Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2006-073 November 9, 2006

A Fast Approximation of the Bilateral Filter using a Signal Processing Approach

Sylvain Paris and Fredo Durand

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

CSAIL

A Fast Approximation of the Bilateral Filter using a Signal Processing Approach

Sylvain Paris Frédo Durand

Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory

Abstract

The bilateral filter is a nonlinear filter that smoothes a signal while preserving strong edges. It has demonstrated great effectiveness for a variety of problems in computer vision and computer graphics, and fast versions have been proposed. Unfortunately, little is known about the accuracy of such accelerations. In this paper, we propose a new signal-processing analysis of the bilateral filter which complements the recent studies that analyzed it as a PDE or as a robust statistical estimator. The key to our analysis is to express the filter in a higher-dimensional space where the signal intensity is added to the original domain dimensions. Importantly, this signal-processing perspective allows us to develop a novel bilateral filtering acceleration using downsampling in space and intensity. This affords a principled expression of accuracy in terms of bandwidth and sampling. The bilateral filter can be expressed as linear convolutions in this augmented space followed by two simple nonlinearities. This allows us to derive criteria for downsampling the key operations and achieving important acceleration of the bilateral filter. We show that, for the same running time, our method is more accurate than previous acceleration techniques. Typically, we are able to process a 2 megapixel image using our acceleration technique in less than a second, and have the result be visually similar to the exact computation that takes several tens of minutes. The acceleration is most effective with large spatial kernels. Furthermore, this approach extends naturally to color images and cross bilateral filtering.

1 Introduction

The bilateral filter is a nonlinear filter proposed by Aurich and Weule [1995], Smith and Brady [1997], and Tomasi and Manduchi [1998] to smooth images. It has been adopted for several applications such as image denoising [Tomasi and Manduchi, 1998; Liu et al., 2006], relighting and texture manipulation [Oh et al., 2001], dynamic range compression [Durand and Dorsey, 2002], illumination correction [Elad, 2005], and photograph enhancement [Eisemann and Durand, 2004; Petschnigg et al., 2004; Bae et al., 2006]. It has also be adapted to other domains such as mesh fairing [Jones et al., 2003; Fleishman et al., 2003], volumetric denoising [Wong et al., 2004], optical flow and motion estimation [Xiao et al., 2006; Sand and Teller, 2006], and video processing [Bennett and McMillan, 2005; Winnemöller et al., 2006]. This large success stems from several origins. First, its formulation and implementation are simple: a pixel is simply replaced by a weighted mean of its neighbors. And it is easy to adapt to a given context as long as a distance can be computed between two pixel values (*e.g.* distance between hair orientations [Paris et al., 2004]). The bilateral filter is also non-iterative, thereby achieving satisfying results with only a single pass. This makes the filter's parameters relatively intuitive since their effects are not cumulated over several iterations.

The bilateral filter has proven to be very useful, however it is slow. It is nonlinear and its evaluation is computationally expensive since traditional accelerations, such as performing convolution after an FFT, are not applicable. Brute-force computation is on the order of tens of minutes. Nonetheless, solutions have been proposed to speed up the evaluation of the bilateral filter [Durand and Dorsey, 2002; Elad, 2002; Pham and van Vliet, 2005; Weiss, 2006]. Unfortunately, most of these methods rely on approximations that are not grounded on firm theoretical foundations, and it is difficult to evaluate the accuracy that is sacrificed.

Overview In this paper, we build on this body of work but we interpret the bilateral filter in terms of signal processing in a higher-dimensional space. This allows us to derive an improved acceleration scheme that yields equivalent running times but dramatically improves accuracy. The key idea of our technique is to analyze the frequency content of this higher-dimensional space. We demonstrate that in this new representation, the signal of interest is mostly low-frequency and can be accurately approximated using coarse sampling, thereby reducing the amount of data to be processed. The quality of the results is evaluated both numerically and visually to characterize the strengths and limitations of the proposed method. Our study shows that our technique is especially fast with large kernels and achieves a close match to the exact computation. As a consequence, our approach is well-suited for applications in the field of computational photography, as illustrated by a few examples. Furthermore, we believe that our new high-dimensional interpretation of images and its low-sampling-rate encoding provide novel and powerful means for edge-preserving image manipulation in general.

This article extends our conference paper [Paris and Durand, 2006]. We provide more detailed description and discussion, including the algorithm pseudo-code. We conducted new conceptual and quantitative comparisons with existing approximations of the bilateral filter. We describe a new and faster implementation based on direct convolution with a small kernel rather than FFT, and demonstrate and discuss the extension of our acceleration scheme to color-image filtering and cross bilateral filtering.

2 Related Work

The bilateral filter was first introduced by Aurich and Weule [1995] under the name "nonlinear Gaussian filter", then by Smith and Brady [1997] as part of the "SUSAN" framework. It was rediscovered later by Tomasi and Manduchi [1998] who called it the "bilateral filter" which is now the most commonly used name. The filter output at each pixel is a weighted average of its neighbors. The weight assigned to each neighbor decreases with both the distance in the image plane (the *spatial domain* $\mathcal{S}$) and the distance on the intensity axis (the *range domain* $\mathcal{R}$). Using a Gaussian G_σ as a decreasing function, and considering a gray-level image I, the result I^{b} of the bilateral filter is defined by:

$$I^{\mathrm{b}}_{\mathbf{p}} = \frac{1}{W^{\mathrm{b}}_{\mathbf{p}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}} \tag{1a}$$

$$\text{with} \quad W^{\mathrm{b}}_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) \tag{1b}$$

The parameter σ_{s} defines the size of the spatial neighborhood used to filter a pixel, and σ_{r} controls how much an adjacent pixel is downweighted because of the intensity difference. W^{b} normalizes the sum of the weights.

The bilateral filter can be defined with other decreasing functions such as the Tukey function. Durand and Dorsey [2002] noted that these functions introduce only minor variations in the results. In this article, we focus on the Gaussian bilateral filter because all the practical applications use this version. We keep the study of other functions as future work.

2.1 Link with Other Filters

Barash [2002] showed that the two weight functions are actually equivalent to a single weight function based on a distance defined on $\mathcal{S}\times\mathcal{R}$. Using this approach, he related the bilateral filter to adaptive smoothing. Our work follows a similar idea and also uses $\mathcal{S}\times\mathcal{R}$ to describe bilateral filtering. Our formulation is nonetheless significantly different because we not only use the higher-dimensional space for the definition of a distance, but we also use convolution in this space. Elad [2002] demonstrated that the bilateral filter is similar to the Jacobi algorithm, with the specificity that it accounts for a larger neighborhood instead of the closest adjacent pixels usually considered. Buades *et al.* [2005] exposed an asymptotic analysis of the Yaroslavsky filter which is a special case of the bilateral filter with a step function as spatial weight [Yaroslavsky, 1985]. They proved that asymptotically, the Yaroslavsky filter behaves as the Perona-Malik filter, *i.e.* it alternates between smoothing and shock formation depending on the gradient intensity. Aurich and Weule [1995] pointed out the link with robust statistics [Huber, 1981; Hampel et al., 1986; Black et al., 1998]. Durand and Dorsey [2002] also cast their study into this framework and showed that the bilateral filter is a w-estimator [Hampel et al., 1986] (p.116). This explains the role of the range weight in terms of sensitivity to outliers. They also pointed out that the bilateral filter can be seen as an extension of the Perona-Malik filter using a larger neighborhood and a single iteration. Weickert *et al.* [1998] and Barash *et al.* [2003] described acceleration techniques for PDE filters. They split the multi-dimensional differential operators into combinations of one-dimensional operators that can be efficiently integrated. The obtained speed-up stems from the small spatial footprint of the 1D operators, and the extension to bilateral filtering is unclear. Van de Weijer and van den Boomgaard [2001] demonstrated that the bilateral filter is the first iteration of a process that seeks the local mode of the intensity histogram of the adjacent pixels. Mrázek *et al.* [2006] related bilateral filtering to a large family of nonlinear filters. From a single equation, they expressed filters such as anisotropic diffusion and statistical estimators by varying the neighborhood size and the involved functions.

The main difference between our study and existing work is that the previous approaches link bilateral filtering to another nonlinear filter based on PDEs or statistics whereas we cast our study into a signal processing framework. We demonstrate that the bilateral filter can be mainly computed with linear operations, leaving the nonlinearities to be grouped in a final step.

2.2 Variants of the Bilateral Filter

Higher-Order Filters The bilateral filter implicitly assumes that the desired output should be piecewise constant: such an image is unchanged by the filter when the edges between constant parts are high enough. Several articles [Elad, 2002; Choudhury and Tumblin, 2003; Buades et al., 2005] extended the bilateral filter to a *piecewise-linear* assumption. They share the same idea and characterize the local "slope" of the image intensity to better represent the local shape of the signal. Thus, they define a modified filter that better preserves the image characteristics. In particular, they avoid the formation of shocks. We have not explored this direction but it is an interesting avenue for future work.

Cross Bilateral Filter In computational photography applications, it is often useful to decouple the data I to be smoothed from the data E defining the edges to be preserved. For instance, in a "flash no-flash" scenario [Eisemann and Durand, 2004; Petschnigg et al., 2004], a picture P^{nf} is taken in a dark environment without flash and another picture P^{f} is taken with flash. Directly smoothing P^{nf} is hard because of the high noise level typical of low-light images. To address this problem, Eisemann and Durand [2004] and Petschnigg *et al.* [2004] introduced the cross bilateral filter (*a.k.a.* joint bilateral filter) as a variant of the classical bilateral filter. This filter smoothes the no-flash picture $P^{\mathrm{nf}} = I$ while relying on the flash version $P^{\mathrm{f}} = E$ to locate the edges to preserve. The definition is similar to Equation 1 except that E replaces I in the range weight G_{σ_r}:

$$\begin{aligned} I^{\mathrm{c}}_{\mathbf{p}} &= \frac{1}{W^{\mathrm{c}}_{\mathbf{p}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_\mathrm{r}}(|E_{\mathbf{p}}-E_{\mathbf{q}}|)\, I_{\mathbf{q}} \\ \text{with} \quad W^{\mathrm{c}}_{\mathbf{p}} &= \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_\mathrm{r}}(|E_{\mathbf{p}}-E_{\mathbf{q}}|) \end{aligned}$$

Aurich and Weule [1995] introduced ideas related to the cross bilateral filter, but for a single input image when the filter is iterated. After a number of iterations of bilateral filtering, they filter the original images using range weights derived from the last iteration.

The method described in this article also applies to cross bilateral filtering.

Channel Smoothing Felsberg *et al.* [2006] described an efficient smoothing method based on a careful design of the intensity weighting function. They showed that B-splines enable the discretization of the intensity range into a small set of *channels*. Filtering these channels yields smooth images with preserved edges akin to the output of the bilateral filter. B-splines allowed for a precise theoretical characterization of their filter using robust statistics. The downside of B-splines is the higher computational effort required to handle them. This approach and ours are complementary and closely related on a number of points that we discuss in Section 9.2.

2.3 Fast Methods for Bilateral Filtering

The work most related to ours are the techniques that speed up the evaluation of the bilateral filter. There are two categories of acceleration schemes: specialized filters that perform an exact computation but are restricted to a specific scenario, and approximated filters that are more general but do not produce exact results.

Exact Computation Elad [2002] used Gauss-Seidel iterations to accelerate the convergence of iterative filtering. This technique is only useful when the filter is iterated to reach the stable point, which is not the standard use of the bilateral filter (one iteration or only a few). Weiss [2006] describes an efficient algorithm to incrementally compute the intensity histogram of the square windows surrounding each pixel. This technique is primarily used for median filtering. As an extension, Weiss showed that integrating these histograms weighted by a range function G_{σ_r} (cf. Equation 1) is equivalent to calculating a bilateral filter where a step function is used on a square window instead of the isotropic Gaussian G_{σ_s}. This filter actually corresponds to a Yaroslavsky filter computed on square neighborhoods. The achieved computation times are on the order of a few seconds for 8 megapixel images. The downside is that the spatial weighting is restricted to a step function that incurs defects such as ripples near strong edges because its Fourier transform contains numerous zeros. In addition, this technique can only handle color images channel-per-channel, which can introduce color-bleeding artifacts. Although these two techniques are fast and produce an exact result, they are too specialized for many applications such as image editing as shown later.

Approximate Computation At the cost of an approximate result, several authors propose fast methods that address more general scenarios. For instance, Weiss [2006] iterated his filter based on square windows to obtain a smoother profile, thereby removing the ripple defects. To avoid shocks that sharpen edges and result in a cartoon look, the range weights G_{σ_r} are kept constant through the iterations *i.e.* they are always evaluated according to the original input picture. Van de Weijer and van den Boomgaard [2001] showed that it corresponds to a search for the closest local mode in the neighborhood histogram.

Durand and Dorsey [2002] linearized the bilateral filter which makes possible the use of fast Fourier transforms. They also downsample the data to accelerate the computation to a second or less for one-megapixel images. Although their article mentions FFT computation for the linearized bilateral filter, once the data is downsampled, a direct convolution is more efficient because the kernel is small enough. While their article does not emphasize it, their final results are obtained with direct convolution, without FFT. Our technique is related to their work in that we also express the bilateral filter with linear operations and draw much of our speedup from downsampling. However, our formulation relies on a more principled expression based on a new higher dimensional interpretation of images. This affords a solid signal processing perspective on the bilateral filter and improved accuracy.

Pham and van Vliet [2005] applied a 1D bilateral filter independently on each image row and then on each column. It produces smooth results that still preserve edges. The convolutions are performed in the spatial domain (without FFT). All these approximations yield short running times suitable for interactive applications and even real-time processing using modern graphics cards [Winnemöller et al., 2006]. However, no theoretical study is proposed, and the accuracy of these approximations is unclear. In contrast, we base our technique on signal processing grounds which helps us to define a new and meaningful numerical scheme. Our algorithm performs low-pass filtering in a higher-dimensional space. We show that our approach extends naturally to color images and can also be used to speed up cross-bilateral filtering [Eisemann and Durand, 2004; Petschnigg et al., 2004]. The cost of a higher-dimensional convolution is offset by downsampling the data without significant accuracy loss, thereby yielding a better precision for running times equivalent to existing methods.

2.4 Contributions

This paper introduces the following contributions:

- An interpretation of the bilateral filter in a signal processing framework. Using a higher dimensional space, we formulate the bilateral filter as a convolution followed by simple nonlinearities.
- Using this higher dimensional space, we demonstrate that the convolution computation can be downsampled without significant impact on the resulting accuracy. This approximation technique enables a speed-up of several orders of magnitude while controlling the induced error.
- We evaluate the accuracy and performance of the proposed acceleration over several scenarios. The obtained results are compared to existing techniques, thereby characterizing the strengths and limitations of our approach.
- We show that this method naturally handles color images and can be easily adapted to cross bilateral filtering.

2.5 Notation

Table 1 summarizes the main notation we use throughout this article. All the vectors in this paper are column vectors. We sometimes use a row notation and omit the transpose sign to prevent clutter in the equations.

$\mathcal{S}$	spatial domain	$\mathcal{R}$	range domain
I, W...	2D functions defined on $\mathcal{S}$	i, w...	3D functions defined on $\mathcal{S}\times\mathcal{R}$
$\mathbf{p}$, $\mathbf{C}$...	vectors	$\mathbf{p} \in \mathcal{S}$	pixel position (2D vector)
$\|\mathbf{x}\|$	L_2 norm of vector $\mathbf{x}$	$I_\mathbf{p} \in \mathcal{R}$	image intensity at $\mathbf{p}$
$\otimes$	convolution operator	$\delta(x)$	Kronecker symbol (1 if $x=0$, 0 otherwise)
G_σ	1D Gaussian: $x \mapsto \exp(-\frac{x^2}{2\sigma^2})$	$g_{\sigma_\mathrm{s},\sigma_\mathrm{r}}$	3D Gaussian: $(\mathbf{x},\zeta) \in \mathcal{S}\times\mathcal{R} \mapsto \exp(-\frac{\mathbf{x}\cdot\mathbf{x}}{2\sigma_\mathrm{s}{}^2} - \frac{\zeta^2}{2\sigma_\mathrm{r}{}^2})$
s_s, s_r	sampling rates (space and range)	σ_s, σ_r	Gaussian parameters (space and range)
I^b	result of the bilateral filter	W^b	normalization factor

Table 1: Notation used in the paper.

3 Signal Processing Approach

We propose a new interpretation of the bilateral as a higher-dimensional convolution followed by two nonlinearities. For this, we propose two important re-interpretations of the filter that respectively deal with its two nonlinear components: the normalization division and the data-dependent weights (Equation 1). First, we define a homogeneous intensity that will allow us to obtain the normalization term $W^\mathrm{b}_\mathbf{p}$ as a homogeneous component. Second, we introduce an additional dimension to the 2D image domain, corresponding to the image intensity *a.k.a.* range. While the visualization of images as height fields in such 3D space is not new, we actually go further and interpret filtering using functions over this full 3D domain, which allows us to express the bilateral filter as a linear shift-invariant convolution in 3D. This convolution is followed by simple pixel-wise nonlinearities to extract the relevant output and perform the normalization by our homogeneous component.

