Guided Image Filtering

Kaiming He, *Member*, *IEEE*, Jian Sun, *Member*, *IEEE*, and Xiaoou Tang, *Fellow*, *IEEE*

Abstract—In this paper, we propose a novel explicit image filter called *guided filter*. Derived from a local linear model, the guided filter computes the filtering output by considering the content of a guidance image, which can be the input image itself or another different image. The guided filter can be used as an edge-preserving smoothing operator like the popular *bilateral filter* [1], but it has better behaviors near edges. The guided filter is also a more generic concept beyond smoothing: It can transfer the structures of the guidance image to the filtering output, enabling new filtering applications like dehazing and guided feathering. Moreover, the guided filter naturally has a fast and nonapproximate linear time algorithm, regardless of the kernel size and the intensity range. Currently, it is one of the fastest edge-preserving filters. Experiments show that the guided filter is both effective and efficient in a great variety of computer vision and computer graphics applications, including edge-aware smoothing, detail enhancement, HDR compression, image matting/feathering, dehazing, joint upsampling, etc.

Index Terms—Edge-preserving filtering, bilateral filter, linear time filtering

1 Introduction

Most applications in computer vision and computer graphics involve image filtering to suppress and/or extract content in images. Simple linear translation-invariant (LTI) filters with explicit kernels, such as the mean, Gaussian, Laplacian, and Sobel filters [2], have been widely used in image restoration, blurring/sharpening, edge detection, feature extraction, etc. Alternatively, LTI filters can be implicitly performed by solving a Poisson Equation as in high dynamic range (HDR) compression [3], image stitching [4], image matting [5], and gradient domain manipulation [6]. The filtering kernels are implicitly defined by the inverse of a homogenous Laplacian matrix.

The LTI filtering kernels are spatially invariant and independent of image content. But usually one may want to consider additional information from a given *guidance* image. The pioneer work of *anisotropic diffusion* [7] uses the gradients of the filtering image itself to guide a diffusion process, avoiding smoothing edges. The *weighted least squares* (WLS) filter [8] utilizes the filtering input (instead of intermediate results, as in [7]) as the guidance, and optimizes a quadratic function, which is equivalent to anisotropic diffusion with a nontrivial steady state. The guidance image can also be another image besides the filtering input in many applications. For example, in colorization [9] the chrominance channels should not bleed across luminance edges; in image matting [10] the alpha matte should capture the thin structures in a composite image; in haze removal [11] the depth layer should be consistent with the scene. In these cases, we regard the chrominance/alpha/depth layers as the image to be filtered, and the luminance/composite/scene as the guidance image, respectively. The filtering process in [9], [10], and [11] is achieved by optimizing a quadratic cost function weighted by the guidance image. The solution is given by solving a large sparse matrix which solely depends on the guide. This inhomogeneous matrix implicitly defines a *translation-variant* filtering kernel. While these optimization-based approaches [8], [9], [10], [11] often yield state-of-the-art quality, it comes with the price of expensive computational time.

Another way to take advantage of the guidance image is to explicitly build it into the filter kernels. The *bilateral filter*, independently proposed in [12], [13], and [1] and later generalized in [14], is perhaps the most popular one of such explicit filters. Its output at a pixel is a weighted average of the nearby pixels, where the weights depend on the intensity/color similarities in the guidance image. The guidance image can be the filter input itself [1] or another image [14]. The bilateral filter can smooth small fluctuations and while preserving edges. Though this filter is effective in many situations, it may have unwanted *gradient reversal* artifacts [15], [16], [8] near edges (discussed in Section 3.4). The fast implementation of the bilateral filter is also a challenging problem. Recent techniques [17], [18], [19], [20], [21] rely on quantization methods to accelerate but may sacrifice accuracy.

In this paper, we propose a novel explicit image filter called *guided filter*. The filtering output is locally a linear transform of the guidance image. On one hand, the guided filter has good edge-preserving smoothing properties like the bilateral filter, but it does not suffer from the gradient reversal artifacts. On the other hand, the guided filter can be used beyond smoothing: With the help of the guidance image, it can make the filtering output more structured and less smoothed than the input. We demonstrate that the guided filter performs very well in a great variety of applications, including image smoothing/enhancement, HDR compression, flash/no-flash imaging, matting/feathering, dehazing, and joint upsampling. Moreover, the guided

• *K. He and J. Sun are with the Visual Computing Group, Microsoft Research Asia, Microsoft Building 2, #5 Dan Leng Street, Hai Dian District, Beijing 100080, China. E-mail: {kahe, jiansun}@microsoft.com.*
• *X. Tang is with the Department of Information Engineering, Chinese University of Hong Kong, 809 SHB, Shatin, N.T., Hong Kong. E-mail: xtang@ie.cuhk.edu.hk.*

Manuscript received 13 June 2012; revised 6 Sept. 2012; accepted 16 Sept. 2012; published online 1 Oct. 2012.
Recommended for acceptance by J. Jia.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2012-06-0447.
Digital Object Identifier no. 10.1109/TPAMI.2012.213.

filter naturally has an $\mathrm{O}(N)$ time (in the number of pixels N)[1] *nonapproximate* algorithm for both gray-scale and high-dimensional images, regardless of the kernel size and the intensity range. Typically, our *CPU* implementation achieves 40 ms per mega-pixel performing gray-scale filtering: To the best of our knowledge, this is one of the fastest edge-preserving filters.

A preliminary version of this paper was published in ECCV '10 [22]. It is worth mentioning that the guided filter has witnessed a series of new applications since then. The guided filter enables a high-quality real-time $\mathrm{O}(N)$ stereo matching algorithm [23]. A similar stereo method is proposed independently in [24]. The guided filter has also been applied in optical flow estimation [23], interactive image segmentation [23], saliency detection [25], and illumination rendering [26]. We believe that the guided filter has great potential in computer vision and graphics, given its simplicity, efficiency, and high-quality. We have provided a public code to facilitate future studies [27].

2 Related Work

We review edge-preserving filtering techniques in this section. We categorize them as explicit/implicit weighted-average filters and nonaverage ones.

2.1 Explicit Weighted-Average Filters

The *bilateral filter* [1] is perhaps the simplest and most intuitive one among explicit weighted-average filters. It computes the filtering output at each pixel as the average of neighboring pixels, weighted by the Gaussian of both spatial and intensity distance. The bilateral filter smooths the image while preserving edges. It has been widely used in noise reduction [28], HDR compression [15], multiscale detail decomposition [29], and image abstraction [30]. It is generalized to the *joint bilateral filter* in [14], where the weights are computed from another guidance image rather than the filtering input. The joint bilateral filter is particularly favored when the image to be filtered is not reliable to provide edge information, e.g., when it is very noisy or is an intermediate result, such as in flash/no-flash denoising [14], image upsamling [31], image deconvolution [32], stereo matching [33], etc.

The bilateral filter has limitations despite its popularity. It has been noticed in [15], [16], and [8] that the bilateral filter may suffer from "gradient reversal" artifacts. The reason is that when a pixel (often on an edge) has few similar pixels around it, the Gaussian weighted average is unstable. In this case, the results may exhibit unwanted profiles around edges, usually observed in detail enhancement or HDR compression.

Another issue concerning the bilateral filter is the efficiency. A brute-force implementation is $\mathrm{O}(Nr^2)$ time with kernel radius r. Durand and Dorsey [15] propose a piece-wise linear model and enable FFT-based filtering. Paris and Durand [17] formulate the gray-scale bilateral filter as a 3D filter in a space-range domain, and downsample this domain to speed up if the Nyquist condition is approximately true. In the case of box spatial kernels, Weiss [34] proposes an $\mathrm{O}(N \log r)$ time method based on *distributive histograms*, and Porikli [18] proposes the first $\mathrm{O}(N)$ time method using *integral histograms*. We point out that constructing the histograms is essentially performing a 2D spatial filter in the space-range domain with a 1D range filter followed. Under this viewpoint, both [34] and [18] sample the signal along the range domain but do not reconstruct it. Yang [19] proposes another $\mathrm{O}(N)$ time method which interpolates along the range domain to allow more aggressive subsampling. All of the above methods are linearly complex *w.r.t.* the number of the sampled intensities (e.g., number of linear pieces or histogram bins). They require coarse sampling to achieve satisfactory speed, but at the expense of quality degradation if the Nyquist condition is severely broken.

The space-range domain is generalized to higher dimension for color-weighted bilateral filtering [35]. The expensive cost due to the high dimensionality can be reduced by the *Gaussian kd-trees* [20], the *Permutohedral Lattices* [21], or the *Adaptive Manifolds* [36]. But the performance of these methods is not competitive for gray-scale bilateral filters because they spend much extra time preparing the data structures.

Given the limitations of the bilateral filter, people began to investigate new designs of fast edge-preserving filters. The $\mathrm{O}(N)$ time *Edge-Avoiding Wavelets* (EAW) [37] are wavelets with explicit image-adaptive weights. But the kernels of the wavelets are sparsely distributed in the image plane, with constrained kernel sizes (to powers of two), which may limit the applications. Recently, Gastal and Oliveira [38] propose another $\mathrm{O}(N)$ time filter known as the *Domain Transform* filter. The key idea is to iteratively and separably apply 1D edge-aware filters. The $\mathrm{O}(N)$ time complexity is achieved by integral images or recursive filtering. We will compare with this filter in this paper.

2.2 Implicit Weighted-Average Filters

A series of approaches optimize a quadratic cost function and solve a linear system, which is equivalent to implicitly filtering an image by an inverse matrix. In image segmentation [39] and colorization [9], the affinities of this matrix are Gaussian functions of the color similarities. In image matting, a matting Laplacian matrix [10] is designed to enforce the alpha matte as a local linear transform of the image colors. This matrix is also applied in haze removal [11]. The weighted least squares filter in [8] adjusts the matrix affinities according to the image gradients and produces halo-free edge-preserving smoothing.

