Domain Transform for Edge-Aware Image and Video Processing

Eduardo S. L. Gastal* Manuel M. Oliveira†
Instituto de Informática – UFRGS

(a) *Photograph* **(b)** *Edge-aware smoothing* **(c)** *Detail enhancement*

(d) *Stylization* **(e)** *Recoloring* **(f)** *Pencil drawing* **(g)** *Depth-of-field*

Figure 1: *A variety of effects illustrating the versatility of our domain transform and edge-preserving filters applied to the photograph in* (a).

Abstract

We present a new approach for performing high-quality edge-preserving filtering of images and videos in real time. Our solution is based on a transform that defines an isometry between curves on the 2D image manifold in 5D and the real line. This transform preserves the geodesic distance between points on these curves, adaptively warping the input signal so that 1D edge-preserving filtering can be efficiently performed in linear time. We demonstrate three realizations of 1D edge-preserving filters, show how to produce high-quality 2D edge-preserving filters by iterating 1D-filtering operations, and empirically analyze the convergence of this process. Our approach has several desirable features: the use of 1D operations leads to considerable speedups over existing techniques and potential memory savings; its computational cost is not affected by the choice of the filter parameters; and it is the first edge-preserving filter to work on color images at arbitrary scales in real time, without resorting to subsampling or quantization. We demonstrate the versatility of our domain transform and edge-preserving filters on several real-time image and video processing tasks including edge-preserving filtering, depth-of-field effects, stylization, recoloring, colorization, detail enhancement, and tone mapping.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement—Filtering

*eslgastal@inf.ufrgs.br
†oliveira@inf.ufrgs.br

Keywords: domain transform, edge-preserving filtering, anisotropic diffusion, bilateral filter.

Links: DL PDF WEB

1 Introduction

Filtering is arguably the single most important operation in image and video processing. In particular, edge-preserving smoothing filters are a fundamental building block for several applications [Fattal 2009; Farbman et al. 2010], having received considerable attention from the research community over the last two decades. The most popular filters in this class are anisotropic diffusion [Perona and Malik 1990] and the bilateral filter [Tomasi and Manduchi 1998]. While anisotropic diffusion requires an iterative solver, bilateral filtering uses a space-varying weighting function computed at a space

of higher dimensionality than the signal being filtered. As a result, such filters have high computational costs. Recently, several techniques have been proposed that either try to accelerate anisotropic diffusion or bilateral filtering, or introduce alternative ways of performing edge-aware smoothing and other related operations. However, these solutions natively only handle grayscale [Chen et al. 2007; Criminisi et al. 2010; Yang et al. 2009], are still not sufficiently fast for real-time performance [Adams et al. 2010; Farbman et al. 2008; Grewenig et al. 2010; Subr et al. 2009; Weickert et al. 1998], or restrict filtering to certain scales [Fattal 2009].

We present a new approach for efficiently performing edge-preserving filtering of images and videos that addresses the main limitations of previous techniques. Our approach is based on a novel **domain transform** and its efficiency derives from a key observation: an RGB image is a 2D manifold in a 5D space, and an edge-preserving filter can be defined as a 5D spatially-invariant kernel, whose response decreases as the distances among pixels increase in 5D; *if these distances are preserved in a space of lower dimensionality, many spatially-invariant filters in this new space will also be edge-preserving*. The new transform defines an isometry between curves on the 2D image manifold and the real line. It preserves the geodesic distances between points on the curve, adaptively warping the input signal so that 1D edge-preserving filtering can be efficiently performed in linear time. We demonstrate three realizations for our 1D edge-preserving filters, based on normalized convolution, interpolated convolution, and recursion. These filters have very distinct impulse responses, making each one more appropriate for specific applications. Finally, although our 1D filters cannot be exactly generalized to higher dimensions, we show how to use them to efficiently produce high-quality 2D edge-preserving filters.

Our approach has several desirable features. First, the use of 1D operations leads to considerable speedups over existing techniques and potential memory savings. For instance, it can filter one megapixel color images in 0.007 seconds on a GPU. Second, its computational cost is not affected by the choice of the filter parameters. Third, it is the first edge-preserving technique capable of working on color images at arbitrary scales in real time, without resorting to subsampling or quantization.

We demonstrate the versatility of our domain transform and edge-preserving filters on several real-time image and video processing tasks including edge-preserving smoothing, depth-of-field effects, stylization, recoloring, colorization, detail enhancement, and tone mapping (Section 8). Examples of some of these effects can be seen in Figure 1, applied to the photograph shown on the far left.

The **contributions** of our work include:

- A novel approach for efficiently performing high-quality edge-aware filtering of images and videos based on a dimensionality-reduction strategy (Sections 3 and 4). Our approach leads to filters with several desirable features and significant speed-ups over existing techniques;
- A technique to perform anisotropic edge-preserving filtering on curves of the 2D image manifold using 1D linear filters. It consists of anisotropically scaling the curve, which is then mapped to the real line using an isometry, followed by the application of a 1D linear filter (Section 4);
- A technique to efficiently implement 2D edge-preserving smoothing filters as a sequence of 1D filtering operations (Section 5). The resulting 2D filters correctly handle color images and behave as expected even in extreme situations (Section 4.2);
- The *first* edge-preserving smoothing filter that simultaneously exhibits the following properties: (i) it supports a continuum of scales; (ii) its processing time is linear in the number of pixels, and is independent of the filter parameters, allowing for real-time computations; (iii) correctly handles color images; and (iv) offers control over the kernel's shape. For this, we show examples of approximate Gaussian and exponential responses (Section 6);
- A demonstration that our approach can be used to create a variety of effects for images and videos in real time (Section 8).

2 Related Work

There has been a significant amount of work on data-dependent filtering. Existing methods are able to produce good results in many practical scenarios, and edge-aware filters are available in several image-processing applications [Kimball et al. 2011; Adobe Systems Inc. 2010]. This section discusses the approaches most related to ours, and points out the aspects that our technique contributes to advance.

Bilateral Filter A bilateral filter [Tomasi and Manduchi 1998] works by weight averaging the colors of neighbor pixel based on their distances in space and range. For (2D) RGB images, it can be interpreted as operating in a 5D space [Barash 2002]. Durand and Dorsey [2002] compute the filter response by linear interpolation of several discretized intensity values, each filtered with a Gaussian kernel in the frequency domain. Porikli [2008] extended this idea by using summed area tables to filter each intensity level, and Yang et al. [2009] further extended it to arbitrary kernels. All of these methods can only be applied to grayscale images. Paris and Durand [2009] represent the bilateral filter in 5D and perform filtering by downsampling. A simplified version of this method was shown to perform in real-time on GPUs for three-dimensional (grayscale) bilateral filtering [Chen et al. 2007]. Adams et al. [2010] proposed the use of uniform simplices to efficiently implement color bilateral filters in 5D.

All these approaches for accelerating bilateral filters derive their performance from the use of quantization or downsampling. This leads to runtime and/or memory costs that are inversely proportional to the kernels sizes defined over space (σ_s) and range (σ_r). As a result, their performances are severely affected by the use of small values of σ_r (required to enforce edge preservation) or of σ_s (needed for small amounts of blurring). Finally, Pham and Vliet [2005] implement the bilateral filter as a separable operation. This approach has been successfully used for tasks such as abstraction; however, its cost is still high for large kernels.

Anisotropic Diffusion & Related Another popular approach for edge-aware image processing is anisotropic diffusion (AD) [Perona and Malik 1990]. It is modeled using partial differential equations (PDEs) and implemented as an iterative process, which is usually slow. Some approaches have been proposed to improve the speed of AD [Weickert et al. 1998; Grewenig et al. 2010]. However, they do so at the cost of accuracy and still hardly achieve interactive performance. Kimmel et al. [1997] generalized many diffusion processes through the use of Beltrami flow.