3.1 Homogeneous Intensity

Similar to any weighted average, the two lines of Equation 1 that define the bilateral filter output and its normalization factor are almost the same. The main difference is the normalizing factor $W^\mathrm{b}_\mathbf{p}$ and the image intensity $I_\mathbf{q}$ in the first equation. We emphasize this similarity by multiplying both sides of Equation 1a by $W^\mathrm{b}_\mathbf{p}$. We then rewrite the equations using two-dimensional vectors:

$$\begin{pmatrix} W^\mathrm{b}_\mathbf{p}\, I^\mathrm{b}_\mathbf{p} \\ W^\mathrm{b}_\mathbf{p} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_\mathrm{r}}(|I_\mathbf{p}-I_\mathbf{q}|) \begin{pmatrix} I_\mathbf{q} \\ 1 \end{pmatrix} \tag{2}$$

where G_{σ_s} and G_{σ_r} are Gaussian functions, $\mathcal{S}$ is the spatial domain, I the input image, and I^b the result of the bilateral filter. To maintain the property that the bilateral filter is a weighted mean, we assign a weight $W=1$ to the input values:

$$\begin{pmatrix} W^\mathrm{b}_\mathbf{p}\, I^\mathrm{b}_\mathbf{p} \\ W^\mathrm{b}_\mathbf{p} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_\mathrm{r}}(|I_\mathbf{p}-I_\mathbf{q}|) \begin{pmatrix} W_\mathbf{q}\, I_\mathbf{q} \\ W_\mathbf{q} \end{pmatrix} \tag{3}$$

By assigning a couple $(W_{\mathbf{q}}\ I_{\mathbf{q}}, W_{\mathbf{q}})$ to each pixel $\mathbf{q}$, we express the filtered pixels as linear combinations of their adjacent pixels. Of course, we have not "removed" the division since to access the actual value of the intensity, the first coordinate (WI) still has to be divided by the second one (W). This is similar to homogeneous coordinates used in projective geometry. Adding an extra coordinate to our data makes most of the computation pipeline computable with linear operations; a division is made only at the final stage. Inspired by this parallel, we call the two-dimensional vector (WI, W) the *homogeneous intensity*. It is also related to the use of pre-multiplied alpha in image algebra [Porter and Duff, 1984]. We discuss this aspect in Section 7.1.1.

An important aspect of this homogeneous formulation is its *exact equivalence* with the original formulation (Equation 1). Using a homogeneous representation enables a simpler formulation of the filter. However, although Equation 3 is a linear combination, this does not define a linear filter yet since the weights depend on the actual values of the pixels. The next section addresses this issue.

3.2 The Bilateral Filter as a Convolution

If we ignore the term $G_{\sigma_\mathrm{r}}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)$, Equation 3 is a classical convolution by a Gaussian kernel: $(W^\mathrm{b}\, I^\mathrm{b}, W^\mathrm{b}) = G_{\sigma_\mathrm{s}} \otimes (WI, W)$. Furthermore, it has been pointed out that the product of the spatial and range Gaussian defines a higher dimensional Gaussian in the 3D product space of the domain and range of the image. However, this 3D Gaussian interpretation has so far only been used to illustrate the weights of the filter, not to linearize computation. Since these weights are in 3D but the summation in Equation 1 is only over the 2D spatial domain, it does not define a convolution. To overcome this, we push the 3D interpretation further and define an intensity value for each point of the product space so that we can define a summation over this full 3D space.

Formally, we introduce an additional dimension ζ and define the intensity I for each point (x, y, ζ). With the Kronecker symbol $\delta(\zeta)$ ($\delta(0) = 1$, $\delta(\zeta) = 0$ otherwise) and $\mathcal{R}$ the interval on which the intensity is defined, we rewrite Equation 3 using $\delta(\zeta - I_{\mathbf{q}})$ such that the terms are cancelled when $\zeta \neq I_{\mathbf{q}}$:

$$\begin{pmatrix} W^\mathrm{b}_{\mathbf{p}}\ I^\mathrm{b}_{\mathbf{p}} \\ W^\mathrm{b}_{\mathbf{p}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} \sum_{\zeta\in\mathcal{R}} G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_\mathrm{r}}(|I_{\mathbf{p}} - \zeta|)\, \delta(\zeta - I_{\mathbf{q}}) \begin{pmatrix} W_{\mathbf{q}}\ I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix} \tag{4}$$

Equation 4 is a sum over the product space $\mathcal{S}\times\mathcal{R}$. We now focus on this space. We use lowercase names for the functions defined on $\mathcal{S}\times\mathcal{R}$. The product $G_{\sigma_\mathrm{s}} G_{\sigma_\mathrm{r}}$ defines a separable Gaussian kernel $g_{\sigma_\mathrm{s},\sigma_\mathrm{r}}$ on $\mathcal{S}\times\mathcal{R}$:

$$g_{\sigma_\mathrm{s},\sigma_\mathrm{r}} : \quad (\mathbf{x}\in\mathcal{S}, \zeta\in\mathcal{R}) \quad \mapsto \quad G_{\sigma_\mathrm{s}}(\|\mathbf{x}\|)\, G_{\sigma_\mathrm{r}}(|\zeta|) \tag{5}$$

From the remaining part of Equation 4, we build two functions i and w:

$$i : \quad (\mathbf{x}\in\mathcal{S}, \zeta\in\mathcal{R}) \quad \mapsto \quad I_{\mathbf{x}} \tag{6a}$$

$$w : \quad (\mathbf{x}\in\mathcal{S}, \zeta\in\mathcal{R}) \quad \mapsto \quad \delta(\zeta - I_{\mathbf{x}}) \;=\; \delta(\zeta - I_{\mathbf{x}})\, W_{\mathbf{x}} \qquad \text{since} \qquad W_{\mathbf{x}} = 1 \tag{6b}$$

With Definitions 6, we rewrite the right side of Equation 4:

$$\delta(\zeta - I_{\mathbf{q}}) \begin{pmatrix} W_{\mathbf{q}}\ I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} \delta(\zeta - I_{\mathbf{q}})\, W_{\mathbf{q}}\ I_{\mathbf{q}} \\ \delta(\zeta - I_{\mathbf{q}})\, W_{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} w(\mathbf{q},\zeta)\ i(\mathbf{q},\zeta) \\ w(\mathbf{q},\zeta) \end{pmatrix} \tag{7}$$

Then with Definition 5, we get:

$$\begin{pmatrix} W^{\mathrm{b}}_{\mathbf{p}}\ I^{\mathrm{b}}_{\mathbf{p}} \\ W^{\mathrm{b}}_{\mathbf{p}} \end{pmatrix} = \sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}}(\mathbf{p}-\mathbf{q}, I_p-\zeta) \begin{pmatrix} w(\mathbf{q},\zeta)\ i(\mathbf{q},\zeta) \\ w(\mathbf{q},\zeta) \end{pmatrix} \tag{8}$$

The above formula corresponds to the value at point $(\mathbf{p}, I_{\mathbf{p}})$ of a convolution between $g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}}$ and the two-dimensional function (wi, w):

$$\begin{pmatrix} W^{\mathrm{b}}_{\mathbf{p}}\ I^{\mathrm{b}}_{\mathbf{p}} \\ W^{\mathrm{b}}_{\mathbf{p}} \end{pmatrix} = \left[g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}} \otimes \begin{pmatrix} wi \\ w \end{pmatrix} \right] (\mathbf{p}, I_{\mathbf{p}}) \tag{9}$$

According to the above equation, we introduce the functions i^{b} and w^{b}:

$$(w^{\mathrm{b}}\ i^{\mathrm{b}}, w^{\mathrm{b}}) = g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}} \otimes (wi, w) \tag{10}$$

Thus, we have reached our goal. The bilateral filter is expressed as a convolution followed by nonlinear operations:

linear: 3D convolution

$$(w^{\mathrm{b}}\ i^{\mathrm{b}}, w^{\mathrm{b}}) = g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}} \otimes (wi, w) \tag{11a}$$

nonlinear: slicing+division

$$I^{\mathrm{b}}_{\mathbf{p}} = \frac{w^{\mathrm{b}}(\mathbf{p}, I_{\mathbf{p}})\ i^{\mathrm{b}}(\mathbf{p}, I_{\mathbf{p}})}{w^{\mathrm{b}}(\mathbf{p}, I_{\mathbf{p}})} \tag{11b}$$

The nonlinear section is actually composed of two operations. The functions $w^{\mathrm{b}} i^{\mathrm{b}}$ and w^{b} are evaluated at point $(\mathbf{p}, I_{\mathbf{p}})$. We name this operation *slicing*. The second nonlinear operation is the division that retrieves the intensity value from the homogeneous vector. This division corresponds to applying the normalization that we delayed earlier. In our case, slicing and division commute *i.e.* the result is independent of their order because $g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}}$ is positive and w values are 0 and 1, which ensures that w^{b} is positive.

3.3 Intuition

To gain more intuition about our formulation of the bilateral filter, we propose an informal description of the process before further discussing its consequences.

The spatial domain $\mathcal{S}$ is a classical xy image plane and the range domain $\mathcal{R}$ is a simple axis labelled ζ. The w function can be interpreted as "the plot in the $xy\zeta$ space of $\zeta = I(x,y)$" *i.e.* w is null everywhere except on the points $(x, y, I(x,y))$ where it is equal to 1. The wi product is similar to w. Instead of using binary values 0 or 1 to "plot I", we use 0 or $I(x,y)$ *i.e.* it is a plot with a pen whose brightness equals the plotted value. An example is shown in Figure 1.

Using these two functions wi and w, the bilateral filter is computed as follows. First, we "blur" wi and w, that is we convolve wi and w with a Gaussian defined on $xy\zeta$. This results in the functions $w^{\mathrm{b}}\ i^{\mathrm{b}}$ and w^{b}. For each point of the $xy\zeta$ space, we compute $i^{\mathrm{b}}(x,y,\zeta)$ by dividing $w^{\mathrm{b}}(x,y,\zeta)\ i^{\mathrm{b}}(x,y,\zeta)$ by $w^{\mathrm{b}}(x,y,\zeta)$. The final step is to get the value of the pixel (x,y) of the filtered image I^{b}. This corresponds directly to the value of i^{b} at $(x, y, I(x,y))$ which is the point where the input image I was "plotted". Figure 1 illustrates this process on a simple 1D image.

Note that although the 3D functions get smoothed agressively, the slicing nonlinearity makes the output of the filter piecewise-smooth and the strong edges of the input are preserved.

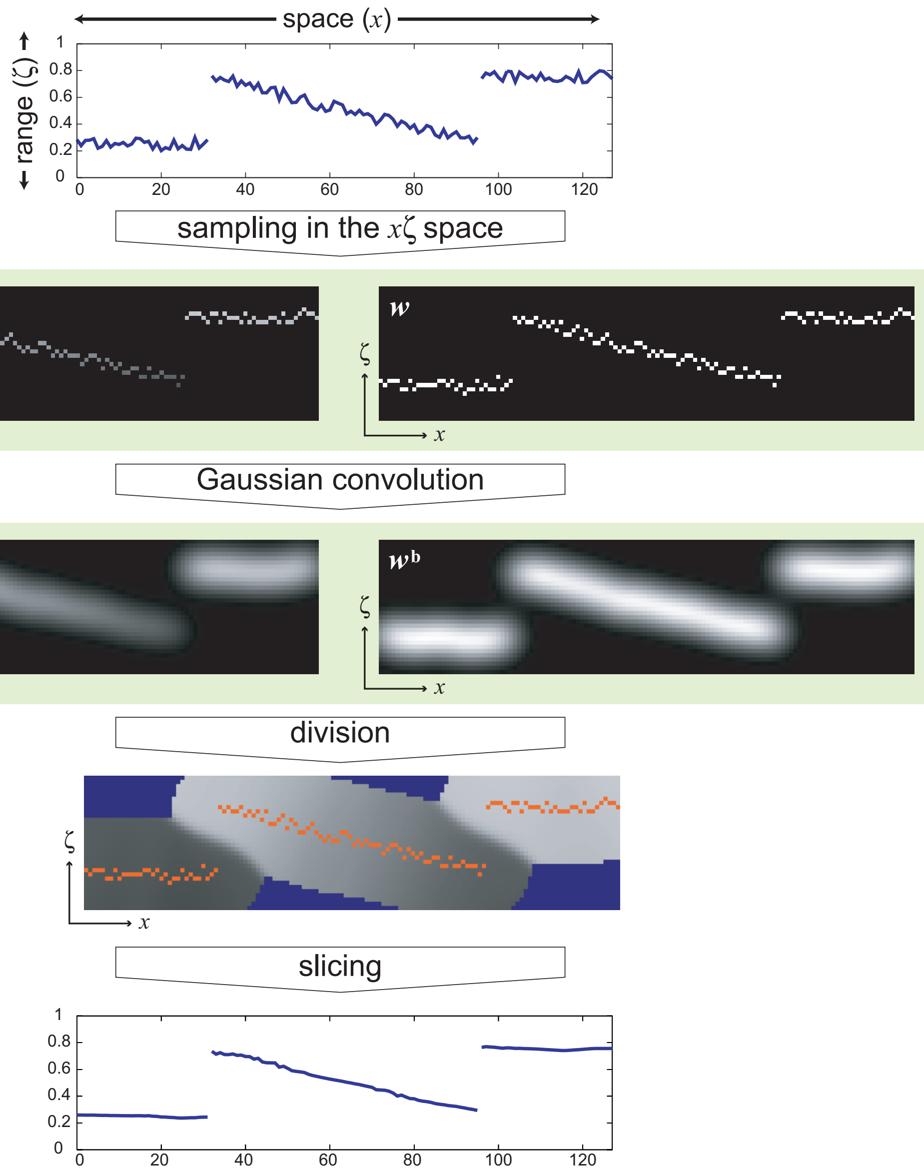


Fig. 1: Our computation pipeline applied to a 1D signal. The original data (top row) are represented by a two-dimensional function (wi, w) (second row). This function is convolved with a Gaussian kernel to form $(w^b\, i^b, w^b)$ (third row). The first component is then divided by the second (fourth row, blue area is undefined because of numerical limitation, $w^b \approx 0$). Then the final result (last row) is extracted by sampling the former result at the location of the original data (shown in red on the fourth row).

4 Fast Approximation

The key idea which speeds up the computation is computing the 3D convolution at a coarse resolution. For this, we demonstrate that the wi and w functions can be downsampled without introducing significant errors. In fact, we never construct the full-resolution product space. This ensures the good memory and speed performance of our method. We discuss the practical implementation of this strategy and analyze the accuracy and performance of the proposed technique.

4.1 Downsampling the Data

We have shown that the bilateral filter can be interpreted as a Gaussian filter in a product space. Our acceleration scheme directly follows from the fact that this operation is a low-pass filter. $(w^{\mathrm{b}}\ i^{\mathrm{b}}, w^{\mathrm{b}})$ is a band-limited function which is well approximated by its low frequencies. According to the sampling theorem [Smith, 2002] (p.35), it is sufficient to sample with a rate at least half of the smallest wavelength considered. We exploit this idea by performing the convolution at a lower resolution. Formally, we downsample (wi, w), perform the convolution, and upsample the result as indicated by the following equations. Note, however, that our implementation never stores full-resolution data: the high-resolution data is built at each pixel and downsampled on the fly, and we upsample only at the slicing location. The notion of using a high-resolution 3D space which we then downsample is used only for formal exposition (cf. Section 4.2 for implementation details):

$$(w_{\downarrow} i_{\downarrow}, w_{\downarrow}) \quad = \quad \text{downsample}(wi, w) \qquad \text{[computed on the fly]} \tag{12a}$$

$$(w^{\mathrm{b}}_{\downarrow}\ i^{\mathrm{b}}_{\downarrow}, w^{\mathrm{b}}_{\downarrow}) \quad = \quad g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}} \otimes (w_{\downarrow} i_{\downarrow}, w_{\downarrow}) \tag{12b}$$

$$(w^{\mathrm{b}}_{\downarrow\uparrow}\ i^{\mathrm{b}}_{\downarrow\uparrow}, w^{\mathrm{b}}_{\downarrow\uparrow}) \quad = \quad \text{upsample}(w^{\mathrm{b}}_{\downarrow}\ i^{\mathrm{b}}_{\downarrow}, w^{\mathrm{b}}_{\downarrow}) \qquad \text{[evaluated only at slicing location]} \tag{12c}$$

The rest of the computation remains the same except that we slice and divide $(w^{\mathrm{b}}_{\downarrow\uparrow}\ i^{\mathrm{b}}_{\downarrow\uparrow}, w^{\mathrm{b}}_{\downarrow\uparrow})$ instead of $(w^{\mathrm{b}}\ i^{\mathrm{b}}, w^{\mathrm{b}})$, using the same $(\mathbf{p}, I_{\mathbf{p}})$ points. Since slicing occurs at points where $w = 1$, it guarantees $w^{\mathrm{b}} \geq g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}}(0)$, which ensures that we do not divide by small numbers that would degrade our approximation.

We use box-filtering for the prefilter of the downsampling (*a.k.a.* average downsampling), and linear upsampling. While these filters do not have perfect frequency responses, they offer much better performances than schemes such as tri-cubic filters. We name s_{s} and s_{r} the sampling rates of $\mathcal{S}$ and $\mathcal{R}$, *i.e.* we use a box function that measures $s_{\mathrm{s}} \times s_{\mathrm{s}}$ pixels and s_{r} intensity units.

4.2 Implementation

This section details the actual implementation of our algorithm. We have not performed low-level optimization. Although some optimization may be introduced by compilers (we used GCC 3.3.5), our code does not explicitly rely on vector instructions of modern CPU nor on the streaming capacities of recent GPU. Exploring such optimizations is nevertheless an exciting avenue for future work. Our code is publicly available on our webpage: `http://people.csail.mit.edu/sparis/#code`. The software is open-source and under the MIT license.

4.2.1 Design Overview

In order to achieve high performance, we never build the $\mathcal{S}\times\mathcal{R}$ space at the fine resolution. We only deal with the downsampled version. In practice, this means that we directly construct the downsampled $\mathcal{S}\times\mathcal{R}$ domain from the image, *i.e.* we store each pixel directly into the corresponding coarse bin. At the slicing stage, we evaluate the upsampled values only at the points $(\mathbf{p}, I_{\mathbf{p}})$, *i.e.* we do not upsample the entire $\mathcal{S}\times\mathcal{R}$ space, only the points corresponding to the final result.