Although these optimization-based approaches often generate high quality results, solving the linear system is time-consuming. Direct solvers like Gaussian Elimination are not practical due to the memory-demanding "filled in" problem [40], [41]. Iterative solvers like the Jacobi method, Successive Over-Relaxation (SOR), and Conjugate Gradients [40] are too slow to converge. Though carefully designed preconditioners [41] greatly reduce the iteration number, the computational cost is still high. The multigrid method [42] is proven $\mathrm{O}(N)$ time complex for homogeneous Poisson equations, but its quality degrades when the matrix becomes more inhomogeneous. Empirically, the implicit weighted-average filters take at least a few seconds to process a one megapixel image either by preconditioning [41] or by multigrid [8].

1. In the literature, "$\mathrm{O}(N)$/linear time" algorithms are sometimes referred to as "$\mathrm{O}(1)$/constant time" (per pixel). We are particularly interested in the complexity of filtering the entire image, and we will use the way of "$\mathrm{O}(N)$ time" throughout this paper.

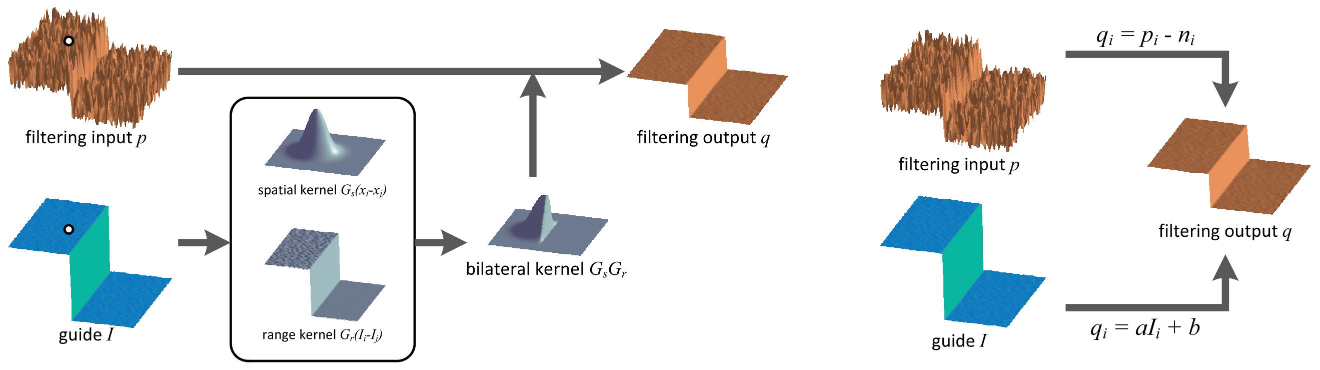


Fig. 1. Illustrations of the bilateral filtering process (left) and the guided filtering process (right).

It has been observed that these implicit filters are closely related to the explicit ones. In [43], Elad shows that the bilateral filter is one Jacobi iteration in solving the Gaussian affinity matrix. The Hierarchical Local Adaptive Preconditioners [41] and the Edge-Avoiding Wavelets [37] are constructed in a similar manner. In this paper, we show that the guided filter is closely related to the matting Laplacian matrix [10].

2.3 Nonaverage Filters

Edge-preserving filtering can also be achieved by nonaverage filters. The median filter [2] is a well-known edge-aware operator, and is a special case of local histogram filters [44]. Histogram filters have $O(N)$ time implementations in a way as the bilateral grid. The Total-Variation (TV) filters [45] optimize an L_1-regularized cost function, and are shown equivalent to iterative median filtering [46]. The L_1 cost function can also be optimized via *half-quadratic split* [47], alternating between a quadratic model and *soft shrinkage* (thresholding). Recently, Paris et al. [48] proposed manipulating the coefficients of the Laplacian Pyramid around each pixel for edge-aware filtering. Xu et al. [49] propose optimizing an L_0-regularized cost function favoring piecewise constant solutions. The nonaverage filters are often computationally expensive.

3 Guided Filter

We first define a general linear translation-variant filtering process, which involves a guidance image I, an filtering input image p, and an output image q. Both I and p are given beforehand according to the application, and they can be identical. The filtering output at a pixel i is expressed as a weighted average:

$$q_i = \sum_j W_{ij}(I) p_j, \tag{1}$$

where i and j are pixel indexes. The filter kernel W_{ij} is a function of the guidance image I and independent of p. This filter is linear with respect to p.

An example of such a filter is the joint bilateral filter [14] (Fig. 1 (left)). The bilateral filtering kernel W^{bf} is given by

$$W^{\mathrm{bf}}_{ij}(I) = \frac{1}{K_i} \exp\left(-\frac{\| \mathbf{x}_i - \mathbf{x}_j \|^2}{\sigma_{\mathrm{s}}^2}\right) \exp\left(-\frac{\| I_i - I_j \|^2}{\sigma_{\mathrm{r}}^2}\right), \tag{2}$$

where $\mathbf{x}$ is the pixel coordinate and K_i is a normalizing parameter to ensure that $\sum_j W^{\mathrm{bf}}_{ij} = 1$. The parameters σ_{s} and σ_{r} adjust the sensitivity of the spatial similarity and the *range* (intensity/color) similarity, respectively. The joint bilateral filter degrades to the original bilateral filter [1] when I and p are identical.

The implicit weighted-average filters (in Section 2.2) optimize a quadratic function and solve a linear system in this form:

$$\mathrm{A}\mathrm{q} = \mathrm{p}, \tag{3}$$

where q and p are N-by-1 vectors concatenating $\{q_i\}$ and $\{p_i\}$, respectively, and A is an N-by-N matrix only depends on I. The solution to (3), i.e., $\mathrm{q} = \mathrm{A}^{-1}\mathrm{p}$, has the same form as (1), with $W_{ij} = (\mathrm{A}^{-1})_{ij}$.

3.1 Definition

Now we define the guided filter. The key assumption of the guided filter is a local linear model between the guidance I and the filtering output q. We assume that q is a linear transform of I in a window ω_k centered at the pixel k:

$$q_i = a_k I_i + b_k, \forall i \in \omega_k, \tag{4}$$

where (a_k, b_k) are some linear coefficients assumed to be constant in ω_k. We use a square window of a radius r. This local linear model ensures that q has an edge only if I has an edge, because $\nabla q = a \nabla I$. This model has been proven useful in image super-resolution [50], image matting [10], and dehazing [11].

To determine the linear coefficients (a_k, b_k), we need constraints from the filtering input p. We model the output q as the input p subtracting some unwanted components n like noise/textures:

$$q_i = p_i - n_i. \tag{5}$$

We seek a solution that minimizes the difference between q and p while maintaining the linear model (4). Specifically, we minimize the following cost function in the window ω_k:

$$E(a_k, b_k) = \sum_{i \in \omega_k} \left((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2 \right). \tag{6}$$

Here, ϵ is a regularization parameter penalizing large a_k. We will investigate its intuitive meaning in Section 3.2. Equation (6) is the *linear ridge regression* model [51], [52] and its solution is given by

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}, \tag{7}$$

$$b_k = \bar{p}_k - a_k \mu_k. \tag{8}$$

Here, μ_k and σ_k^2 are the mean and variance of I in ω_k, $|\omega|$ is the number of pixels in ω_k, and $\bar{p}_k = \frac{1}{|\omega|}\sum_{i\in\omega_k} p_i$ is the mean of p in ω_k. Having obtained the linear coefficients (a_k, b_k), we can compute the filtering output q_i by (4). Fig. 1 (right) shows an illustration of the guided filtering process.

However, a pixel i is involved in all the overlapping windows ω_k that covers i, so the value of q_i in (4) is not identical when it is computed in different windows. A simple strategy is to average all the possible values of q_i. So after computing (a_k, b_k) for all windows ω_k in the image, we compute the filtering output by

$$q_i = \frac{1}{|\omega|}\sum_{k|i\in\omega_k}(a_k I_i + b_k). \tag{9}$$

Noticing that $\sum_{k|i\in\omega_k} a_k = \sum_{k\in\omega_i} a_k$ due to the symmetry of the box window, we rewrite (9) by

$$q_i = \bar{a}_i I_i + \bar{b}_i, \tag{10}$$

where $\bar{a}_i = \frac{1}{|\omega|}\sum_{k\in\omega_i} a_k$ and $\bar{b}_i = \frac{1}{|\omega|}\sum_{k\in\omega_i} b_k$ are the average coefficients of all windows overlapping i. The averaging strategy of overlapping windows is popular in image denoising (see [53]) and is a building block of the very successful BM3D algorithm [54].

With the modification in (10), ∇q is no longer scaling of ∇I because the linear coefficients $(\bar{a}_i, \bar{b}_i)$ vary spatially. But as $(\bar{a}_i, \bar{b}_i)$ are the output of a mean filter, their gradients can be expected to be much smaller than that of I near strong edges. In this situation we can still have $\nabla q \approx \bar{a}\nabla I$, meaning that abrupt intensity changes in I can be mostly preserved in q.

Equations (7), (8), and (10) are the definition of the guided filter. A pseudocode is in Algorithm 1. In this algorithm, f_{mean} is a mean filter with a window radius r. The abbreviations of correlation (corr), variance (var), and covariance (cov) indicate the intuitive meaning of these variables. We will discuss the fast implementation and the computation details in Section. 4.

Algorithm 1. Guided Filter.

```
Input: filtering input image $p$, guidance image $I$, radius $r$,
    regularization $\epsilon$
Output: filtering output $q$.
1: $\text{mean}_I = f_{\text{mean}}(I)$
   $\text{mean}_p = f_{\text{mean}}(p)$
   $\text{corr}_I = f_{\text{mean}}(I.*I)$
   $\text{corr}_{Ip} = f_{\text{mean}}(I.*p)$
2: $\text{var}_I = \text{corr}_I - \text{mean}_I.*\text{mean}_I$
   $\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I.*\text{mean}_p$
3: $a = \text{cov}_{Ip}./(\text{var}_I + \epsilon)$
   $b = \text{mean}_p - a.*\text{mean}_I$
4: $\text{mean}_a = f_{\text{mean}}(a)$
   $\text{mean}_b = f_{\text{mean}}(b)$
5: $q = \text{mean}_a.*I + \text{mean}_b$
/* $f_{\text{mean}}$ is a mean filter with a wide variety of O($N$) time
methods. */
```

3.2 Edge-Preserving Filtering

Given the definition of the guided filter, we first study the edge-preserving filtering property. Fig. 2 shows an example of the guided filter with various sets of parameters. Here we investigate the special case where the guide I is identical to the filtering input p. We can see that the guided filter behaves as an edge-preserving smoothing operator (Fig. 2).