Farbman et al. [2008] perform edge-preserving smoothing using a weighted least squares framework. Their approach requires the solution of a sparse linear system, which limits the performance of the technique. The solution of linear systems has also been employed by Levin et al. [2004] for image colorization, and by Subr et al. [2009] for multiscale image decomposition. More recently, Fattal [2009] proposed a new family of edge avoiding wavelets (EAW). This multiscale representation can be quickly computed, but constrains the sizes of the smoothing kernels (in pixels) to powers of two. Criminisi et al. [2010] presented a geodesic framework for edge-aware filtering defined for grayscale images

that employs quantization of the luma channel. Finally, Farbman et al. [2010] proposed the use of diffusion distances for calculating the affinity among pixels, which can be seamlessly integrated with our approach.

In contrast to all these techniques, our approach performs real-time edge-preserving smoothing directly on color images, using kernels of any size, without resorting to sub-sampling or quantization. It can be understood as reducing the dimensionality of the input signal prior to filtering. Lee and Verleysen [2010] present a comprehensive survey on dimensionality-reduction techniques.

3 Transform for Edge-Preserving Filtering

Our approach is inspired by the multi-dimensional interpretation of edge-preserving filters [Barash 2002]. Let $I : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3$ be a 2D RGB color image, defining a 2D manifold M_I in $\mathbb{R}^5$ [Kimmel et al. 1997]. Also, let $\hat{p} = (x_p, y_p, r_p, g_p, b_p) \in M_I$ be a point on this manifold. $\hat{p}$ has a corresponding pixel in I with spatial coordinates $p = (x_p, y_p)$ and range coordinates $I(p) = (r_p, g_p, b_p)$. Let $F(\hat{p}, \hat{q})$ be an edge-preserving filter kernel in 5D. J, the image obtained when filtering I with F can be expressed as

$$J(p) = \int_\Omega I(q)\, F(\hat{p}, \hat{q})\, dq, \tag{1}$$

where $\int_\Omega F(\hat{p}, \hat{q})\, dq = 1$. For example, disregarding normalization, the bilateral filter kernel is given by

$$F(\hat{p}, \hat{q}) = G_{\sigma_s}(\|p - q\|)\, G_{\sigma_r}(\|I(p) - I(q)\|), \tag{2}$$

where $\|\cdot\|$ is the ℓ_2 norm. G_{σ_s} and G_{σ_r} are typically Gaussian spatial and range filters, with supports given by σ_s and σ_r, respectively. Since the bilateral filter works in 5D space, its naive implementation is too slow for many practical uses.

Problem Statement Our work addresses the fundamental question of *whether there exists a transformation $t : \mathbb{R}^5 \rightarrow \mathbb{R}^l$, $l < 5$, and a filter kernel H defined over $\mathbb{R}^l$ that, for any input image I, produce an equivalent result as the 5D edge-preserving kernel F*:

$$J(p) = \int_\Omega I(q)\, F(\hat{p}, \hat{q})\, dq = \int_\Omega I(q)\, H\left(\, t(\hat{p}),\, t(\hat{q})\, \right)\, dq. \tag{3}$$

This construction becomes attractive when evaluating t plus H is more efficient than evaluating the original kernel F. In our case, we are interested in replacing the evaluation of a computationally expensive edge-preserving filter defined in 5D with a domain transformation t and a lower-dimensional linear filter H, evaluated in Euclidean space ($\mathbb{R}^l$). While one could try to exactly mimic the response of a specific filter F (*e.g.*, anisotropic diffusion or the bilateral filter), in this paper we instead focus on finding a transformation that *maintains the edge-preserving property of the filter*.

3.1 Distance-Preserving Transforms

When performing edge-preserving smoothing, the amount of mixing between two pixels should be inversely proportional to their distance, which can be expressed in any metric in the 5D space [Farbman et al. 2010]. For example, the bilateral filter uses the ℓ_2 norm [Chen et al. 2007], while Criminisi et al. [2010] use the intrinsic (geodesic) distance on the image manifold. If the transformation t preserves the original distances from $\mathbb{R}^5$ in $\mathbb{R}^l$, it will also maintain the edge-preserving property of a filter defined in the lower-dimensional space.

A distance-preserving transformation is known as an *isometry* [O'Neill 2006], and finding one is not an easy task. Let us consider the case of mapping a grayscale image to a plane, which involves finding an isometry $t : \mathbb{R}^3 \rightarrow \mathbb{R}^2$. This can be visualized as trying to flatten a heightfield without introducing any metric distortions. Unfortunately, it is known that such mappings do not exist in general (they only exist for surfaces with zero Gaussian curvature) [O'Neill 2006], and only approximate solutions can be found.

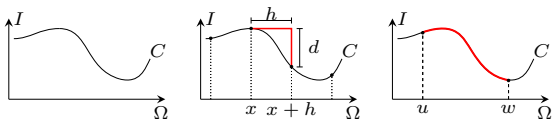


Figure 2: *Curve C defined by the graph $(x, I(x))$, $x \in \Omega$* (left). *In ℓ_1 norm, $\|(x + h, I(x + h)) - (x, I(x))\| = h + d = h + |I(x + h) - I(x)|$* (center). *Arc length of C, from u to w* (right).

For our purpose of edge-aware filtering, preserving the distances among pixels is essential. Approximate solutions introduce hard to predict and image-dependent errors (see supplementary materials). Furthermore, existing approaches from the dimensionality-reduction [Belkin and Niyogi 2003] and texture-mapping [Lévy et al. 2002] literature use optimization methods, which are too slow for our use in real-time edge-preserving filtering. While a solution for a 2D domain does not exist in general, Section 4 shows that an isometric transform exists for a 1D domain. Section 5 then shows how this 1D transform can be effectively used to filter 2D color images.

4 Domain Transform

For deriving an isometric 1D transform, let $I : \Omega \rightarrow \mathbb{R}$, $\Omega = [0, +\infty) \subset \mathbb{R}$, be a 1D signal, which defines a curve C in $\mathbb{R}^2$ by the graph $(x, I(x))$, for $x \in \Omega$ (Figure 2, left). Our goal is to find a transform $t : \mathbb{R}^2 \rightarrow \mathbb{R}$ which preserves, in $\mathbb{R}$, the original distances between points on C, given by some metric. Thus, let $S = \{x_0, x_1, \ldots, x_n\}$ be a sampling of Ω, where $x_{i+1} = x_i + h$, for some sampling interval h. We seek a transform t that satisfies $|t(x_i, I(x_i)) - t(x_j, I(x_j))| = \|(x_i, I(x_i)) - (x_j, I(x_j))\|$, where $x_i, x_j \in S$, $|.|$ is the absolute value operator, and $\|.\|$ is some chosen metric. For simplicity, we use the nearest-neighbor ℓ_1 norm; thus, t only needs to preserve the distances between neighboring samples x_i and x_{i+1}. As we will soon show, this choice gives rise to the geodesic metric. Finally, let $ct(x) = t(\hat{x}) = t(x, I(x))$. To be isometric, the desired transform must satisfy the following equality (in ℓ_1 norm) (Figure 2, center):

$$ct(x + h) - ct(x) = h + |I(x + h) - I(x)|\,, \tag{4}$$

which states that the Euclidean distance between neighboring samples in the new domain ($\mathbb{R}$) must equal the ℓ_1 distance between them in the original domain ($\mathbb{R}^2$). To avoid the need for the absolute value operator on the left of (4), we constrain ct to be monotonically increasing — *i.e.*, $ct(x + h) \geq ct(x)$. Dividing both sides of (4) by h and taking the limit as $h \rightarrow 0$ we obtain

$$ct'(x) = 1 + \left|I'(x)\right|, \tag{5}$$

where $ct'(x)$ denotes the derivative of $ct(x)$ with respect to x. Integrating (5) on both sides and letting $ct(0) = 0$, we get

$$ct(u) = \int_0^u 1 + \left|I'(x)\right|\, dx, \quad u \in \Omega. \tag{6}$$

Intuitively, ct is "unfolding" the curve C defined in $\mathbb{R}^2$ (Figure 2, left) into $\mathbb{R}$, while preserving the distances among neighboring samples. Moreover, for any two points u and w in Ω, $w \geq u$, the distance between them in the new domain is given by

$$ct(w) - ct(u) = \int_u^w 1 + \left|I'(x)\right|\, dx, \tag{7}$$

which is the arc length of curve C in the interval $[u, w]$, under the ℓ_1 norm (Figure 2, right). As such, the transformation given by Equation 6 *preserves the geodesic distance between all points on the curve*. A similar derivation is possible for the ℓ_2 norm or perhaps other metrics.

