Accelerating Spatially Varying Gaussian Filters

Jongmin Baek David E. Jacobs
Stanford University*

Figure 1: *Use of spatially varying Gaussian filters.* **Left***: A noisy signal (left) is filtered with a bilateral filter (middle) and with a bilateral filter whose kernel is oriented along the signal gradient (right). The bilateral filter tends to create piecewise-flat regions.* **Middle***: Tone mapping with the bilateral filter (left) and our proposed algorithm (right). The bilateral filter suffers from false edges and blooming at high contrast regions, such as the perimeter of the lamp.* **Right***: Range image upsampling with the bilateral filter (top) and our proposed algorithm (bottom). The bilateral filter introduces spurious detail on the road at color edges—e.g. the center divider and the shadows on the road.*

Abstract

High-dimensional Gaussian filters, most notably the bilateral filter, are important tools for many computer graphics and vision tasks. In recent years, a number of techniques for accelerating their evaluation have been developed by exploiting the separability of these Gaussians. However, these techniques do not apply to the more general class of *spatially varying* Gaussian filters, as they cannot be expressed as convolutions. These filters are useful because the underlying data—e.g. images, range data, meshes or light fields—often exhibit strong local anisotropy and scale. We propose an acceleration method for approximating spatially varying Gaussian filters using a set of spatially invariant Gaussian filters each of which is applied to a segment of some non-disjoint partitioning of the dataset. We then demonstrate that the resulting ability to locally tilt, rotate or scale the kernel improves filtering performance in various applications over traditional spatially invariant Gaussian filters, without incurring a significant penalty in computational expense.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement—Filtering; I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Sensor fusion

Keywords: bilateral filter, anisotropic gaussian filter, trilateral filter, depth upsampling.

*e-mail: {jbaek, dejacobs}@cs.stanford.edu

1 Introduction

The bilateral filter, first termed over a decade ago [Tomasi and Manduchi 1998], has become popular in computer graphics, vision and computational photography. It is a non-iterative, non-linear filter that blurs pixels spatially while preserving sharp edges. If the weights are Gaussian, it can be expressed neatly as a Gaussian blur in an elevated space that encompasses both spatial location and intensity [Paris and Durand 2006]. Other variations, such as the joint bilateral filter [Eisemann and Durand 2004; Petschnigg et al. 2004] and non-local means [Buades et al. 2005], also can be expressed in this framework of high-dimensional Gaussian filtering.

However, there is no inherent requirement that the Gaussian kernel be spatially invariant. For instance, an extension was proposed by Choudhury and Tumblin [2003] in the context of tone mapping high dynamic range images. In this so-called trilateral filter, the Gaussian kernel is tilted in range as to follow the gradients in the input, leading to better preservation of image details.

The bilateral filter and its relatives have enjoyed a speed-up of several orders of magnitude since their inception [Chen et al. 2007; Adams et al. 2009; Adams et al. 2010]. Unfortunately, acceleration of spatially varying Gaussian filters has lagged far behind, as all these methods rely on the separability of spatially invariant Gaussian kernels. Efforts to evaluate anisotropic Gaussian filters on a regular 2D grid [Geusebroek and Smeulders 2003; Lampert and Wirjadi 2006] also presuppose spatial invariance. No algorithm currently exists for accelerating spatially varying Gaussian filters.

We describe an efficient method for evaluating spatially varying Gaussian filters. We exploit the observation that even in cases that call for spatially varying kernels, the kernel may be considered *locally* invariant. This allows the application of existing methods for spatially invariant Gaussian filters, such as the algorithm of Adams et al. [2010], on segments of the dataset, each admitting a nearly constant kernel. Each data point is filtered in multiple segments, and the results are interpolated in order to suppress seam artifacts.

The rest of the paper is organized as follows: in Section 2, we review the formulation of the bilateral filter as a high-dimensional Gaussian filter, and generalize the formulation to allow spatially varying kernels; in Section 3, we consider a number of schemes that enable the use of existing acceleration techniques; in Section 4, we demonstrate the versatility of our algorithm by applying it to the tasks of contrast management and sensor fusion.

2 Review of Related Work

The bilateral filter is a non-linear filter whose weight depends not only on the spatial distance between pixels but also on the distance in their intensity. For a grayscale image I, we define its filtered image I' at position $\vec{p} = (x, y)$ as

$$I'(\vec{p}) = \frac{1}{K} \sum_{\vec{q}} I(\vec{q}) \cdot N_{\sigma_s}(\|\vec{p} - \vec{q}\|) \cdot N_{\sigma_t}(I(\vec{p}) - I(\vec{q})), \quad (1)$$

where $K = \sum_{\vec{q}} N_{\sigma_s}(\|\vec{p} - \vec{q}\|) \cdot N_{\sigma_t}(I(\vec{p}) - I(\vec{q}))$ is a normalization factor, and $N_\sigma(\cdot)$ is a Gaussian kernel with standard deviation σ. For a color bilateral filter, the image I becomes a three-valued function $\vec{I}(\vec{p}) = (r(\vec{p}), g(\vec{p}), b(\vec{p}))$ instead.