The edge-preserving filtering property of the guided filter can be explained intuitively as following. Consider the case where $I \equiv p$. In this case, $a_k = \sigma_k^2/(\sigma_k^2 + \epsilon)$ in (7) and $b_k = (1 - a_k)\mu_k$. It is clear that if $\epsilon = 0$, then $a_k = 1$ and $b_k = 0$. If $\epsilon > 0$, we can consider two cases.

Case 1: "High variance." If the image I changes a lot within ω_k, we have $\sigma_k^2 \gg \epsilon$, so $a_k \approx 1$ and $b_k \approx 0$.

Case 2: "Flat patch." If the image I is almost constant in ω_k, we have $\sigma_k^2 \ll \epsilon$, so $a_k \approx 0$ and $b_k \approx \mu_k$.

When a_k and b_k are averaged to get $\bar{a}_i$ and $\bar{b}_i$, combined in (10) to get the output, we have that if a pixel is in the middle of a "high variance" area, then its value is unchanged $(a \approx 1, b \approx 0, q \approx p)$, whereas if it is in the middle of a "flat patch" area, its value becomes the average of the pixels nearby $(a \approx 0, b \approx \mu, q \approx \bar{\mu})$.

More specifically, the criterion of a "flat patch" or a "high variance" one is given by the parameter ϵ. The patches with variance (σ^2) much smaller than ϵ are smoothed, whereas those with variance much larger than ϵ are preserved. The effect of ϵ in the guided filter is similar to the range variance σ_{r}^2 in the bilateral filter (2): Both determine "what is an edge/a high variance patch that should be preserved."

Further, in a flat region the guided filter becomes a cascade of two box mean filters whose radius is r. Cascades of box filters are good approximations of Gaussian filters. Thus, we empirically set up a "correspondence" between the guided filter and the bilateral filter: $r \leftrightarrow \sigma_s$ and $\epsilon \leftrightarrow \sigma_r^2$. Fig. 2 shows the results of both filters using corresponding parameters. The table "PSNR" in Fig. 2 shows the quantitative difference between the guided filter results and the bilateral filter results of the corresponding parameters.[2] It is often considered as visually insensitive when the PSNR ≥ 40 dB [18].

3.3 Filter Kernel

It is easy to show that the relationships among I, p, and q, given by (7), (8), and (10), are in the weighted-average form as (1). In fact, a_k in (7) can be rewritten as a weighted sum of $p: a_k = \sum_j A_{kj}(I)p_j$, where A_{ij} are the weights only dependent on I. For the same reason, we also have $b_k = \sum_j B_{kj}(I)p_j$ from (8) and $q_i = \sum_j W_{ij}(I)p_j$ from (10). We can prove that the kernel weights is explicitly expressed by

$$W_{ij}(I) = \frac{1}{|\omega|^2}\sum_{k:(i,j)\in\omega_k}\left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right). \tag{11}$$

Proof. Due to the linear dependence between p and q, the filter kernel is given by $W_{ij} = \partial q_i/\partial p_j$. Putting (8) into (10) and eliminating b, we obtain

2. Note that we do not intend to approximate the bilateral filer, and the bilateral filter results are not considered as "ground-truth." So the "PSNR" measure is just analogous to those used in the bilateral filter approximations [18].

	r=2	r=4	r=8
ε = 0.1²	39.38	38.25	37.23
ε = 0.2²	38.11	37.78	37.09
ε = 0.4²	38.09	39.04	37.72

"PSNR" (dB)

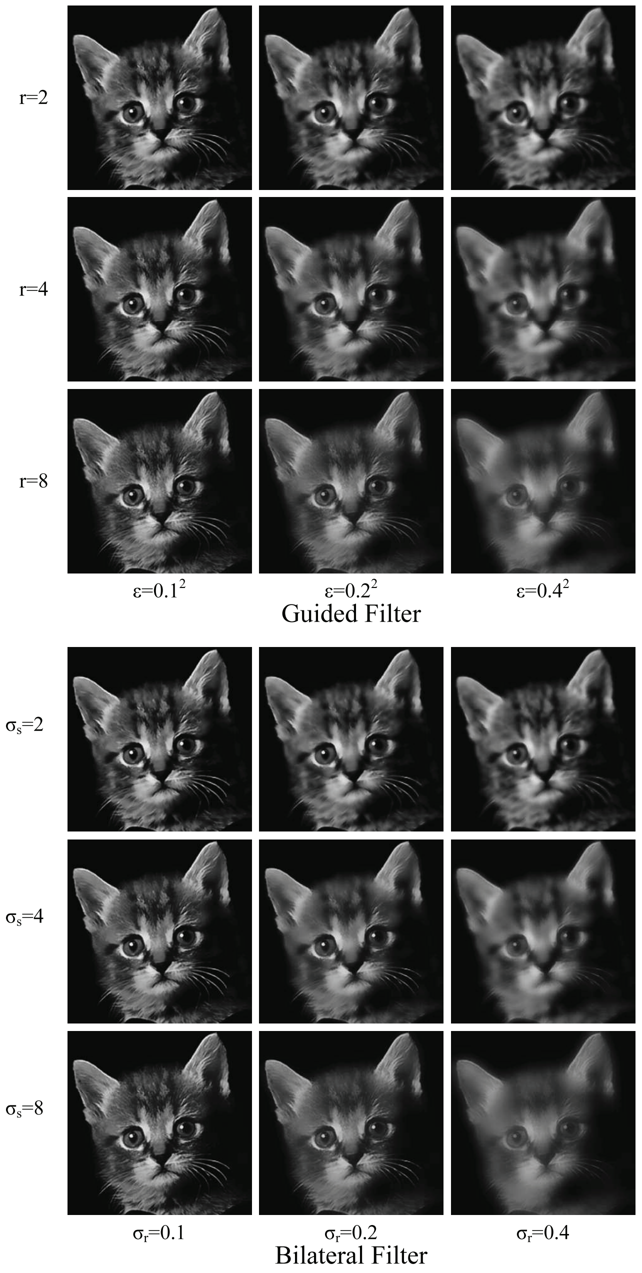


Fig. 2. Edge-preserving filtering results of a gray-scale image using the guided filter (top) and the bilateral filter (bottom). In this example, the guidance I is identical to the input p. The input image is scaled in $[0, 1]$. The table "PSNR" shows the quantitative difference between the guided filter results and the bilateral filter results using corresponding parameters. The input image is from [1].

$$q_i = \frac{1}{|\omega|}\sum_{k\in\omega_i}(a_k(I_i - \mu_k) + \bar{p}_k). \tag{12}$$

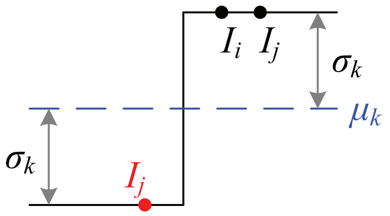


Fig. 3. A 1D example of an ideal step edge. For a window that exactly centers on the edge, the variables μ and σ are as indicated.

The derivative gives

$$\frac{\partial q_i}{\partial p_j} = \frac{1}{|\omega|}\sum_{k\in\omega_i}\left(\frac{\partial a_k}{\partial p_j}(I_i - \mu_k) + \frac{\partial \bar{p}_k}{\partial p_j}\right). \tag{13}$$

In this equation, we have

$$\frac{\partial \bar{p}_k}{\partial p_j} = \frac{1}{|\omega|}\delta_{j\in\omega_k} = \frac{1}{|\omega|}\delta_{k\in\omega_j}, \tag{14}$$

where $\delta_{j\in\omega_k}$ is one when j is in the window ω_k, and is zero otherwise. On the other hand, the partial derivative $\partial a_k/\partial p_j$ in (13) can be computed from (7):

$$\begin{aligned}\frac{\partial a_k}{\partial p_j} &= \frac{1}{\sigma_k^2+\epsilon}\left(\frac{1}{|\omega|}\sum_{i\in\omega_k}\frac{\partial p_i}{\partial p_j}I_i - \frac{\partial \bar{p}_k}{\partial p_j}\mu_k\right)\\ &= \frac{1}{\sigma_k^2+\epsilon}\left(\frac{1}{|\omega|}I_j - \frac{1}{|\omega|}\mu_k\right)\delta_{k\in\omega_j}.\end{aligned} \tag{15}$$

Putting (14) and (15) into (13), we obtain

$$\frac{\partial q_i}{\partial p_j} = \frac{1}{|\omega|^2}\sum_{k\in\omega_i, k\in\omega_j}\left(1 + \frac{(I_i-\mu_k)(I_j-\mu_k)}{\sigma_k^2+\epsilon}\right). \tag{16}$$

This is the expression of the filter kernel W_{ij}. □

Some further algebraic manipulations show that $\sum_j W_{ij}(I) \equiv 1$. No extra effort is needed to normalize the weights.

The edge-preserving smoothing property can also be understood by investigating the filter kernel (11). Take an ideal step edge of a 1D signal as an example (Fig. 3). The terms $I_i - \mu_k$ and $I_j - \mu_k$ have the same sign (+/-) when I_i and I_j are on the same side of an edge, while they have opposite signs when the two pixels are on different sides. So in (11) the term $1 + \frac{(I_i-\mu_k)(I_j-\mu_k)}{\sigma_k^2+\epsilon}$ is much smaller (and close to zero) for two pixels on different sides than on the same sides. This means that the pixels across an edge are almost not averaged together. We can also understand the smoothing effect of ϵ from (11). When $\sigma_k^2 \ll \epsilon$ ("flat patch"), the kernel becomes $W_{ij}(I) \approx \frac{1}{|\omega|^2}\sum_{k:(i,j)\in\omega_k} 1$: This is an LTI low-pass filter (it is a cascade of two mean filters).

Fig. 4 illustrate some examples of the kernel shapes in real images. In the top row are the kernels near a step edge. Like the bilateral kernel, the guided filter kernel assigns nearly zero weights to the pixels on the opposite side of the edge. In the middle row are the kernels in a patch with small scale textures. Both filters average almost all the nearby pixels together and appear as low-pass filters. This is more apparent in a constant region (Fig. 4 (bottom row)), where the guided filter degrades to a cascade of two mean filters.

