) = -1$ for $x \in A^{j,k}_2$. Second, $\phi(x)\phi(y)$ represents a 2D box function $\ddot{B}(\boldsymbol{x})$ with the support region $A \times A$ as $\phi(x)$ is constant in the region A. Unlike $\phi(x)\phi(y)$, $\phi(x)\psi_{j_2,k_2}(y)$ and $\psi_{j_1,k_1}(x)\phi(y)$ are binary functions equal to 1 in the regions $A \times A^{j_2,k_2}_1$, $A^{j_1,k_1}_1 \times A$ and obtain -1 in the regions $A \times A^{j_2,k_2}_2$, $A^{j_1,k_1}_2 \times A$, respectively. Hence putting $\ddot{B}^1_{0,j_2,k_2}(\boldsymbol{x})$, $\ddot{B}^2_{0,j_2,k_2}(\boldsymbol{x})$, $\ddot{B}^1_{j_1,k_1,0}(\boldsymbol{x})$ and $\ddot{B}^2_{j_1,k_1,0}(\boldsymbol{x})$ be four 2D box functions defined on the support regions $A \times A^{j_2,k_2}_1$, $A \times A^{j_2,k_2}_2$, $A^{j_1,k_1}_1 \times A$ and $A^{j_1,k_1}_2 \times A$ respectively, we thus have $\phi(x)\psi_{j_2,k_2}(y) = \ddot{B}^1_{0,j_2,k_2}(\boldsymbol{x}) - \ddot{B}^2_{0,j_2,k_2}(\boldsymbol{x})$, $\psi_{j_1,k_1}(x)\phi(y) = \ddot{B}^1_{j_1,k_1,0}(\boldsymbol{x}) - \ddot{B}^2_{j_1,k_1,0}(\boldsymbol{x})$. Similar to $\phi(x)\psi_{j_2,k_2}(y)$ and $\psi_{j_1,k_1}(x)\phi(y)$, $\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)$ is also a binary function. Unlike them, $\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)$ equals to 1 on the regions $A^{j_1,k_1}_1 \times A^{j_2,k_2}_1$, $A^{j_1,k_1}_2 \times A^{j_2,k_2}_2$ and obtains -1 on the region $A^{j_1,k_1}_1 \times A^{j_2,k_2}_2$, $A^{j_1,k_1}_2 \times A^{j_2,k_2}_1$. Let $\ddot{B}^1_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$, $\ddot{B}^2_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$, $\ddot{B}^3_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$ and $\ddot{B}^4_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$ be four 2D box functions with the support regions $A^{j_1,k_1}_1 \times A^{j_2,k_2}_1$, $A^{j_1,k_1}_1 \times A^{j_2,k_2}_2$, $A^{j_1,k_1}_2 \times A^{j_2,k_2}_2$ and $A^{j_1,k_1}_2 \times A^{j_2,k_2}_1$, respectively, we can reformulate $\psi_{j_1,k_1}(x)\psi_{j_2,k_2}(y)$ as $\ddot{B}^1_{j_1,k_1,j_2,k_2}(\boldsymbol{x}) - \ddot{B}^2_{j_1,k_1,j_2,k_2}(\boldsymbol{x}) + \ddot{B}^3_{j_1,k_1,j_2,k_2}(\boldsymbol{x}) - \ddot{B}^4_{j_1,k_1,j_2,k_2}(\boldsymbol{x})$. Finally, we list the support regions and the size of the box filters used to represent 2D Haar functions in Table VI for reference.

Given all this, we can reexpress $K_s(\|\boldsymbol{x}\|)$ in (30) as

$$
\begin{aligned}
K_s(\|\boldsymbol{x}\|) \approx &\sum_{c_0 \in \Lambda^s_1} c_0 B(\boldsymbol{x}) \\
&+ \sum_{c_{0,j_2,k_2} \in \Lambda^s_2} c_{0,j_2,k_2} \sum_{i=1}^{2} (-1)^{i+1} B^i_{0,j_2,k_2}(\boldsymbol{x}) \\
&+ \sum_{c_{j_1,k_1,0} \in \Lambda^s_3} c_{j_1,k_1,0} \sum_{i=1}^{2} (-1)^{i+1} B^i_{j_1,k_1,0}(\boldsymbol{x}) \\
&+ \sum_{c_{j_1,k_1,j_2,k_2} \in \Lambda^s_4} c_{j_1,k_1,j_2,k_2} \sum_{i=1}^{4} (-1)^{i+1} B^i_{j_1,k_1,j_2,k_2}(\boldsymbol{x})
\end{aligned}
$$

$$(37)$$

which is consisted of several box filters. Further, let $\ddot{B}_j(\boldsymbol{x})$ present the 2D box functions used in (37), c_j be the corresponding coefficients of $\ddot{B}_j(\boldsymbol{x})$ and Λ^s stands for the collection of c_j, $N^j_{\boldsymbol{x}}$ be the support region of $\ddot{B}_j(\boldsymbol{x})$, we have (31).