The bilateral filter is useful in a number of applications, including tone mapping, style transfer, relighting, denoising, upsampling, and data fusion. Paris et al. [2008] discuss these applications in detail.

2.1 High-Dimensional Gaussian Filtering

As demonstrated in the literature [Paris and Durand 2006; Adams et al. 2009], the bilateral filter can be ensconced in a more general framework of high-dimensional Gaussian filtering: given a set of position-value pairs $(\vec{P}_i, \vec{V}_i)$ indexed by i, where $\vec{P}_i, \vec{V}_i$ are respectively d_p- and d_v-dimensional signals, we define the output $\vec{V'}_i$:

$$\forall i, \quad \vec{V'}_i := \sum_j \vec{V}_j \exp\left\{ -\frac{(\vec{P}_i - \vec{P}_j)^T D (\vec{P}_i - \vec{P}_j)}{2} \right\}, \quad (2)$$

where D is a d_p-by-d_p diagonal correlation matrix of the form

$$D = \begin{bmatrix} {\sigma_1}^{-2} & & \\ & \ddots & \\ & & {\sigma_{d_p}}^{-2} \end{bmatrix},$$

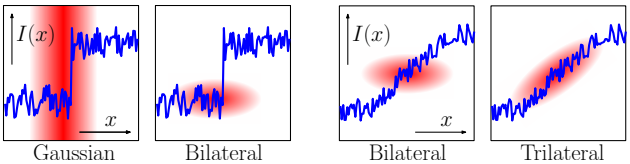
controlling the standard deviation of the blur in each dimension of the position vectors. For instance, bilaterally filtering an RGB image $\vec{I}_i = (r_i, g_i, b_i)$ in this framework is equivalent to setting

$$\vec{P}_i := (x_i, y_i, r_i, g_i, b_i) \qquad (d_p = 5),$$
$$\vec{V}_i := (r_i, g_i, b_i, 1) \qquad (d_v = 4).$$

The last dimension in $\vec{V}$ has been added as a homogeneous coordinate in anticipation of the normalization. Finally, $D := \text{diag}\{{\sigma_s}^{-2}, {\sigma_s}^{-2}, {\sigma_s}^{-2}, {\sigma_t}^{-2}, {\sigma_t}^{-2}\}$ where σ_s and σ_t are the spatial and tonal standard deviations. The R, G, B values of the bilaterally filtered output can be obtained by dividing the first three components of $\vec{V'}_i$ by its last component. This homogeneous coordinate value can also be thought of as a measure of confidence in the bilateral filter, as it represents the total weight of all data points contributing to the blurred value.

We briefly discuss recent techniques that rapidly evaluate Eq. (2).

Bilateral Grid: The bilateral grid [Chen et al. 2007] is a dense data structure that voxelizes the space of the position vectors into a regular hypercubic lattice. It is perhaps the most natural way to express images whose samples are arranged in a grid. To evaluate Eq. (2), convolution with a windowed 1D Gaussian is carried out along each of the d_p-axes. However, as images are typically elevated to a higher-dimensional space before filtering, a dense data structure wastes memory, and as a result, the runtime scales exponentially with d_p, the dimensionality of the position vectors.



Gaussian Bilateral Bilateral Trilateral

(a) Filtering a noisy step signal. (b) Filtering a noisy ramp signal.

Figure 2: *Comparison of various filters operating on a 1D signal. The filter weight is shown as an intensity map.* **Left***: The bilateral filter improves upon a spatial Gaussian filter, avoiding blurring across edges.* **Right***: When the signal has a linear ramp, the support of the bilateral filter becomes limited, leading to artifacts seen in Figure 1; the trilateral filter adapts to the local gradient.*

Gaussian KD-Tree: The Gaussian KD-tree [Adams et al. 2009] allows importance sampling of the space of the position vectors. Because most position vectors generate negligible weights, it is sufficient to find a few j's in Eq. (2) that contribute the most. To evaluate this equation, one builds a KD-tree of the position vectors, and makes a query for k samples nearby to each position. At each splitting hyperplane, the query is split into two subqueries, one for each subtree, such that the samples are divided proportionally to the expected weights (with respect to Eq. (2)) of the two subtrees. The Gaussian KD-tree can handle sparse data, and its runtime grows log-linearly with the size of the dataset and linearly with d_p.