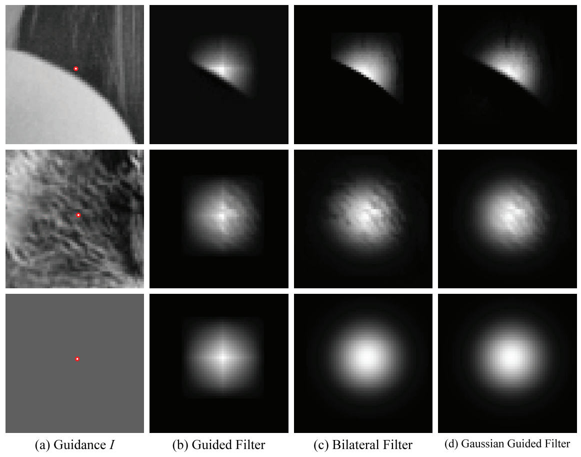


Fig. 4. Filter kernels. Top: A realistic step edge (guided filter: $r=7, \epsilon=0.1^2$, bilateral filter: $\sigma_s=7, \sigma_r=0.1$). Middle and Bottom: A textured patch and a constant patch (guided filter: $r=8, \epsilon=0.2^2$, bilateral filter: $\sigma_s=8, \sigma_r=0.2$). The kernels are centered at the pixels denoted by the red dots. The Gaussian guided filter replaces the mean filter in Algorithm 1 by a Gaussian filter with $\sigma_g=\sigma_s/\sqrt{2}$.

It can be observed in Fig. 4b that the guided filter is rotationally *asymmetric* and slightly biases to the x/y-axis. This is because we use a box window in the filter design. This problem can be solved by using a Gaussian weighted window instead. Formally, we can introduce the weights $w_{ik}=\exp(-\parallel \mathbf{x}_i-\mathbf{x}_k \parallel^2/\sigma_g^2)$ in (6):

$$E(a_k,b_k)=\sum_{i\in\omega_k} w_{ik}\big((a_kI_i+b_k-p_i)^2+\epsilon a_k^2\big). \quad (17)$$

It is straightforward to show that the resulted Gaussian guided filter can be computed by replacing all the mean filters f_{mean} in Algorithm 1 with Gaussian filters f_{Gauss}. The resulting kernels are rotationally symmetric as in Fig. 4d.[3] In Section 4, we will show that the Gaussian guided filter is still $\mathrm{O}(N)$ time like the original guided filter. But because in practice we find that the original guided filter is always good enough, we use it in all the remaining experiments.

3.4 Gradient-Preserving Filtering

Though the guided filter is an edge-preserving smoothing operator like the bilateral filter, it avoids the gradient reversal artifacts that may appear in detail enhancement and HDR compression. A brief introduction to the detail enhancement algorithm is as follows (see also Fig. 5). Given the input signal p (black in Fig. 5), its edge-preserving smoothed output is used as a *base layer* q (red). The difference between the input signal and the base layer is the *detail layer* (blue): $d=p-q$. It is magnified to boost the details. The enhanced signal (green) is the combination of the boosted detail layer and the base layer. An elaborate description of this method can be found in [15].

For the bilateral filter (Fig. 5 (top)), the base layer is not consistent with the input signal at the edge pixels (see the zoom-in). Fig. 6 illustrates the bilateral kernel of an edge pixel. Because this pixel has no similar neighbors, the Gaussian weighted range kernel unreliably averages a group of pixels. Even worse, the range kernel is biased due to the abrupt change of the edge. For the example edge pixel in Fig. 6, its filtered value is smaller than its original value, making the filtered signal q *sharper* than the input p. This sharpening effect has been observed in [15], [16], [8]. Now suppose the gradient of the input p is positive: $\partial_x p>0$ (as in Figs. 5 and 6). When q is sharpened, it gives: $\partial_x q>\partial_x p$. The detail layer d thus has a negative gradient $\partial_x d=\partial_x p-\partial_x q<0$, meaning it has a *reversed* gradient direction *w.r.t.* the input signal (see Fig. 5 (top)). When the detail layer is magnified and recombined with the input signal, the gradient reversal artifact on the edge appears. This artifact is inherent and cannot be safely avoided by tuning parameters because natural images usually have edges at all kinds of scales and magnitudes.

On the other hand, the guided filter performs better on avoiding gradient reversal. Actually, if we use the patch-wise model (4), it is guaranteed to not have gradient reversal in the case of self-guided filtering ($I\equiv p$). In this case, (7) gives $a_k=\sigma_k^2/(\sigma_k^2+\epsilon)<1$ and b_k is a constant. So we have $\partial_x q=a_k\partial_x p$ and the detail layer gradient $\partial_x d=\partial_x p-\partial_x q=(1-a_k)\partial_x p$, meaning that $\partial_x d$ and $\partial_x p$ are always in the same direction. When we use the overlapping model (9) instead of (4), we have $\partial_x q=\bar{a}\partial_x p+p\partial_x\bar{a}+\partial_x\bar{b}$. Because $\bar{a}$ and $\bar{b}$ are low-pass filtered maps, we obtain $\partial_x q\approx\bar{a}\partial_x p$ and the above conclusion is still approximately true. In practice, we do not observe the gradient reversal artifacts in all experiments. Fig. 5 (bottom) gives an example. In Fig. 6, we show the guided filter kernel of an edge pixel. Unlike the bilateral kernel, the guided filter assigns some small but essential weights to the weaker side of the kernel. This makes the guided kernel is less biased, avoiding reducing the value of the example edge pixel in Fig. 5.

We notice that the gradient reversal problem also appears in the recent edge-preserving *Domain Transform* filter [38] (Fig. 7). This very efficient filter is derived from the (1D) bilateral kernel, so it does not safely avoid gradient reversal.

3.5 Extension to Color Filtering

The guided filter can be easily extended to color images. In the case when the filtering input p is multichannel, it is straightforward to apply the filter to each channel independently. In the case when the guidance image I is multichannel, we rewrite the local linear model (4) as

$$q_i=\mathbf{a}_k^{\mathrm{T}}\mathbf{I}_i+b_k, \quad \forall i\in\omega_k. \quad (18)$$

Here $\mathbf{I}_i$ is a 3×1 color vector, $\mathbf{a}_k$ is a 3×1 coefficient vector, q_i and b_k are scalars. The guided filter for color guidance images becomes

$$\mathbf{a}_k=(\Sigma_k+\epsilon\mathrm{U})^{-1}\left(\frac{1}{|\omega|}\sum_{i\in\omega_k}\mathbf{I}_ip_i-\mu_k\bar{p}_k\right), \quad (19)$$

$$b_k=\bar{p}_k-\mathbf{a}_k^{\mathrm{T}}\mu_k, \quad (20)$$

$$q_i=\bar{\mathbf{a}}_i^{\mathrm{T}}\mathbf{I}_i+\bar{b}_i. \quad (21)$$

Here, Σ_k is the 3×3 covariance matrix of $\mathbf{I}$ in ω_k, and U is a 3×3 identity matrix.

A color guidance image can better preserve the edges that are not distinguishable in gray-scale (see Fig. 8). This is also the case in bilateral filtering [20]. A color guidance

3. Because the Gaussian guided filter becomes a cascade of two Gaussian filters in constant regions, we set the Gaussian parameter $\sigma_g=\sigma_s/\sqrt{2}$ to ensure the same response in constant regions as the bilateral filter.

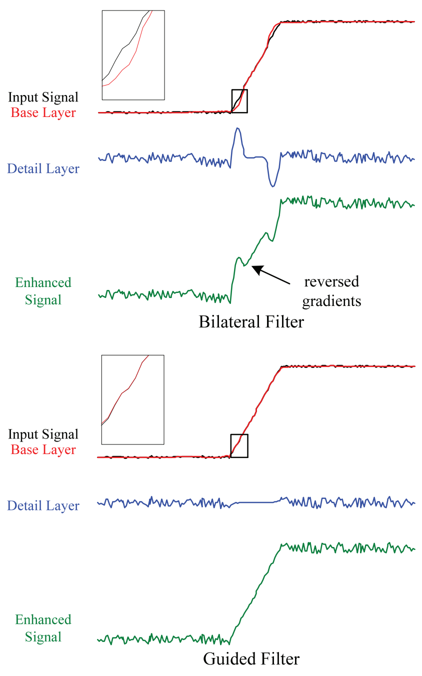


Fig. 5. 1D illustration for detail enhancement. See the text for explanation.

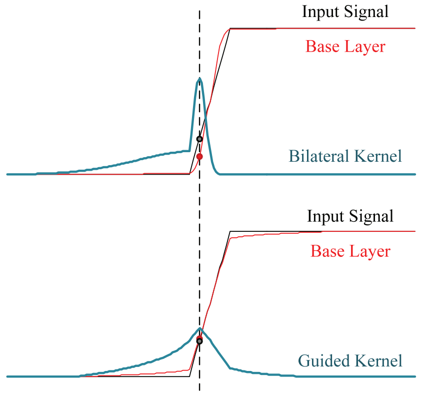


Fig. 6. The filtering results and the filter kernels at a pixel on a clean edge. In this example, the edge height is 1 and the edge slope spans 20 pixels. The parameters are $r = 30, \epsilon = 0.15^2$ for the guided filter and $\sigma_s = 30, \sigma_r = 0.15$ for the bilateral filter.

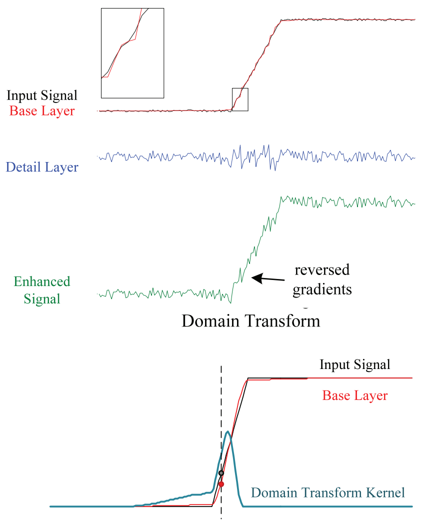


Fig. 7. Gradient reversal artifacts in the domain transform filter [38].