4.2.2 Pseudo-code

Our algorithm is summarized in Figure 2. Step 3 directly computes the downsampled versions of wi and w from the input image I. The high-resolution value (wi, w) computed in Step 3a is a temporary variable. Downsampling is performed on the fly: We compute (wi, w) (Step 3a) and directly add it into the low-resolution array ($w_\downarrow\, i_\downarrow$) (Step 3c). In Step 3b, we offset the $\mathcal{R}$ coordinate to ensure that the array indices start at 0. Note that in Step 3c, we do not need to normalize the values by the size of the downsampling box. This results in a uniform scaling that propagates through the convolution and upsampling operations. It is cancelled by the final normalization (Step 5b) and thus does not affect the result. Step 4 performs the convolution. We discuss two different options in the following section. Steps 5a and 5b correspond respectively to the slicing and division nonlinearities described previously.

For completeness, Figure 3 gives the pseudo-code of a direct implementation of Equation 1.

FAST BILATERAL FILTER

input: image I
Gaussian parameters σ_{s} and σ_{r}
sampling rates s_{s} and s_{r}

output: filtered image I^{b}

1. Initialize all $w_\downarrow\, i_\downarrow$ and $w_\downarrow$ values to 0.
2. Compute the minimum intensity value:

$$I_{\min} \quad \leftarrow \quad \min_{(X,Y)\in\mathcal{S}} I(X,Y)$$

3. For each pixel $(X,Y) \in \mathcal{S}$ with an intensity $I(X,Y) \in \mathcal{R}$
   (a) Compute the homogeneous vector (wi, w):

$$(wi, w) \quad \leftarrow \quad \big(I(X,Y), 1\big)$$

   (b) Compute the downsampled coordinates (with $[\,\cdot\,]$ the rounding operator)

$$(x, y, \zeta) \quad \leftarrow \quad \left(\left[\frac{X}{s_{\mathrm{s}}}\right], \left[\frac{Y}{s_{\mathrm{s}}}\right], \left[\frac{I(X,Y) - I_{\min}}{s_{\mathrm{r}}}\right]\right)$$

   (c) Update the downsampled $\mathcal{S}\times\mathcal{R}$ space

$$\begin{pmatrix} w_\downarrow\, i_\downarrow(x,y,\zeta) \\ w_\downarrow(x,y,\zeta) \end{pmatrix} \quad \leftarrow \quad \begin{pmatrix} w_\downarrow\, i_\downarrow(x,y,\zeta) \\ w_\downarrow(x,y,\zeta) \end{pmatrix} \quad + \quad \begin{pmatrix} wi \\ w \end{pmatrix}$$

4. Convolve $(w_\downarrow\, i_\downarrow, w_\downarrow)$ with a 3D Gaussian g whose parameters are $\sigma_{\mathrm{s}}/s_{\mathrm{s}}$ and $\sigma_{\mathrm{r}}/s_{\mathrm{r}}$

$$(w^{\mathrm{b}}_\downarrow\, i^{\mathrm{b}}_\downarrow, w^{\mathrm{b}}_\downarrow) \quad \leftarrow \quad (w_\downarrow\, i_\downarrow, w_\downarrow) \ \otimes \ g$$

5. For each pixel $(X,Y) \in \mathcal{S}$ with an intensity $I(X,Y) \in \mathcal{R}$
   (a) Tri-linearly interpolate the functions $w^{\mathrm{b}}_\downarrow\, i^{\mathrm{b}}_\downarrow$ and $w^{\mathrm{b}}_\downarrow$ to obtain $W^{\mathrm{b}}\, I^{\mathrm{b}}$ and W^{b}:

$$W^{\mathrm{b}}\, I^{\mathrm{b}}(X,Y) \quad \leftarrow \quad \text{interpolate}\left(w^{\mathrm{b}}_\downarrow\, i^{\mathrm{b}}_\downarrow, \frac{X}{s_{\mathrm{s}}}, \frac{Y}{s_{\mathrm{s}}}, \frac{I(X,Y)}{s_{\mathrm{r}}}\right)$$

$$W^{\mathrm{b}}(X,Y) \quad \leftarrow \quad \text{interpolate}\left(w^{\mathrm{b}}_\downarrow, \frac{X}{s_{\mathrm{s}}}, \frac{Y}{s_{\mathrm{s}}}, \frac{I(X,Y)}{s_{\mathrm{r}}}\right)$$

   (b) Normalize the result

$$I^{\mathrm{b}}(X,Y) \quad \leftarrow \quad \frac{W^{\mathrm{b}}\, I^{\mathrm{b}}(X,Y)}{W^{\mathrm{b}}(X,Y)}$$

Fig. 2: Pseudo-code of our algorithm. The algorithm is designed such that we never build the full-resolution $\mathcal{S}\times\mathcal{R}$ space.

Brute Force Bilateral Filter	input:	image I
		Gaussian parameters σ_{s} and σ_{r}
	output:	filtered image I^{b}

1. Initialize all I^{b} and W^{b} values to 0.
2. For each pixel $(X, Y) \in \mathcal{S}$ with an intensity $I(X, Y) \in \mathcal{R}$
   (a) For each pixel $(X', Y') \in \mathcal{S}$ with an intensity $I(X', Y') \in \mathcal{R}$
      i. Compute the associated weight:
      $$\text{weight} \quad \leftarrow \quad \exp\left(-\frac{(I(X',Y') - I(X,Y))^2}{2\sigma_{\mathrm{r}}{}^2} - \frac{(X'-X)^2 + (Y'-Y)^2}{2\sigma_{\mathrm{s}}{}^2}\right)$$
      ii. Update the weight sum $W^{\mathrm{b}}(X, Y)$:
      $$W^{\mathrm{b}}(X,Y) \quad \leftarrow \quad W^{\mathrm{b}}(X,Y) \; + \; \text{weight}$$
      iii. Update $I^{\mathrm{b}}(X, Y)$:
      $$I^{\mathrm{b}}(X,Y) \quad \leftarrow \quad I^{\mathrm{b}}(X,Y) \; + \; \text{weight} \times I^{\mathrm{b}}(X',Y')$$
   (b) Normalize the result:
   $$I^{\mathrm{b}}(X,Y) \quad \leftarrow \quad \frac{I^{\mathrm{b}}(X,Y)}{W^{\mathrm{b}}(X,Y)}$$

Fig. 3: Pseudo-code of the brute force algorithm.

4.2.3 Efficient Convolution

Convolution is at the core of our technique and its efficiency is important to the overall performance of our algorithm. We describe two options that will be evaluated in Section 5.

Full Kernel One can use the fast Fourier transform to efficiently compute the convolution. This has the advantage that the computation does not depend on the Gaussian size, but on only the domain size. Thus, we can use an exact Gaussian kernel. In this case, the approximation comes only from the downsampling and upsampling applied to the data before and after the convolution and from cross-boundary artifacts inherent in FFT. To minimize these artifacts, the domain is padded with zeros over 2σ.

Truncated Kernel When the convolution kernel is small, an explicit computation in the spatial domain is an effective alternative since only a limited number of samples are involved. Although the Gaussian kernel is not compact, its tail falls off quickly. We therefore use the classical approximation by truncating the kernel beyond 2σ. We only apply this technique for the case where the sampling rate equals the Gaussian standard deviation (*i.e.* $s = \sigma$) since, in this case, the downsampled kernel is isotropic and has a variance equal to 1. The truncated kernel then covers a $5 \times 5 \times 5$ region which is compact enough to ensure fast computation. Since the kernel is shift-invariant and separable, we further shorten the running times by replacing the 3D kernel by three 1D kernels. This well-known technique reduces the number of points to an average of 15 ($= 3 \times 5$) instead of 125 ($= 5^3$). Unlike the separable approximation [Pham and van Vliet, 2005], this separation is exact since the original 3D kernel is Gaussian and therefore separable.

We shall see that this convolution in the spatial domain with a truncated kernel yields a better ratio of numerical accuracy over running time than the frequency-space convolution with the full kernel. This latter option remains nonetheless useful to achieve high numerical accuracy, at the cost of slower performances.

5 Evaluation of our Approximation

This section investigates the accuracy and performance of our technique compared to the exact computation. The timings are measured on an Intel Xeon 2.8GHz with 1MB cache using double-precision floating-point numbers.

On Ground Truth In practical applications such as photograph enhancement [Bae et al., 2006] or user-driven image editing [Weiss, 2006], the notions of numerical accuracy and ground truth are not always well defined. Running times can be objectively measured, but other aspects are elusive and difficult to quantify. Furthermore, whether the bilateral filter is the ideal filter for an application is a separate question.

Since our method achieves acceleration through approximation, we have chosen to measure the numerical accuracy by comparing the outputs of our technique and of other existing approximations with the result of the original bilateral filter. This comparison pertains only to the "numerical quality" of the approximation, and the readers should keep in mind that numerical differences do not necessarily produce unsatisfying outputs. To balance this numerical aspect, we also provide visual results to let the readers examine by themselves the outputs. Important criteria are then the regularity of the achieved smoothing, artifacts that add visual features which do not exist in the input picture, tone (or color) faithfulness, and so on.

5.1 Numerical Accuracy

To evaluate the error induced by our approximation, we compare the result $I^{\mathrm{b}}_{\downarrow\uparrow}$ from our fast algorithm to the exact result I^{b} obtained from Equation 1. We have chosen three images as different as possible to cover a broad spectrum of content (Figure 12):

- An artificial image with various edges, frequencies, and white noise.
- An architectural picture structured along two main directions.
- And a photograph of a natural scene with more stochastic structure.

To express numerical accuracy, we compute the peak signal-to-noise ratio (PSNR) considering $\mathcal{R} = [0;1]$: $\mathrm{PSNR}(I^{\mathrm{b}}_{\downarrow\uparrow}) = -10\ \log_{10}\left(\frac{1}{|\mathcal{S}|}\sum_{\mathbf{p}\in\mathcal{S}}\left|I^{\mathrm{b}}_{\downarrow\uparrow}(\mathbf{p}) - I^{\mathrm{b}}(\mathbf{p})\right|^2\right)$. For instance, considering intensity values encoded on 8 bits, if two images differ from one gray level at each pixel, the resulting PSNR is 48dB. As a guideline, PSNR values above 40dB often corresponds to limited differences, almost invisible. This should be confirmed by a visual inspection since a high PSNR can "hide" a few large errors because it is a mean over the whole image.

The box downsampling and linear upsampling schemes yield very satisfying results while being computationally efficient. We tried other techniques such as tri-cubic downsampling and upsampling.

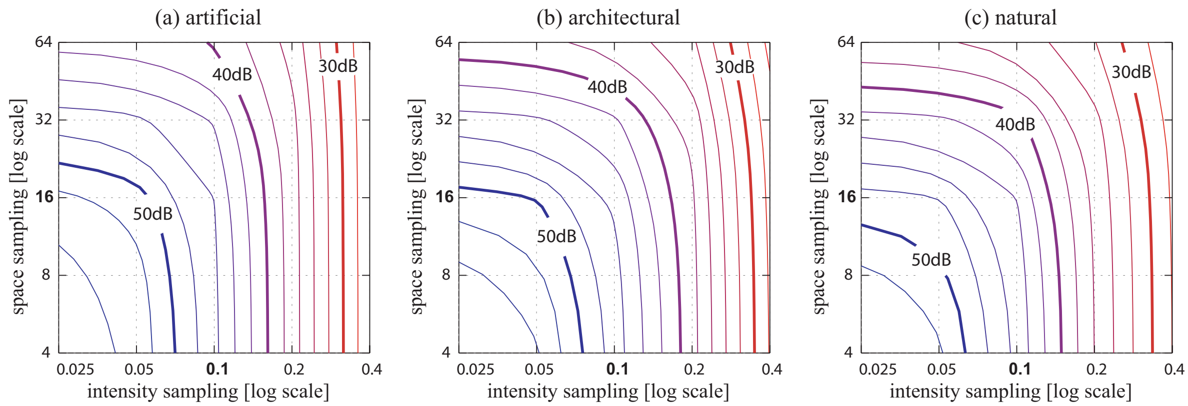


Fig. 4: Accuracy evaluation. All the images are filtered with ($\sigma_s = 16, \sigma_r = 0.1$). The PSNR in dB is evaluated at various sampling of $\mathcal{S}$ and $\mathcal{R}$ (greater is better). Our approximation scheme is more robust to space downsampling than range downsampling. It is also slightly more accurate on structured scenes (a,b) than on stochastic ones (c). This test uses the full-kernel implementation of the 3D convolution based on FFT.

Our tests showed that their computational cost exceeds the accuracy gain, *i.e.* a similar gain can be obtained in a shorter time using a finer sampling of the $\mathcal{S}\times\mathcal{R}$ space. The results presented in this paper use box downsampling and linear upsampling. We experimented with several sampling rates (s_s, s_r) for $\mathcal{S}\times\mathcal{R}$. The meaningful quantities to consider are the ratios $\left(\frac{s_s}{\sigma_s}, \frac{s_r}{\sigma_r}\right)$ that indicate the relative position of the frequency cutoff due to downsampling with respect to the bandwidth of the filter we apply. Small ratios correspond to limited approximations and high ratios to more aggressive downsamplings. A consistent approximation is a sampling rate proportional to the Gaussian bandwidth (*i.e.* $\frac{s_s}{\sigma_s} \approx \frac{s_r}{\sigma_r}$) to achieve similar accuracy on the whole $\mathcal{S}\times\mathcal{R}$ domain. The results plotted in Figure 4 show that this remark is globally valid in practice. A closer look at the plots reveals that $\mathcal{S}$ can be slightly more downsampled than $\mathcal{R}$. This is probably due to the nonlinearities and the anisotropy of the signal.

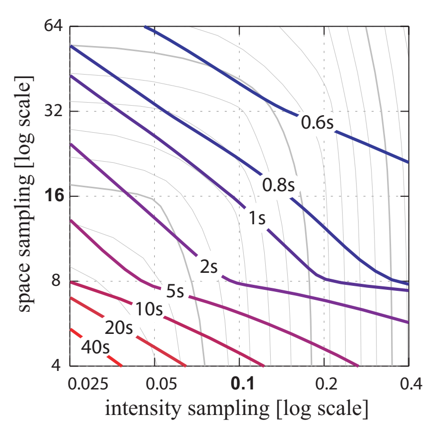


Fig. 5: Running times on the architectural picture with ($\sigma_s = 16, \sigma_r = 0.1$). The PSNR isolines are plotted in gray. Exact computation takes several tens of minutes (varying with the actual implementation). This test uses the full-kernel implementation of the 3D convolution based on FFT.

5.2 Running Times

Figure 5 shows the running times for the architectural picture with the same settings. In theory, the gain from space downsampling should be twice the one from range downsampling since $\mathcal{S}$ is two-dimensional and $\mathcal{R}$ one-dimensional. In practice, the nonlinearities and caching issues induce minor deviations. Combining this plot with the PSNR plot (in gray under the running times) allows for selecting the best sampling parameters for a given error tolerance or a given time budget. As a simple guideline, using sampling steps equal to σ_s and σ_r produces results without visual difference with the exact computation (see Figure 12). Our scheme achieves a dramatic speed-up since direct computation of Equation (1) lasts several tens of minutes (varying with the actual implementation). Our approximation requires one second.

Effect of the Kernel Size An important aspect of our approximation is visible on Figure 6-right. Our technique runs faster with larger kernels. Indeed, when σ increases, keeping the level of approximation constant (the ratio $\frac{s}{\sigma}$ is fixed) allows for a more important downsampling, that is, larger sampling steps s. The rationale of this behavior is that the more the image is smoothed, the more the high frequencies of $\mathcal{S}\times\mathcal{R}$ are attenuated, and the more we can afford to discard them without incurring visible differences.

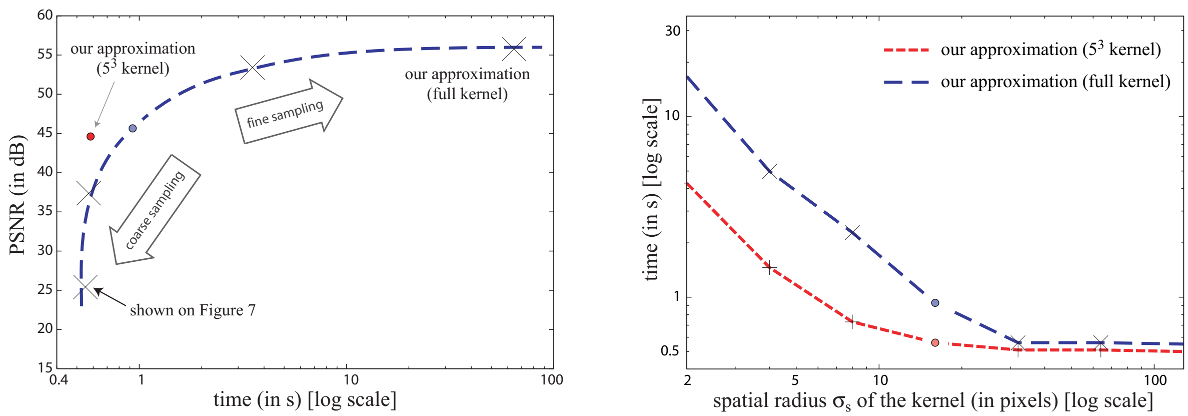


Fig. 6: Left: Accuracy-versus-time comparison. Our approximations are tested on the architectural picture (1600 × 1200) using $\sigma_s = 16$ and $\sigma_r = 0.1$ as parameters. The full-kernel can achieve variable degrees of accuracy and speed-up by varying the sampling rates of $\mathcal{S}\times\mathcal{R}$. We tested it with the following sampling rates of $\mathcal{S}\times\mathcal{R}$ (from left to right): (4;0.025) (8;0.05) (16;0.1) (32,0.2) (64,0.4). The result with the coarsest sampling ($s_s = 64, s_r = 0.4$) is shown on Figure 7. • Right: Time-versus-radius comparison. Our approximations are tested on the architectural image using $\sigma_r = 0.1$. The full- kernel technique uses a sampling rate equal to $(\sigma_s;\sigma_r)$. Remember that an exact computation lasts at least several tens of minutes (varying with the implementation). • The color dots in the left and right plots correspond to the same measurements.

