# Recursive Bilateral Filtering*

Qingxiong Yang

Department of Computer Science,
City University of Hong Kong, Hong Kong, China
http://www.cs.cityu.edu.hk/~qiyang/

**Abstract.** This paper proposes a recursive implementation of the bilateral filter. Unlike previous methods, this implementation yields an bilateral filter whose computational complexity is linear in both input size and dimensionality. The proposed implementation demonstrates that the bilateral filter can be as efficient as the recent edge-preserving filtering methods, especially for high-dimensional images. Let the number of pixels contained in the image be $N$, and the number of channels be $D$, the computational complexity of the proposed implementation will be $O(ND)$. It is more efficient than the state-of-the-art bilateral filtering methods that have a computational complexity of $O(ND^2)$ [1] (linear in the image size but polynomial in dimensionality) or $O(N\log(N)D)$ [2] (linear in the dimensionality thus faster than [1] for high-dimensional filtering). Specifically, the proposed implementation takes about 43 ms to process a one megapixel color image (and about 14 ms to process a 1 megapixel grayscale image) which is about 18× faster than [1] and 86× faster than [2]. The experiments were conducted on a MacBook Air laptop computer with a 1.8 GHz Intel Core i7 CPU and 4 GB memory. The memory complexity of the proposed implementation is also low: as few as the image memory will be required (memory for the images before and after filtering is excluded). This paper also derives a new filter named *gradient domain bilateral filter* from the proposed recursive implementation. Unlike the bilateral filter, it performs bilateral filtering on the gradient domain. It can be used for edge-preserving filtering but avoids sharp edges that are observed to cause visible artifacts in some computer graphics tasks. The proposed implementations were proved to be effective for a number of computer vision and computer graphics applications, including stylization, tone mapping, detail enhancement and stereo matching.

## 1 Introduction

The bilateral filter is a robust edge-preserving filter introduced by Tomasi and Manduchi [3]. It has been used in many computer vision and computer graphics tasks, and a general overview of the applications can be found in [4]. A bilateral

* This work was supported by a startup grant from the City University of Hong Kong under Project 7200250. The source code is available on the author's webpage.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part I, LNCS 7572, pp. 399–413, 2012.

filter has two filter kernels: a spatial filter kernel and a range kernel for measuring the spatial and range distance between the center pixel and its neighbors, respectively. The two filter kernels are traditionally based on a Gaussian distribution [3], [4], [5], [6]. Being non-linear, the brute force implementations of the bilateral filter are slow when the kernel is large (larger than $5 \times 5$ [5]).

In the literature, several techniques have been proposed for fast bilateral filtering. Pham and Vliet [7] implemented the bilateral filter as a separable operation. The cost of this approach is still high for large kernels. By constraining the spatial filter kernel to box filter kernel, Weiss [8] showed that the result depends only on the histogram of the neighborhood, and exposed an efficient algorithm to compute the histogram of the square neighborhoods of an image. Unfortunately, this method works efficiently only on grayscale images. Durand and Dorsey [6] linearized the bilateral filter by quantizing the range domain to a small number of discrete values. This method can process a one megapixel grayscale image in about one second with additional quantization on the spatial domain. Paris and Durand [5] extended this method and represented the grayscale image in a volumetric data structure that they named the *bilateral grid*. The bilateral filtering then exactly corresponds to convolving the grid with a 3D Gaussian. This bilateral filtering method can be directly extended to 5D for color images. The use of bilateral grid increases the accuracy of [6] when the spatial domain quantization is included. Its parallel implementation [9] demonstrated real-time grayscale image filtering performance even for high-resolution images. However, the memory cost maybe unacceptable when filter kernel is small.

The computational complexity of all above methods still depends on the size of the filter kernel. Use integral histogram [10], Porikli [11] is the first to remove this dependency by demonstrating that the computational complexity of local histogram is independent of the region (filter) size. Hence, Weiss's method [8] can be computed linearly in the number of image pixels for grayscale images. Porikli [11] also proposed a Taylor series based solution to remove the box filter kernel constraint. Unfortunately, high-order derivatives are required for shape range functions. Yang *et al.* [12] also demonstrated that Durand and Dorsey's [6] bilateral filtering method can be implemented using recursive Gaussian filter so that its computational complexity will be independent of the filter kernel size. Methods proposed in [11] and [12] can be extended for color images, but the computational complexity will be exponential in the color dimension. Adams *et al.* [2] then proposed to use Gaussian KD-trees for efficient high-dimensional Gaussian filtering. Let $N$ denote the number of image pixels, and $D$ denote the number of channels, its computational complexity is $O(N \log(N) D)$ and the memory complexity is $O(ND)$. This method can be directly integrated with Paris and Durand [5]'s method for fast bilateral filtering. Adams *et al.* [13], [1] later proposed to use permutohedral lattice for bilateral filtering, which has a computational complexity of $O(ND^2)$ and is faster than their Gaussian KD-trees [2] for relatively lower dimensionality but has a higher memory cost. Nevertheless, these methods all rely on quantization and may sacrifice accuracy for speed.