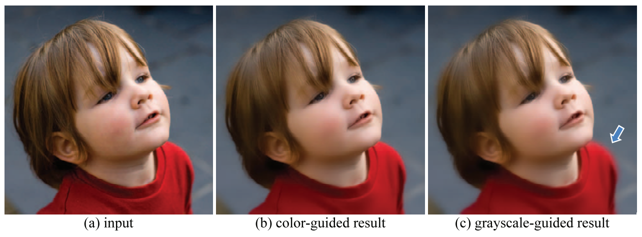


Fig. 8. Guided filtering results guided by the color image (b) and guided by its gray-scale version (c). The result in (c) has halos because the edge is not undistinguishable in gray-scale.

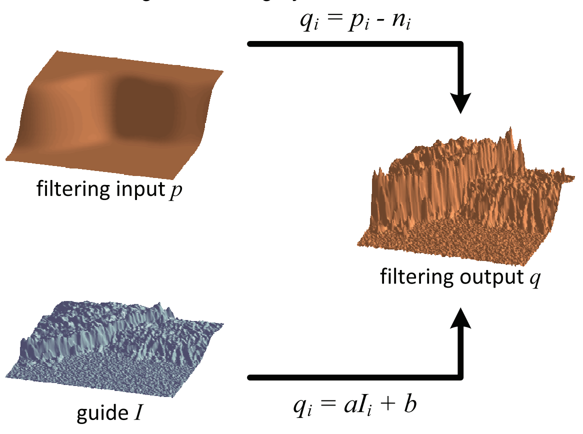


Fig. 9. Structure-transferring filtering.

image is also essential in the matting/feathering and dehazing applications, as we show later, because the local linear model is more likely to be valid in the RGB color space than in gray-scale [10].

3.6 Structure-Transferring Filtering

Interestingly, the guided filter is not simply a smoothing filter. Due to the local linear model of $q = aI + b$, the output q is locally a scaling (plus an offset) of the guidance I. This makes it possible to transfer structure from the guidance I to the output q, even if the filtering input p is smooth (see Fig. 9).

To show an example of structure-transferring filtering, we introduce an application of *guided feathering*: A binary mask is refined to appear an alpha matte near the object boundaries (Fig. 10). The binary mask can be obtained from graph-cut or other segmentation methods, and is used as the filter input p. The guidance $\mathbf{I}$ is the color image. Fig. 10 shows the behaviors of three filters: guided filter, (joint) bilateral filter, and a recent domain transform filter [38]. We observe that the guided filter faithfully recovers the hair, even though the filtering input p is binary and very rough. The bilateral filter may lose some thin structures (see zoom-in). This is because the bilateral filer is guided by pixel-wise color difference, whereas the guided filter has a patch-wise model. We also observe that the domain transform filter does not have a good structure-transferring ability and simply smooths the result. This is because this filter is based on geodesic distance of pixels, and its output is a series of 1D box filters with adaptive spans [38].

The structure-transferring filtering is an important property of the guided filter. It enables new filtering-based applications, including feathering/matting and dehazing (Section 5). It also enables high-quality filtering-based stereo matching methods in [23] and [24].

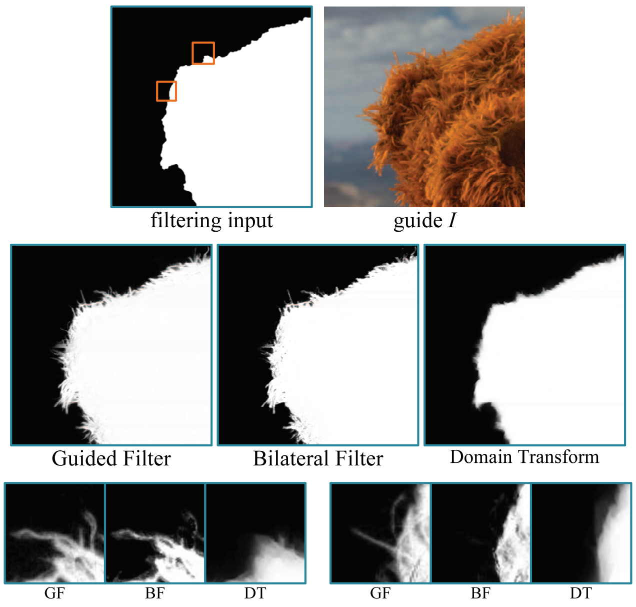


Fig. 10. Comparisons on structure-transferring filtering. The parameters are $r = 60$, $\epsilon = 0.01^2$ for the guided filter, $\sigma_s = 60$, $\sigma_r = 0.01$ for the (joint) bilateral filter, and $\sigma_s = 60$, $\sigma_r = 0.5$ for the domain transform filter (a smaller σ_r would make the domain transform result more similar to the input mask and does not improve the quality).

3.7 Relation to Implicit Methods

The guided filter is closely related to the matting Laplacian matrix [10]. This casts new insights into the understanding of this filter.

In a closed-form solution to matting [10], the matting Laplacian matrix is derived from a local linear model. Unlike the guided filter, which computes the local optimal for each window, the closed-form solution seeks a global optimal. To solve for the unknown alpha matte, this method minimizes the following cost function:

$$E(\mathrm{q}) = (\mathrm{q} - \mathrm{p})^{\mathrm{T}}\Lambda(\mathrm{q} - \mathrm{p}) + \mathrm{q}^{\mathrm{T}}\mathrm{L}\mathrm{q}. \tag{22}$$

Here, q is an N-by-1 vector denoting the unknown alpha matte, p is the constraint (e.g., a trimap), L is an $N \times N$ matting Laplacian matrix, and Λ is a diagonal matrix encoded with the weights of the constraints. The solution to this optimization problem is given by solving a linear system

$$(\mathrm{L} + \Lambda)\mathrm{q} = \Lambda\mathrm{p}. \tag{23}$$

The elements of the matting Laplacian matrix are given by

$$\mathrm{L}_{ij} = \sum_{k:(i,j)\in\omega_k}\left(\delta_{ij} - \frac{1}{|\omega|}\left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right)\right). \tag{24}$$

where δ_{ij} is the Kronecker delta. Comparing (24) with (11), we find that the elements of the matting Laplacian matrix can be directly given by the guided filter kernel:

$$\mathrm{L}_{ij} = |\omega|(\delta_{ij} - \mathrm{W}_{ij}), \tag{25}$$

Following the strategy in [43], we prove that the output of the guided filter is one Jacobi iteration in optimizing (22):

$$\mathrm{q}_i \approx \sum_j W_{ij}(I)\mathrm{p}_j. \tag{26}$$

Proof. The matrix form of (25) is

$$\mathrm{L} = |\omega|(\mathrm{U} - \mathrm{W}), \tag{27}$$

where U is a unit matrix of the same size as L. To apply the Jacobi method [40] on the linear system (23), we require the diagonal/off-diagonal parts of the matrices. We decompose W into a diagonal part W_d and an off-diagonal part W_o, so $\mathrm{W} = \mathrm{W}_d + \mathrm{W}_o$. From (27) and (23) we have

$$(|\omega|\mathrm{U} - |\omega|\mathrm{W}_d - |\omega|\mathrm{W}_o + \Lambda)\mathrm{q} = \Lambda\mathrm{p}. \tag{28}$$

Note that only W_o is off-diagonal here. Using p as the initial guess, we compute one iteration of the Jacobi method:

$$\begin{aligned} \mathrm{q} &\approx (|\omega|\mathrm{U} - |\omega|\mathrm{W}_d + \Lambda)^{-1}(|\omega|\mathrm{W}_o + \Lambda)\mathrm{p} \\ &= \left(\mathrm{U} - \mathrm{W}_d + \frac{\Lambda}{|\omega|}\right)^{-1}\left(\mathrm{W} - \mathrm{W}_d + \frac{\Lambda}{|\omega|}\right)\mathrm{p}. \end{aligned} \tag{29}$$

In (29), the only convolution is the matrix multiplication Wp. The other matrices are all diagonal and point-wise operations. To further simplify (29), we let the matrix Λ satisfy: $\Lambda = |\omega|\mathrm{W}_d$ or, equivalently,

$$\Lambda_{ii} = \frac{1}{|\omega|}\sum_{k\in\omega_i}\left(1 + \frac{(I_i - \mu_k)^2}{\sigma_k^2 + \epsilon}\right). \tag{30}$$

The expectation value of Λ_{ii} in (30) is 2, implying that the constraint in (22) is soft. Equation (29) is then reduced to

$$\mathrm{q} \approx \mathrm{W}\mathrm{p}. \tag{31}$$

This is the guided filter. □

In [55], we have shown another relationship between the guided filter and the matting Laplacian matrix through the Conjugate Gradient solver [40].

In Section 5, we apply this property to image matting/feathering and haze removal, which provide some reasonably good initial guess p. This is another perspective of the structure-transferring property of the filter.

4 Computation and Efficiency

A main advantage of the guided filter over the bilateral filter is that it naturally has an $\mathrm{O}(N)$ time nonapproximate algorithm, independent of the window radius r and the intensity range.

The filtering process in (1) is a translation-variant convolution. Its computational complexity increases when the kernel becomes larger. Instead of directly performing the convolution, we compute the filter output from its definition (7), (8), and (10) by Algorithm 1. The main computational burden is the mean filter f_{mean} with box windows of radius r. Fortunately, the (unnormalized) box filter can be efficiently computed in $\mathrm{O}(N)$ time using the *integral image* technique [57] or a simple *moving sum* method (see Algorithm 2). Considering the separability of the box filter, either method takes two operations (addition/subtraction) per pixel along each x/y direction. Thus the mean filter f_{mean} takes, per pixel, five addition/subtraction operations and one @ division (to normalize).