Truncated Kernel versus Full Kernel Figure 6 also shows that, for a limited loss of accuracy, the direct convolution with a truncated kernel yields running times significantly shorter than the Fourier-domain convolution with a full kernel. We believe that the direct convolution with the 5^3 kernel is a better choice for most applications.

High and Low Resolution Filtering To evaluate the usability of our technique in a professional environment, we processed a 8 megapixel image and obtained a running time of 2.5s. This is in the same order of magnitude as the iterated-box filter and orders of magnitude faster than currently available professional software (see [Weiss, 2006] for details). We also experimented with an image at DVD resolution, *i.e.* $576 \times 320 \approx 0.2$ megapixel, and we obtained a running time of about 57ms near the 40ms required to achieve real-time performances, *i.e.* a 25Hz frame rate. This motivated us to run our algorithm on our most recent machine, an AMD Opteron 252 at 2.6MHz with 1MB of cache, in order to simulate a professional setup. With this machine, the running time is 38ms. This means that our algorithm achieves real-time video processing in software. Although this does not leave any time to apply other effects as demonstrated by Winnemöller *et al.* [2006] using the GPU, we believe that our technique paves the way toward interactive video applications that are purely software-based.

Bottlenecks Figures 6 and 7 show that when the sampling rate becomes too coarse, the accuracy suffers dramatically but the running time does not improve. This is because slicing becomes the bottleneck of the algorithm: The trilinear interpolation necessary at each output pixel becomes significantly more costly than the operations performed on the 3D data (see Table 2 on page 34). On the other extreme, when the sampling rate becomes very fine, the running time grows significantly but the plot shows little improvement because the errors become too small to be reliably measured after the quantization on 8 bits.

(a) input (b) exact bilateral filter (c) full-kernel approximation with extreme downsampling (d) difference

Fig. 7: We filtered the architectural picture (a) and applied an extreme downsampling to the $\mathcal{S}\times\mathcal{R}$ domain, four times the filter bandwidth, *i.e.* $(\sigma_s, \sigma_r) = (16, 0.1)$ and $(s_s, s_r) = (64, 0.4)$. The result (c) exhibits numerous visual artifacts: the whole image is overly smoothed and the smoothing strength varies spatially (observe the roof railing for instance). Compared to the exact computation (b), the achieved PSNR is 25dB. The difference image (with a 10× multiplier) shows that large errors cover most of the image (d). We do not recommend such extreme downsampling since the speed gain is limited while the accuracy loss is important as shown on Figure 6-left (this result corresponds to the lower-left measured point on the dashed curve).

5.3 Effect of the Downsampling Grid

Our method uses a coarse 3D grid to downsample the data. The position of this grid is arbitrary and can affect the produced results as pointed out by Weiss [2006]. To evaluate this effect, we filter 1000 times the same image and applied each time a different random offset to the grid. We average all the outputs to get the mean result. For each grid position, we measured at each pixel the difference between the intensity obtained for this offset and the mean value over all offsets. Figure 8-left shows the distribution of these variations caused by the grid offsets. We can see that there are a few large variations (under 40dB) but that most of them are small (≈ 50dB and more). These variations have to be compared with the accuracy of the approximation: Figure 8-right shows that they are significantly smaller than the error incurred by the approximation. In other words, the potential variations caused by the position of the downsampling grid are mostly negligible in regards of the error stemming from the approximation itself.

Although these variations have never been a problem in our applications (cf. Section 8), they might be an issue in other contexts. To better appreciate the potential defects that may appear, we built a worst-case scenario with the statistics of the 1000 runs. We computed the minimum and maximum results:

$$I^{\mathrm{b}}_{\min}(\mathbf{p}) \;=\; \min_{\text{all runs}} I^{\mathrm{b}}_{\downarrow\uparrow}(\mathbf{p}) \qquad \text{and} \qquad I^{\mathrm{b}}_{\max}(\mathbf{p}) \;=\; \max_{\text{all runs}} I^{\mathrm{b}}_{\downarrow\uparrow}(\mathbf{p}) \tag{14}$$

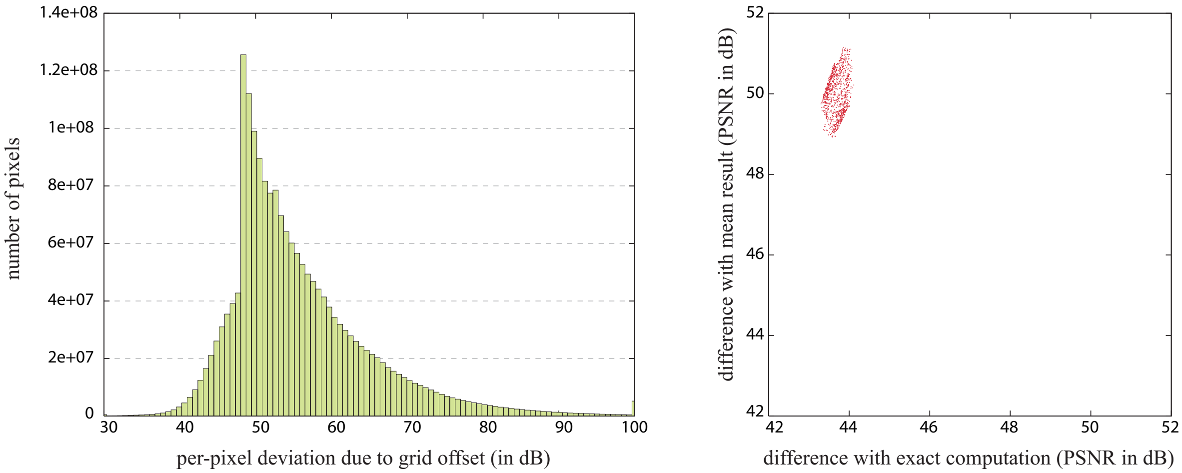


Fig. 8: Left: Per-pixel distance to the mean. We filter 1000 times the architectural picture using the 5^3 kernel with a different offset of the downsampling grid. For each pixel $\mathbf{p}$, we computed the mean value $M_{\mathbf{p}}$. Then for each run, we measured the distance between the result value $I^{\mathrm{b}}_{\downarrow\uparrow}(\mathbf{p})$ and the mean $M(\mathbf{p})$. The histogram shows the distribution of these distances across the 1000 runs. • Right: Comparison between the distance to the mean and the distance to the exact result. For each run, we measured the PSNR between the approximated result $I^{\mathrm{b}}_{\downarrow\uparrow}$ and the mean result M. We also measured the PSNR between $I^{\mathrm{b}}_{\downarrow\uparrow}$ and the exact result I^b. The plot shows these two measures for each run. The approximated results are one order of magnitude closer to the mean M. This demonstrates that the position of the downsampling grid a limited influence on the produced results. • These tests use the 5^3-kernel implementation.

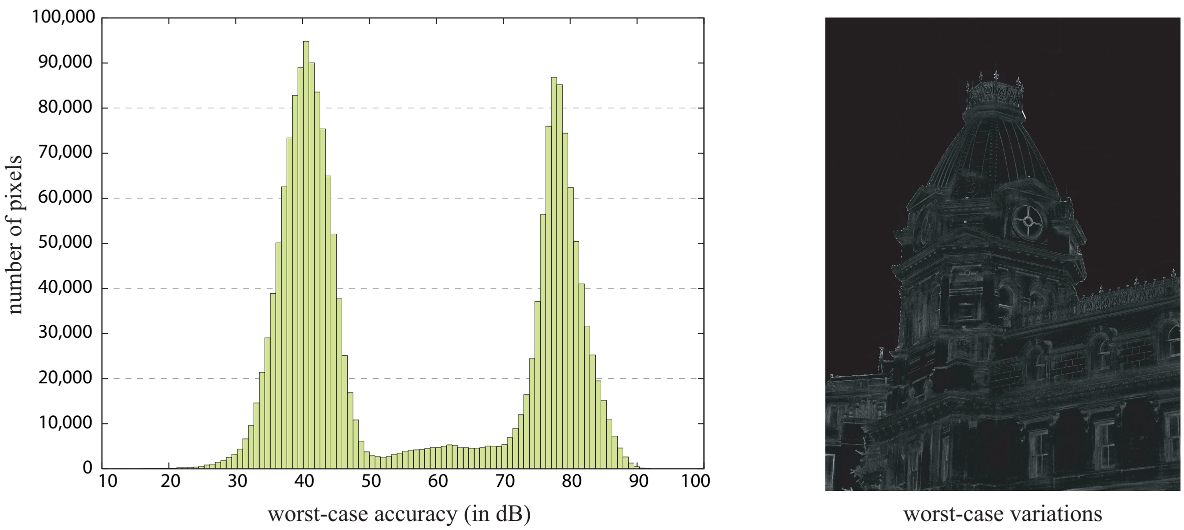


Fig. 9: Left: Influence of the grid position in a worst-case scenario. The histogram shows the distances between the maximum and minimum of all the runs (Equation 14). We used the architectural picture. The two modes of the distribution correspond to the building area that contains many contrasted features likely to be affected by the grid position, and to the smooth sky region with almost no such features. • Right: Variation map. The thin features are the most affected by the downsampling grid position whereas the smooth areas are mostly unaltered. The variation amplitudes are mapped to gray levels after a 10× multiplication. Depending on your printer, some details may not appear properly in the figure. We recommend the electronic version for a better resolution.

These two images correspond to the worst-case hypotheses where all the negative (resp. positive) deviations happened at the same time which is unlikely. Figure 9-left plots the histogram of the differences between $I^{\mathrm{b}}_{\max}$ and $I^{\mathrm{b}}_{\min}$: Even in this worst-case scenario, most variations remain above 40dB. Globally, the maximum and minimum results still achieve satisfying accuracies of 42dB and 43dB compared to the exact computation. Figure 9-right reveals that the errors are larger on discontinuities (edges, corners) and textured regions. As a consequence, the variation distribution (Figure 9-left) is bimodal with a high-accuracy mode corresponding to the smooth sky and a lower-accuracy mode stemming from the building. Nonetheless, if higher accuracy is required, one can refine the downsampling grid at the cost of longer running times as shown earlier (Figure 5).

6 Comparisons with Other Acceleration Techniques

We now compare our method to other fast approximations of the bilateral filter. To better understand the common points and differences with our approach, we look into the details of these methods. We selected the methods that explicitly aim for bilateral filtering [Durand and Dorsey, 2002; Pham and van Vliet, 2005; Weiss, 2006]. We did not test the Gauss-Seidel scheme proposed by Elad [2002] since it requires a large number of iterations whereas practical applications use non-iterative bilateral filtering. As we shall see, our technique performs especially well on large kernels and has the advantage to be general and simple to implement.

6.1 Comparison with the Piecewise-Linear Approximation

Durand and Dorsey [2002] describe a piecewise-linear approximation of the bilateral filter that is closely related to our approach. Our framework provides better understanding their method: As we will see, the main difference with our technique lies in the downsampling strategy.

Using evenly spaced intensity values $\zeta_1..\zeta_n$ that cover $\mathcal{R}$, the piecewise-linear scheme can be summarized as (for convenience, we also name G_{σ_s} the 2D Gaussian kernel):

$$\iota_\downarrow \;=\; \mathrm{downsample}(I) \qquad \text{[image downsampling]} \tag{15a}$$

$$\forall k \in \{1..n\} \qquad \omega_{\downarrow k}(\mathbf{p}) \;=\; G_{\sigma_\mathrm{r}}(|\iota_\downarrow(\mathbf{p}) - \zeta_k|) \qquad \text{[range weight evaluation]} \tag{15b}$$

$$\forall k \in \{1..n\} \qquad \omega\iota_{\downarrow k}(\mathbf{p}) \;=\; \omega_{\downarrow k}(\mathbf{p})\; \iota_\downarrow(\mathbf{p}) \qquad \text{[intensity multiplication]} \tag{15c}$$

$$\forall k \in \{1..n\} \qquad (\omega\iota^\mathrm{b}_{\downarrow k}, \omega^\mathrm{b}_{\downarrow k}) \;=\; G_{\sigma_\mathrm{s}} \otimes_{\mathcal{S}} (\omega\iota_{\downarrow k}, \omega_{\downarrow k}) \qquad \text{[spatial convolution on } \mathcal{S}\text{]} \tag{15d}$$

$$\forall k \in \{1..n\} \qquad \iota^\mathrm{b}_{\downarrow k} \;=\; \omega\iota^\mathrm{b}_{\downarrow k} \;/\; \omega^\mathrm{b}_{\downarrow k} \qquad \text{[normalization]} \tag{15e}$$

$$\forall k \in \{1..n\} \qquad \iota^\mathrm{b}_{\downarrow\uparrow k} \;=\; \mathrm{upsample}(\iota^\mathrm{b}_{\downarrow k}) \qquad \text{[layer upsampling]} \tag{15f}$$

$$I^\mathrm{b}_\mathrm{pl}(\mathbf{p}) \;=\; \mathrm{interpolation}(\iota^\mathrm{b}_{\downarrow\uparrow k})(\mathbf{p}) \qquad \text{[linear layer interpolation]} \tag{15g}$$

Without downsampling (*i.e.* $\{\zeta_k\} = \mathcal{R}$ and Steps 15a,f ignored), the piecewise-linear scheme is equivalent to ours because Steps 15b,c,d correspond to a convolution on $\mathcal{S}\times\mathcal{R}$. Indeed, $\iota_\downarrow(\mathbf{p}) = I_\mathbf{p}$, Steps 15b and 15c can be rewritten in a vectorial form, and the value of the Gaussian can be expressed as a convolution on $\mathcal{R}$ with a Kronecker symbol:

$$\begin{matrix}\text{[Step 15c]} \\ \text{[Step 15b]}\end{matrix} \quad \begin{pmatrix} \omega\iota_{\downarrow k}(\mathbf{p}) \\ \omega_{\downarrow k}(\mathbf{p}) \end{pmatrix} = \begin{pmatrix} I_\mathbf{p} \\ 1 \end{pmatrix} G_{\sigma_\mathrm{r}}(|I_\mathbf{p} - \zeta_k|) \;=\; \begin{pmatrix} I_\mathbf{p} \\ 1 \end{pmatrix} \Big[\delta_{I_\mathbf{p}} \otimes_{\mathcal{R}} G_{\sigma_\mathrm{r}}\Big](\zeta_k) \tag{16a}$$

$$\text{with} \quad \delta_{I_\mathbf{p}}(\zeta \in \mathcal{R}) \;=\; \delta(I_\mathbf{p} - \zeta) \tag{16b}$$

With Step 15d, the convolution on $\mathcal{S}$, these three steps perform a 3D convolution using a separation between $\mathcal{R}$ and $\mathcal{S}$.

The main differences comes from the downsampling approach, where Durand and Dorsey downsample in 2D while we downsample in 3D. They also interleave linear and nonlinear operations differently from us: Their division is done after the convolution 15d but before the upsampling 15f. There is no simple theoretical ground to estimate the error. More importantly, the piecewise-linear strategy is such that the intensity ι and the weight ω are functions defined on $\mathcal{S}$ only. A given spatial pixel in the downsampled image has only one intensity and one weight. After downsampling, both sides of a discontinuity may be represented by the same values of ι and ω. This is a poor representation of discontinuities since they inherently involve several values. In comparison, we define functions on $\mathcal{S}\times\mathcal{R}$. For a given image point in $\mathcal{S}$, we can handle several values on the $\mathcal{R}$ domain. The advantage of working in $\mathcal{S}\times\mathcal{R}$ is that this characteristic is not altered by downsampling (cf. Figure 10). It is the major reason why our scheme is more accurate than the piecewise-linear technique, especially around discontinuities.

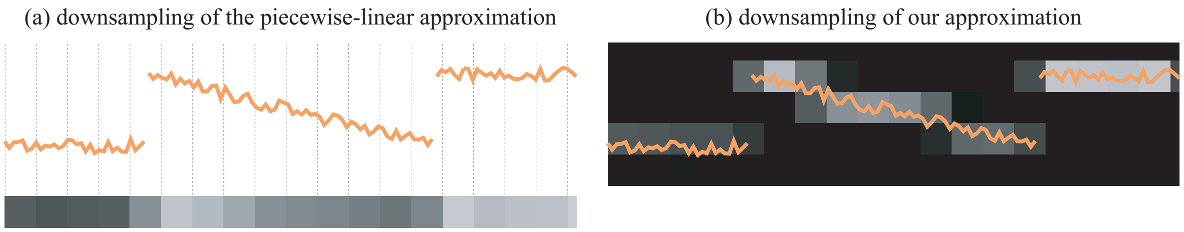


Fig. 10: (a) The piecewise-linear approximation is such that only a single value can representation at each position. After downsampling discontinuities are represented by only one intensity value which poorly approximates the discontinuity. (b) With our scheme, the discontinuities are represented by two distinct values in the downsampled $\mathcal{S}\times\mathcal{R}$ domain, even after downsampling. The original function (in red) is the same as in Figure 1. The corresponding downsampled representation of the intensity is shown under.