TABLE VII

THE CORRESPONDING RELATIONSHIP BETWEEN N AND M FOR SPATIAL KERNEL $G_5(x)$ AND RANGE KERNEL $G_{40}(x)$

Spatial kernel	N	1	2	3	4	5	6	7	8
	M	1	19	19	19	19	19	19	19
Range kernel	N	1	2	3	4	5	6	7	8
	M	2	2	3	4	5	6	7	8

APPENDIX B
HOW TO COMPUTE THE BEST N-TERM APPROXIMATION?

Theoretically, finding the best N-term approximation of a given function requires us to minimize the objective function (11). However, in practice the minimizing is not necessary because the best N-term approximation for arbitrary functions can be obtained by selecting the first N largest coefficients and the amplitude of coefficients attenuates very fast. The two points imply that the first N largest coefficients must reside in the first M coefficients, where M is a constant which is equal or greater than N. More clearly speaking, to find the best N-term approximation, we only need to calculate the first M coefficients and then pick up the first N largest coefficients from the first M coefficients.

As an example, Fig 12 illustrates the coefficients on the Haar basis and trigonometric basis for the Gaussian function respectively, where the abscissa axis denotes the linear order of each basis functions, the ordinate axis represents the amplitude of coefficient of each basis functions. Note that since each Haar function $\psi_{i,j}$ is identified by two indices i, j, we order each Haar function according to following rules: for two Haar functions ψ_{i_1,j_1}, ψ_{i_2,j_2}, if $i_1 < i_2$ or $i_1 = i_2$ and $j_2 < j_2$, ψ_{i_1,j_1} is putted before ψ_{i_2,j_2}; otherwise, ψ_{i_1,j_1} is located after ψ_{i_2,j_2}. As for the trigonometric basis, all coefficients of sine functions are zeros, we thus sort the cosine functions $\cos(k\omega x)$ according the index k. From Figs 12a 12b, we can observe that both the coefficients of Haar basis functions and the coefficients of trigonometric basis functions decay dramatically. The corresponding relationship between N and M for spatial kernel $G_5(x)$ and range kernel $G_{40}(x)$ is listed in Table VII which tells us the minimal number M of the first M coefficients containing the first N largest coefficients. From this table we can reasonably conclude that finding the best N-term approximation only needs to figure out the first $20N$ coefficients and then to select the first N largest coefficients and their corresponding coefficients.
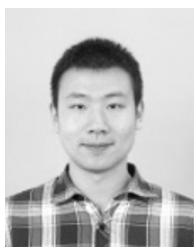
REFERENCES

[1] S. Paris, P. Kornprobst, and J. Tumblin, *Bilateral Filtering*. Hanover, MA, USA: Now Publishers, 2009.

[2] V. Aurich and J. Weule, "Non-linear Gaussian filters performing edge preserving diffusion," in *Proc. DAGM-Symp.*, 1995, pp. 538–545.

[3] S. M. Smith and J. M. Brady, "SUSAN—A new approach to low level image processing," *Int. J. Comput. Vis.*, vol. 23, no. 1, pp. 45–78, May 1997.

[4] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. Int. Conf. Comput. Vis.*, Jan. 1998, pp. 839–846.

[5] R. A. Haddad and A. N. Akansu, "A class of fast Gaussian binomial filters for speech and image processing," *IEEE Trans. Signal Process.*, vol. 39, no. 3, pp. 723–727, Mar. 1991.

[6] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 257–266, Jul. 2002.

[7] Q. Yang, "Hardware-efficient bilateral filtering for stereo matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 1026–1032, May 2014.

[8] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 664–672, Aug. 2004.

[9] M. Elad, "Retinex by two bilateral filters," in *Proc. Scale-Space Conf.*, 2005, pp. 217–229.

[10] A. Ansar, A. Castano, and L. Matthies, "Enhanced real-time stereo using bilateral filtering," in *Proc. 2nd Int. Symp. 3D Data Process., Visualizat., Transmiss.*, Sep. 2004, pp. 455–462.

[11] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," *Trans. Graph.*, vol. 25, no. 3, pp. 1221–1226, Jul. 2006.

[12] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, New York, NY, USA, 1984, pp. 207–212.

[13] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high-dimensional filtering," *Trans. Graph.*, vol. 28, no. 3, pp. 21:1–21:12, Aug. 2009.

[14] A. Adams, J. Baek, and M. Davis, "Fast high-dimensional filtering using the permutohedral lattice," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 753–762, 2010.