Permutohedral Lattice: The permutohedral lattice [Adams et al. 2010] is a sparse lattice that tessellates the space with simplices. By exploiting the fact that the number of vertices in a simplex grows slowly with d_p, combined with sparsity, it avoids the exponential growth of runtime that the bilateral grid suffers. Its runtime grows linearly with the size of the dataset and quadratically with d_p.

All these techniques rely on the separability of the Gaussian kernel.

2.2 Anisotropic Gaussian Filtering

A d_p-variate Gaussian kernel need not be isotropic in order to be separable; as long as it is spatially invariant, a suitable rotation or tilt of the space makes the kernel decomposable into a number of 1D blurs along the standard axes. Several recent works have focused on accurately evaluating spatially invariant anisotropic Gaussian filters on a regular 2D grid [Geusebroek and Smeulders 2003; Lampert and Wirjadi 2006; Lam and Shi 2007].

Mathematically, a spatially varying kernel means that the correlation matrix D is no longer fixed and that it may be non-diagonal. However, D must be symmetric and positive-definite to ensure that it can be approximated with a finite support. The first instance of spatially varying Gaussian filters used in graphics of which we are aware is the trilateral filter [Choudhury and Tumblin 2003] that operates on a grayscale image I. For each pixel label i, define a tilted image $\hat{I}_i$ as follows,

$$\hat{I}_i(x, y) := I(x, y) - G_i^1 \cdot (x - x_i) - G_i^2 \cdot (y - y_i),$$

where G_i^1, G_i^2 are the image gradients in x- and y-directions at (x_i, y_i), respectively. Then the output of the trilateral filter is

$$I'(x_i, y_i) := \frac{1}{K} \sum_j \hat{I}_i(x_j, y_j) \exp\left\{ -\frac{\Delta x^2 + \Delta y^2}{2{\sigma_s}^2} - \frac{\hat{I}_i(x_j, y_j)^2}{2{\sigma_t}^2} \right\},$$

where $\Delta x := x_j - x_i$, $\Delta_y = y_j - y_i$. This formulation resembles Eq. (1), but the image has been locally approximated as a plane, and then tilted to be flat before being bilaterally filtered. This simple modification leverages the fact that the bilateral filter is best applied to piecewise flat regions, as illustrated in Figure 2.

It is straightforward to show that the trilateral filter can be expressed in terms of Eq. (2): we set $\vec{P}_i = (x_i, y_i, I(x_i, y_i))$ and $\vec{V}_i = (I(x_i, y_i), x_i, y_i, 1)$, and let

$$D_i := \begin{bmatrix} {\sigma_s}^{-2} + \left(G_i^1\right)^2 {\sigma_t}^{-2} & -G_i^1 {\sigma_t}^{-2} & \\ -G_i^1 {\sigma_t}^{-2} & {\sigma_s}^{-2} + \left(G_i^2\right)^2 {\sigma_t}^{-2} & -G_i^2 {\sigma_t}^{-2} \\ & -G_i^2 {\sigma_t}^{-2} & {\sigma_t}^{-2} \end{bmatrix}. \quad (3)$$

Note that D_i is now explicitly non-diagonal and spatially varying. As before, denote by $\vec{V'}_i$ the filtered output values with respect to the position-value pairs $(\vec{P}_i, \vec{V}_i)$ and spatially varying correlation matrix D_i. The desired output of the trilateral filter is given by

$$I'(x_i, y_i) = \frac{(\vec{V'}_i)_1 - G_i^1 (\vec{V'}_i)_2 - G_i^2 (\vec{V'}_i)_3}{(\vec{V'}_i)_4} + G_i^1 x_i + G_i^2 y_i.$$

See Section A of the supplement for the derivation of D_i and the above equation.

The trilateral filter was originally implemented in a brute-force fashion that scales poorly with d_p and cannot handle unstructured data. Shen et al. [2009] improves runtime by downsampling the image, but does not address the fundamental issue of spatial variance.

There are other applications besides tone mapping that can benefit from spatially varying Gaussian filters. For instance, object geometries are also typically piecewise linear, and existing geometry smoothing algorithms fit a tangent plane before attempting to smooth the data [Choudhury and Tumblin 2003; Fleishman et al. 2003]. Similarly, the usefulness of spatially varying bilateral filters for images with spatially varying uncertainty (e.g. shot noise) has already been acknowledged [Zhang and Allebach 2008; Granados et al. 2010]. Other unexplored datasets with strong local anisotropy or scale variation include range data, videos and light fields.