In this paper, a recursive implementation of the bilateral filter is proposed. Unlike previous methods, this implementation yields an exact bilateral filter. The proposed method is related to Weiss's method [8] in that instead of the spatial filter kernel, the range filter kernel is constrained in the proposed implementation. A traditional range kernel measures the range distance between the center pixel $p$ and another pixel $q$ based on their color difference. However, the proposed method measures the range distance between $p$ and $q$ by accumulating the color difference between every two neighboring pixels on the path between $p$ and $q$. For any bilateral filter containing this new range filter kernel and *any spatial filter kernel that can be recursively implemented*, an exact recursive implementation can be obtained by simply altering the coefficients of the recursive system defined by the spatial filter kernel at each pixel location.

The computational and memory complexity of the proposed recursive implementation are both $O(ND)$. It is more efficient than the state-of-the-art bilateral filtering methods that have a computational complexity of $O(ND^2)$ [1] or $O(N\log(N)D)$ [2]. Specifically, the proposed implementation takes about 43 ms to process a one megapixel color image (and about 14 ms to process a 1 megapixel grayscale image) which is about 18× faster than [1] and 86× faster than [2]. The experiments were conducted on a MacBook Air laptop computer with a 1.8 GHz Intel Core i7 CPU and 4 GB memory. The memory complexity of the proposed implementation is also low: as few as the image memory will be required (memory for the images before and after filtering is excluded).

Inspired by the bilateral filter, a number of edge-preserving filtering methods that have similar applications but lower computational complexity emerged recently. For instance, Fattal's [14] EAW method, He *et al.*'s [15] guided filtering method, and Gastal and Oliveira's [16] Domain transform filtering method. Domain transform filtering is the fastest among all (about 15× faster than the guided filter). Let $\mathcal{T}$ denote the runtime of applying a $1^{st}$-order recursive filter to an input image, the amount of time spent on the domain transform computation will be about $2.24\mathcal{T}$. The computational complexity of the proposed recursive bilateral filter is the same as the domain transform filtering method. In a $1^{st}$-order recursive bilateral filter, the coefficients of the $1^{st}$-order recursive filter have to be modified at each pixel location, and the runtime of this extra computation will be as few as $0.11\mathcal{T}$. As a result, it can be even faster than the domain transform filtering.

Besides demonstrating that the bilateral filter has an exact algorithm that can be as efficient as the recent edge-preserving filtering methods, a new filter named *gradient domain bilateral filter* (**GDBF**) is derived in this paper. It is obtained with a small modification of the proposed $1^{st}$-order recursive bilateral filter. As the name suggests, GDBF essentially performs bilateral filtering on the gradient domain, while both the original $1^{st}$-order recursive bilateral filter and the traditional bilateral filter perform bilateral filtering on the range domain. GDBF enforces smoothness in the gradient domain, thus obtains more natural results that are required in some computer graphics tasks like detail enhancement. This is not true for some computer vision tasks requiring share edges like depth from stereo. In this case, the proposed recursive bilateral filters are preferred. Recursive bilateral filters also outperform the traditional bilateral filter on standard stereo benchmark [17] due to the use of new range filter kernel.

## 2 Recursive Filtering

This section gives a brief overview of recursive filters [18] with a focus on two: $1^{st}$-order recursive filter that is the simplest recursive filter and $2^{nd}$-order recursive filter that can be used to approximate the Gaussian filter.

### 2.1 One-Dimensional Recursive Filtering

Let $x$ denote the one-dimensional (1D) input signal of a causal recursive system of order $n$, and $y$ denote the output, then

$$y_i = \sum_{l=0}^{n-1}(a_l \cdot x_{i-l}) - \sum_{k=1}^{n}(b_k \cdot y_{i-k}). \tag{1}$$

This recursive system is then characterized by the following transfer function [19]

$$H^a(Z) = \frac{\sum_{l=0}^{n-1} a_l \cdot Z^{-l}}{1 + \sum_{k=1}^{n} b_k \cdot Z^{-k}}, \tag{2}$$

$$= \sum_{k=0}^{+\infty} h_k^a \cdot Z^{-k}, \tag{3}$$

where $\{h_k^a\}$ denote the impulse response of the recursive system whose Z-transform is $H^a(Z)$.

Given any user-specified filter with impulse response sequence $\{h_k\}$, the problem of recursive implementation of this filter deals with the determination of the coefficients $a_l$ and $b_k$ of Eq. (2) such that $\{h_k^a\}$ in Eq. (3) is exactly, or best approximates $\{h_k\}$ by minimizing

$$E = \sum_{k=0}^{+\infty}(h_k^a - h_k)^2. \tag{4}$$

Let $x$ denote a scanline extracted from a 2D grayscale image, then the response $y_i$ computed from Eq. (1) at pixel $i$ only takes into account supports coming from the pixels on the left side of pixel $i$. To include supports from the pixels on the right side of pixel $i$, an anticausal recursive filter of the same order is required for computing responses from right to the left but is omitted here for simplicity.