TABLE 1
Number of Mean Filters ($\#f_{\rm m}$) and Guided Filter Running Time in Different Scenarios

dimensionality	$d_I = 1, d_p = 1$	$d_I = 3, d_p = 1$	$d_I = 3, d_p = 3$	$d_I = 3, d_p = K$	$I \equiv p, d_p = 1$	$I \equiv p, d_p = 3$
example applications	flash/no-flash (gray-scale)	feathering/matting, dehazing	flash/no-flash (color)	stereo [23]	edge-preserving smoothing (gray-scale)	edge-preserving smoothing (color)
$\#f_{\rm m}(I)$	1	3	3	3	1	3
$\#f_{\rm m}(I^2)$	1	6	6	6	1	6
$\#f_{\rm m}(p)$	1	1	3	K	0	0
$\#f_{\rm m}(Ip)$	1	3	9	$3K$	0	0
$\#f_{\rm m}(a)$	1	3	9	$3K$	1	6
$\#f_{\rm m}(b)$	1	1	3	K	1	3
total $\#f_{\rm m}$	6	17	33	9+8K	4	18
total running time (ms/Mp)	60	140	300	~100+70K (~1200 if K=16)	40	150

Algorithm 2. 1D Box Filter via Moving Sum
Input: input signal p, radius r
Output: output signal s.

```
1: $s_0 = \sum_{i=0}^{r} p_i$
2: for $i = 1$ to end do
3:     $s_i = s_{i-1} + p_{i+r} - p_{i-r-1}$
4: end for
```

With the $\mathrm{O}(N)$ time mean filter, the guided filter in Algorithm 1 is naturally $\mathrm{O}(N)$ time. Likewise, the color-guidance version in (19), (20), and (21) can be computed in a similar $\mathrm{O}(N)$ time manner.[4] A public Matlab code is available in [22], including both gray-scale and color versions.

In Table 1, we summarize the number of mean filters needed in different scenarios. Here, the d_I and d_p are the number of channels in I and p, respectively. We also list the special case of $I \equiv p$ because the duplication of I and p saves some mean filters. The case of $I \equiv p$ is most concerned in practice.

We experiment with the running time in a PC with an Intel core i7 3.0 GHz CPU and 8 GB RAM. The implementation is in C++. All the algorithms are single-core based and *without* SIMD instructions (e.g., SSE) unless specified. In our experiments, a mean filter takes about 5-7 ms/Mp. The running time is listed in Table 1.

We would like to highlight that the gray-scale image edge-preserving smoothing ($I \equiv p, d_p = 1$) takes only 40 ms/Mp. As a comparison (see Table 2), the O(N) time bilateral filter in [18] is reported 155 ms/Mp using 32-bin histograms ($B = 32$) and 310 ms/Mp using 64-bin as reported in [18]. The method as described in [18] uses *integral histograms*, requiring $6B$ addition/subtraction operations per pixel to build the histogram. Instead, we can adopt the *moving histogram* in [56], which requires $2B + 2$ operations per pixel. With SSE our implementation of [18]+[56] achieves 40 ms/Mp ($B = 32$) and 80 ms/Mp ($B = 64$). Because the moving histogram in [56] is proposed only for median filtering, the combination of [18]+[56], to the best of our knowledge, is an unpublished state-of-the-art of bilateral filtering in the literature. Yang's $\mathrm{O}(N)$ algorithm [19] takes about 120 ms/Mp when $B = 8$ (using the author's public code, with box spatial kernels).

4. In (19), we have to invert a 3-by-3 symmetric matrix. The solution can be computed explicitly in about 30 operations ($+ - */$) per matrix.

Note that the $\mathrm{O}(N)$ time guided filter is nonapproximate and applicable for intensity of any range. On the contrary, the $\mathrm{O}(N)$ time bilateral filter may have noticeable quantization artifacts due to range subsampling. Fig. 11 shows an example where the signal to be filtered is in high dynamic range. Porikli's method [18] has apparent quantization artifacts even when $B = 32$. Similar artifacts are less visible in Yang's method [19] when $B = 8$ thanks to the range interpolation (but takes more time, see Table 2), but still obvious when $B = 4$ because the Nyquist sampling condition becomes severely broken.

For color image filtering (see Table 3), the guided filter takes 300 ms/Mp when $I \neq p$ and 150 ms/Mp when $I \equiv p$. This is substantially faster than the high-dimensional bilateral filter algorithms, like the Gaussian kd-tree [20] (> 10 s/Mp) and the state-of-the-art Permutohedral Lattice [21] (> 1 s/Mp). After the publication of the guided filter in [22], most recently [38] proposed the $\mathrm{O}(N)$ time Domain Transform filters. Its Normalized Convolution (NC) version takes 160 ms/Mp, and its Recursive Filter (RF) version takes 60 ms/Mp for color image filtering, as report in [38]. Though

TABLE 2
Time Comparisons with $\mathrm{O}(N)$ Bilateral Filters for Gray-Scale Image Filtering

Method	Time (ms/Mp)
Porikli [18] (B=32)	155†
Porikli [18] (B=64)	310†
Porikli+[56]+SSE (B=32)	40
Porikli+[56]+SSE (B=64)	80
Yang [19] (B=4, box spatial)	65
Yang [19] (B=8, box spatial)	120
Yang [19] (B=8, Gaussian spatial)	1,000
Ours	40

†*: reported by the authors.*

TABLE 3
Time Comparisons on Color Image Filtering

Method	Time (ms/Mp)
Gaussian kd-tree [20]	>10,000†
Permutohedral Lattice [21]	≥1,000†
Domain Transform (NC) [38]	160†
Domain Transform (RF) [38]	60†
Ours ($I \neq p$)	300
Ours ($I \equiv p$)	150

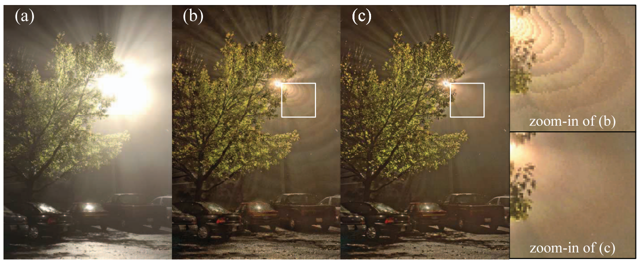


Fig. 11. Quantization artifacts of O(N) time bilateral filter. (a) Input HDR image (32 bit float, displayed by linear scaling). (b) Compressed image using Porikli's O(N) bilateral filter (32 bins) [18]. (c) Compressed image using the guided filter. Note both methods have comparable running time.

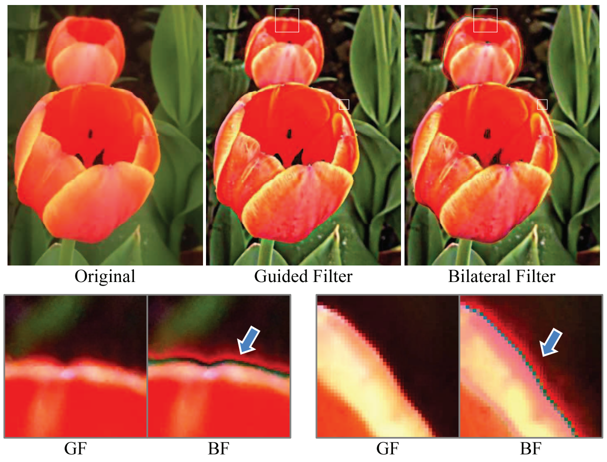


Fig. 12. Detail enhancement. The parameters are $r = 16$, $\epsilon = 0.1^2$ for the guided filter, and $\sigma_s = 16$, $\sigma_r = 0.1$ for the bilateral filter. The detail layer is boosted $\times 5$.

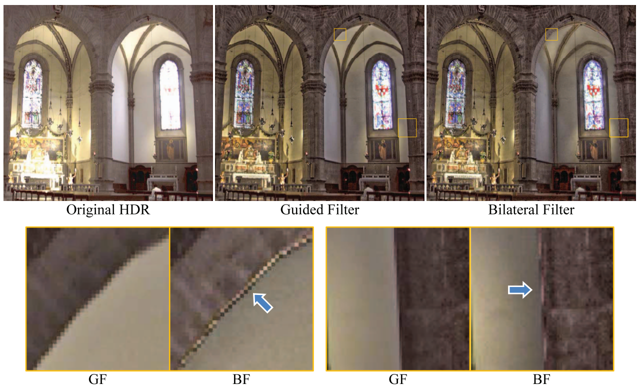


Fig. 13. HDR compression. The parameters are $r = 15$, $\epsilon = 0.12^2$ for the guided filter, and $\sigma_s = 15$, $\sigma_r = 0.12$ for the bilateral filter.

the Domain Transform is very fast, it does not avoid gradient reversal (Fig. 7) and not suitable for transferring structures (Fig. 10).

With the O(N) time recursive Gaussian filter [58], the Gaussian guided filter discussed in Section 3.3 is also O(N) time. The recursive Gaussian filter is more expensive than the box filter (15 operations versus two operations per pixel per x/y direction).

5 EXPERIMENTS

Next, we experiment with the guided filter in a great variety of computer vision and graphics applications.

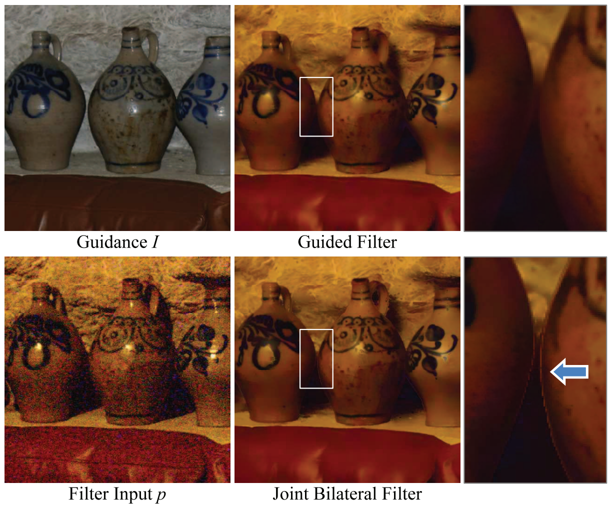


Fig. 14. Flash/no-flash denoising. The parameters are $r = 8$, $\epsilon = 0.2^2$ for the guided filter, and $\sigma_s = 8$, $\sigma_r = 0.2$ for the joint bilateral filter.