Numerical and Visual Comparisons We have implemented the piecewise-linear technique with the same code base as our technique. Figures 11 show the precision and running time achieved by both approaches. Both techniques exhibit similar profiles but our approach achieves significantly higher accuracies for the same running times (except for extreme downsampling, but these results are not satisfying as shown on Figure 7 on page 16). Figure 12 confirms visually this measure: Our method approximates better the exact result. As a consequence, our technique can advantageously replace the piecewise-linear approximation since it is both simpler and more precise.

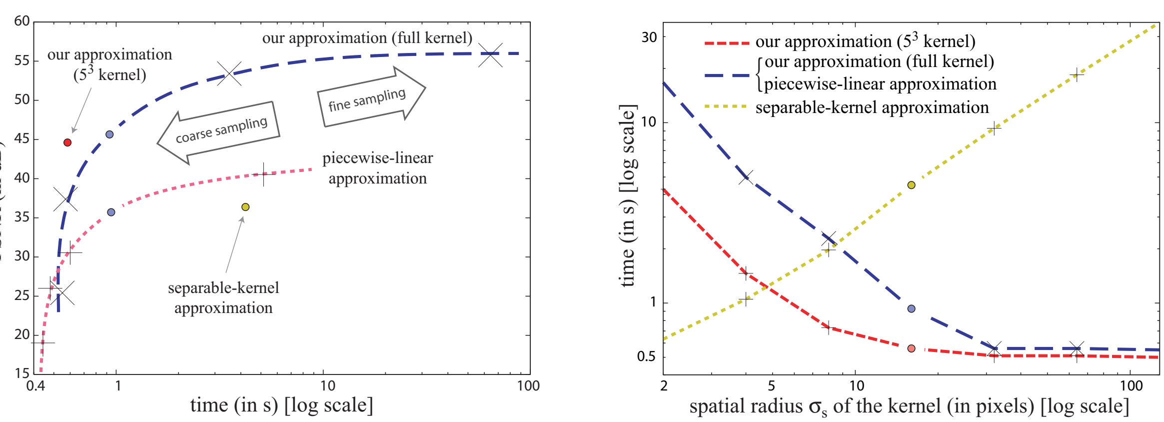


Fig. 11: Left: Accuracy-versus-time comparison. The methods are tested on the architectural picture (1600 × 1200) using $\sigma_s = 16$ and $\sigma_r = 0.1$ as parameters. The piecewise-linear approximation and our method using an untruncated kernel can achieve variable degrees of accuracy and speed-up by varying the sampling rates of $\mathcal{S}\times\mathcal{R}$. We tested them with the same sampling rates of $\mathcal{S}\times\mathcal{R}$ (from left to right): (4;0.025) (8;0.05) (16;0.1) (32,0.2) (64,0.4). • Right: Time-versus-radius comparison. The methods are tested on the architectural image using $\sigma_r = 0.1$. Our method using the full kernel uses a sampling rate equal to $(\sigma_s; \sigma_r)$. Its curve and the piecewise-linear curve are identical because we allow the same computation budget to both methods. Remember that an exact computation lasts at least several tens of minutes (varying with the implementation). • The color spots in the left and right plots correspond to the same measurements.

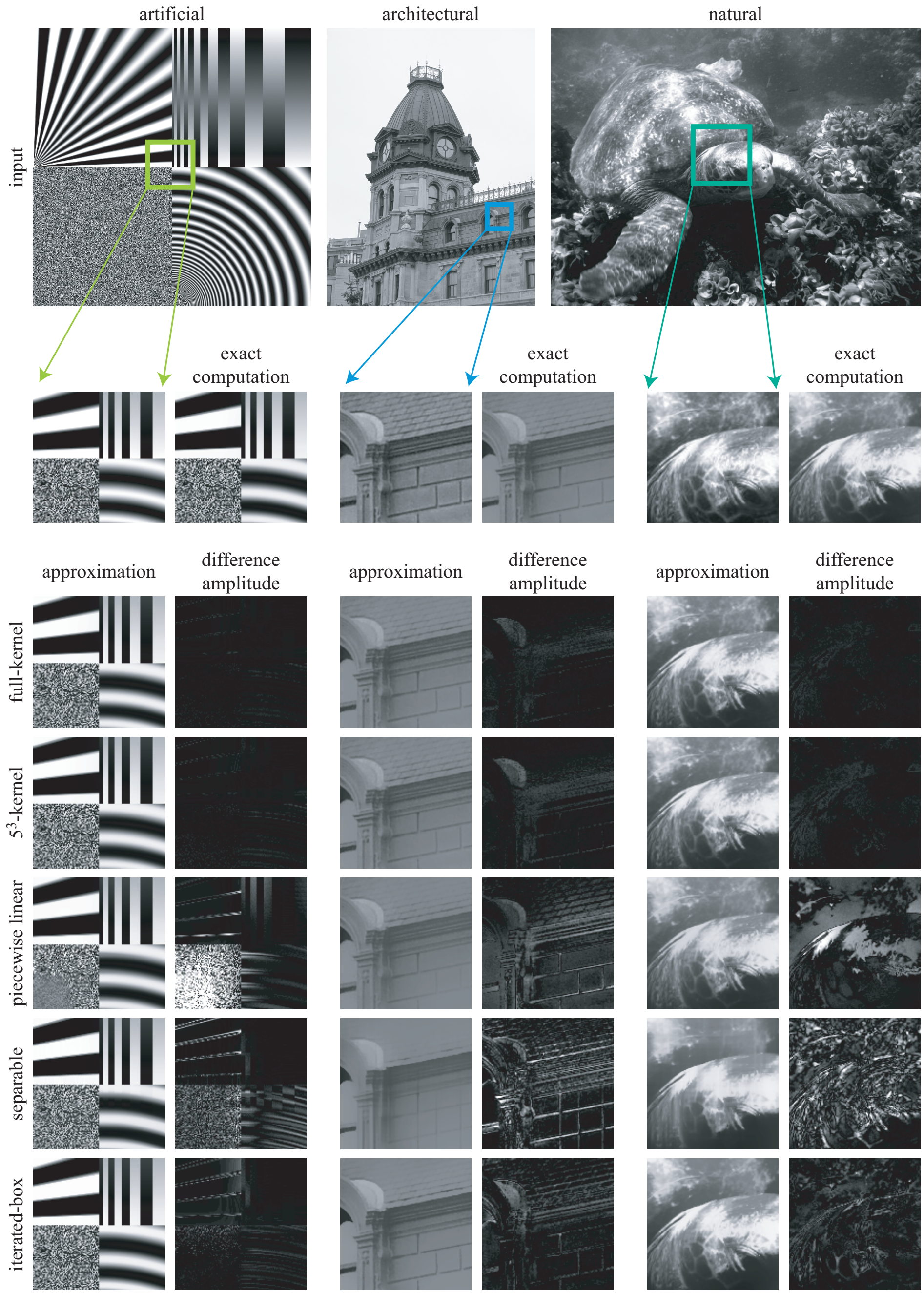


Fig. 12: We have tested our approximated scheme on three images (first row): an artificial image (512×512) with different types of edges and a white noise region, an architectural picture (1600×1200) with strong and oriented features, and a natural photograph (800×600) with more stochastic textures. For clarity, we present representative close-ups. Full resolution images are as supplemental material. Our approximations produces results visually similar to the exact computation. Although we applied a $10\times$ multiplier to the difference amplitudes, the image differences do not show significant errors. In comparison, the piecewise-linear approximation introduces large visual discrepancies: small yet contrasted features are washed out. Furthermore this defect depends the neighborhood. For instance, on the artificial image, the noise region is less smoothed near the borders because of the adjacent white and black bands. Similar artifacts are visible on the brick shadows of the architectural image. The separable kernel introduces axis-aligned streaks which are visible on the bricks of the architectural picture. In comparison, the iterated-box kernel does not incur artifacts but the filter does not actually approximate a Gaussian kernel (Section 6.3). This can be seen on the large differences. All the filters are computed for $\sigma_s = 16$ and $\sigma_r = 0.1$. Our filter uses a sampling rate of (16,0.1). The piecewise-linear filter is allocated the same time budget as our full-kernel approximation: its sampling rate is chosen in order to achieve the same (or slightly superior) running time. The iterated-box method is set with 2 additional iterations as described by Weiss [2006] and $\sigma_r' = \frac{0.1}{\sqrt{3}} \approx 0.0577$ (Section 6.3). Depending on your printer, some details may not appear properly in the figure. We recommend the electronic version for a better resolution.

6.2 Comparison with the Separable-Kernel Approximation

Pham and van Vliet [2005] approximate the 2D bilateral filter with a separable kernel by first filtering the image rows and then the columns. This dramatically shortens the processing times since the number of pixels to process is proportional to the kernel radius instead of its area in the 2D case. Pham [2006] also describes an extension to his work where the separation axes are oriented to follow the image gradient. However, we knew about this extension too late to include it in our tests. Our experiments focus on the axis-aligned version that has been shown to be extremely efficient by Winnemöller *et al.* [2006]. We have implemented this technique in C++ using the same code base as for our method. Our C++ code on 2.8GHz Intel Xeon achieves a 5× speed-up compared to the experiment in Matlab on an 1.47GHz AMD reported in the original article.

As in the original paper, our tests confirm that the separable-kernel strategy is useful for small kernels. Figure 11-right shows that this advantage applies to radii that do not exceed a few pixels ($\sigma_s \approx 5$ pixels with our implementation). Winnemöller *et al.* [2006] have shown that it is a suitable choice to simplify the content of videos by applying several iterations of the bilateral filter using a small kernel. This approximation is less suitable for larger kernels as shown by the running times on Figure 11-right and because artifacts due to the approximation become visible (Figures 12 and 15f). As the kernel becomes bigger, the pixel neighborhoods contain more and more complex features, *e.g.* several edges and corners. These features are poorly handled by the separable kernel because it considers the rows and columns separately. This results in axis-aligned artifacts which may be undesirable in a number of cases. In comparison, our approach handles the complete neighborhoods. The incurred error is significantly lower and follows the image structure, yielding mostly imperceptible differences (Figures 12 and 15g).

6.3 Comparison with the Iterated-Box Method

Weiss [2006] describes an algorithm that computes efficiently the exact bilateral filter using a square box function to define the spatial influence. It is based on the maintenance of local intensity histograms, and Section 9.5 relates this concept with our higher-dimensional interpretation. In this section, we discuss performances and accuracy. The complexity of Weiss's algorithm evolves as the logarithm of the kernel radius, thereby achieving running times in the order of a few seconds even for large kernels (100 pixels and above for an 8 megapixel image).

However, the box function is known to introduce visual artifacts because its Fourier transform is not band-limited and it introduces Mach bands. To address this point, Weiss iterates the filter three times while keeping the range weights constant, *i.e.* the range values are always computed according to the input image. Weiss motivates this approach by the fact that, on regions with smooth intensity, the range weights have little influence and the iteration corresponds to spatially convolving a box function β^0 with itself, *i.e.* $\beta^0 \otimes_{\mathcal{S}} \beta^0 \otimes_{\mathcal{S}} \beta^0$. This results in a smooth quadratic B-spline β^2 that approximates a Gaussian function. This interpretation does not hold on non-uniform areas because the spatial component interacts with the range function. Van de Weijer and van den Boomgaard [2001] have shown that this iteration scheme that keeps the weights constant eventually leads to the nearest local maximum of the local histogram. A downside of this sophisticated scheme is that the algorithm is more difficult to adapt. For instance, extension to color images seems nontrivial.

6.3.1 Running Times

A performance comparison with our method is difficult because Weiss' implementation relies on the vector instructions of the CPU. Implementing this technique as efficiently as the original article could not be done in a reasonable amount of time. And since the demonstration plug-in[1] does not run on our platform, we cannot make side-by-side timing comparisons. Nonetheless, we performed some tests on a 8 megapixel image as Weiss uses in his article. For a small kernel ($\sigma_s = 2$ pixels), our software runs in 18s whereas the iterated-box technique runs in ≈ 2s. Although the CPU and the implementation are not the same, the difference is significant enough to conclude that the iterated-box method is faster on small kernels. We also experimented with bigger kernels (recall that our running times decrease): our running times are under 3s for radii over 10 pixels and stabilize at 2.5s for larger radii ($\sigma_s \geq 30$ pixels) because of the fixed time required by downsampling, upsampling, and nonlinearities. These performances are in the same order as the iterated-box technique. In such situations, the running times depend more on the implementation than on the actual algorithm. Our code can be clearly improved to exploit the vector and multi-core capacities of modern architectures. Weiss also describes avenues to speed up his implementation. We believe that both techniques are approximately equivalent in terms of performance for intermediate and large kernels. The major difference is that our running times decrease with big kernels whereas the ones of the iterated-box slowly increase.

6.3.2 Numerical and Visual Evaluation

Comparing the accuracy is not straightforward either since the iterated version of this filter approximates a Gaussian profile only on constant regions while the kernel has not been studied in other areas. To better understand this point, we conducted a series of tests to characterize the produced results. Weiss [2006] points out that convolving a box function n times with itself produces a B-spline β^n, and as n increases, β^n approximates a Gaussian profile. The iterated-box method is however more complex since at each iteration, the spatial box function β^0 is multiplied by the Gaussian range weight G_{σ_r} and then normalized. This motivated us to investigate the link between the iterated-box scheme and the non-iterative bilateral filter with a B-spline as spatial weight.

Test Setup We have coded a brute-force implementation of the iterated-box filter using classical "for loops". Although our piece of software is slow, it uses exactly the same space and range functions as the iterated-box method. Thus we can perform visual and numerical comparisons safely as long as we do not consider the running times.

We computed the iterated-box filter with increasing numbers of iterations and compared the results with the exact computation of the bilateral filter using a Gaussian as spatial weight as well as with the exact computation of the bilateral filter using a B-spline β^n with n matching the number of iterations. In both cases, the range weight remains a Gaussian as described by Weiss. We set the radius r_{box} of the box function so as to achieve the same standard deviation σ_s for the all the tests. For a square box function (*i.e.* no iteration), a classical variance computation gives $\sigma_s^2 = \frac{2}{3}\, r_{\text{box}}^2$. For n iterations, we use $r_{\text{box}} = \sigma_s \sqrt{\frac{3}{2(n+1)}}$.

Setting the range influence is equally important. A first solution is to use the same value σ_r independently of the number of iterations. As a consequence, the smoothing strength varies with the number of iterations (Figure 14). Our interpretation of the bilateral filter shows that the range domain also undergoes a convolution. This suggests to set $\sigma_r' = \sigma_r \sqrt{\frac{1}{n+1}}$ when the filter is applied a total of $n+1$ times. It would result in a range Gaussian of size σ_r if the filter was purely a convolution. Let's use our framework to examine the effects of the iterations with fixed weights. Because the weights are kept constant, the same 3D kernel and normalization factors are applied at each iteration. However, the normalization factors vary spatially and slicing occurs after each pass which corresponds to setting

[1] `http://www.shellandslate.com/fastmedian.html`

to 0 all the $\mathcal{S} \times \mathcal{R}$ points $(\mathbf{x}, \zeta)$ such that $\zeta \neq I_{\mathbf{x}}$. Hence, the process is not purely a succession of convolutions. Nevertheless, we shall see that adapting σ_r to the total number of iterations has some visual advantages over the fixed value.

Note that Weiss experimented with adapting both the space and range sigmas to the total number of iterations although his article did not focus on approximating any kernel and showed only results from three iterations.

Numerical Evaluation Figure 13 shows that the iterated-box scheme does not converge either toward Gaussian bilateral filtering or toward spline bilateral filtering, at least with the parameters that we used. It might not be a problem depending on the application, since it does not affect the visual quality of the result. For instance, this scheme is well-suited for applications such as denoising or smoothing [Weiss, 2006]. A formal characterization of the achieved kernel would nevertheless be a valuable contribution since it would enable a better comparison with the other existing methods and allows for motivated choices.

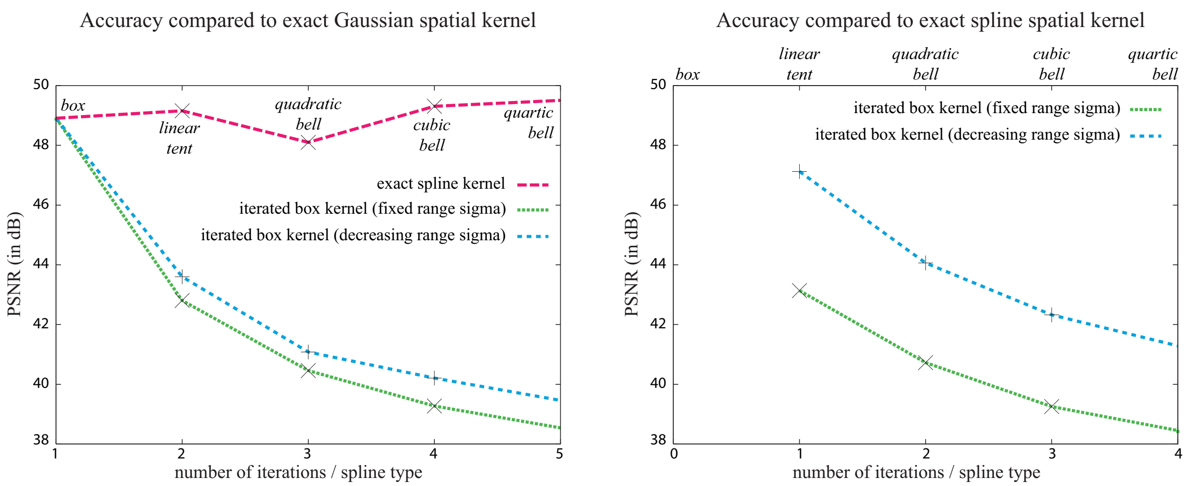


Fig. 13: Comparisons with exact bilateral filtering using a Gaussian spatial kernel (left) and using a B-spline spatial kernel (right). As the number of iterations increases, the iterated-box filter becomes more different from the non-iterated bilateral filters, for both spatial profiles (Gaussian and spline) and both range settings. On the left, the spline profile converges toward a Gaussian kernel with higher spline order although numerical precision issues limit the convergence speed. The spline shape is indicated on the curve. On the right, the iterated-box kernel does not converge towards a spline kernel. The shape of the spline used for the comparison is indicated on the top axis.