### 2.2 $1^{st}$-Order and $2^{nd}$-Order Recursive Filtering

The simplest recursive filter is the $1^{st}$-order recursive filter. According to Eq. (1), the output of a $1^{st}$-order recursive filter is

$$y_i = a_0 \cdot x_i - b_1 \cdot y_{i-1}. \tag{5}$$

According to [16], $a_0 = 1 - a$ and $b_1 = -a$ , Eq. (5) can then be rewritten as

$$y_i = (1 - a) \cdot x_i + a \cdot y_{i-1} \tag{6}$$

where coefficient $a \in [0, 1]$ is a feedback coefficient [18].

Using $2^{nd}$-order recursive implementation, Deriche [20] demonstrated that Gaussian filter can be computed linearly in the number of pixels in an image. The equation of a Gaussian function in one dimension is

$$G(i) = \frac{1}{\sqrt{2\pi\sigma_S^2}} exp(-\frac{i^2}{2\sigma_S^2}), \tag{7}$$

where $\sigma_S$ is the standard deviation of the Gaussian distribution.

According to Eq. (1), the output of a $2^{nd}$-order causal recursive filter is

$$y_i = a_0 \cdot x_i + a_1 \cdot x_{i-1} - b_1 \cdot y_{i-1} - b_2 \cdot y_{i-2}, \tag{8}$$

and the coefficients suggested by Deriche [20] are

$$a_0 = (1 - e^{-\frac{1.695}{\sigma_S}})^2/(1 + 3.39e^{-\frac{1.695}{\sigma_S}}/\sigma_S - e^{-\frac{3.39}{\sigma_S}}), \tag{9}$$

$$a_1 = (1.695/\sigma_S - 1)e^{-\frac{1.695}{\sigma_S}} a_0, \tag{10}$$

$$b_1 = -2e^{-\frac{1.695}{\sigma_S}}, \tag{11}$$

$$b_2 = e^{-\frac{3.39}{\sigma_S}}. \tag{12}$$

As discussed in Sec. 2.1, an anticausal recursive filter of the same order with the following output will be required

$$y_i^a = a_2 \cdot x_{i+1} + a_3 \cdot x_{i+2} - b_1 \cdot y_{i+1}^a - b_2 \cdot y_{i+2}^a, \tag{13}$$

where $a_2 = (1.695/\sigma_S + 1)e^{-\frac{1.695}{\sigma_S}} a_0$ and $a_3 = -a_0 b_2$. Results obtained from the causal and anticausal recursive filters will finally be merged together to form the output of a symmetric 1D Gaussian filter.

## 3 Recursive Bilateral Filtering

A brief overview of the bilateral filter [3], [4] is given in Sec. 3.1 and a recursive implementation of the bilateral filter is proposed in Sec. 3.2-3.3. The complexity of the proposed implementation is then discussed in Sec. 3.4.

### 3.1 Bilateral Filtering

The bilateral filtering is the combination of the spatial and range filtering by enforcing both geometric and photometric locality. Let $x$ denote a scanline of a 2D grayscale image, the 1D bilateral filtered value of $x$ at pixel $i$ is

$$y_i = \sum_{k=0}^{i} R_{k,i}\mathcal{S}_{k,i} \cdot x_k, \tag{14}$$

where $R_{k,i} = R(x_k, x_i)$ is the range filter kernel for measuring the range similarity of pixel $k$ and $i$ and $\mathcal{S}_{k,i} = \mathcal{S}(k,i)$ is the spatial filter kernel for measuring their spatial similarity. Note that the normalization factor $\sum_{k=0}^{i} R_{k,i}\mathcal{S}_{k,i}$ is omitted in Eq. (14) as it can be directly computed from Eq. (14) by setting $x_k$ to be all ones.

### 3.2 1D Recursive Bilateral Filtering

In this section, a recursive implementation of the bilateral filter is proposed by confining the range filtering kernel with the following property:

$$R_{k,i} = R_{i,k} = R_{k,k+1}R_{k+1,k+2}\cdots R_{i-2,i-1}R_{i-1,i} = \prod_{j=k}^{i-1} R_{j,j+1}. \tag{15}$$

In the literature [3], [4], [5], [6], the range filtering kernel is often Gaussian

$$R_{j,j+1} = R_{j+1,j} = exp(-\frac{|x_j - x_{j+1}|^2}{2\sigma_R^2}), \tag{16}$$

where $|x_j - x_{j+1}|^2$ denotes the range cost of traveling from pixel $j$ to $j+1$ (or from $j+1$ to $j$) and $R_{j,j} = 1$, then

$$R_{k,i} = \prod_{j=k}^{i-1} R_{j,j+1} = \prod_{j=k}^{i-1} exp(-\frac{|x_j - x_{j+1}|^2}{2\sigma_R^2}) = exp(-\frac{\sum_{j=k}^{i-1} |x_j - x_{j+1}|^2}{2\sigma_R^2}). \tag{17}$$

As can be seen in Eq. (15), the new range filter kernel $R_{k,i}$ measures the range distance between pixel $k$ and $i$ by accumulating the range distance between every two neighboring pixels on the path between $k$ and $i$.