Multichannel Signals For edge-preserving filtering, it is important to process all channels of the input signal at once, as processing them independently should introduce artifacts around the edges [Tomasi and Manduchi 1998]. For a 1D signal $I : \Omega \to \mathbb{R}^c$ with c channels defining a curve C in $\mathbb{R}^{c+1}$, one can apply a similar derivation to obtain the multichannel transformation:

$$ct(u) = \int_0^u 1 + \sum_{k=1}^{c} \left|I_k'(x)\right| \, dx, \tag{8}$$

where I_k is the k-th channel of signal I. In the case I is an image, I_k can be a color channel in some color space (*e.g.*, RGB or CIE Lab), or a more complex representation, such as a diffusion map [Farbman et al. 2010]. Equation 8 defines a warping $ct : \Omega \to \Omega_w$ of the signal's 1D spatial domain by the isometric transform $t : \mathbb{R}^{c+1} \to \mathbb{R}$, where $ct(u) = t(\hat{u}) = t(u, I_1(u), \ldots, I_c(u))$. We call ct a *domain transform*.

4.1 Application to Edge-Preserving Filtering

Equation 8 reduces the evaluation domain of the filter from $\mathbb{R}^{c+1}$ to $\mathbb{R}$. Thus, the filter H (see Equation 3) is one-dimensional. Since our transformation is isometric, any filter H, whose response decreases with distance at least as fast as F's, will be edge-preserving. Section 6 discusses some choices of H. By reducing the dimensionality of the filter from $c + 1$ to 1, it may seem that we lost the ability to control its support over the signal's space and range (*i.e.*, control the values of σ_s and σ_r, in bilateral filter notation). But, as we show, one can encode the values of σ_s and σ_r in the transformation itself.

Given a 1D signal I and a 1D filtering kernel H (with unit area), we can define $I_a(u) = I(u/a)$, which stretches/shrinks I by a, and $H_{1/a}(u-\tau) = H(au-\tau)\, a$, which shrinks/stretches H by $1/a$ and renormalizes it to unit area, where τ is a translation. Representing the convolution operator by $*$,

$$(I * H_{1/a})(\tau) = \int_{-\infty}^{+\infty} I(u)\, H_{1/a}(u-\tau)\, du = (I_a * H)(\tau). \tag{9}$$

Thus, under convolution, scaling the filter's support by $1/a$ is equivalent to scaling the signal's support by a (and vice-versa). Therefore, to encode the filter's support onto the domain transform, we: (i) derive a_i, for each dimension d_i of the signal I, from the desired support of filter F over d_i; (ii) scale each d_i by its corresponding a_i; (iii) apply the domain transform to the scaled signal; and (iv) filter the signal in 1D using H. This is an important observation, since it shows that the support of the original multi-dimensional kernel F can be completely encoded in 1D. We will refer to the desired variances of the filter F over the signal's spatial domain Ω as σ_s^2, and over the signal's range as $\sigma_{r_k}^2$, $k \in \{1, \ldots, c\}$, for all channels k.

Obtaining the scaling factors By the scaling property of variances [Loeve 1977]:

$$\sigma_{r_k}^2 = Var(H_{1/a}) = Var(H/a) = Var(H)/a^2 = \sigma_H^2/a^2;$$

which solves to $a = \sigma_H/\sigma_{r_k}$, where σ_H^2 is the variance of filter H. The scaling factor for the spatial domain Ω is given by $a = \sigma_H/\sigma_s$. Note that the scaling factor 'a' may vary for each dimension in $\mathbb{R}^{c+1}$, allowing the definition of anisotropic filtering in 1D. These results are valid and produce correct filtering for *any* value $\sigma_H > 0$.

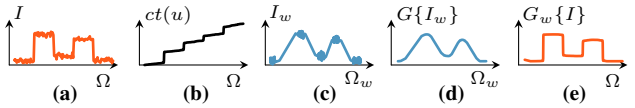


Figure 3: *1D edge-preserving filtering using $ct(u)$.* (a) *Input signal I.* (b) *$ct(u)$.* (c) *Signal I plotted in the transformed domain (Ω_w). Signal I filtered in Ω_w with a 1D Gaussian* (d) *and plotted in Ω* (e).

Scaling the Signal According to Equation 9, before it can be filtered by H, the signal I should be scaled by 'a' prior to evaluating the domain transform. Scaling the distances in the right-hand side of Equation 4 by the appropriate a factors (*i.e.*, $a_s h + a_r \left|I(x+h) - I(x)\right|$) and carrying out the derivation, one obtains

$$ct(u) = \int_0^u \frac{\sigma_H}{\sigma_s} + \sum_{k=1}^{c} \frac{\sigma_H}{\sigma_{r_k}} \left|I_k'(x)\right| \, dx. \tag{10}$$

Since σ_H is a free parameter, we let $\sigma_H = \sigma_s$ and obtain our final **domain transform**, where a single value of σ_r has been used for all channels for simplicity:

$$ct(u) = \int_0^u 1 + \frac{\sigma_s}{\sigma_r} \sum_{k=1}^{c} \left|I_k'(x)\right| \, dx. \tag{11}$$

Filtering the signal in the transformed domain is done through 1D convolution with H. Further details are presented in Section 6. Figure 3 illustrates the use of a domain transform for filtering the 1D signal I, shown in (a) in its original domain Ω. (b) shows the associated domain transform $ct(u)$ computed using Equation 11. (c) shows signal I in the transformed domain Ω_w or, more compactly, $I_w(ct(u)) = I(u)$. The result of filtering I with a Gaussian filter H in Ω_w is shown in (d). (e) shows the desired filtered signal obtained by reversing $ct(u)$ for the signal shown in (d). The small-scale variations were eliminated and the strong edges preserved.

4.2 Analysis

This section analyzes the filtering-related properties of our domain transform (Equation 11). As $ct(x)$ is applied to a 1D signal I, its domain is locally scaled by

$$ct'(x) = 1 + \frac{\sigma_s}{\sigma_r} \left|I'(x)\right|, \tag{12}$$

where the summation over all channels has been omitted for simplicity. According to Equation 9, scaling the input signal I by $ct'(x)$ is equivalent to scaling the support of the filter H by $1/ct'(x)$. Thus, the amount of local smoothing introduced by H in the signal at $I(x)$, can be expressed as

$$\text{smoothing}_H(x) \propto \left(\sigma_H \,/ ct'(x)\right) = \sigma_s \Big/ \left(1 + \frac{\sigma_s}{\sigma_r} \left|I'(x)\right|\right). \tag{13}$$

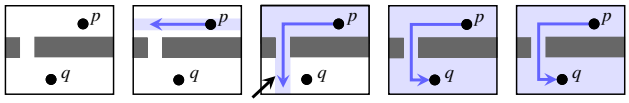
Using (13), we analyze the relationship of H's response with the parameters σ_s and σ_r, as well as with $I(x)$:

$$\lim_{\sigma_r \to \infty} \text{smoothing}_H(x) = \sigma_s, \qquad \lim_{\sigma_r \to 0} \text{smoothing}_H(x) = 0,$$

$$\lim_{\sigma_s \to \infty} \text{smoothing}_H(x) = \frac{\sigma_r}{|I'(x)|}, \qquad \lim_{\sigma_s \to 0} \text{smoothing}_H(x) = 0,$$

$$\lim_{|I'(x)| \to \infty} \text{smoothing}_H(x) = 0, \qquad \lim_{|I'(x)| \to 0} \text{smoothing}_H(x) = \sigma_s.$$

Relationship to σ_r When σ_r approaches infinity, $ct(x) = x$, and, as expected, H's response will be no longer edge-preserving, but a smoothing one proportional to σ_s. When σ_r approaches zero, $ct'(x)$ goes to infinity, and any filter H with compact support will produce a filtered signal identical to the input, as expected.