Visual Results Figure 14 illustrates the typical defect appearing with a square box function as spatial weight and a single iteration: A faint band appears above the dark cornice. It is worth mentioning that we tested several images and this defect was less common than we expected. As soon as the contrast is large enough or when the picture is textured, no defect is visible. As an example, no banding is visible around the dome contour and near the windows. However, when this defect appears, it impacts the overall image quality. The solution is to apply a smoother kernel, either by using a higher-order spline, or by iterating the filter. Our experiment shows that using a spline of order 1 (a linear tent) solves the problem while a single additional iteration only attenuates the band without completely removing it. As suggested by Weiss [2006], three iterations yield results without visible artifacts. Furthermore, this figure shows that the results are progressively washed out if the range setting is not adapted according to the number of iterations. Adapting σ_r as the square root of the number of iterations prevents this effect and produces more stable results. The downside of iterating can be seen on the wall corner at the bottom right end of the close-up: Stacking up the nonlinearities

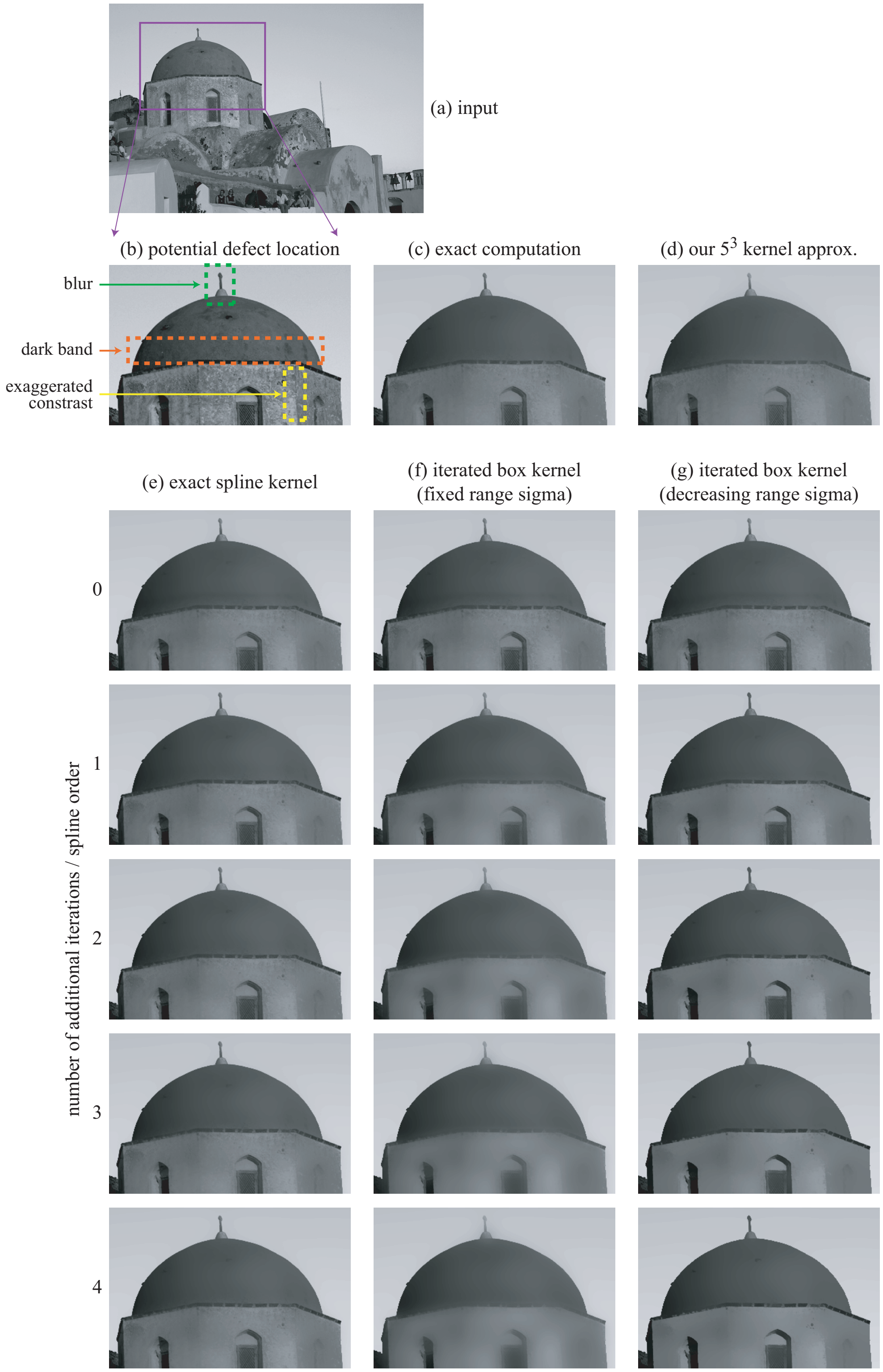


Fig. 14: Results from the spline and iterated-box bilateral filters. Several defects may appear depending on the chosen scenario; their location is shown on the close-up (b). Using a square box function without iteration produces a faint band above the cornice (first row (e,f,g)). Iterating the filter makes this band disappear. If the range Gaussian is not adapted according to the number of iterations, the results become blurry (see the cross and the cornice in (f)). Decreasing σ_r as the square root of the number of iterations prevents blurriness and produces more stable results (g). But iterating the filter can introduce exaggerated contrast variations (see the edge of the wall). The contrast of the close-ups has been increased for clarity purpose. Depending on your printer, some details may not appear properly in the figure. We recommend the electronic version for a better resolution. Original images are available in supplemental material.

results in an exaggerated contrast variation because the top part of the corner has been smoothed at each iteration while the bottom part has been preserved. The non-iterated versions of the filter (either with a spline or a Gaussian kernel) do not present such contrast variations. This effect may be a concern depending on the application.

7 Extensions

An advantage of our approach is that it can be straightforwardly adapted to handle color images and cross bilateral filtering. In the following sections, we describe the details of these extensions.

7.1 Color Images

For color images, the range domain $\mathcal{R}$ is typically a three-dimensional color space such as RGB or CIE-Lab. We name $\mathbf{C} = (C_1, C_2, C_3)$ the vector in $\mathcal{R}$ describing the color of a pixel. The bilateral filter is then defined as:

$$\mathbf{C}^{\mathrm{b}}_{\mathbf{p}} = \frac{1}{W^{\mathrm{b}}_{\mathbf{p}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(\|\mathbf{C}_{\mathbf{p}}-\mathbf{C}_{\mathbf{q}}\|)\, \mathbf{C}_{\mathbf{q}} \tag{17a}$$

$$\text{with} \quad W^{\mathrm{b}}_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(\|\mathbf{C}_{\mathbf{p}}-\mathbf{C}_{\mathbf{q}}\|) \tag{17b}$$

The definition is similar to Equation 1, except that we handle color vectors $\mathbf{C}$ instead of scalar intensities I. We derive formulae that isolate the nonlinearities similarly to Equation 11 by propagating this change through all the equations. We give the main steps in the following paragraphs and the details are identical to the gray-level case.

Akin to the homogeneous intensity defined in Section 3.1, we name *homogeneous color* the 4D homogeneous vector $(W\mathbf{C}, W)$. This let us write Equation 17 in a vector form:

$$\begin{pmatrix} W^{\mathrm{b}}_{\mathbf{p}}\, \mathbf{C}^{\mathrm{b}}_{\mathbf{p}} \\ W^{\mathrm{b}}_{\mathbf{p}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(\|\mathbf{C}_{\mathbf{p}}-\mathbf{C}_{\mathbf{q}}\|) \begin{pmatrix} W_{\mathbf{q}}\, \mathbf{C}_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix} \tag{18}$$

Then, we define the functions $\mathbf{c}$ and w on the joint domain $\mathcal{S}\times\mathcal{R}$:

$$\mathbf{c}: \quad (\mathbf{x}\in\mathcal{S}, \boldsymbol{\zeta}\in\mathcal{R}) \mapsto \mathbf{C}_{\mathbf{x}} \tag{19a}$$

$$w: \quad (\mathbf{x}\in\mathcal{S}, \boldsymbol{\zeta}\in\mathcal{R}) \mapsto \delta(\|\boldsymbol{\zeta}-\mathbf{C}_{\mathbf{x}}\|)\, W_{\mathbf{x}} \tag{19b}$$

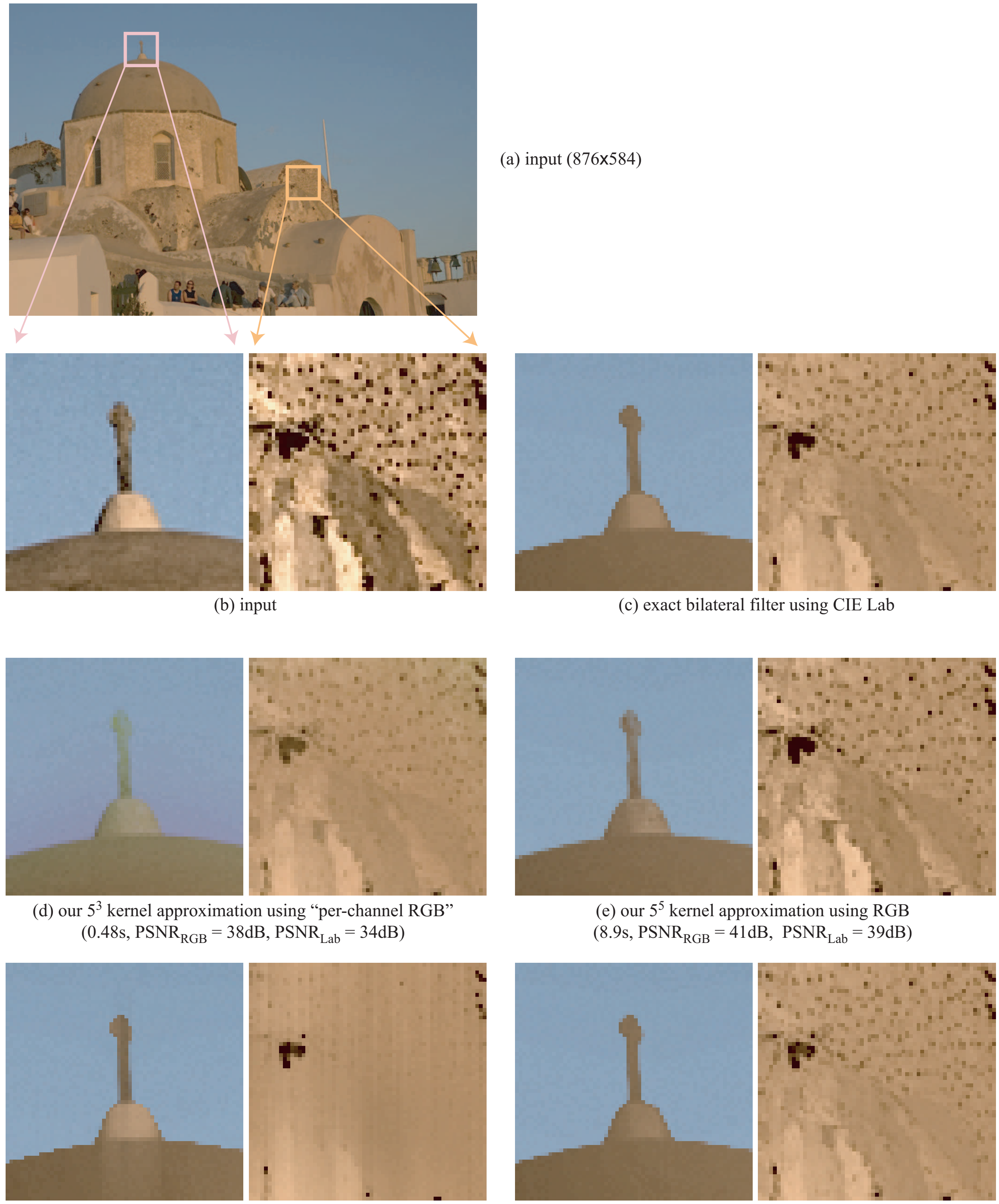
We obtain formulae similar to Equation 11:

$$\text{linear:} \quad (w^{\mathrm{b}}\, \mathbf{c}^{\mathrm{b}}, w^{\mathrm{b}}) = g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}} \otimes (w\mathbf{c}, w) \tag{20a}$$

$$\text{nonlinear:} \quad \mathbf{C}^{\mathrm{b}}_{\mathbf{p}} = \frac{w^{\mathrm{b}}(\mathbf{p}, \mathbf{C}_{\mathbf{p}})\, \mathbf{c}^{\mathrm{b}}(\mathbf{p}, \mathbf{C}_{\mathbf{p}})}{w^{\mathrm{b}}(\mathbf{p}, \mathbf{C}_{\mathbf{p}})} \tag{20b}$$

7.1.1 Discussion

Dimensionality For color images, the $\mathcal{S}\times\mathcal{R}$ domain is 5D: 2 dimensions for the spatial position and 3 dimensions to describe the color. Each point of $\mathcal{S}\times\mathcal{R}$ is a 4D vector since $\mathbf{C}$ is 3D, thus the homogeneous color $(W\mathbf{C}, W)$ is four-dimensional. This dimensionality increase becomes a problem when the $\mathcal{S}\times\mathcal{R}$



(a) input (876x584)

(b) input

(c) exact bilateral filter using CIE Lab

(d) our 5^3 kernel approximation using “per-channel RGB” (0.48s, $\text{PSNR}_{\text{RGB}} = 38\text{dB}$, $\text{PSNR}_{\text{Lab}} = 34\text{dB}$)

(e) our 5^5 kernel approximation using RGB (8.9s, $\text{PSNR}_{\text{RGB}} = 41\text{dB}$, $\text{PSNR}_{\text{Lab}} = 39\text{dB}$)

(f) separable-kernel approximation using CIE Lab (5.8s, $\text{PSNR}_{\text{RGB}} = 42\text{dB}$, $\text{PSNR}_{\text{Lab}} = 42\text{dB}$)

(g) our 5^5 kernel approximation using CIE Lab (10.9s, $\text{PSNR}_{\text{RGB}} = 46\text{dB}$, $\text{PSNR}_{\text{Lab}} = 46\text{dB}$)

Fig. 15: Comparison on a color image. We tested various strategy to filter a color image (a,b). Processing the red, green, and blue channels independently results in color bleeding that makes the cross disappear in the sky (d). Dealing with the RGB vector as described in Equation 20 improves this aspect but some bleeding still occurs (e). In contrast, working in the CIE-Lab space achieves satisfying results (c,g). Comparing our method (g) to the separable-kernel technique (f) shows our technique is slower but produces a result closer to the exact computation (c). Especially, the separable kernel incurs axis-aligned streaks (f) that may undesirable in a number of applications. These remarks are confirmed by the numerical precision evaluated with the PSNR computed the RGB and CIE-Lab color spaces. The contrast of the close-ups has been increased for clarity purpose. Depending on your printer, some details may not appear properly in the figure. We recommend the electronic version for a better resolution. Original images are available in supplemental material.

domain is finely sampled because of the memory required to store the data. Nonetheless, in many cases, the computation is still tractable. The validation section details this aspect.

Pre-multiplied Alpha Transparency is classically represented by an additional image channel named *alpha* [Porter and Duff, 1984]. Pixel values are then 4D vectors $(\mathbf{C}, \alpha)$ and it is well-known that most filtering operations such as blurring must be done on the *pre-multiplied alphas* $(\alpha\mathbf{C}, \alpha)$ in order to obtain correct color and transparency values. Intuitively, transparent pixels contribute less to the output. Although this representation "looks" similar to the homogeneous colors, there are a few important differences. An α value has a physical meaning: the transparency of a pixel. As a consequence, it lies in the $[0; 1]$ interval. In contrast, a W value carries statistical information since it corresponds to a pixel weight. It is non-negative and has no upper bound. A major difference between both representations is that W values can be scaled uniformly across the image without changing their meaning. This is a known property of homogeneous quantities: they are defined up to a scale factor. On the other side, a global scaling on α values alters the result since objects become more or less transparent.

7.1.2 Validation

We tested several alternative strategies on a color image (Figure 15). First, we filtered an RGB image as three independent channels and compared the output to the result of filtering colors as vectors. We found that vector filtering yields better results but color inconsistencies remained. Thus, we experimented with the CIE-Lab color space which is known to be perceptually meaningful.

Per-channel Filtering versus Vector Filtering Processing the RGB channels independently induces inconsistencies when an edge has different amplitudes depending on the channel. In that case, an edge may be smoothed in a channel while it is preserved in another, inducing incoherent results between channels. Figure 15d shows our truncated-kernel approximation on an example where the blue sky "leaks" over the brown cross. Considering the RGB channels altogether using Equation 20 significantly reduces this bleeding defect (Figure 15e). The downside is the longer computation times required to process the 5D space (cf. Section 7.1.1 and Figure 16). It precluded our algorithm from handling small kernels because of the fine sampling required. In that case, the memory usage was over 1GB for an image of 0.5 megapixel with samplings $(s_\mathrm{s}, s_\mathrm{r}) = (8, 0.1)$. We did not measure the running time because it was perturbed by disk swapping. This limitation makes our technique not suitable for small kernels on color images. For these cases, one should prefer per-channel filtering and since the kernel is small, the iterated-box method [Weiss, 2006] seems an appropriate choice. However, even the RGB-vector filtering result exhibits some color leaking (observe the cross in Figure 15e). This motivated us to experiment with the CIE-Lab color space which is known to provide better color management [Margulis, 2005].