Using the new range filtering kernel, a recursive implementation of the bilateral filter can be obtained with a small modification of the coefficients ($a_l$ and $b_k$) of the recursive system defined by the spatial filter kernel at each pixel location:

$$y_i = \sum_{l=0}^{n-1}(a_l^{new} \cdot x_{i-l}) - \sum_{k=1}^{n}(b_k^{new} \cdot y_{i-k}) = \sum_{l=0}^{n-1}(R_{i,i-l} \cdot a_l \cdot x_{i-l}) - \sum_{k=1}^{n}(R_{i,i-k} \cdot b_k \cdot y_{i-k}), \tag{18}$$

where $n \geq 1$.

The output of this modified recursive system is then

$$y_i = \sum_{k=0}^{i} R_{i,k} (\sum_{m=0}^{n-1} \lambda_{i-m-k} a_m) x_k, \tag{19}$$

where

$$\lambda_i = \begin{cases} 1 & i = 0, \\ \sum_{k=1}^{\min(i,n)} -b_k \lambda_{i-k} & i > 0, \\ 0 & i < 0, \end{cases} \tag{20}$$

with the initial condition that $y_0 = a_0 x_0$, and $x_i = 0$ when $i < 0$. Apparently, this is a bilateral filter where $R_{i,k}$ is the range filter kernel and $\mathcal{S}_{i,k} = \sum_{m=0}^{n-1} \lambda_{i-m-k} a_m$ is the spatial filter kernel. Eq. (19) can be proved using mathematical induction, and the detailed proof is provided on the author's webpage due to page limit.

**Claim 1.** *The proposed recursive implementation in Eq. (18) yields an exact bilateral filter with a range filter kernel described in Eq. (15) and a spatial filter kernel described by the recursive filter in Eq. (1).*

**Claim 1** is obtained directly from Eq. (19). It shows that any spatial filter with a recursive implementation (the coefficients $a_l$ and $b_k$ are determined) can be used in Eq. (18) to obtain a bilateral filter containing this spatial filter kernel and the range filter kernel in Eq. (15). However, only the $1^{st}$-order and $2^{nd}$-order recursive implementations are experimentally verified in this paper because the $1^{st}$-order recursive filter is the simplest recursive filter and the $2^{nd}$-order recursive filter can well approximate the Gaussian filter which is maybe the most used filter in computer vision.

### 3.3 2D Recursive Bilateral Filtering

Performing the proposed 1D recursive bilateral filter both horizontally and vertically extends the 1D filter to 2D. Assuming the horizontal pass is performed first, the vertical pass will be applied to the result produced by the horizontal one (and vice-versa). In this case, the range filtering kernel $R_{k,i}$ still corresponds to the range cost of traveling from $k$ to $i$ (or $i$ to $k$). However, there will be two possible traveling paths depending on whether the horizontal or vertical pass is performed first. We can choose the same path for every pixel or choose the path with the lowest traveling cost at each pixel location. The second option will double the runtime, thus the first option is used for all the experiments conducted in this paper.

### 3.4 Complexity Analysis

According to Eq. (1), a causal recursive system of order $n$ requires $2n$ multiplication operations and $2n - 1$ addition/subtraction operations. Because $R_{i,i-k} = R_{i,i-(k-1)} \cdot R_{i-(k-1),i-k}$ can be computed recursively, the $n^{th}$-order recursive implementation of the bilateral filter in Eq. (18) only requires $1 + (n-1) \cdot 3 =$

$3n-2$ additional multiplication operations and $n$ operations for measuring the range distance between two neighboring pixels ($j$ and $j-1$) with the assumption that all possible $R_{j,j-1}$ values are pre-computed and stored as a lookup table. As discussed in Sec. 2.1, most of the spatial filter has symmetric impulse responses, thus an additional anticausal recursive filter of the same order and same computational complexity will be required. For 2D signals, an extra pass (horizontally or vertically) will be required. Nevertheless, as a recursive implementation, the proposed method will be independent of the filter kernel size and only depends on the number of pixels in an image. Let the 2D signal be a 2D image with $N$ pixels and $D$ channels, the new range filter kernel in Eq. (16) is then

$$R_{j,j+1} = exp(-\frac{\sum_{k=0}^{D-1} |x_j^k - x_{j+1}^k|^2/D}{2\sigma_R^2}), \tag{21}$$

where $x^k$ is the $k^{th}$ channel of image $x$. As can be seen from Eq. (21), the computational complexity of proposed recursive implementation is $O(ND)$.

The memory cost is low for the proposed implementation. Besides a memory buffer for the input image, only two extra memory buffers of the same size will be required to store the outputs of the causal and anticausal recursive filters, and one of them will also be used to store the output image.