3 Acceleration

We consider three distinct approaches for evaluating spatially varying Gaussian filters. Each takes advantage of a unique property of an existing acceleration technique for spatially invariant Gaussian filters. The first method modifies the sampling function of the Gaussian KD-tree with a numerical integration step to directly evaluate anisotropic filter kernels. The second method embeds the data in a yet-higher-dimensional space where anisotropic kernels become separable, enabling spatially varying evaluation using a standard Gaussian KD-tree. The third method leverages the speed of the spatially invariant permutohedral lattice by segmenting the dataset into overlapping regions of constant anisotropic kernel. In practice we find this final approach, which we call *kernel sampling*, to be the best suited for common filtering tasks, and demonstrate its use in Section 4.

3.1 Importance Sampling

The Gaussian KD-tree [Adams et al. 2009] performs importance sampling of the dataset in order to evaluate Eq. (2) quickly. It is briefly summarized in Algorithm 1 and Figure 3. The separability of a Gaussian kernel is exploited when the integrals q_0, q_1 over bounding boxes R_0, R_1 are evaluated: all that matters is their ratio, and since the two bounding boxes differ only along one dimension (the axis normal to the splitting hyperplane), it suffices to consider only this dimension, and the ratio q_0/q_1 can be evaluated in $O(1)$.

We note that because the evaluation of Eq. (2) for a particular index i is not tied to that of another index, the Gaussian KD-tree is well-suited to the task of spatially varying blurs. In fact, if the spatially varying Gaussian kernel is still axis-aligned—e.g. D is diagonal—the extension is trivial. However, if D is not diagonal, we then must solve the problem of evaluating the integral of an artibrary d_p-variate Gaussian over a d_p-dimensional box:

When $d_p = 2$, the problem can be transformed to the evaluation of rectangular bivariate normal (BVN) probabilities [Genz 2004] in the applied mathematics literature:

$$\psi(\vec{b}, \rho) = \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^{b_1} \int_{-\infty}^{b_2} e^{\left\{-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right\}} dy dx. \quad (4)$$

The BVN problem can be solved efficiently using numerical methods. Because we seek to blur pixels, high precision is not required, and we find that the Gauss-Legendre quadrature of order 6 suffices in practice. By the Fundamental Theorem of Calculus, we may calculate the integral of the kernel over the bounding box with 2^{d_p} calls to the BVN routine.

When $d_p = 3$, the problem of evaluating rectangular trivariate normal (TVN) probabilities is still tractable. For larger d_p, however, techniques for evaluating multi-variate normal probabilities quickly become prohibitively expensive as they succumb to the curse of dimensionality. Nonetheless, importance sampling with the Gaussian KD-tree remains a very general solution, albeit impractical for high dimensional problems.

Algorithm 1 Evaluation of Eq. (2) with importance sampling.

Require: $\vec{P},\ \vec{V},\ i,\ D$
  $k \leftarrow$ the number of samples desired.
  $\mathcal{R} \leftarrow$ the root of the KD-tree.
  Call QUERY$(\vec{P}_i, \mathcal{R}, k, D)$.
  for all j where $\vec{P}_j$ is a leaf node found **do**
    $w_j \leftarrow$ the kernel evaluated at $\vec{P}_j$.
    $w_j \leftarrow w_j$ divided by the likelihood of having sampled $\vec{P}_j$.
  end for
  return $\sum_j w_j \vec{V}_j$.

QUERY$(\vec{X}, R, k, D)$
  If R is a leaf node, store R and return.
  $R_0 \leftarrow$ the left child of R.
  $R_1 \leftarrow$ the right child of R.
  for all $i \in \{0, 1\}$ **do**
    $q_i \leftarrow$ the integral of the kernel corresponding to D (centered at $\vec{x}$) over the bounding box of the subtree rooted at R_i.
  end for
  for all $i \in \{0, 1\}$ **do**
    Call QUERY $\left(\vec{X}, R_i, k \cdot \frac{q_i}{q_0 + q_1}\right)$.
  end for
  return

3.2 Dimensionality Elevation

We saw in the previous section that the Gaussian KD-tree can perform spatially varying blurs as long as D is diagonal. With this in mind, consider the problem of anisotropic Gaussian blur with $d_p = 2$. The correlation matrix D is a 2-by-2 matrix of the form

$$D = \begin{bmatrix} A & C/2 \\ C/2 & B \end{bmatrix} \quad (5)$$

for some A, B, C. The weight of the filter is then,

$$f(x, y) = \exp\left\{-\frac{Ax^2 + By^2 + Cxy}{2}\right\}. \quad (6)$$

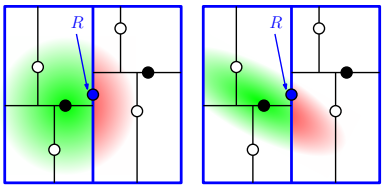


Figure 3: *Importance sampling with the Gaussian KD-tree. The node being queried and its bounding box are marked in blue. The query is split into the two children nodes, marked in black, based on the ratio of the integrals of the filter over their bounding boxes.* **Left***: When D is diagonal, the Gaussian is separable along the axes of the KD-tree, so the ratio can be computed easily.* **Right***: When D is not, integrating the kernel over the bounding boxes is non-trivial.*