(a) **(b)** *horz.* **(c)** *1 itr.* **(d)** *1 itr.+horz.* **(e)** *2 itr.*

Figure 4: *2D edge-preserving smoothing using two-pass 1D filtering.* (a) *p and q belong to same region.* (b) *One horizontal pass.* (c) *One complete iteration (*i.e.*, horz.+vert. pass). Information from p cannot yet reach q.* (d) *After an additional horizontal pass.* (e) *Two complete iterations (*i.e.*, horz.+vert.+horz.+vert. passes).*

Relationship to σ_s Interestingly, as σ_s approaches infinity, H does not produce unbounded smoothing in the image. This is exactly what is expected from an edge-preserving filter when σ_r is held constant. Furthermore, the amount of smoothing is inversely proportional to the gradient magnitude of the signal, which is the most commonly used estimator of image edges. Finally, when σ_s approaches zero, no smoothing is performed, as expected.

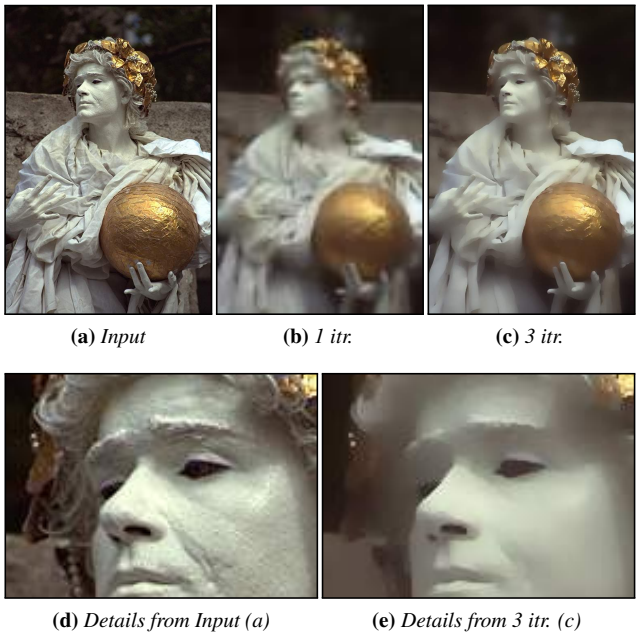
Relationship to I When the gradient magnitude of the input signal is very large, no smoothing is performed. On the other hand, in regions where the gradient magnitude is not significant, smoothing is performed with the same response of a linear smoothing filter. Note that in both cases, our filter behaves as an edge-preserving one.

5 Filtering 2D Signals

Equation 11 defines a domain transform for 1D signals. Ideally, an inherently 2D transform $ct(x, y)$ should be used for 2D signals, directly mapping the content at positions (x, y) in the original domain to positions (u, v) in the transformed domain. Unfortunately, as discussed in Section 3, $ct(x, y)$ (*i.e.*, $t : \mathbb{R}^{c+2} \rightarrow \mathbb{R}^2$) does not exist in general [O'Neill 2006]. Since it is not possible to simultaneously satisfy all the distance requirements in $\mathbb{R}^2$, the use of a space with higher-dimensionality would be needed, implying additional computational and memory costs. To avoid these extra costs, we use our 1D transform to perform 2D filtering.

The most common approach for filtering 2D signals using 1D operations is to perform separate passes along each dimension of the signal. For an image, this means performing a (*horizontal*) pass along each image row, and a (*vertical*) pass along each image column [Smith 1987; Oliveira et al. 2000]. Assuming the horizontal pass is performed first, the vertical pass is applied to the result produced by the vertical one (and vice-versa). This construction is extensively used with standard separable linear filters [Dougherty 1994] and anisotropic diffusion [Weickert et al. 1998], and is also related to the computation of geodesic distances on the image manifold using raster-scan algorithms [Criminisi et al. 2010].

One caveat is that filtering a 2D signal using a 1D domain transform is not a separable operation; otherwise, this would be equivalent to performing $ct(x, y)$ in 2D. Since edge-preserving filters should not propagate information across strong edges, they cannot be implemented using a single *iteration* of a two-pass 1D filtering process (*i.e.*, a horizontal pass followed by a vertical one, or vice-versa). This situation is illustrated in Figure 4, where pixels p and q belong to a same region (represented in white in (a)) and, therefore, should have their information combined. Figure 4 (b) shows, in blue, the region reachable from p after one horizontal pass; and (c) after one complete iteration (assuming that the horizontal pass is performed first). The region reachable from q is analogous. By not reaching the entire white region after one iteration, this process may introduce visual artifacts perceived as "stripes" (indicated by the black arrow in Figure 4 (c)). For this example, one additional horizontal pass would be needed to propagate p's information to the entire white region, thus eliminating the stripe (Figure 4 (d)). Further passes do not alter the result (e).



(a) *Input* **(b)** *1 itr.* **(c)** *3 itr.*

(d) *Details from Input (a)* **(e)** *Details from 3 itr. (c)*

Figure 5: *Two-pass 1D filtering ($\sigma_H = \sigma_s = 40$ and $\sigma_r = 0.77$).* (a) *Input image.* (b) *One filtering iteration.* (c) *Three filtering iterations.* (d) *Details from (a).* (e) *Details from (c). The image content has been smoothed while its edges have been preserved.*

The required number of horizontal and vertical passes depends on (the geometry of) the image content, and, therefore, is hard to predict. However, we use two key observations to make these artifacts unnoticeable in the filtered images: (i) stripes are only present along the last filtered dimension: a horizontal (vertical) step removes stripes introduced by the previous vertical (horizontal) step; and (ii) the length of the stripes is proportional to the size of the filter support used in the last pass. Thus, we interleave a sequence of vertical and horizontal passes, such that the two 1D filters used in each iteration (consisting of a vertical and a horizontal filter) have a σ value that is half of the one used in the previous iteration. This progressively reduces the extension of the artifacts, making them virtually unnoticeable. In practice, three iterations usually suffice to achieve good results (Section 5.1). During a horizontal pass, I' in Equation 11 is the partial derivative computed along the rows of image I, while, in a vertical pass, I' represents the partial derivative computed along the image columns.

Since variances (and not standard deviations) add [Loeve 1977], care must be taken when computing the σ_H value for each iteration: we must use standard deviations that halve at each step *and* whose squared sum matches the original desired variance σ_H^2. This is achieved by the following expression:

$$\sigma_{H_i} = \sigma_H \sqrt{3} \frac{2^{N-i}}{\sqrt{4^N - 1}}, \tag{14}$$

where σ_{H_i} is the standard deviation for the kernel used in the i-th iteration, N is the total number of iterations, and σ_H is the standard deviation of the desired kernel. The image resulting from the i-th iteration is used as input for the $(i + 1)$-th iteration. The domain transform $ct(x)$ is computed only once (for the original image) and used with all the different scales of the filter H.

Figure 6: *Filtering of diagonal edges.* (Left) *Input image (1280×960 pixels).* (Center) *Filtered image with two iterations of our two-pass 1D filter ($\sigma_H = \sigma_s = 50$ and $\sigma_r = 0.5$).* (Right) *Detail from the filtered image.*

Figures 5 (b) and (c) illustrate the results of performing, respectively, one and three 1D edge-preserving filtering iterations on the image shown in (a). Figure 5 (d) and (e) compare the face of the statue before and after the filtering operation, and shows that small scale details have been smoothed while the important edges have been preserved. Although our filter is performed as a series of 1D operations along rows and columns, it correctly handles diagonal edges. Figure 6 illustrates this property on an example containing several sharp edges at various slopes. The image on the center shows the filtered result obtained using only two iterations of our two-pass 1D filter. The original edges have been faithfully preserved, while the colors have been properly filtered.

The decomposition of a 2D edge-preserving filter as a sequence of 1D filtering operations can be generalized to higher dimensions. Unfortunately, this also causes the filter not to be rotationally invariant. However, this is also true for other fast edge-preserving filters [Farbman et al. 2008; Fattal 2009].

5.1 Convergence Analysis

Artifact-free filtered images can be obtained by increasing the number of iterations. Here, we describe an experiment designed to empirically analyze the convergence of the 2D filtering process. For color images with channels in the $[0, 1]$ range, ten to twelve iterations are sufficient to cause the mean-square difference between the results of subsequent iterations to fall below the threshold of 10^{-4}, defined experimentally. The quality of a filtered result obtained after n iterations is evaluated by comparing it to the result obtained for the same image after 15 iterations, which, for practical purposes, can be considered artifact free. The comparison is performed using the Structural Similarity (SSIM) index [Wang et al. 2004]. SSIM is an image-quality metric consistent with human perception. Its structural nature makes it appropriate for detecting "stripes". Since the SSIM index detects similarity, we use its complement $(1 - SSIM)$ as an error measure.