RGB versus CIE-Lab The CIE-Lab space solves the color-bleeding problem. Visual and numerical comparisons show that our technique produces results close to the exact computation (Figure 15c,g). In contrast, the separable-kernel technique [Pham and van Vliet, 2005] applied in the CIE-Lab space yields results degraded by axis-aligned streaks (Figure 15f). The iterated-box method is described only for per-channel processing, and the extension to vector filtering is unclear.

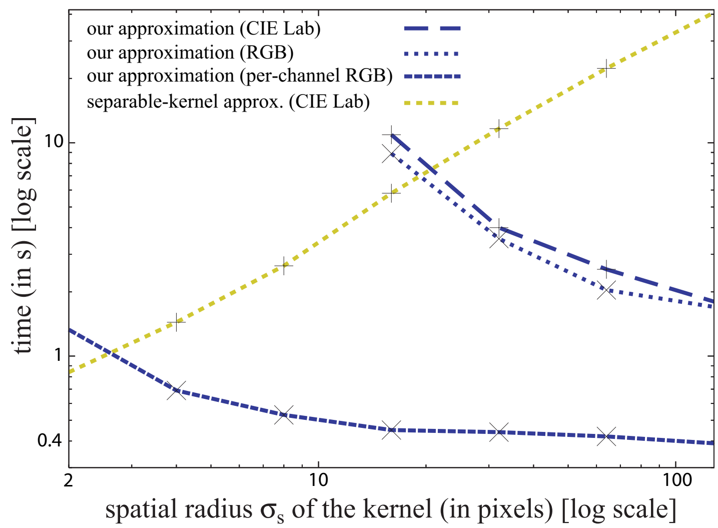


Fig. 16: Time-versus-radius comparison. We processed the image shown in Figure 15 with $\sigma_\mathrm{r} = 0.1$ in RGB color space, and $\sigma_\mathrm{r} = 10$ in the CIE-Lab color space. Both settings correspond to 10% of the intensity range. The plots show the running times of various options depending on the spatial extent of the kernel. Our approximation is not able to process small kernels because of memory limitations (see the text for details). We used a truncated kernel for our method.

7.2 Cross Bilateral Filter

Another advantage of our approach is that it can be extended to cross bilateral filtering. The cross bilateral filter has been simultaneously discovered by Eisemann and Durand [2004] and Petschnigg *et al.* [2004] (who named it "joint bilateral filter"). It smoothes an image I while respecting the edges of another image E:

$$I^\mathrm{c}_\mathbf{p} = \frac{1}{W^\mathrm{c}_\mathbf{p}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_\mathrm{r}}(|E_\mathbf{p}-E_\mathbf{q}|)\, I_\mathbf{q} \tag{21a}$$

$$\text{with} \quad W^\mathrm{c}_\mathbf{p} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_\mathrm{r}}(|E_\mathbf{p}-E_\mathbf{q}|) \tag{21b}$$

The only difference with a classical bilateral filter is the range weight G_{σ_r} which uses E instead of I. Applying this change to all the equations results in a new definition for the w function while the function i remains unmodified (Equation 6):

$$i: \quad (\mathbf{x}\in\mathcal{S}, \zeta\in\mathcal{R}) \quad \mapsto \quad I_\mathbf{x} \tag{22a}$$

$$w: \quad (\mathbf{x}\in\mathcal{S}, \zeta\in\mathcal{R}) \quad \mapsto \quad \delta(\zeta - E_\mathbf{x})\, W_\mathbf{x} \tag{22b}$$

These new functions are used to obtain a set of equations that isolate the nonlinearities. Note that the slicing has to be done at the points $(\mathbf{p}, E_\mathbf{p})$:

$$\text{linear:} \quad (w^\mathrm{c}\, i^\mathrm{c}, w^\mathrm{b}) = g_{\sigma_\mathrm{s},\sigma_\mathrm{r}} \otimes (wi, w) \tag{23a}$$

$$\text{nonlinear:} \quad I^\mathrm{c}_\mathbf{p} = \frac{w^\mathrm{c}(\mathbf{p}, E_\mathbf{p})\, i^\mathrm{c}(\mathbf{p}, E_\mathbf{p})}{w^\mathrm{c}(\mathbf{p}, E_\mathbf{p})} \tag{23b}$$

Intuition Following the example of a pen whose ink varies (cf. Section 3.3), the function wi can be interpreted as the plot of E using a pen with a brightness equal to I (cf. Equation 22). The brightness of the plot depends on I and its shape is controlled by E.

Validation Figure 17 illustrates the effects of the bilateral filter on a flash / no-flash pair. The flash picture has a low level of noise but an unpleasant illumination. The no-flash image has a better illumination but is very noisy. We denoised it with the cross bilateral filter by using the flash image to define the range influence. The achieved result exhibits crisper details than a direct bilateral filter based on the no-flash image alone. Our approximation of the cross bilateral filter achieves the same accuracy and performances as demonstrated for bilateral filtering. We refer the reader to the articles by Petschnigg *et al.* [2004] and Eisemann and Durand [2004] for more advanced approaches to processing flash / no-flash pairs. Bae *et al.* [2006] propose another use of the cross bilateral filter to estimate the local amount of texture in a picture. All the results of this article have been computed with our technique.

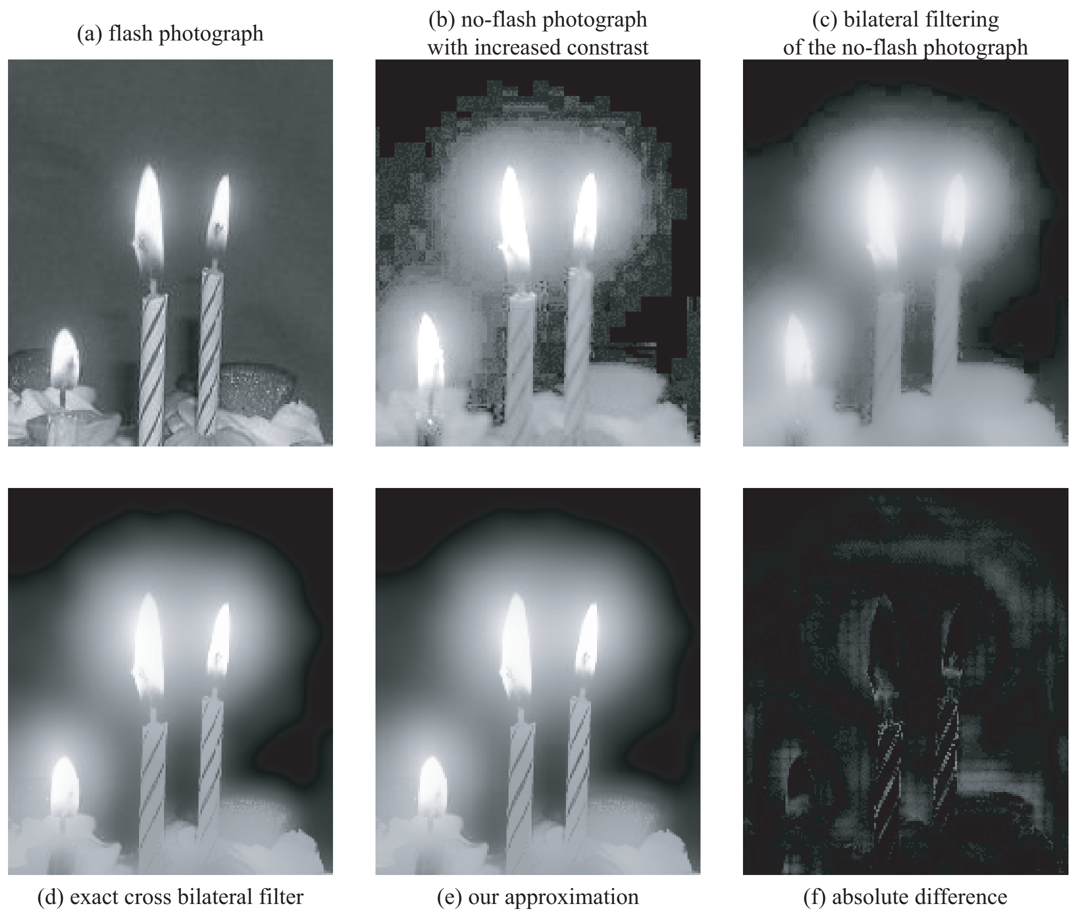


Fig. 17: Example of cross bilateral filtering with a flash / no-flash pair of photographs (a,b). Directly denoising the no-flash picture is hard because of the high level of noise (c). Relying on the flash picture to define the edges to preserve greatly improves the result quality (d,e). Our approximation scheme produces a result (e) visually similar to the exact computation (d). The difference image (f) reveals subtle deviations (a 10× multiplication has been applied). The PSNR is 41dB. We used $\sigma_s = 6$ for all the experiments, and $\sigma_r = 0.25$ for direct bilateral filtering and $\sigma_r = 0.1$ for cross bilateral filtering. These values produce the most pleasing results from all our tests. Depending on your printer, some details may not appear properly in the figure. We recommend the electronic version for a better resolution. The input images are courtesy of Elmar Eisemann.

8 Applications

In this section, we illustrate the capacities of our technique to achieve quality results and meet high standards. We reproduce several of results published previously by the authors.

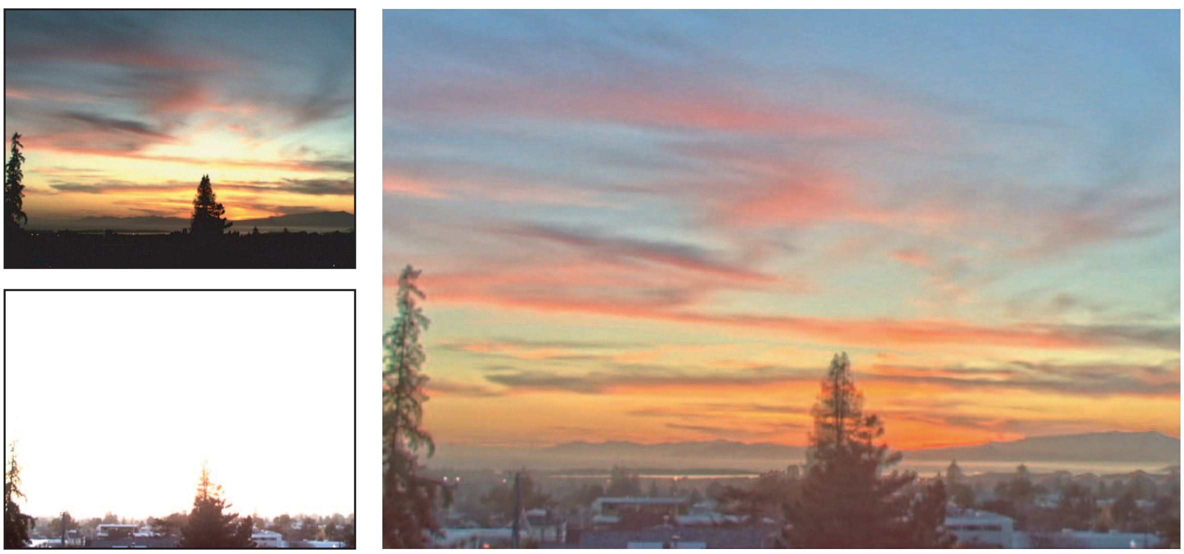
Tone Mapping We used our technique to manipulate the tone distribution of pictures. First, we implemented a simplified version of the tone-mapping operator described by Durand and Dorsey [2002]. Given a high dynamic range image H whose intensity values span a range too broad to be displayed properly on a screen, the goal is to produce a low dynamic range image L that fits the display capacities. The technique can be summarized as follows (see [Durand and Dorsey, 2002] for details):

1. Compute the logarithmic image $\log(H)$ and apply the bilateral filter to split it into a large-scale component (*a.k.a. base*) $B = bf(\log(H))$ and a small-scale component (*a.k.a. detail*) $D = \log(H) - B$.

2. Compress the base: $B' = \gamma B$ where $\gamma = \text{contrast}/\big(\max(B) - \min(B)\big)$. 'contrast' is set by the user to achieve the desired rendition.

3. Compute the result $L = \exp(B' + D)$.

Figure 18 shows a sample result that confirms the ability of our technique to achieve high-quality outputs.



(a) HDR input (b) tone-mapped result

Fig. 18: Example of tone mapping. The input has a dynamic range too large to be displayed correctly, either the sky is over-exposed or the city is under-exposed. Tone-mapping computes a new image that contains all the details in the sky and in the city and that can viewed on a standard display. We implemented a simplified version of the tone-mapping operator described by Durand and Dorsey [2002]. Our approximation does not incur any artifact in the result. The input image is courtesy of Paul Debevec.

Tone Management We also used our method within a more complex pipeline to manipulate the "look" of digital photographs. This technique starts from the same idea as the tone-mapping operator by splitting the image into two layers but then the two layers undergo much stronger transformations in order to adjust both the global contrast and the local amount of the texture of the picture. The bilateral filter is used early in the pipeline to separate the picture into a large-scale layer and a small-scale layer. Cross bilateral filtering is applied later to compute a map that quantifies the local amount of texture at each pixel. Our fast scheme allows this application to be interactive since the result is computed in a few seconds. The details of the algorithm are given in a dedicated paper [Bae et al., 2006]. A sample result is shown in Figure 19. Notice how the result is free form artifacts although an extreme increase of contrast has been applied.

(a) input (b) our result

Fig. 19: Example of tone management. Our approximation of the bilateral filter and of the cross bilateral filter has been used to enhance digital photographs. See the original article for details [Bae et al., 2006].

9 Discussion

9.1 Dimensionality

Our approach may seem counterintuitive at first since to speed up the computation, we increase the dimensionality of the problem. Our separation into linear and nonlinear parts comes at the cost of additional range dimensions (one for gray-level images, three for color images). One has to be careful before increasing the dimensionality of a problem since the incurred performance overhead may exceed the gains, restricting our study to a theoretical discussion. The key of our performances is the possibility to perform an important downsampling of the $\mathcal{S}\times\mathcal{R}$ domain without incurring significant errors. Our tests have shown that the dimensionality is a limiting factor only with color images when using small kernels. In that case, the 5D $\mathcal{S}\times\mathcal{R}$ space is finely sampled and the required amount of memory becomes prohibitive (Section 7.1). Adapting the *narrow band* technique [Adalsteinsson and Sethian, 1995] to bilateral filtering would certainly help on this aspect. Nevertheless, in all the other scenarios (color images with bigger kernels, and single-channel images with kernel of any size), we have demonstrated that our formalism allows for a computation scheme that is several orders of magnitude faster than a straightforward application of the bilateral filter. This advocates performing the computation in the $\mathcal{S}\times\mathcal{R}$ space instead of the image plane. This strategy is reminiscent of level sets [Osher and Sethian, 1988] which alleviate topology management by representing surfaces in a higher-dimensional space. In comparison, we introduce a higher-dimensional image representation that enables dramatic speed-ups through signal downsampling.

Note that using the homogeneous intensities and colors does not increase the dimensionality since Equation 1 and 17 compute the W^{b} function in addition to I^{b} or $\mathbf{C}^{\mathrm{b}}$.

9.2 Comparison with *Channel Smoothing*

Felsberg *et al.* [2006] describe an edge-preserving filter that represents an image as a set of *channels* corresponding to regularly spaced intensity values. For each pixel, the three closest channels are assigned the value a second-order B-spline centered on the pixel intensity. Then, the channels are smoothed independently. The output values are computed by reconstructing a B-spline from three channel values. In our bilateral filter framework, this technique can be interpreted as the replacement of the Gaussian by a B-spline to define the range influence. The channels are similar to our downsampling strategy applied only to the range domain except that we use a box function and a Gaussian instead of a B-spline. This suggests that further speed-up can be obtained by downsampling the channels as well, akin to our space downsampling. The strength of this approach is that aspects such as the *influence function* [Huber, 1981; Hampel et al., 1986; Black et al., 1998; Durand and Dorsey, 2002] can be analytically derived and studied. The downside is the computational complexity introduced to deal with splines. In particular, a relatively complex process is run at each pixel during the final reconstruction to avoid aliasing. In comparison, we cast our approach as an approximation problem and characterize the achieved accuracy with signal processing arguments. Thus, we do not define a new filter and focus on bilateral filtering to improve dramatically its computational efficiency. For instance, we prevent aliasing with a simple linear interpolation and downsample both the range and space domains. Further study of both approaches would be a valuable research contribution.

9.3 Comparison with Image Manifolds

Sochen *et al.* [1998] describe the *geometric framework* that handles images as manifolds in the $\mathcal{S}\times\mathcal{R}$ space. For instance, a gray-level image I is seen as a 2D surface embedded in a 3D space, *i.e.* $z = I(x, y)$. This representation leads to techniques that define functions on the manifold itself instead of the xy plane, and that can be interpreted as deformations of the image manifold. In this context, the bilateral filter is shown to be related to the *short-time kernel* of the heat equation defined directly on the image manifold: Sochen *et al.* [2001] demonstrate that bilateral filtering using a small window and blurring using a Gaussian kernel embedded in the image manifold yield results close to bilateral filtering. This interpretation based on small neighborhoods is related to other results linking bilateral filtering to anisotropic diffusion using partial differential equations [Durand and Dorsey, 2002; Elad, 2002; Buades et al., 2005] since these filters involve only the eight neighbors of a given pixel. In a similar spirit, Barash [2002] uses points in the $\mathcal{S}\times\mathcal{R}$ domain to interpret the bilateral filter. He handles $\mathcal{S}\times\mathcal{R}$ to compute distances and express the difference between adaptive smoothing and bilateral filtering as a difference of distance definitions.