[15] S. Yoshizawa, A. Belyaev, and H. Yokota, "Fast Gauss bilateral filtering," *Comput. Graph. Forum*, vol. 29, no. 1, pp. 60–74, Mar. 2010.

[16] T. Q. Pham and L. J. van Vliet, "Separable bilateral filtering for fast video preprocessing," in *Proc. Int. Conf. Multimedia Expo*, Jul. 2005, pp. 1–4.

[17] Q. Yang, K.-H. Tan, and N. Ahuja, "Real-time O(1) bilateral filtering," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 557–564.

[18] R. Deriche, "Recursively implementing the Gaussian and its derivatives," INRIA, France, Tech. Rep. RR-1893, 1993. [Online]. Available: https://hal.inria.fr/inria-00074778

[19] P. Getreuer, "A survey of Gaussian convolution algorithms," *Image Process. Line*, vol. 3, pp. 286–310, Dec. 2013.

[20] K. Zhang, G. Lafruit, R. Lauwereins, and L. Van Gool, "Constant time joint bilateral filtering using joint integral histograms," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4309–4314, Sep. 2012.

[21] B. Gunturk, "Fast bilateral filter with arbitrary range and domain kernels," *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2690–2696, Sep. 2011.

[22] S. Pan, X. An, and H. He, "Optimal O(1) bilateral filter with arbitrary spatial and range kernels using sparse approximation," *Math. Problems Eng.*, vol. 2014, Feb. 2014, Art. no. 289517.

[23] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," CS Technion, Israel, Tech. Rep. vol.40,no.8, 2008.

[24] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations," *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, Nov. 1988.

[25] F. Porikli, "Constant time O(1) bilateral filtering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.

[26] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Int. J. Comput. Vis.*, vol. 81, no. 1, pp. 24–52, Jan. 2009.

[27] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *Trans. Graph.*, vol. 26, no. 3, Jul. 2007, Art. no. 103.

[28] S. An, F. Boussaid, M. Bennamoun, and F. Sohel, "Quantitative error analysis of bilateral filtering," *IEEE Signal Process. Lett.*, vol. 22, no. 2, pp. 202–206, Feb. 2015.

[29] K. N. Chaudhury, "Constant-time filtering using shiftable kernels," *IEEE Signal Process. Lett.*, vol. 18, no. 11, pp. 651–654, Nov. 2011.

[30] K. N. Chaudhury, D. Sage, and M. Unser, "Fast O(1) bilateral filtering using trigonometric range kernels," *IEEE Trans. Image Process.*, vol. 20, no. 12, pp. 3376–3382, Dec. 2011.

[31] L. Dai, M. Yuan, and X. Zhang, "Accelerate bilateral filter using Hermite polynomials," *Electron. Lett.*, vol. 50, no. 20, pp. 1432–1434, Sep. 2014.

[32] K. N. Chaudhury, "Acceleration of the shiftable O(1) algorithm for bilateral filtering and nonlocal means," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1291–1300, Apr. 2013.

[33] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," *Math. Ann.*, vol. 69, no. 3, pp. 331–371, Sep. 1910.

IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 25, NO. 6, JUNE 2016

[34] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[35] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient visual event detection using volumetric features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 1. Oct. 2005, pp. 166–173.

[36] E. Tapia, "A note on the computation of high-dimensional integral images," *Pattern Recognit. Lett.*, vol. 32, no. 2, pp. 197–201, Jan. 2011.

[37] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Proc. Eur. Conf. Comput. Vis.*, vol. 3952. 2006, pp. 589–600.

[38] Y. Li, H. Li, and Z. Cai, "Fast orthogonal Haar transform pattern matching via image square sum," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 9, pp. 1748–1760, Sep. 2014.

Mengke Yuan received the B.S. and M.S. degrees in mathematics from Zhengzhou University, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree in computer science with the Institute of Automation, Chinese Academy of Sciences. His research interests lie in computer vision, optimization-based techniques for image analysis and synthesis, and machine learning.

Longquan Dai received the B.S. degree in electronics engineering from the Henan University of Technology, China, in 2006, and the M.S. degree in electronics engineering from Shantou University, China, in 2010. He is currently pursuing the Ph.D. degree in computer science with the Institute of Automation, Chinese Academy of Sciences, China. His research interests lie in computer graphics, computer vision, and optimization-based techniques for image analysis and synthesis.

Xiaopeng Zhang (M'11) received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 1984 and 1987, respectively, and the Ph.D. degree in computer science from the Institute of Software, Chinese Academy of Sciences, Beijing, China, in 1999. He is currently a Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His main research interests are computer graphics and computer vision.