The graph in Figure 7 summarizes the errors measured for various numbers of filtering iterations. These results represent the maximum errors obtained while filtering 31 natural images with various contents. Each curve corresponds to a fixed value of σ_r. For each point along a σ_r curve, we plot the maximum error obtained among all values of $\sigma_s \in \{1, 10, 20, 40, 60, 80, 100, 200, 500, 1000, 3000\}$. The graph shows that the dissimilarity metric decreases quickly with the first three iterations, which defines a good tradeoff between filtering quality and computational time.

6 Filtering in the Transformed Domain

Given a domain transform $ct : \Omega \rightarrow \Omega_w$, the transformed signal $I_w(ct(x)) = I(x)$ is then filtered using the 1D kernel H. This section discusses alternatives for performing this filtering operation on digital signals, where I_w will likely be non-uniformly sampled.

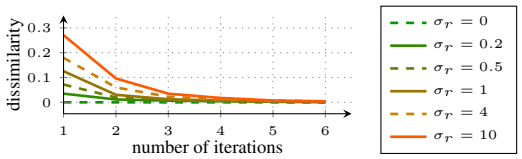


Figure 7: *Maximum measure of dissimilarity (according to SSIM) between filtered images and their corresponding "ideal" results as a function of number of iterations, for different values of σ_r.*

6.1 Normalized Convolution (NC)

Filtering the non-uniformly sampled signal $I_w(ct(x))$ in Ω_w can be seen as filtering a uniformly sampled signal with missing samples (Figure 8, left). This scenario has been studied by Knutsson and Westin [1993] in the context of data uncertainty, where they showed that optimal filtering results, in the mean square sense, are obtained by *normalized convolution* (NC). For a uniform discretization $D(\Omega)$ of the original domain Ω, NC describes the filtered value of a sample $p \in D(\Omega)$ as

$$J(p) = (1/K_p) \sum_{q \in D(\Omega)} I(q)\ H\left(\, t(\hat{p}),\, t(\hat{q})\, \right), \qquad (15)$$

where $K_p = \sum_{q \in D(\Omega)} H(\, t(\hat{p}),\, t(\hat{q})\,)$ is a normalization factor for p, and $t(\hat{p}) = ct(p)$. For N samples and an arbitrary kernel H, the cost of evaluating Equation 15 for all p is $O(N^2)$. However, as $ct(x)$ is monotonically increasing (Equation 11), we use an efficient *moving-average* approach [Dougherty 1994] to perform NC with a box filter in $O(N)$ time. The box kernel is defined as

$$H\left(\, t(\hat{p}),\, t(\hat{q})\, \right) = \delta\{\ |t(\hat{p}) - t(\hat{q})| \leq r\ \}, \qquad (16)$$

where $r = \sigma_H \sqrt{3}$ is the filter radius, and δ is a boolean function that returns 1 when its argument is true, and 0 otherwise. This box kernel has a constant radius in Ω_w, but a space-varying and non-symmetric radius in Ω, where its size changes according to the similarity between p and its neighborhood in the image manifold M_I (Figure 8, right, in blue). This can be interpreted as an estimate of which neighbors belong to the same population as p. The box kernel is then a robust estimator of the population mean, with connections to robust anisotropic diffusion [Black et al. 1998] and bilateral filtering [Durand and Dorsey 2002].

The cost of evaluating Equation 15 using the box kernel from Equation 16 is linear in the number of samples. We use it for the 1D filtering iterations described in Section 5, with σ_{H_i} defined by Equation 14. For three iterations, the resulting filter produces an indistinguishable approximation to a Gaussian filter (PSNR > 40) when $\sigma_r = \infty$. Figure 11 compares this result to the ones obtained with several other filters.

CPU Implementation Since samples are not uniformly spaced in Ω_w, the number of samples added to and removed from the kernel window as it slides from one sample to the next is not constant. Thus, performing box filtering in Ω_w requires updating K_p, plus one additional memory read per sample to check its domain coordinate. One only needs to perform convolution at positions in Ω_w that contain samples, as other positions will not contribute to the filtered image in the (discrete) original domain. Finally, derivatives are estimated using backward differences.

GPU Implementation Our domain transform is highly parallel: each thread calculates the value of $ct'(x)$ (Equation 12) for one sample, and a scan operation performs the integration. For filtering, each thread computes the filtered value of one pixel. To find the first and last pixels inside the current 1D kernel window, we perform two

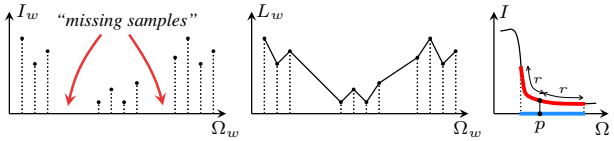


Figure 8: *Filtering in the transformed domain.* (Left) *Normalized convolution (NC).* (Center) *Interpolated convolution (IC).* (Right) *Their interpretation: NC box kernel in blue, IC box kernel in red.*

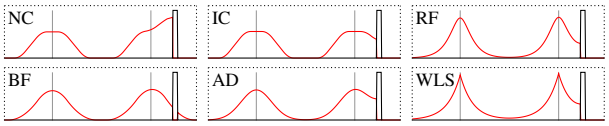


Figure 9: *Impulse response for various filters at neighborhoods without (left) and with strong edges (right). NC: Normal. Convolution; IC: Interp. Convolution; RF: Recursive Filter; BF: Bilateral Filter; AD: Anisotropic Diffusion; WLS: Weighted Least Squares.*

binary searches on the transformed domain (Ω_w) coordinates. Once the first and last pixels under the 1D kernel have been identified, the sum of the colors of all contributing pixels is calculated using a 1D summed area table (per color channel). These tables need to be updated before each horizontal/vertical pass.

6.2 Interpolated Convolution (IC)

Another option when dealing with irregularly sampled data is to use interpolation for approximating the original continuous function [Piroddi and Petrou 2004]. Figure 8 (center) shows a reconstructed signal L_w obtained by linear interpolation (in Ω_w) of the samples shown in Figure 8 (left). Filtering L_w is performed by continuous convolution:

$$J(p) = \int_{\Omega_w} L_w(x)\ H\left(\, t(\hat{p}),\, x\,\right)\ dx, \tag{17}$$

where H is a normalized kernel. Interpolated convolution has an interesting interpretation: a *linear* diffusion process working *on the signal*. Figure 8 (right) shows this interpretation for a box filter of radius r, where the kernel window is shown in red. This is the same interpretation as the 1D Beltrami flow PDE [Sochen et al. 2001].

Implementation For a box filter, Equation 17 can be evaluated for all pixels in $O(N)$ time. This is achieved by a *weighted moving-average* [Dougherty 1994]. The normalized box kernel is given by

$$H\left(\, t(\hat{p}),\, x\,\right) = \delta\big\{\ |t(\hat{p}) - x| \le r\ \big\} \big/\, 2\,r, \tag{18}$$

where $r = \sigma_H \sqrt{3}$ is the filter radius. Substituting (18) in (17):

$$J(p) = \frac{1}{2\,r} \int_{t(\hat{p})-r}^{t(\hat{p})+r} L_w\,(x)\ dx. \tag{19}$$

The linearly-interpolated signal L_w does not need to be uniformly resampled, since the area under its graph can be explicitly computed using the trapezoidal rule.