Unfortunately, it is not possible to express the exponent $Ax^2 + By^2 + Cxy$ as a linear sum of x^2 and y^2 terms. These two terms correspond to blurring along the x- and y-axis, and no combination of them can generate the correlation term Cxy. Now, if it were possible to blur along some 1D axis other than those standard axes, the correlation term may be expressible. This notion was explored by Lam and Shi [2007] in the context of performing spatially invariant anisotropic Gaussian on a 2D grid. Consider elevating the position vectors by the following mapping:

$$M : (x, y) \mapsto \left(\frac{x}{\sqrt{2}}, \frac{y}{\sqrt{2}}, \frac{x+y}{2}, \frac{x-y}{2} \right).$$

One can easily show that M is an isometry:

$$\forall x_i, y_i, x_j, y_j,\ \|(x_i, y_i) - (x_j, y_j)\| = \|M(x_i, y_i) - M(x_j, y_j)\|.$$

An axis-aligned Gaussian blur in this space has the form

$$f(x, y) = \exp\left\{ -\left(\frac{ax^2 + by^2}{4} + \frac{c(x+y)^2 + d(x-y)^2}{8} \right) \right\},$$

where $a, b, c, d \geq 0$ are controllable. Therefore, it is possible to simulate Eq. (6) by setting a, b, c, d such that the exponent above matches the desired quadratic expression, and the blur can be performed in this elevated space, as visualized in Figure 4. Solving for a, b, c, d is described in Section B of the supplement. However, there is a bound on the maximum eccentricity of D that can be obtained this way for some orientations.

Algorithm 2 Evaluation of Eq. (2) with dimensionality elevation ($d_p = 2$).

Require: $\vec{P},\ \vec{V},\ i,\ D = [A, C/2; C/2, B]$
Require: The KD-tree has been created in the elevated space.
$k \leftarrow$ the number of approximate nearest neighbors desired.
$\mathcal{R} \leftarrow$ the root of the KD-tree.
$a, b, c, d \leftarrow$ the solution of

$$\frac{Ax^2 + By^2 + Cxy}{2} = \frac{ax^2 + by^2}{4} + \frac{c(x+y)^2 + d(x-y)^2}{8}.$$

$D^* \leftarrow \text{diag}\{a, b, c, d\}$.
Call QUERY$(\vec{P}_i, \mathcal{R}, k, D^*)$.
Follow the rest of Algorithm 1.

Algorithm 2 summarizes the overall procedure. The isometry M is equivalent to adding the ability to blur along the two main diagonals $x = \pm y$. When $d_p = 3$, diagonals $x = \pm y, y = \pm z, z = \pm x, x = \pm y = \pm z$ can be added, resulting in an isometric mapping $M' : \mathbb{R}^3 \to \mathbb{R}^{13}$. Once again, the method scales exponentially with d_p, simply because the set of possible axes grows combinatorially.

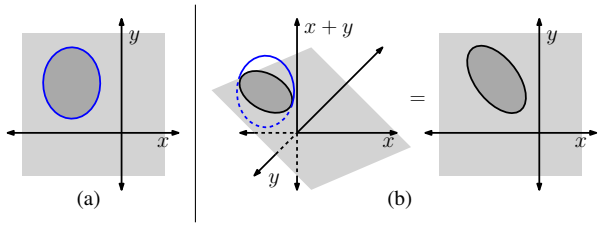


Figure 4: *The effect of elevating the image space via isometry.* **Left***: The Gaussian KD-tree allows axis-aligned Gaussian blurs.* **Right***: Once the image is elevated via an isometry, an axis-aligned Gaussian blur in the elevated space is equivalent to a tilted Gaussian blur in the original image space.*

3.3 Kernel Sampling and Segmentation

We observe that while the two previous proposed methods are general methods that deal with arbitrary spatially varying Gaussian kernels, they necessarily incur the curse of dimensionality. In order to avoid this cost, we introduce two additional assumptions.

- The kernel is locally constant. In other words, local anisotropy exists, but when it does exist, it affects a sizable region instead of a single point.
- While the space of possible kernels is very large—D has $O(d_p^2)$ degrees of freedom—often restricting the kernels to a smaller subspace is sufficient. For instance, the correlation matrix for the trilateral filter in Eq. (3) has only 2 degrees of freedom, set by G_i^1 and G_i^2. Let d_r signify the dimension of this restricted space of kernels.