The main difference between these techniques and our interpretation stems from the image representation: in image manifolds, images remain fundamentally two-dimensional – but embedded in a 3D space, while our representation stores values in the whole 3D space. In the geometric framework, each pixel is mapped to a point in $\mathcal{S}\times\mathcal{R}$. Given a point in $\mathcal{S}\times\mathcal{R}$, either it belongs to the manifold and its $\mathcal{S}$ and $\mathcal{R}$ coordinates directly indicate its position and intensity, or it is not on the manifold and is ignored by the algorithm. These methods also use the intrinsic metric of the image manifold. In contrast, we deal with the entire $\mathcal{S}\times\mathcal{R}$ domain, we use its Euclidean metric, we define functions on it, resample it, perform convolutions, and so on. Another major difference is that we define the intensity through a function (i or i^{b}), and that in general, the intensity of a point $(\mathbf{x}, \zeta)$ is *not* its range coordinate ζ, *e.g.* $i^{\mathrm{b}}(\mathbf{x}, \zeta) \neq \zeta$. To our knowledge, this use of the $\mathcal{S}\times\mathcal{R}$ domain has not been described before and opens new avenues to deal with images. We have shown that it enables the use signal-processing techniques and theory to better compute and understand the bilateral filter. Our approach is complementary to existing frameworks and has potential to inspire new image processing techniques.

9.4 Frequency Content

Our interpretation shows that the bilateral filter output is obtained from smooth low-frequency data which may seem incompatible with the feature-preserving aspect of bilateral filtering. The convolution step of our algorithm indeed smoothes the functions defined on the $\mathcal{S}\times\mathcal{R}$ domain. Nevertheless, the final result exhibits edges and corners because during the slicing nonlinearity, two adjacent pixels, *i.e.*

two adjacent points in the spatial domain, can sample this smooth signal at points distant in the range domain, thereby generating discontinuities in the output.

9.5 Comparison with Local-Histogram Approaches

Local histograms are classical intensity histograms where each pixel contributes only a fraction defined by a spatial influence function. Van de Weijer and van den Boomgaard [2001] and Weiss [2006] demonstrated that the result of the bilateral filter at a given pixel is the average intensity of its local histogram with each bin weighted by the range function. This average is normalized by the sum of the weights. Our technique can be interpreted in this framework by remarking that the ζ axis represents histograms. When we first build the $\mathcal{S}\times\mathcal{R}$ domain, the homogeneous coordinate of the w function is a trivial histogram at each pixel $\mathbf{p}$: $w(\mathbf{p},\zeta) = 1$ at $\zeta = I_{\mathbf{p}}$ and 0 at $\zeta \neq I_{\mathbf{p}}$. The bins are point-wise, *i.e.* there is a single intensity value per bin. Then when we downsample the $\mathcal{R}$ domain by a factor s_{r}, the bins become wider and cover an intensity interval of size s_{r}. At this stage, each pixel has still its own histogram. After downsampling the $\mathcal{S}$ domain with a box function of size s_{s}, each group of $s_{\mathrm{s}} \times s_{\mathrm{s}}$ pixels shares the same histogram, and in general, several bins are occupied, *i.e.* $w_{\downarrow}(\mathbf{p},\zeta) > 0$ for several ζ values. Then applying the spatial Gaussian approximates a Gaussian window for the histograms, and the range Gaussian is equivalent to weighting the histogram bins.

Let's have a look at a few specifics of our approach. First, although groups of pixels share the same histogram, there are no blocks in the results since the linear interpolation applied at the end of the algorithm assigns to each pixel its own histogram. In addition, we compute the wi function that stores the mean intensity of each bin. Using this value, we adapt the intensity associated to each bin to the data it contains instead of using a generic value. For instance, if we put a single sample into a bin, this bin is represented by the sample value and not by its midpoint, thereby preventing any error. If several samples are stored together, the bin is assigned their mean. Coupled with the linear interpolation on the range domain, this ensures that our technique does not suffer from intensity aliasing. Figure 20 shows a simple example that confirms that our approximated schemes do not introduce blocks (the diagonal edge is preserved) and that there is no intensity aliasing (constant intensity regions are unaltered).

Fig. 20: A simple edge picture. Our approximations produce the exact result. Although we downsample both the space and range domains, there are no spatial blocking artifacts nor intensity aliasing defects. See the text for details.

9.6 Complexity

Our algorithm operates on two types of data: the original 2D full-resolution image of size $|\mathcal{S}|$ and the low-resolution 3D representation of size $\frac{|\mathcal{S}|}{s_{\mathrm{s}}^2} \times \frac{|\mathcal{R}|}{s_{\mathrm{r}}}$ where $|\cdot|$ indicates the cardinal of a set, and s_{s} and s_{r} are the sampling rates of the space and range domains. The complexity of the method depends on operations performed on both types. The complexity of the convolution is:

- $\mathcal{O}\left(\frac{|\mathcal{S}|}{s_\mathrm{s}^2}\frac{|\mathcal{R}|}{s_\mathrm{r}}\log\left(\frac{|\mathcal{S}|}{s_\mathrm{s}^2}\frac{|\mathcal{R}|}{s_\mathrm{r}}\right)\right)$ for the full-kernel option computed with fast Fourier transform and multiplication in the frequency domain.
- $\mathcal{O}\left(\frac{|\mathcal{S}|}{s_\mathrm{s}^2}\frac{|\mathcal{R}|}{s_\mathrm{r}}\right)$ for the 5^3-kernel option computed explicitly in the spatial domain. Since in this case, we have $(s_\mathrm{s}, s_\mathrm{r}) = (\sigma_\mathrm{s}, \sigma_\mathrm{r})$, the complexity can be expressed as $\mathcal{O}\left(\frac{|\mathcal{S}|}{\sigma_\mathrm{s}^2}\frac{|\mathcal{R}|}{\sigma_\mathrm{r}}\right)$.

Downsampling, upsampling, slicing, and dividing are done pixel by pixel and are linear in the image size *i.e.* $\mathcal{O}(|\mathcal{S}|)$. The total algorithm complexity is thus $\mathcal{O}\left(|\mathcal{S}| + \frac{|\mathcal{S}|}{s_\mathrm{s}^2}\frac{|\mathcal{R}|}{s_\mathrm{r}}\log\left(\frac{|\mathcal{S}|}{s_\mathrm{s}^2}\frac{|\mathcal{R}|}{s_\mathrm{r}}\right)\right)$ with the full-kernel convolution and $\mathcal{O}\left(|\mathcal{S}| + \frac{|\mathcal{S}|}{s_\mathrm{s}^2}\frac{|\mathcal{R}|}{s_\mathrm{r}}\right)$ using a truncated kernel. Hence, depending on the sampling rates, the algorithm's order of growth is dominated either by the convolution or by the downsampling, upsampling, and nonlinearities. Table 2 illustrates the impact of the sampling rates on the running times: Convolution takes most of the time with small sampling values whereas it becomes negligible for large values. This complexity also characterizes the effects of large kernels when the sampling rates are equal to the Gaussian sigmas: The per-pixel operations are not affected but the convolution becomes "cheaper". This is confirmed in practice on the running times shown in Figure 16 on page 28, and Figure 11 on page 19. Finally, photography applications set the kernel spatial radius as a portion of the image size [Durand and Dorsey, 2002; Eisemann and Durand, 2004] and the range sigma can also be proportional the intensity range [Petschnigg et al., 2004; Bae et al., 2006]. In the latter case, the ratios $\frac{|\mathcal{S}|}{\sigma_\mathrm{s}^2}$ and $\frac{|\mathcal{R}|}{\sigma_\mathrm{r}}$ are fixed. This leads to a constant-cost convolution and a global complexity linear in the image size $\mathcal{O}(|\mathcal{S}|)$. For these applications, the running time variations are purely due to the per-pixel operations (downsampling, upsampling, and division).

sampling (s_s,s_r)	(4,0.025)	(8,0.05)	(16,0.1)	(32,0.2)	(64,0.4)
downsampling	1.3s	0.23s	0.09s	0.07s	0.06s
convolution	63s	2.8s	0.38s	0.02s	0.01s
slicing, upsampling, division	0.48s	0.47s	0.46s	0.47s	0.46s

Table 2: Time used by each step at different sampling rates of the architectural image. Upsampling is reported with the nonlinearities because our implementation computes $i^\mathrm{b}_{\downarrow\uparrow}$ only at the $(\mathbf{x}, I_\mathbf{x})$ points rather than upsampling the whole $\mathcal{S}\times\mathcal{R}$ space (cf. Section 4.2). We use the full-kernel implementation with $\sigma_\mathrm{s} = 16$ and $\sigma_\mathrm{r} = 0.1$.

9.7 Practical Use

Our experiments have highlighted the differences between a number of accelerations of the bilateral filter. In practice, the choice of a method will depend on the specific application, and in particular on the spatial kernel size, the need for accuracy vs. performance, the need for extensions such as cross-bilateral filtering and color filtering, as well as ease of implementation constraints.

Brute-force Bilateral Filter The brute-force bilateral filter is a practical choice only for small kernels (in the 5×5 range) and when accuracy is paramount. It is trivial to implement but can be extremely slow (minutes per megapixel).

Separable Kernel The separable kernel approximation should be used only with small spatial kernels (no more than 10 pixels). It is simple to implement and can be adapted to graphcis hardware. However, processing time and accuracy suffer dramatically with bigger kernels. This technique is thus well adapted to noise-removal tasks that typically use small spatial and range sigmas.

Iterated Box Weiss's iterated box method is very efficient for kernels of all sizes. It is only an approximation of the Gaussian bilateral filter, but for most applications such as computational photography, this is probably not an issue. The speed of Weiss's implementation draws heavily from the vector instruction set of modern CPUs and the algorithm implementation might be more complex than the one in this paper. The main drawbacks of Weiss's method are the restriction to box spatial weights, to single-channel images, and the difficulty in extend it to cross bilateral filtering.

Our Method Our method is most efficient for large kernels such as the ones used in computational photography to extract a large-scale component of an image, e.g. [Durand and Dorsey, 2002; Eisemann and Durand, 2004; Petschnigg et al., 2004; Bae et al., 2006]. It is very easy to implement and to extend to color images and cross bilateral filtering. The main weakness of our technique is for small kernels where the size of the higher-dimensional representation becomes prohibitive. Our method subsumes Durand and Dorsey's fast bilateral filter [Durand and Dorsey, 2002] and, in our opinion, there is no compelling reason to use that older method since the new version is simpler conceptually and implementation-wise, and has better accuracy-performance tradeoffs.

10 Conclusions

We have presented a fast approximation of the bilateral filter based on a signal processing interpretation. From a theoretical point of view, we have introduced the notion of homogeneous intensity and demonstrated a new approach of the space-intensity domain: We define intensities through functions that are resampled and convolved in this space whereas existing frameworks use it to represent images as manifolds. Although smooth functions are at the core of our approach, the results exhibit sharp features because of the slicing nonlinearity. We believe that these concepts can be applied beyond bilateral filtering, and we hope that these contributions will inspire new studies. From a practical point of view, our approximation technique yields results visually similar to the exact computation with interactive running times. We have demonstrated that this technique enables interactive applications relying on quality image smoothing. Our experiments characterize the strengths and limitations of our technique compared to existing approaches. Our study also casts a new light on these other methods, leading for instance to a consistent strategy to set the parameters of the iterated-box technique, and pointing out its lack of convergence toward Gaussian and B-spline kernels. We have listed a few guidelines stemming from these tests to select an appropriate bilateral filter implementation depending on the targeted application. Our technique is best at dealing with big kernels. Furthermore, our method is extremely simple to implement and can be easily extended to cross bilateral filtering and color images.

Acknowledgement We are grateful to Ben Weiss, Tuan Pham, and Lucas van Vliet for their remarks and feedback about our experiments, and to Michael Cohen, Todor Georgiev, Pierre Kornprobst, Bruno Lévy, Richard Szeliski, and Matthew Uyttendaele for their insightful discussions and suggestions to extend our conference paper. We thank Tilke Judd, Paul Green, and Sara Su for their help with the article.

This work was supported by a National Science Foundation CAREER award 0447561 "Transient Signal Processing for Realistic Imagery," an NSF Grant No. 0429739 "Parametric Analysis and Transfer of Pictorial Style," a grant from Royal Dutch/Shell Group, and the Oxygen consortium. Frédo Durand acknowledges a Microsoft Research New Faculty Fellowship, and Sylvain Paris was partially supported by a Lavoisier Fellowship from the French "Ministère des Affaires Étrangères."

References

David Adalsteinsson and James A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118:269–277, 1995.

Volker Aurich and Jörg Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Proceedings of the DAGM Symposium*, 1995.

Soonmin Bae, Sylvain Paris, and Frédo Durand. Two-scale tone management for photographic look. *ACM Transactions on Graphics*, 25(3):637 – 645, 2006. Proceedings of the SIGGRAPH conference.

Danny Barash. A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (6):844, 2002.

Danny Barash, Tamar Schlick, Moshe Israeli, and Ron Kimmel. Multiplicative operator splittings in non-linear diffusion: from spatial splitting to multiplicative timesteps. *Journal of Mathematical Imaging and Vision*, 19:33–48, 2003.

Eric P. Bennett and Leonard McMillan. Video enhancement using per-pixel virtual exposures. *ACM Transactions on Graphics*, 24(3):845 – 852, July 2005. Proceedings of the SIGGRAPH conference.

Michael J. Black, Guillermo Sapiro, David H. Marimont, and David Heeger. Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):421–432, March 1998.

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Neighborhood filters and PDE's. Technical Report 2005-04, CMLA, 2005.

Prasun Choudhury and John Erwin Tumblin. The trilateral filter for high contrast images and meshes. In *Proceedings of the Eurographics Symposium on Rendering*, 2003.

Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3), 2002. Proceedings of the SIGGRAPH conference.

Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference.

Michael Elad. On the bilateral filter and ways to improve it. *IEEE Transactions On Image Processing*, 11(10):1141–1151, October 2002.

Michael Elad. Retinex by two bilateral filters. In *Proceedings of the Scale-Space conference*, 2005.

Michael Felsberg, Per-Erik Forssén, and Hanno Scharr. Channel smoothing: Efficient robust smoothing of low-level signal features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 (2):209–222, February 2006.

Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference.

Frank R. Hampel, Elvezio M. Ronchetti, Peter M. Rousseeuw, and Werner A. Stahel. *Robust Statistics – The Approach Based on Influence Functions*. Wiley Interscience, 1986. ISBN 0-471-73577-9.

Peter J. Huber. *Robust Statistics*. Probability and Statistics. Wiley-Interscience, February 1981.

Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3), July 2003. Proceedings of the SIGGRAPH conference.

Ce Liu, William T. Freeman, Richard Szeliski, and Sing Bing Kang. Noise estimation from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE, 2006.

Dan Margulis. *Photoshop LAB Color: The Canyon Conundrum and Other Adventures in the Most Powerful Colorspace*. Peachpit Press, 2005. ISBN: 0321356780.

Pavel Mrázek, Joachim Weickert, and Andres Bruhn. *Geometric Properties from Incomplete Data*, chapter On Robust Estimation and Smoothing with Spatial and Tonal Kernels. Springer, 2006.

Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *Proceedings of the SIGGRAPH conference*. ACM, 2001.

Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

Sylvain Paris, Hector Briceño, and François Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference.

Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. In *Proceedings of the European Conference on Computer Vision*, 2006.

Georg Petschnigg, Maneesh Agrawala, Hugues Hoppe, Richard Szeliski, Michael Cohen, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3), July 2004. Proceedings of the SIGGRAPH conference.

Tuan Q. Pham. *Spatiotonal adaptivity in Super-Resolution of Undersampled Image Sequences*. PhD thesis, Delft University of Technology, 2006.

Tuan Q. Pham and Lucas J. van Vliet. Separable bilateral filtering for fast video preprocessing. In *International Conference on Multimedia and Expo*. IEEE, 2005.

Thomas Porter and Tom Duff. Compositing digital images. *Computer Graphics*, 18(3):253–259, 1984.

Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2006.

Stephen M. Smith and J. Michael Brady. SUSAN – a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

Steven Smith. *Digital Signal Processing*. Newnes, 2002. ISBN: 075067444X.

Nir Sochen, Ron Kimmel, and Alfred M. Bruckstein. Diffusions and confusions in signal and image processing. *Journal of Mathematical Imaging and Vision*, 14(3):237–244, 2001.

Nir Sochen, Ron Kimmel, and Ravi Malladi. A general framework for low level vision. *IEEE Transactions in Image Processing*, 7:310–318, 1998.

Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the International Conference on Computer Vision*, pages 839–846. IEEE, 1998.

Joost van de Weijer and Rein van den Boomgaard. Local mode filtering. In *Proceedings of the conference on Computer Vision and Pattern Recognition*, 2001.

Joachim Weickert, Bart M. ter Haar Romeny, and Max A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7:398–410, 1998.

Ben Weiss. Fast median and bilateral filtering. *ACM Transactions on Graphics*, 25(3):519 – 526, 2006. Proceedings of the SIGGRAPH conference.

Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM Transactions on Graphics*, 25(3):1221 – 1226, 2006. Proceedings of the SIGGRAPH conference.

Wilbur C. K. Wong, Albert C. S. Chung, and Simon C. H. Yu. Trilateral filtering for biomedical images. In *Proceedings of the International Symposium on Biomedical Imaging*. IEEE, 2004.

Jiangjian Xiao, Hui Cheng, Harpreet Sawhney, Cen Rao, and Michael Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *Proceedings of the European Conference on Computer Vision*, 2006.

Leonid P. Yaroslavsky. *Digital Picture Processing. An Introduction.* Springer Verlag, 1985.