6.3 Recursive Filtering (RF)

For a discrete signal $I[n] = I(x_n)$, non edge-preserving filtering can be performed using a 1st-order recursive filter as

$$J[n] = (1-a)\ I[n] + a\ J[n-1], \tag{20}$$

where $a \in [0,1]$ is a *feedback coefficient* [Smith 2007]. This filter has an infinite impulse response (IIR) with exponential decay: an impulse of magnitude m at position i generates a response of magnitude $m\ (1-a)\ a^{j-i}$ at $j \ge i$. Note that $j-i$ can be interpreted as the *distance* between samples x_i and x_j, assuming a unitary sampling interval. Based on this observation, a recursive edge-preserving filter can be defined in the transformed domain as

$$J[n] = (1-a^d)\ I[n] + a^d\ J[n-1], \tag{21}$$

where $d = ct(x_n) - ct(x_{n-1})$ is the distance between neighbor samples x_n and x_{n-1} in the transformed domain (Ω_w). As d increases, a^d goes to zero, stopping the propagation chain and, thus, preserving edges. This can be interpreted as a geodesic propagation on the image lattice. The impulse response of (21) is not symmetric, since it only depends on previous inputs and outputs (it is a causal filter). A symmetric response is achieved by applying the filter twice: for a 1D signal, (21) is performed left-to-right (top-to-bottom) and then right-to-left (bottom-to-top).

The feedback coefficient of this filter is computed from the desired filter variance as $a = \exp(-\sqrt{2}/\sigma_H)$ (see the appendix for a derivation). Since $a \in [0,1]$, the filter is stable [Smith 2007], and its implementation in $O(N)$ time is straightforward.

7 Comparison to Other Approaches

We compare our edge-preserving filters based on normalized convolution (NC), interpolated convolution (IC), and recursion (RF) against previous works: brute-force bilateral filter (BF) [Tomasi and Manduchi 1998]; anisotropic diffusion (AD) [Perona and Malik 1990]; edge-avoiding wavelets (EAW) [Fattal 2009]; weighted least squares filter (WLS) [Farbman et al. 2008], which has been shown to produce optimal results for tone and detail manipulation; and finally the permutohedral lattice BF (PLBF) [Adams et al. 2009] and constant time BF (CTBF) [Yang et al. 2009], which are, respectively, the fastest color and grayscale bilateral filter approximations.

Filter Response Figure 9 shows a comparison of the impulse response of our three filters NC, IC and RF (all performed using three iterations) against the impulse response of BF, AD and WLS. The NC and IC filters have Gaussian-like response, similar to AD and BF. In the presence of strong edges, the IC filter behaves similarly to AD. The NC filter has a higher response near strong edges, which is a direct implication of its interpretation as a robust mean: pixels near edges have less neighbors in the same population, and will weight their contribution strongly. Finally, our recursive filter (RF) has an exponential impulse response which is completely attenuated by strong edges, like the WLS's response.

The NC filter is ideal for stylization and abstraction, since it accurately smoothes similar image regions while preserving and sharpening relevant edges. For applications where sharpening of edges is not desirable (*e.g.*, tone mapping and detail manipulation), the IC and RF filters produce results of equal quality as the state-of-the-art techniques [Farbman et al. 2008; Fattal 2009]. Finally, for edge-aware interpolation (*e.g.*, colorization and recoloring), the RF filter produces the best results due to its infinite impulse response, which propagates information across the whole image lattice. Section 8 illustrates the use of our filters for all these applications.

Smoothing Quality Figure 10 shows a side-by-side comparison of edge-aware smoothing applied to a portion of the photograph shown in Figure 1 (a). For small amounts of smoothing, the bilateral filter (b) and our NC filter (c) produce visually similar results. For further smoothing, the bilateral filter may incorrectly mix colors, as observed in the window frame (Figure 10 (d)). In contrast, our filter manages to continuously smooth image regions while preserving strong edges. This effect, illustrated in Figure 10 (e), is sim-

(a) *Input* **(b)** *BF,* $\sigma_s = 17$ **(c)** *NC,* $\sigma_s = 17$ **(d)** *BF,* $\sigma_s = 40$

(e) *NC,* $\sigma_s = 80$ **(f)** *WLS* **(g)** *AD* **(h)** *EAW*

Figure 10: *Qualitative comparison of the results of bilateral filters (BF) with* $\sigma_r = 0.2$ *(*b *and* d*), our normalized convolution filter (NC) with* $\sigma_r = 0.8$ *(*c *and* e*), weighted least squares filter (WLS)* (f) *with* $\lambda = 0.15$ *and* $\alpha = 2$*, anisotropic diffusion (AD)* (g)*, and the edge-avoiding wavelets (EAW)* (h)*.*

ilar to the results obtained with WLS, shown in (f), and anisotropic diffusion (using [D'Almeida 2004]), shown in (g). Since the scale of EAW cannot be freely controlled, the technique is not ideal for edge-preserving smoothing. Figure 10 (h) shows the result produced by EAW with a maximum decomposition depth of 5 and coefficients for each detail level defined by $0.6^{(5-level)}$, which preserve some high-frequency details. Setting these coefficients to zero results in distracting artifacts. Additional comparisons among these techniques can be found in the supplementary materials.

Our filters converge to standard linear smoothing filters on regions with weak edges, or when the range support σ_r is set to a large value (see Section 4.2). This feature is desirable, for instance, in joint filtering for performing depth-of-field effects, as shown in Figure 1 (g) and discussed in Section 8. Figure 11 shows that previous techniques have difficulty to simulate a regular smoothing filter, either because their kernels cannot be explicitly controlled, as in the cases of WLS (b) and EAW (c); or because the downsampling required for performance introduces structural artifacts, as in the case of PLBF (d). In contrast, our filters provide an indistinguishable approximation to a Gaussian (f) for $\sigma_r = \infty$, as shown in (e).

7.1 Performance Evaluation

This section reports performance numbers obtained on a 2.8 GHz Quad Core PC with 8 GB of memory and a GeForce GTX 280.

Filtering on CPU We implemented our NC and RF filters on CPU using C++. For the IC filter we have a MATLAB implementation, but its performance is expected to be similar to NC's. On a single CPU core, the typical runtimes of our NC and RF filters for processing a 1 megapixel color image using three iterations are 0.16 and 0.06 seconds, respectively. Their performances scale linearly with image size, filtering 10 megapixel color images in under 1.6 and 0.6 seconds. On a quad-core CPU, we achieve a 3.3× speedup.

We compare the performance of our edge-aware filters against the fastest filters from previous works: EAW, PLBF and CTBF. The PLBF and our NC and RF filters process all three color channels simultaneously, while CTBF only processes grayscale. Thus, CTBF is actually performing one third of the work done by the other three methods. We measured the reported results on a single CPU core. For PLBF and CTBF we used source code provided by the authors.

(a) *Input* **(b)** *WLS* **(c)** *EAW* **(d)** *PLBF* **(e)** *Our NC* **(f)** *Gaussian*

Figure 11: *Approximating a Gaussian filter. Parameters from all approaches where tuned to best approximate a Gaussian with* $\sigma = 15$*. For the NC filter,* $\sigma_s = 15$*, and* $\sigma_r = \infty$*. Note that WLS and EAW were not designed to approximate Gaussian blur. Best viewed in the electronic version.*

The runtimes for both PLBF and CTBF are inversely proportional to the value of σ_r. For σ_r approaching zero, their runtimes are above 10 seconds. The runtimes of our filters are independent of the σ_s and σ_r parameters, and they use no simplifications to improve performance. In our experience, to achieve good edge-preserving smoothing with PLBF or CTBF values of $\sigma_r < 0.15$ should be used. In this range, our filters with three iterations are 5 to 15× faster than these approaches. For small amounts of smoothing, one can obtain good results using two or even one iteration of our filter (Figure 7), with speed-ups of 25 to 40× over PLBF for color filtering. According to Fattal [2009], EAW can smooth a 1 megapixel grayscale image in 0.012 seconds on a 3.0 GHz CPU. Since it generates decompositions at only a few scales, it is not generally applicable for edge-preserving smoothing. WLS takes 3.5 seconds to solve its sparse linear system using a fast CPU implementation. A graph comparing the performances of these techniques can be found in the supplementary materials.

Filtering on GPU We implemented our NC filter on GPU using CUDA. The total time required for filtering a 1 megapixel color image is 0.7 msec for computing the domain transform plus 2 msec for each 2D filtering iteration. This gives a total runtime of approximately 0.007 seconds for three iterations of our filter— a speedup of 23× compared to our one-core CPU implementation. Since our filter scales linearly with the image size, our GPU implementation is able to filter 10 megapixel color images in under 0.07 seconds.