The first assumption is useful because now our kernel is spatially invariant at least locally, and we know how to apply spatially invariant Gaussian filters efficiently. The second is useful because restricting the space of possible kernels makes sampling it feasible.

Armed with these assumptions, we begin by sparsely sampling the space of kernels. The space of kernels may be regularly sampled, or kernels that appear often in the dataset can be chosen via a clustering algorithm. Let $\mathcal{D} = \{D^1, D^2, \ldots\}$ be the set of kernels we sample. The naïve approach here is to filter the entire dataset with each kernel in $\mathcal{D}$, and to interpolate the results to simulate kernels that were not sampled, as shown in Figure 5(b). Unfortunately, this method scales linearly with $|\mathcal{D}|$, which may be large.

Instead, we note that the interpolation in the kernel space only requires a few samples (as few as $d_r + 1$ samples for the barycentric interpolation used by [Adams et al. 2010]). It suffices then to apply each kernel only to the parts of the dataset that are relevant, as shown in Figure 5(c). Formally, for each kernel $D^l \in \mathcal{D}$, define the segment S^l as the set of position vectors satisfying the following: $\vec{P}_i$ is an element of S^l only if blurring $\vec{P}_i$ with D^l is necessary for interpolating D_i. Each segment S^l is then filtered separately: it is rotated or sheared so that D^l is diagonal, and an existing method for spatially invariant Gaussian filtering can be applied. We choose the permutohedral lattice over the Gaussian KD-tree because it is more efficient for low-dimensional, spatially invariant blurs (its runtime grows linearly with input size instead of log-linearly.) The Gaussian KD-tree's ability to locally vary the query parameters is unnecessary here, so we can afford the loss of flexibility that the lattice

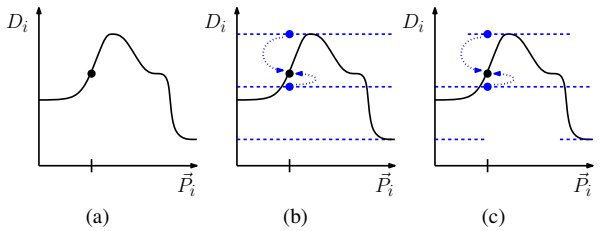


Figure 5: *Kernel sampling.* **Left***: The kernel D_i may vary as a function of the position vector $\vec{P}_i$.* **Middle***: Three samples in the space of possible kernels are chosen, and the entire dataset is blurred with each kernel. Now the result of blurring with a kernel that had not been chosen can be estimated via interpolation. However, the runtime scales linearly with the number of kernel samples.* **Right***: Three samples as chosen as before, but only parts of the dataset necessary for interpolation are blurred with each kernel.*

brings. The total runtime of this approach is equal to that of a spatially invariant Gaussian filter, multiplied by the "complexity" of the scene: the average number of segments to which each position vector belongs.

In order to blur accurately, each data point requires a reasonable spatial support. The assumption of locally constant kernel is again useful; if $\vec{P}_i$ belongs to S^l, it is highly likely that its support does as well. However, this assumption does not hold at segmentation boundaries, and causes artifacts as shown in Figure 6(b).

To alleviate this problem, it is necessary to spatially dilate each segment. Because morphological operators on sparse sets are expensive, we take a sampling approach to bound the runtime: for each $\vec{P}_i$, randomly sample a fixed number k of neighbors within a spatial radius, and append $\vec{P}_i$ to the segments corresponding to these neighbors. (Each neighbor $\vec{P}_j$ has a corresponding kernel D_j. Find the kernel $D^l \in \mathcal{D}$ closest to D_j. Add $\vec{P}_i$ to S^l. See Figure 6(a) for a visualization.) This process ensures that if there are sufficiently large number of neighbors belonging to a particular segment, $\vec{P}_i$ will also be added to the segment to provide spatial support for these neighbors, suppressing seam artifacts. In practice, this dilation has meaningful effects only on those points along a segmentation boundary, and does not increase the total number of points filtered significantly. The method is summarized in Algorithm 3.

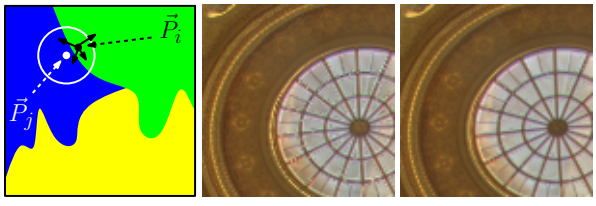


Figure 6: *Spatial dilation by sampling.* **Left***: The colors designate distinct segments. For each $\vec{P}_i$, k random neighbors are chosen, and $\vec{P}_i$ is added to their segments. Therefore, after dilation, $\vec{P}_i$ will likely be included in the segment containing $\vec{P}_j$, and help its filtering.* **Middle, Right***: Insets from tone mapping result with $k = 0$ and $k = 16$, respectively.*

Algorithm 3 Evaluation of Eq. (2) with kernel sampling.