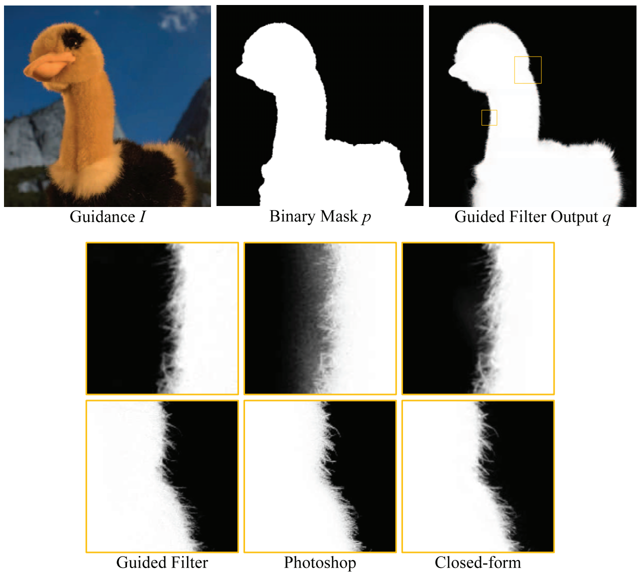


Fig. 15. Guided feathering. A binary mask p is filtered under the guidance of I. In the zoom-in patches, we compare with the photoshop refine edge function and the closed-form matting. For closed-form matting, we erode and dilate the mask to obtain a trimap. The parameters are $r = 60$, $\epsilon = 10^{-6}$ for the guided filter.

Detail enhancement and HDR compression. The method for detail enhancement is described in Section 3.4. The HDR compression is done in a similar way, but compressing the base layer instead of magnifying the detail layer (see [15]). Fig. 12 shows an example for detail enhancement and Fig. 13 shows an example for HDR Compression. The results using the bilateral filter are also provided. As shown in the zoom-in patches, the bilateral filter leads to gradient reversal artifacts. Notice that gradient reversal artifacts often introduce new profiles around edges.

Flash/no-flash denoising. In [14], it is proposed to denoise a no-flash image under the guidance of its flash version. Fig. 14 shows a comparison of using the joint bilateral filter and the guided filter. The gradient reversal artifacts are noticeable near some edges in the joint bilateral filter result.

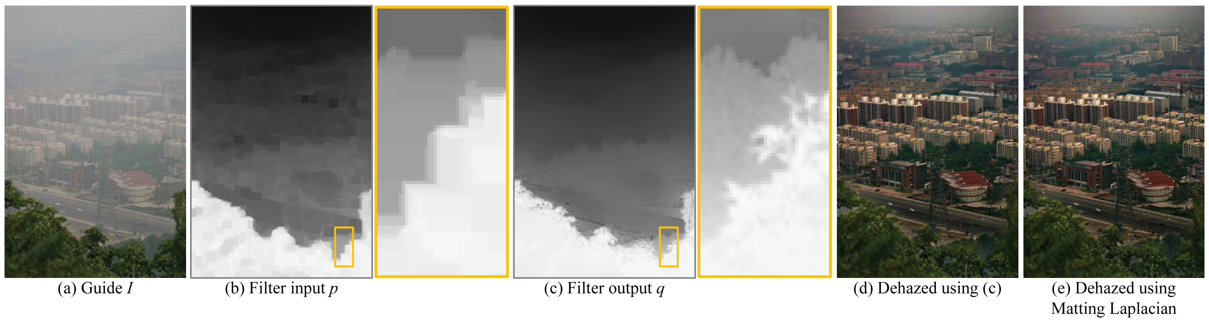


Fig. 16. Haze removal. (a) Hazy image. (b) Raw transmission map [11]. (c) The raw transmission map is refined by the guided filter ($r = 20$, $\epsilon = 10^{-3}$). (d) Using the matting Laplacian in [11]. (e) Recovered image using (c).

Guided feathering/matting. We have introduced the *guided feathering* application in Section 3.6. A similar tool, called "Refine Edge," is provided by the commercial software Adobe Photoshop CS4. An accurate matte can also be computed through the closed-form matting [10]. Fig. 15 shows the comparisons. Our result is visually comparable with the closed-form solution in this short hair case. Both our method and Photoshop provide fast feedback (< 1 s) for this 6-mega-pixel image, while the closed-form solution takes about two minutes to solve a huge linear system.

In the general matting cases the fuzzy region is large; we can adopt color sampling strategies [59] to estimate a more reliable initial guess before filtering. Combined with the global sampling method [59], the guided filter is the best performed filtering-based matting method in the alphamatting benchmark (www.alphamatting.com, performance reported in June 2012).

Single image haze removal. In [11], a haze transmission map is roughly estimated using a dark channel prior, and is refined by solving the matting Laplacian matrix. On the contrary, we simply filter the raw transmission map under the guidance of the hazy image (we first apply a max filter to counteract the morphological effects of the min filter (see [11]), and consider this as the filtering input of the guided filter). The results are visually similar (Fig. 16). The zoom-in windows in Fig. 16 also demonstrate the structure-transferring property of the filter. The running time of the guided filter is about 40 ms for this 600×400 image, in contrast to 10 seconds using the matting Laplacian matrix as in [11].

Joint upsampling. Joint upsampling [31] is to upsample an image under the guidance of another image. Taking the application of colorization [9] as an example. A gray-scale luminance image is colorized through an optimization process. To reduce the running time, the chrominance channels are solved at a coarse resolution and upsampled under the guidance of the full resolution luminance image by the joint bilateral filter [31].

This upsampling process can be performed by the guided filter. The algorithm is slightly different with Algorithm 1 because now we have a guidance image (e.g., luminance) at two scales and a filtering input (e.g., chrominance) at the coarse scale only. In this case, we compute the linear coefficient a and b in using (7) and (8) at the coarse scale, bilinearly upsample them to the fine scale (replacing the mean filter on a and b), and compute the output by $q = aI + b$ at the fine scale. The result is visually comparable to the joint bilateral upsampling (Fig. 17). In our implementation, the joint bilateral upsampling takes 180 ms per mega-pixel output (reported 2s/Mp in [31]), whereas the guided filter upsampling takes about 20 ms/Mp.

Limitations. The guided filter has a common limitation of other explicit filters—it may exhibit halos near some edges. "Halos" refer to the artifacts of unwanted *smoothing* of edges. (On the contrary, "gradient reversal" refers to the artifacts of unwanted sharpening of edges. In the literature, some studies do not distinguish these two kinds of artifacts and simply refer to them as "halos." We discuss them separately in this paper because the reasons for these artifacts are different.) Halos are unavoidable for local filters when the filters are forced to smooth some edges. For example, if strong textures are to be smoothed (see Fig. 18), the weaker edges would also be smoothed. Local filters like guided/bilateral filters would concentrate the blurring near these edges and introduce halos (Fig. 18). Global optimization-based filters (e.g.the WLS filter [8]) would distribute such blurring more globally. The halos are suppressed at the price of global intensity shifting (see Fig. 18 (right)).

6 Conclusion

In this paper, we have presented a novel filter which is widely applicable in computer vision and graphics. Differently from the recent trend toward accelerating the

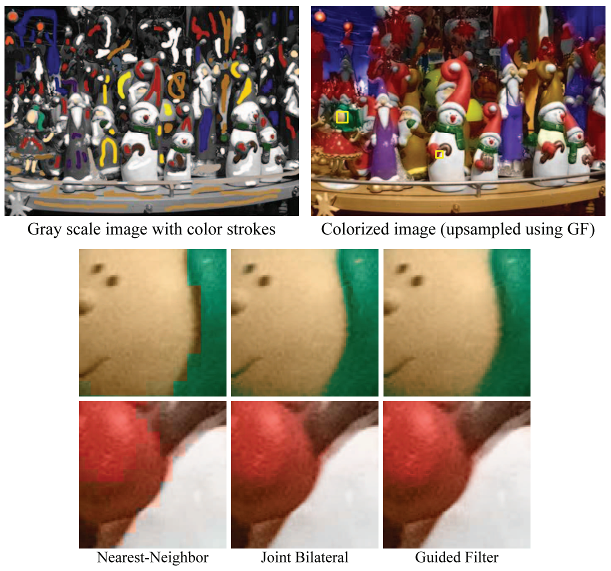

Gray scale image with color strokes
Colorized image (upsampled using GF)
Nearest-Neighbor
Joint Bilateral
Guided Filter

Fig. 17. Joint upsampling for colorization.

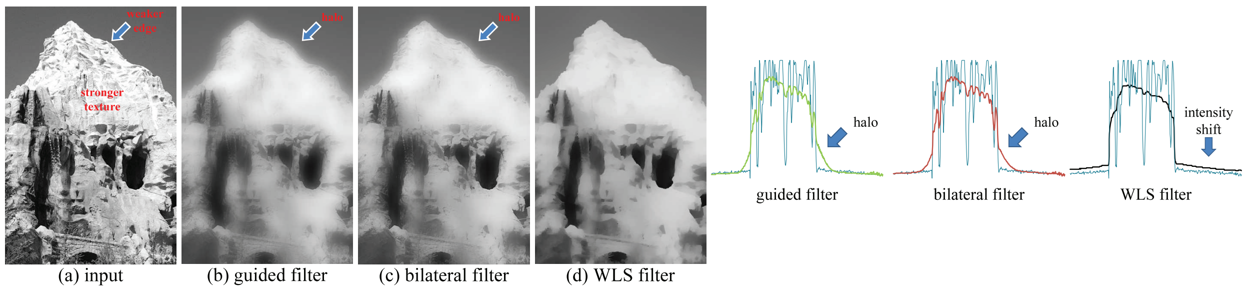


Fig. 18. The halo artifacts. The parameters are $r = 16$, $\epsilon = 0.4^2$ for the guided filter, $\sigma_s = 16$, $\sigma_r = 0.4$ for the bilateral filte, and $\alpha = 1.2$, $\lambda = 5$ (see [8]) for the WLS filter. On the right we show the input signals and filtering results on a scanline.

bilateral filter [17], [18], [19], [20], [21], we design a new filter that exhibits the nice property of edge-preserving smoothing but which can be computed efficiently and nonapproximately. Our filter is more generic than "smoothing" and is applicable for structure-transferring, enabling novel applications of filtering-based feathering/matting and dehazing. Since the local linear model (4) is a kind of patch-wise unsupervised learning, other advanced models/features might be applied to obtain new filters. We leave this for future studies.

References

[1] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," *Proc. IEEE Int'l Computer Vision Conf.*, 1998.