## 4 Gradient Domain Bilateral Filtering

This section shows that the simplest recursive implementation of the bilateral filter - $1^{st}$-order recursive bilateral filter - can be adjusted for performing the bilateral filtering on the gradient domain with a small modification of the coefficients ($a_l$ and $b_k$) of the $1^{st}$-order recursive system. The resulted filter is named the *gradient domain bilateral filter* (**GDBF**).

According to Eq. (18) and Eq. (6), the implementation of a $1^{st}$-order recursive bilateral filter with parameter $a \in [0,1]$ can be expressed as follows:

$$\begin{aligned} y_i &= R_{i,i} \cdot (1-a) \cdot x_i + R_{i,i-1} \cdot a \cdot y_{i-1} = (1-a) \cdot x_i + R_{i,i-1} \cdot a \cdot y_{i-1} \quad (22) \\ &= \sum_{k=0}^{i} R_{i,k} \cdot ((1-a)a^{i-k}) \cdot x_k \\ &= (1-a) \cdot \sum_{k=0}^{i} R_{i,k} \cdot a^{i-k} \cdot x_k. \quad (23) \end{aligned}$$

The proposed GDBF is obtained with a small modification to the recursive implementation in Eq. (22) so that

$$y_i = (1 - R_{i,i-1} \cdot a) \cdot x_i + R_{i,i-1} \cdot a \cdot y_{i-1}, \tag{24}$$

the gradient of the output of this modified filter is then

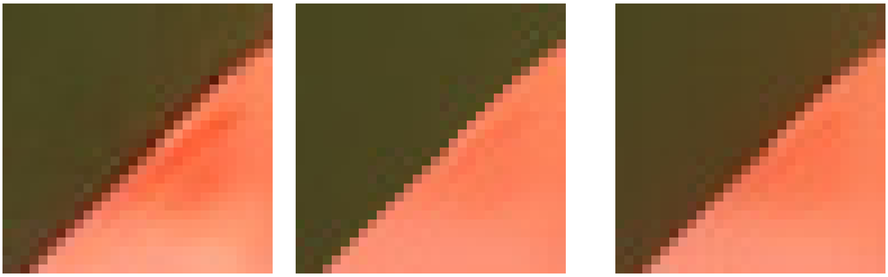
$$\begin{aligned} dy_i &= y_{i+1} - y_i \\ &= (1 - R_{i+1,i} \cdot a) \cdot x_{i+1} + (R_{i+1,i} \cdot a - 1) \cdot y_i \\ &= (1 - R_{i+1,i} \cdot a) \cdot (x_{i+1} - y_i). \end{aligned} \tag{25}$$

According to Eq. (24),

$$x_{i+1} - y_i = x_{i+1} - ((1 - R_{i,i-1} \cdot a) \cdot x_i + (R_{i,i-1} \cdot a) \cdot y_{i-1})$$
$$= (x_{i+1} - x_i) + (R_{i,i-1} \cdot a)(x_i - y_{i-1})$$
$$= dx_i + (R_{i,i-1} \cdot a)(x_i - y_{i-1}) \quad (26)$$
$$= \sum_{k=0}^{i} (R_{i,k} \cdot a^{i-k}) dx_k, \quad (27)$$

Substitute Eq. (27) into Eq. (25), the gradient is then

$$dy_i = (1 - R_{i+1,i} \cdot a) \cdot \sum_{k=0}^{i} (R_{i,k} \cdot a^{i-k}) \cdot dx_k. \quad (28)$$



(a) Input. (b) BF ($\sigma_R = 0.1$, $\sigma_S = 0.02$). (c) GDBF ($\sigma_R = 0.1$, $a = 0.95$).

**Fig. 1.** Gradient domain bilateral filtering. The traditional bilateral filter results in an unnatural color edge in (b). Note: the details will be lost in hard copy.

Comparing Eq. (28) and Eq. (23), we can conclude that GDBF actually performs bilateral filtering on the gradient domain, while both the original $1^{st}$-order recursive bilateral filter and the traditional bilateral filter perform bilateral filtering on the range domain. Low-pass filtering on the gradient domain results in more natural edge-preserving results as shown in Fig. 1 (c). Note that the traditional bilateral filter results in a sharp color edge in Fig. 1 (b). The proposed GDBF enforce smoothness in the gradient domain, thus obtains more natural results that are required in some computer graphics tasks like detail enhancement (see Sec. 5.3).

## 5 Applications

In this section, the effectiveness of the proposed recursive implementations (*with order up to two*) of the bilateral filter (RBF) are experimentally verified for a variety of computer vision and computer graphics applications. Visual or numerical comparisons with the traditional bilateral filter (BF) [3], [4] are presented.