Require: $\vec{P}$, $\vec{V}$.
  Fix a set $\mathcal{D}$ of samples in the kernel space.
  For each $D^l \in \mathcal{D}$, set $S^l \leftarrow \emptyset$
  for all i **do**
    Find samples in $\mathcal{D}$ closest to D_i (for interpolation.)
    For each sample D^l found above, $S^l \leftarrow S^l \bigcup \{\vec{P}_i\}$.
    Find a fixed number k of random position vectors near $\vec{P}_i$.
    for all $\vec{P}_j$ found above **do**
      Let $D^m \in \mathcal{D}$ be the kernel closest to D_j.
      $S^m \leftarrow S^m \bigcup \{\vec{P}_i\}$.
    end for
  end for
  for all $D^l \in D$ **do**
    Filter S^l with the kernel D^l using the permutohedral lattice.
  end for
  for all i **do**
    Calculate $\vec{V'}_i$ by interpolating between the values in the segments of kernels close to D_i.
  end for

3.4 Comparison of Methods

We have described three methods for adopting the existing algorithms for spatially invariant Gaussian filters, and using them to approximate spatially varying Gaussian filters. Table 1 compares their properties. Based on the runtime, as long as the assumptions made are valid, kernel sampling outperforms the others in efficiency.

Method	Import. sampling	Dim. elevation	Kernel sampling
Generality	Yes	Yes	No
Growth with d_p	Exponential	Exponential	Cubic
Growth with dataset	Log-linear	Log-linear	Linear

Table 1: *Comparison of proposed methods. Importance sampling and dimensionality elevation incur the curse of dimensionality. Kernel sampling uses the permutohedral lattice which grows quadraticallly with d_p, and calls it an average of $O(d_p)$ times for each data point, resulting in a cubic growth.*

4 Applications

We now evaluate the utility of the kernel sampling method described above for two trilateral filtering applications: high dynamic range image tone mapping and sparse depth image upsampling. We focus on trilateral applications because spatially varying filters for which the filter aligns with the data tend to be the most immediately useful applications, though our techniques can be applied to accelerate any spatially varying blur. These two specific applications lie in very different domains (photography and vision) and manipulate distinct types of data (dense and unstructured), but can both be addressed easily using our generalized framework.

4.1 Tone Mapping

Tone mapping is a commonly tackled problem in computational photography. Low dynamic range (LDR) display media such as computer monitors or photographic prints are unable to reproduce the full range of brightnesses present in some scenes. Accordingly, any high dynamic range (HDR) image data must be somehow compressed for viewing on the LDR display.

A number of algorithms have been proposed for compressing the dynamic range of such images [Larson et al. 1997; Fattal et al. 2002]. Among these, the bilateral tone mapping algorithm has proven popular [Durand and Dorsey 2002]. This approach represents HDR images as a sum of two layers: an HDR base layer that represents the overall brightness of any particular image region and an LDR detail layer that encodes local texture variations from the base. If we scale down the base layer and then recomposite it with the detail layer, we can achieve a tone mapped result that compresses dynamic range but still maintains image details. The base layer is obtained by filtering the log-of-luminance channel of the HDR image. The bilateral filter works well for this task because it removes detail while respecting strong intensity edges, thus preventing the haloing artifacts one would expect from a plain Gaussian blur.

One consequence of using the bilateral filter, however, is the base layer's tendency to contain piecewise constant brightnesses. Image regions with smooth intensity ramps are flattened in the base layer, effectively accentuating the ramps in the detail layer. The result is an increased presence of blooming artifacts. This problem was first recognized by Choudhury and Tumblin [2003] who showed that the trilateral filter yields better base layers for tone mapping.

In this section, we describe how to adapt the trilateral tone mapping technique for use with our new acceleration method.

Spatial Variation

In this application our goal is to produce a piecewise linear base layer from an HDR image. Accordingly, we would like to filter the image with anisotropic Gaussians kernels oriented along dominant image gradients. To find these gradients, we first compute local gradients on the log-of-luminance channel of the image using simple forward differences. We perform a 4D bilateral filter on the gradients in (x, y, G^1, G^2) to eliminate the effects of local texture. We then threshold the filtered gradients by bilateral filter confidence and inpaint low confidence regions with nearby gradient values.