We compare the performance of our GPU filter against the GPU Bilateral Grid [Chen et al. 2007]. While their implementation is as fast as ours, it only processes luminance values, which may generate undesired color-ghosting artifacts. Their approach draws its efficiency from downsampling, which is not possible for small spatial and range kernels. The GPU implementation of PLBF can filter a 0.5 megapixel image in 0.1 sec on a GeForce GTX 280 [Adams et al. 2010]. A GPU implementation of WLS filters a 1 megapixel grayscale image in about 1 second [Farbman et al. 2008].

8 Real-Time Applications

We show a variety of applications that demonstrate the versatility of our domain transform and filters for image processing. Given its speed, our approach can be performed on the fly on high-resolution images and videos. This improved performance provides users with instant feedback when tuning filter parameters. Examples of video applications are included in the supplementary materials.

(a) *Input* **(b)** *Ours (IC)* **(c)** *EAW*

Figure 12: *Fine detail manipulation. (a) Input image. (b) Our result. J_1 was obtained with the IC filter ($\sigma_s = 20$ and $\sigma_r = 0.08$). (c) EAW result by Fattal [2009].*

(a) *Ours (RF)* **(b)** *WLS*

Figure 13: *Tone mapping results: (a) using our RC filter and (b) using the WLS filter.*

Detail Manipulation Edge-preserving filters can be used to decompose image details into several scales, which can be manipulated independently and recombined to produce various effects [Farbman et al. 2008; Fattal 2009]. Let $J_0, \ldots, J_k$ be progressively smoother versions of an image $I = J_0$. Several *detail layers* capturing progressively coarser details are constructed as $D_i = J_i - J_{i+1}$. Figure 12 shows an example of fine-scale detail enhancement applied to the flower image in (a). The result in (b) was created by filtering the image in (a) once using our IC filter ($\sigma_s = 20$ and $\sigma_r = 0.08$) and by manipulating the detail layer D_0 using a sigmoid function described by Farbman et al. [2008]. Figure 12 (c) shows the result produced by an EAW filter of Fattal [2009]. The images (b) and (c) present similar visual quality. Our filter, however, allows for extra flexibility when decomposing image details, since it can produce a continuum of smoothed images J_i, by varying the values of the parameters σ_s and σ_r.

Tone Mapping Edge-aware tone mapping avoids haloing and other artifacts introduced in the compression process. Figure 13 compares the result of a tone mapping operator implemented using our RF filter (a) and the WLS filter by Farbman et al. [2008] (b). The quality of these results are similar, but our filter is significantly faster, resulting in the fastest high-quality tone-mapping solution available. The result in Figure 13 (a) was obtained by manipulating three detail layers from the HDR image's log-luminance channel. Each layer was obtained in 12 msec using two iterations of our RF filter with: $\sigma_s = 20$ and $\sigma_r = 0.33$ for J_1; $\sigma_s = 50$ and $\sigma_r = 0.67$ for J_2; and $\sigma_s = 100$ and $\sigma_r = 1.34$ for J_3. The compressed luminance channel L_C was obtained as:

$$L_C = 0.12 + \mu + 0.9\ (B - \mu) + 0.3\ D_0 + 0.2\ D_1 + 0.2\ D_2;$$

where B is a linear compression of J_3 to the range $[0, 1]$ (*i.e.*, $B = (J_3 - min(J_3))/(max(J_3) - min(J_3))$), and μ is the mean value of B. J_i and D_i were defined in the previous paragraph.

Figure 14: *High-contrast edges around the most salient features of an edge-aware filtered image, producing a stylized look.*

Stylization Stylization aims to produce digital imagery with a wide variety of effects not focused on photorealism. Edge-aware filters are ideal for stylization, as they can abstract regions of low-contrast while preserving, or enhancing, high-contrast features. Figure 14 illustrates an application of our NC filter to produce abstracted results. Given an input image (top left), the magnitude of the gradient of the filtered image (top right) is superimposed to the filtered image itself to produce high-contrast edges around the most salient features. Another interesting stylization effect can be obtained by assigning to each output pixel a scaled version of the value of the normalization factor K_p from Equation 15. This produces a pencil-like non-photorealistic drawing, such as the one shown in Figure 1 (f), obtained scaling K_p by 0.11.

Joint Filtering Our approach can also be used for *joint filtering*, where the content of one image is smoothed based on the edge information from a second image. For instance, by using the values of an alpha matte [Gastal and Oliveira 2010] as the the image derivatives in Equation 11, one can simulate a depth-of-field effect (Figure 1 (g)). This example emphasizes why converging to a Gaussian-like response is an important property of our filter. Alpha mattes can also be combined with other maps to create some *localized* or *selective stylization*, as shown in Figure 15. In this example, an alpha matte and a structure resembling a Voronoi diagram have been added to define new edges to be preserved (Figure 15, bottom left). The resulting filter produces a unique stylized depth-of-field effect. The edges of the diagram were then superimposed to the filtered image to create the result shown on the right. A similar result is shown in Figure 1 (d), where the edges to be preserved were identified by a Canny edge detector.

Colorization Similar to previous approaches [Levin et al. 2004; Fattal 2009], we propagate the colors S from user-supplied scribbles by blurring them using the edge information from a grayscale image I (Figure 16, left). To keep track of how much color propagates to each pixel, we also blur a normalization function N, which is defined to be one at pixels where scribbles are provided and zero otherwise. Let $\tilde{S}$ and $\tilde{N}$ be the blurred versions of S and N, respectively. The final color of each pixel p is obtained as $\tilde{S}(p)/\tilde{N}(p)$. This value is combined with the original luminance from the grayscale image to produce the colorized result. Figure 16 compares the results obtained with our RF filter ($\sigma_s = 100$ and

Figure 15: *Stylized depth-of-field effect.* (Top Left) *Input image.* (Bottom Left) *Alpha matte combined with a Voronoi-like diagram.* (Right) *Result produced by our edge-aware filter using joint filtering. See text for details.*

$\sigma_r = 0.03$) with the ones of Levin et al. [2004]. In our experience, good colorization results can be obtained with values of σ_r from 0.01 to 0.1 and $\sigma_s > 100$.

Recoloring Localized manipulations of image colors is achieved using soft segmentation. Color scribbles define a set of regions of interest, where each region R_i is defined by a color c_i. For each region R_i, an *influence map* [Lischinski et al. 2006] is obtained by blurring its associated normalization function N_{R_i} defined by all scribbles with color c_i. The contribution of R_i for the recoloring of pixel p is defined as $\tilde{N}_{R_i}(p) / \sum_j \tilde{N}_{R_j}(p)$. Figure 1 (e) shows a recoloring example obtained using this technique.

Our filters are temporally stable and can be applied to videos in real time. Please, see the accompanying video for examples.

9 Conclusions and Future Work

We have presented a new approach for performing high-quality edge-preserving filtering of images and videos in real time. Our solution is based on a transform that defines an isometry between curves on the 2D image manifold in 5D and the real line. This transform preserves the geodesic distance between points on the curve, adaptively warping the input signal so that 1D edge-preserving filtering can be efficiently performed in linear time. We demonstrated three realizations for our 1D edge-preserving filters, based on normalized convolution, interpolated convolution, and recursion. These filters have very distinct impulse responses, making each one more appropriate for specific applications.

We have shown how to produce high-quality 2D edge-preserving filtering by iterating 1D-filtering operations. We analyzed the convergence of this process and showed that three iterations provide a good compromise between image quality and computational time.

Our approach has several desirable features. First, it is applied to the signal's original samples, correctly handling color images. Second, the use of 1D operations leads to considerable speedups over existing techniques and potential memory savings. Third, its computational cost is not affected by the choice of the filter parameters. Finally, it is the first edge-preserving filter capable of working on color images at arbitrary scales in real time, without resorting to subsampling or quantization. It can filter a 1 megapixel color image at approximately 150 fps using three iterations, which is significantly faster than previous techniques.