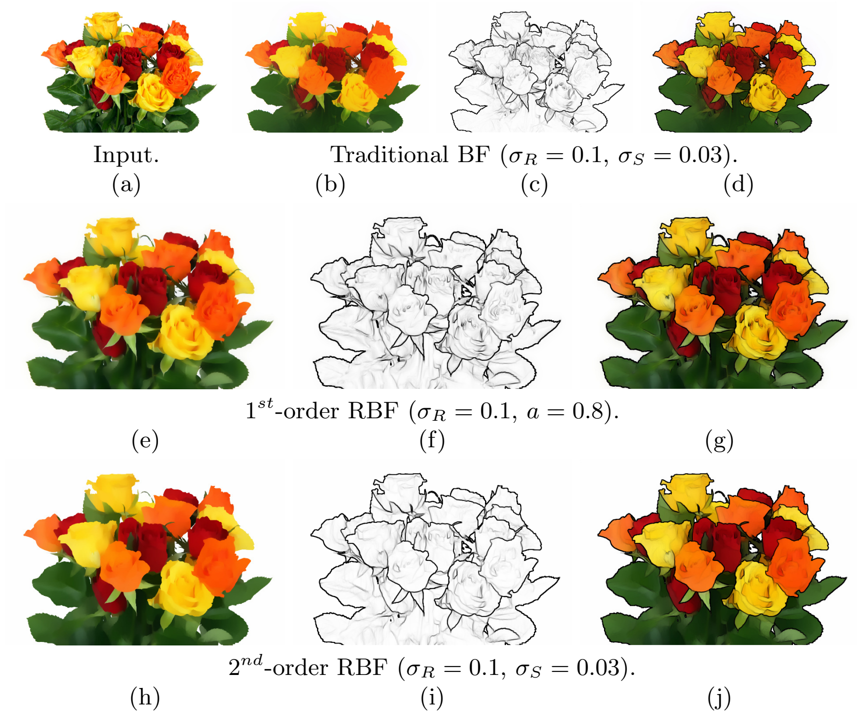



**Fig. 2.** NPR from bilateral filtering. (a) is the input image, (b)-(d) are computed using traditional BF, while (e)-(g) and (h)-(j) are computed using the proposed RBF of order 1 and 2, respectively. Note that all filters extract the most salient features and produce a non-photorealism look with high-contrast edges obtained from the gradient magnitudes of the filtered images.

### 5.1 Non-Photorealistic Rendering

Non-Photorealistic Rendering (NPR) aims to produce a wide variety of expressive styles for digital art not focused on photorealism. As an edge-preserving filter, the bilateral filter abstracts regions of low-contrast while preserving, or enhancing, high-contrast features. Fig. 2 presents an application of using bilateral filter for NPR. Fig. 2 (a) is an input image; (b), (e) and (h) are the filtering images; (c), (f) and (i) are the gradient magnitudes of the filtered images; (d), (g) and (j) are the abstracted images by superimposing the gradient magnitudes of the filtered images to themselves. Visual comparison shows that both traditional BF and the proposed RBFs produce a non-photorealism look with high-contrast edges around the most salient features of the input image.

### 5.2 Tone Mapping

Durand and Dorsey [6] is the first to use the bilateral filter for tone mapping. The bilateral filter is used to smooth a HDR image to produce a base and a detail layer of the HDR image. Only the base layer has its contrast reduced, thereby preserving detail. Such an example is presented in Fig. 3. (a) is obtained from traditional bilateral filter and (c) is obtained from the proposed GDBF. Note that (a) and (c) are similar but the proposed GDBF is significantly faster.

(a)Traditional BF. (b) Close-Up of (a). (c)GDBF (Sec. 4). (d) Close-Up of (c).
$\sigma_R = 0.2$, $\sigma_S = 0.02$. $\sigma_R = 0.2$, $a = 0.95$.

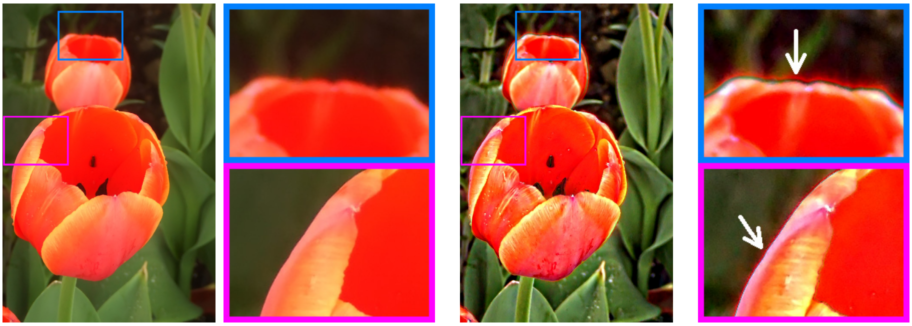
**Fig. 3.** Tone mapping results using bilateral filtering. Visually, the performance of the traditional bilateral filter and the proposed GDBF are similar.