[2] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, second ed. Prentice Hall, 2002.

[3] R. Fattal, D. Lischinski, and M. Werman, "Gradient Domain High Dynamic Range Compression," *Proc. ACM Siggraph*, 2002.

[4] P. Pérez, "Poisson Image Editing," *Proc. ACM Siggraph*, 2003.

[5] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, "Poisson Matting," *Proc. ACM Siggraph*, 2004.

[6] P. Bhat, B. Curless, M. Cohen, and C.L. Zitnick, "Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems," *Proc. European Conf. Computer Vision*, pp. 114-128, 2008.

[7] P. Perona and J. Malik, "Scale-Space and Edge Detection Using Anisotropic Diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629-639, July 1990.

[8] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation," *Proc. ACM Siggraph*, 2008.

[9] A. Levin, D. Lischinski, and Y. Weiss, "Colorization Using Optimization," *Proc. ACM Siggraph*, 2004.

[10] A. Levin, D. Lischinski, and Y. Weiss, "A Closed Form Solution to Natural Image Matting," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[11] K. He, J. Sun, and X. Tang, "Single Image Haze Removal Using Dark Channel Prior," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

[12] V. Aurich and J. Weule, "Non-Linear Gaussian Filters Performing Edge Preserving Diffusion," *Mustererkennung 1995, 17. DAGM-Symp.*, pp. 538-545, 1995.

[13] S.M. Smith and J.M. Brady, "Susan—A New Approach to Low Level Image Processing," *Int'l J. Computer Vision*, vol. 23, pp. 45-78, 1995.

[14] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama, "Digital Photography with Flash and No-Flash Image Pairs," *Proc. ACM Siggraph*, 2004.

[15] F. Durand and J. Dorsey, "Fast Bilateral Filtering for the Display of High-Dynamic-Range Images," *Proc. ACM Siggraph*, 2002.

[16] S. Bae, S. Paris, and F. Durand, "Two-Scale Tone Management for Photographic Look," *Proc. ACM Siggraph*, 2006.

[17] S. Paris and F. Durand, "A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach," *Proc. European Conf. Computer Vision*, 2006.

[18] F. Porikli, "Constant Time O(1) Bilateral Filtering," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[19] Q. Yang, K.-H. Tan, and N. Ahuja, "Real-Time O(1) Bilateral Filtering," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 557-564, 2009.

[20] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-Trees for Fast High-Dimensional Filtering," *Proc. ACM Siggraph*, pp. 21:1-21:12, 2009.

[21] A. Adams, J. Baek, and M.A. Davis, "Fast High-Dimensional Filtering Using the Permutohedral Lattice," *Computer Graphics Forum*, vol. 29, no. 2, pp. 753-762, 2010.

[22] K. He, J. Sun, and X. Tang, "Guided Image Filtering," *Proc. European Conf. Computer Vision*, pp. 1-14, 2010.

[23] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast Cost-Volume Filtering for Visual Correspondence and Beyond," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3017-3024, 2011.

[24] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza, "Linear Stereo Matching," *Proc. IEEE Int'l Computer Vision Conf.* pp. 1708-1715, 2011.

[25] Y. Ding, J. Xiao, and J. Yu, "Importance Filtering for Image Retargeting," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 89-96, 2011.

[26] P. Bauszat, M. Eisemann, and M. Magnor, "Guided Image Filtering for Interactive High-Quality Global Illumination," *Computer Graphics Forum*, vol. 30, no. 4, pp. 1361-1368, June 2011.

[27] K. He, "Guided Image Filtering (Matlab Code)," http://research.microsoft.com/en-us/um/people/kahe/, 2010.

[28] C. Liu, W.T. Freeman, R. Szeliski, and S.B. Kang, "Noise Estimation from a Single Image," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[29] R. Fattal, M. Agrawala, and S. Rusinkiewicz, "Multiscale Shape and Detail Enhancement from Multi-Light Image Collections," *Proc. ACM Siggraph*, 2007.

[30] H. Winnemöller, S.C. Olsen, and B. Gooch, "Real-Time Video Abstraction," *Proc. ACM Siggraph*, 2006.

[31] J. Kopf, M. Cohen, D. Lischinski, and M. Uyttendaele, "Joint Bilateral Upsampling," *Proc. ACM Siggraph*, 2007.

[32] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum, "Progressive Inter-Scale and Intra-Scale Non-Blind Image Deconvolution," *Proc. ACM Siggraph*, pp. 74:1-74:10, 2008.

[33] K.-J. Yoon and I.S. Kweon, "Adaptive Support-Weight Approach for Correspondence Search," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650-656, Apr. 2006.

[34] B. Weiss, "Fast Median and Bilateral Filtering," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 519-526, July 2006.

[35] J. Chen, S. Paris, and F. Durand, "Real-Time Edge-Aware Image Processing with the Bilateral Grid," *ACM Trans. Graphics*, vol. 26, no. 3, 2007.

[36] E.S.L. Gastal and M.M. Oliveira, "Adaptive Manifolds for Real-Time High-Dimensional Filtering," *Proc. ACM Siggraph*, 2012.

[37] R. Fattal, "Edge-Avoiding Wavelets and Their Applications," *Proc. ACM Siggraph*, 2009.

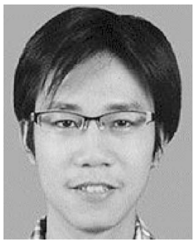
[38] E.S.L. Gastal and M.M. Oliveira, "Domain Transform for Edge-Aware Image and Video Processing," *ACM Trans. Graphics*, vol. 30, no. 4, pp. 69:1-69:12, 2011.

[39] L. Grady, "Random Walks for Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768-1783, Nov. 2006.

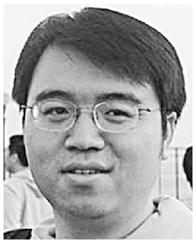
[40] Y. Saad, *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[41] R. Szeliski, "Locally Adapted Hierarchical Basis Preconditioning," *Proc. ACM Siggraph*, 2006.

[42] W.L. Briggs, V.E. Henson, and S.F. McCormick, *A Multigrid Tutorial,* second ed. SIAM, 2000.
[43] M. Elad, "On the Origin of the Bilateral Filter and Ways to Improve It," *IEEE Trans. Image Processing,* vol. 11, no. 10, pp. 1141-1151, Oct. 2002.
[44] M. Kass and J. Solomon, "Smoothed Local Histogram Filters," *ACM Trans. Graphics,* vol. 29, pp. 100:1-100:10, July 2010.
[45] L.I. Rudin, S. Osher, and E. Fatemi, "Nonlinear Total Variation Based Noise Removal Algorithms," *Physica D,* vol. 60, nos. 1-4, pp. 259-268, Nov. 1992.
[46] Y. Li and S. Osher, "A New Median Formula with Applications to PDE Based Denoising," *Comm. Math. Sciences,* vol. 7, pp. 741-753, 2009.
[47] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A New Alternating Minimization Algorithm for Total Variation Image Reconstruction," *SIAM J. Imaging Science,* vol. 1, no. 3, pp. 248-272, 2008.
[48] S. Paris, S.W. Hasinoff, and J. Kautz, "Local Laplacian Filters: Edge-Aware Image Processing with a Laplacian Pyramid," *ACM Trans. Graphics,* vol. 30, no. 4, pp. 68:1-68:12, July 2011.
[49] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image Smoothing via l0 Gradient Minimization," *Proc. ACM Siggraph Asia,* pp. 174:1-174:12, 2011.
[50] A. Zomet and S. Peleg, "Multi-Sensor Super Resolution," *Proc. IEEE Workshop Applications of Computer Vision,* 2002.
[51] N. Draper and H. Smith, *Applied Regression Analysis,* second ed. John Wiley, 1981.
[52] T. Hastie, R. Tibshirani, and J.H. Friedman, *The Elements of Statistical Learning.* Springer, 2003.
[53] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, "From Local Kernel to Nonlocal Multiple-Model Image Denoising," *Int'l J. Computer Vision,* vol. 86, no. 1, pp. 1-32, 2010.
[54] K. Dabov, R. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3D Transform-Domain Collaborative Filtering," *IEEE Trans. Image Processing,* vol. 16, no. 8, pp. 2080-2095, Aug. 2007.
[55] K. He, J. Sun, and X. Tang, "Fast Matting Using Large Kernel Matting Laplacian Matrices," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2165-2172, 2010.
[56] S. Perreault and P. Hébert, "Median Filtering in Constant Time," *IEEE Trans. Image Processing,* vol. 16, no. 9, pp. 2389-2394, Sept. 2007.
[57] F. Crow, "Summed-Area Tables for Texture Mapping," *Proc. ACM Siggraph,* 1984.
[58] R. Deriche, "Recursively Implementing the Gaussian and Its Derivatives," 1993.
[59] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, "A Global Sampling Method for Alpha Matting," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2049-2056, 2011.



Kaiming He received the BS degree from the Academic Talent Program, Physics Department, Tsinghua University in 2007, and the PhD degree from the Department of Information Engineering, the Chinese University of Hong Kong in 2011. He joined Microsoft Research Asia in August 2011. His research interests include computer vision and computer graphics. He has received the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He is a member of the IEEE.



Jian Sun received the BS, MS, and PhD degrees from Xian Jiaotong University in 1997, 2000, and 2003, respectively. He joined Microsoft Research Asia in July 2003. His current two major research interests are interactive computer vision (user interface + vision) and Internet computer vision (large image collection + vision). He is also interested in stereo matching and computational photography. He is a member of the IEEE.

Xiaoou Tang received the BS degree from the University of Science and Technology of China, Hefei, in 1990, the MS degree from the University of Rochester, New York, in 1991, and the PhD degree from the Massachusetts Institute of Technology, Cambridge, in 1996. He is a professor in the Department of Information Engineering, the Chinese University of Hong Kong. He worked as the group manager of the Visual Computing Group at Microsoft Research Asia from 2005 to 2008. He was a program chair of the IEEE International Conference on Computer Vision (ICCV) 2009 and is an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* and the *International Journal of Computer Vision (IJCV)*. His research interests include computer vision, pattern recognition, and video processing. He received the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He is a fellow of the IEEE.