We have demonstrated the versatility of our domain transform and edge-preserving filters on several real-time image and video processing tasks including edge-preserving filtering, depth-of-field effects, stylization, recoloring, colorization, detail enhancement, and

(a) *Input* **(b)** *Ours* **(c)** *Levin et al. [2004]*

Figure 16: *Colorization example.* (a) *Grayscale input image with user scribbles.* (c) *Our result using RF ($\sigma_s = 100$ and $\sigma_r = 0.03$).* (b) *Result of Levin et al. [2004].*

tone mapping. One feature of our filters is that their responses stop at strong edges. This is in contrast with the bilateral filter, whose kernel can cross edges (Figure 9). We believe both behaviors are valid and may provide optimal results for different kinds of applications. The speed, flexibility, and high-quality of the results achieved with our technique may enable many new applications that have not been previously possible.

Limitations As most other fast edge-preserving filters, our 2D filters are not rotationally invariant (*i.e.*, filtering a rotated image and rotating a filtered image may produce different results). This behaviour may cause problems for applications that rely on content matching.

Our 1D edge-preserving filters can be applied to other kinds of signals and to higher-dimensional data. Other possible directions for exploration involve applying our filters on meshes, and their implementation in the frequency domain.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. This work was sponsored by CNPq-Brazil (fellowships and grants 557814/2010-3, 200284/2009-6, 308936/2010-8, and 480485/2010-0). Input images from Figures 12, 15 and 16 courtesy of Farbman et al [2008], www.alphamatting.com and Levin et al. [2004], respectively. Figure 13 generated from a radiance map courtesy of Paul Debevec. Input images from Figures 6 and 14 are from publicdomainpictures.net (images 8363 and 5298).

References

ADAMS, A., GELFAND, N., DOLSON, J., AND LEVOY, M. 2009. Gaussian kd-trees for fast high-dimensional filtering. In *SIGGRAPH*.

ADAMS, A., BAEK, J., AND DAVIS, M. A. 2010. Fast high-dimensional filtering using the permutohedral lattice. *CGF 29*, 2, 753–762.

ADOBE SYSTEMS INC., 2010. Photoshop CS5. Computer software.

BARASH, D. 2002. A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE TPAMI 24*, 844–847.

BELKIN, M., AND NIYOGI, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation 15*, 6 (June), 1373–1396.

BLACK, M., SAPIRO, G., MARIMONT, D., AND HEEGER, D. 1998. Robust anisotropic diffusion. *IEEE TIP 7*, 3, 421–432.

CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM TOG 26*, 3, 103.

CRIMINISI, A., SHARP, T., ROTHER, C., AND P'EREZ, P. 2010. Geodesic image and video editing. *ACM TOG 29*, 5, 134.

D'ALMEIDA, F., 2004. Nonlinear Diff. Toolbox (mathworks.com/matlabcentral/fileexchange/3710-nonlinear-diffusion-toolbox).

DOUGHERTY, E. 1994. *Digital Image Processing Methods*. Optical engineering. CRC Press.

DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In *SIGGRAPH '02*, 257–266.

FARBMAN, Z., FATTAL, R., LISCHINSKI, D., AND SZELISKI, R. 2008. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM TOG 27*, 3, 67.

FARBMAN, Z., FATTAL, R., AND LISCHINSKI, D. 2010. Diffusion maps for edge-aware image editing. *ACM TOG 29*, 6, 145.

FATTAL, R. 2009. Edge-avoiding wavelets and their applications. *ACM TOG 28*, 3, 22.

GASTAL, E. S. L., AND OLIVEIRA, M. M. 2010. Shared sampling for real-time alpha matting. *CGF 29*, 2, 575–584.

GREWENIG, S., WEICKERT, J., AND BRUHN, A. 2010. From box filtering to fast explicit diffusion. *Pattern Recognition*, 533–542.

KIMBALL, S., MATTIS, P., AND GIMP DEVELOPMENT TEAM, 2011. GNU Image Manipulation Program. Computer software.

KIMMEL, R., SOCHEN, N. A., AND MALLADI, R. 1997. From high energy physics to low level vision. In *Scale-Space Theory in Computer Vision*, Springer-Verlag, 236–247.

KNUTSSON, H., AND WESTIN, C.-F. 1993. Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data. In *CVPR*, 515–523.

LEE, J., AND VERLEYSEN, M. 2010. *Nonlinear Dimensionality Reduction*. Springer.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. *ACM TOG 23*, 689–694.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLO T, J. 2002. Least squares conformal maps for automatic texture atlas generation. In *ACM SIGGRAPH*, 362–371.

LISCHINSKI, D., FARBMAN, Z., UYTTENDAELE, M., AND SZELISKI, R. 2006. Interactive local adjustment of tonal values. *ACM TOG 25*, 3, 646–653.

LOEVE, M. 1977. *Probability Theory*, vol. 1. Springer.

OLIVEIRA, M. M., BISHOP, G., AND MCALLISTER, D. 2000. Relief texture mapping. In *ACM SIGGRAPH*, 359–368.

O'NEILL, B. 2006. *Elementary Differential Geometry*. AP.

PARIS, S., AND DURAND, F. 2009. A fast approximation of the bilateral filter using a signal processing approach. *IJCV 81*, 1, 24–52.

PERONA, P., AND MALIK, J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE TPAMI 12*, 7, 629–639.

PHAM, T., AND VAN VLIET, L. 2005. Separable bilateral filtering for fast video preprocessing. *IEEE Intl. Conf. on Multimedia and Expo 0*, 4 pp.

PIRODDI, R., AND PETROU, M. 2004. Analysis of irregularly sampled data: A review. *Advances in Imaging and Electron Physics 132*, 109–165.

PORIKLI, F. 2008. Constant time O(1) bilateral filtering. In *CVPR*, 1–8.

SMITH, A. R. 1987. Planar 2-pass texture mapping and warping. In *Proc. SIGGRAPH 87*, 263–272.

SMITH, J. O. 2007. *Introduction to Digital Filters with Audio Applications*. W3K Publishing.

SOCHEN, N., KIMMEL, R., AND BRUCKSTEIN, A. 2001. Diffusions and confusions in signal and image processing. *Journal of Mathematical Imaging and Vision 14*, 3, 195–209.

SUBR, K., SOLER, C., AND DURAND, F. 2009. Edge-preserving multiscale image decomposition based on local extrema. *ACM TOG 28*, 147:1–147:9.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *ICCV*, 839–846.

WANG, Z., BOVIK, A., SHEIKH, H., AND SIMONCELLI, E. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. on Image Processing 13*, 4, 600–612.

WEICKERT, J., ROMENY, B., AND VIERGEVER, M. 1998. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE TIP 7*, 3, 398–410.

YANG, Q., TAN, K. H., AND AHUJA, N. 2009. Real-time O(1) bilateral filtering. In *CVPR*, 557–564.

Appendix: Derivation of RF feedback coefficient a from the desired variance σ_H^2

The continuous equivalent of the recursive kernel is

$$f(x) = (1-a)\,a^x, \quad x \in [0,+\infty),\ a \in (0,1);$$

where x represents the distance between samples and a is the feedback coefficient. $f(x)$ is not normalized since

$$\int_0^\infty f(x)\,dx = \left.\frac{(1-a)a^x}{log(a)}\right|_0^\infty = -\frac{1-a}{log(a)}.$$

Normalizing f we obtain $f(x) = -log(a)\,a^x$. The first and second moments of f are, respectively:

$$\begin{aligned}\langle f\rangle &= -log(a)\int_0^\infty x\,a^x dx &&= -\frac{1}{log(a)} \quad \text{and}\\ \langle f^2\rangle &= -log(a)\int_0^\infty x^2\,a^x dx &&= \frac{2}{log(a)^2}.\end{aligned}$$

The variance of f is then given by

$$Var(f) = \langle f^2\rangle - \langle f\rangle^2 = \frac{1}{log(a)^2}.$$

Since the signal is filtered twice with f (left-to-right and right-to-left), the total variance of the filter is $2\ Var(f)$. Given the desired variance σ_H^2:

$$\sigma_H^2 = 2\ Var(f) = \frac{2}{log(a)^2};$$

we solve for a and find

$$a = \exp(-\sqrt{2}/\sigma_H) \quad \text{and} \quad a = \exp(\sqrt{2}/\sigma_H);$$

where the former is our solution since $a \in (0,1)$.