### 5.3 Detail Enhancement

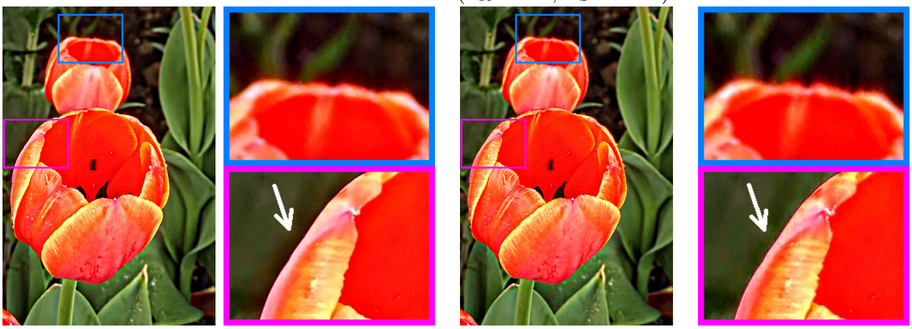
The based and detail layer separation method for tone mapping in Sec. 5.2 can be directly used for detail enhancement by combining the base layer and a boosted detail layer. Fig. 4 (a)-(d) present an input image and the enhanced result obtained from the bilateral filter. Fig. 4 (b) and (d) show the close-ups of the input image in (a) and the enhanced result in (b). Note that there are noticeable gradient reversal artifacts around color edges (shown with white arrows) in the enhanced result in Fig. 4 (d) because the shape of the edges are not maintained in the base layer.

Comparing to bilateral filter, the guided image filter better preserves the gradients around the edges, thus outperforms the bilateral filter for the detail enhancement application as shown in Fig. 4 (f), which contains close-ups of Fig. 4 (e). However, there is noticeable color bleeding artifacts around the color edges (shown with white arrows) in the pink boxes in Fig. 4 (f).

As discussed in Sec. 4, the proposed GDBF performs bilateral filtering on the gradient domain, thus is more suitable for this detail enhancement application as demonstrated in Fig. 4 (g) and (h). GDBF can remove the gradient reversal artifacts introduced by the bilateral filter and the color bleeding artifacts introduced by the guided image filter.



(a) Input. (b) Close-Up of (a). (c) From BF. ($\sigma_R = 0.1$, $\sigma_S = 0.02$). (d) Close-Up of (c).



(e) From GF. ($\epsilon = 0.1^2$, radius= 16) (f) Close-Up of (e). (g) From GDBF (Sec. 4). ($\sigma_R = 0.1$, $a = 0.95$). (h) Close-Up of (g).

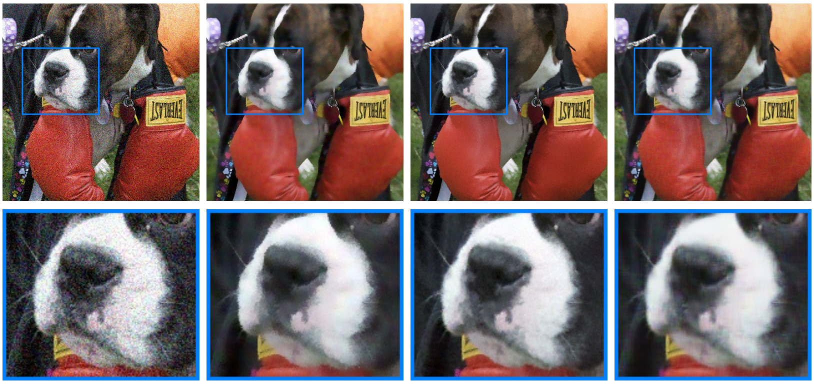
**Fig. 4.** Detail enhancement. (g)-(h) show that the proposed GDBF removes the gradient reversal artifacts (shown with white arrows) introduced by the bilateral filter in (c)-(d) and the color bleeding artifacts (shown with white arrow) introduced by the guided image filter in (e)-(f).

### 5.4 Stereo Matching

A local stereo algorithm generally performs (subsets of) the following four steps:

1. matching cost computation;
2. cost (support) aggregation;
3. disparity computation; and
4. disparity refinement.

All local algorithms require cost aggregation (step 2), and usually make implicit smoothness assumptions by aggregating support. Cost aggregation methods are traditionally performed *locally* by summing/averaging matching cost over windows with constant disparity. Yoon and Kweon [21] are the first to demonstrate that the bilateral filter [3], [4] is very effective for preserving depth edges when used for cost aggregation. In this paper, the proposed recursive bilateral filters are also tested using Middlebury benchmark [17]. In table 1, the proposed recursive bilateral filters are demonstrated to outperform the traditional bilateral



(a)Input (b)Traditional BF (c)[2] (d)$1^{st}$-order RBF.

**Fig. 5.** A failure case on non-local means denoising. (a) is the input image, (b)-(d) are computed using traditional bilateral filter, non-local means based on [2], and non-local means based on the proposed $1^{st}$-order RBF. Visual comparison shows that the proposed RBF failed to keep the details like [2].

filter [21] for stereo matching. It is comparable to the state-of-the-art local stereo method [22] but can be about **10**× faster on average. The numbers in the last thirteen columns are the percentages of the pixels with incorrect disparities on different data sets and on average. The error threshold is set to 1 disparity.

**Table 1.** Stereo matching. This table quantitatively compares the performance of different edge-preserving image filters when used for stereo matching. The numbers in the last thirteen columns are the *percentages* of the misestimated pixels. Note that the performance (*average error*) of the proposed RBFs is very close to the state-of-the-art [22] but about **10**× faster on average. The obtained disparity maps are available on Middlebury website [17]. Parameter setting: $\sigma_R = 0.09$ and $\sigma_S = 0.03$ for $2^{nd}$-order RBF; $\sigma_R = 0.13$ and $a = 0.95$ for $1^{st}$-order RBF.

| Algorithm | Tsukuba | Venus | Teddy | Cones | Avg. Error |
|---|---|---|---|---|---|
| CostFilter[22] | 1.51 | 0.20 | 6.16 | 2.71 | **5.55** |
| **$1^{st}$-order RBF** | 1.85 | 0.35 | 6.28 | 2.80 | **5.68** |
| **$2^{nd}$-order RBF** | 1.51 | 0.32 | 6.61 | 2.75 | **5.68** |
| Traditional BF[21] | 1.38 | 0.71 | 7.88 | 3.97 | **6.67** |

## 6 Conclusion

A recursive implementation of the bilateral filter is proposed in this paper. It uses a new range filter kernel and adopts any spatial filter kernel that can be recursively implemented. It is the first bilateral filter whose computational and memory complexity are linear in both input size and dimensionality. A new filter of the same complexity is derived from the proposed recursive implementation for performing bilateral filtering on the gradient domain. Similar to bilateral filter, this gradient domain bilateral filter can be used for edge-preserving filtering but avoids sharp edges that are observed to cause visible artifacts in some computer graphics tasks.

The proposed recursive implements are demonstrated to be effective for a number of computer vision and computer graphics applications, and are shown to outperform the traditional bilateral filter for some specific tasks like detail enhancement and stereo matching. However, the range filter kernel of the proposed recursive bilateral filter takes into account the pixel connectivity like the spatial filter kernel, thus is not suitable for applications ignoring the spatial relationships, like denoising using non-local means. A failure case is presented in Fig. 5. Visual comparison shows that the proposed method cannot preserve the details like [2] after denoising. There is not a clear solution but a hierarchical implementation of the proposed recursive bilateral filter shall result in filter kernel closer to the traditional bilateral filter, and maybe more accuracy when used for non-local means denoising. Another possible solution is pre-filtering the image with a median filter and used it as the guidance image for the proposed recursive bilateral filter to avoid large oscillations in the image gradient.

## References


1. Adams, A., Baek, J., Davis, A.: Fast high-dimensional filtering using the permutohedral lattice. Comput. Graph. Forum 29, 753–762 (2010)
2. Adams, A., Gelfand, N., Dolson, J., Levoy, M.: Gaussian kd-trees for fast high-dimensional filtering. ACM Trans. Graph. 28, 21:1–21:12 (2009)
3. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: ICCV, pp. 839–846 (1998)
4. Paris, S., Kornprobst, P., Tumblin, J., Durand, F.: Bilateral filtering: Theory and applications. Foundations and Trends in Computer Graphics and Vision (4), 1–73
5. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. IJCV 81, 24–52 (2009)
6. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. In: Siggraph, vol. 21 (2002)
7. Pham, T.Q., van Vliet, L.J.: Separable bilateral filtering for fast video preprocessing. In: International Conference on Multimedia and Expo. (2005)
8. Weiss, B.: Fast median and bilateral filtering. In: Siggraph, vol. 25, pp. 519–526 (2006)
9. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. In: Siggraph, vol. 26 (2007)
10. Porikli, F.: Integral histogram: A fast way to extract histograms in cartesian spaces, pp. 829–836 (2005)
11. Porikli, F.: Constant time o(1) bilateral filtering. In: CVPR (2008)
12. Yang, Q., Tan, K.H., Ahuja, N.: Real-time o(1) bilateral filtering. In: CVPR, pp. 557–564 (2009)
13. Adams, A.: High-dimensional gaussian filtering for computational photography. PhD Thesis, Stanford University, California, U.S.A. (2011)
14. Fattal, R.: Edge-avoiding wavelets and their applications. ToG 28, 1–10 (2009)

15. He, K., Sun, J., Tang, X.: Guided Image Filtering. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 1–14. Springer, Heidelberg (2010)
16. Gastal, E., Oliveira, M.: Domain transform for edge-aware image and video processing. ACM TOG 30, 69:1–69:12 (2011), Proceedings of SIGGRAPH 2011
17. Scharstein, D., Szeliski, R.: (Middlebury stereo evaluation), `http://vision.middlebury.edu/stereo/eval/`
18. Smith, J.O.: Introduction to Digital Filters with Audio Applications. W3K Publishing (2007)
19. Orfanidis, S.: Introduction to signal processing. Prentice-Hall, Inc., Upper Saddle River (1995)
20. Deriche, R.: Recursively implementing the gaussian and its derivatives. In: ICIP, pp. 263–267 (1992)
21. Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. PAMI 28, 650–656 (2006)
22. Rhemann, C., Hosni, A., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. In: CVPR, pp. 3017–3024 (2011)