Implementation

Now that we have identified the source of anisotropy in the dataset, we can apply our framework for kernel sampling and segmentation (Algorithm 3). In order to do so, we must also define these three quantities: D_i, $\mathcal{D}$ and k.

Recall that D_i is given by Eq. (3). By design, it is equivalent to an isotropic bilateral filter on the tilted image $\hat{I}_i(x, y) = I(x, y) - G_i^1 \cdot (x - x_i) - G_i^2 \cdot (y - y_i)$.

We choose the set of sampled kernels $\mathcal{D}$ to correspond to a set of gradients whose orientations and magnitudes are uniformly spaced; while gradients in natural scenes follow a sparse prior, we can nonetheless obtain a good *coverage* of their distribution by this sampling scheme. We experimented with clustering techniques in an attempt to better represent the space of gradients present in the image, but found it increased runtime and did not improve results significantly. For the images shown in the following results section, we found 9 angular bins and 10 magnitude bins to be satisfactory.

The sample count k used in spatial dilation has a small but important effect on filter quality (see Figure 6). For small k, filtered images exhibit visible seams at segment boundaries as these regions have less spatial support for filtering. A large k increases the runtime of the algorithm. We found $k = 16$ to be ideal — larger k increased runtime with little improvement in filter quality.

With these values defined, we apply our kernel sampling technique to extract a base layer from an HDR image. We then scale down the dynamic range of the base layer and recompose it with its derived detail layer to produce a tone mapped image.

(a) Images tone mapped with kernel sampling

(b) Inset of memorial church

(c) Inset of design center

Figure 7: *A comparison of tone mapping techniques.* **Top***: Our tone mapping results for memorial church and design center radiance maps.* **Left Column***: Bilateral tone mapping is computationally inexpensive, but suffers from blooming and false edge artifacts.* **Middle Column***: Our kernel sampling algorithm reduces bloom and edge artifacts.* **Right Column***: A naïve trilateral filter generates good results, but is prohibitively expensive for large datasets. Memorial church radiance map courtesy of Paul Debevec, USC. Design center radiance map courtesy of Byong Mok Oh, MIT.*

Results

We apply our technique to HDR images and compare our results against other filtering techniques: bilateral filtering in (x, y, I); a naïve trilateral filter implemented as a direct evaluation of Gaussian kernels (in a spatial window of radius $2.5\sigma_s$.) Each technique uses the same set of standard deviations.

Insets from the tone mapped memorial church and design center datasets are shown in Figure 7. Table 2 compares the runtimes for the different techniques. As expected, the bilateral filter tone mapped results suffer from blooming and false edge artifacts. While both our implementation and the naïve trilateral filter are free of these problems, our algorithm runs two orders of magnitude faster than the naïve counterpart.

When applied to LDR images, the same pipeline acts as a detail enhancing filter. Figure 8 shows one example of this application.

	Memorial	Design
Bilateral	0.35s	0.50s
Naïve trilateral	312.70s	528.99s
[Choudhury and Tumblin 2003]	41.54s	88.08s
Our kernel sampling (Section 3.3)	5.10s	6.30s

Table 2: *Tone mapping runtimes. Bilateral tone mapping generates results very quickly, but is rife with objectionable artifacts. Our trilateral kernel sampling algorithm faithfully approximates the result of the naïve trilateral implementation, but without its prohibitive time complexity. As Choudhury and Tumblin's algorithm includes automatic parameter selection and other optimizations, visual comparison is difficult; thus we report only its runtime.*

(a) Input (b) Bilateral (c) Kernel sampling

Figure 8: *Detail enhancement of LDR images. Our tone mapping pipeline can also accentuate details in LDR images if we reverse the roles of the detail and base layers—hold the base layer constant and instead scale the detail layer. Note that the soft gradients present in the image are removed, revealing clearer outlines of the organs with our technique compared to the bilateral filter. Image courtesy of JRST Digital Image Database.*

4.2 Sparse Range Image Upsampling

Range images encode scene geometry as the per-pixel distance from a camera to its surrounding objects. Accurate, high resolution range images are useful in domains spanning the operation of autonomous vehicles, background segmentation and novel user interfaces. Typically multiple sensors are deployed for these tasks, including high resolution cameras for image data and low resolution laser range finders for unstructured depth samples. Sensor fusion techniques describe how to combine information from these sources to generate high resolution models of the environment.

The joint bilateral filter has recently gained popularity as one such sensor fusion technique [Yang et al. 2007; Dolson et al. 2010]. In this context, the joint bilateral filter mixes homogenized depth values $\vec{V}_i = (z_i, 1)$ at positions $\vec{P}_i = (x_i, y_i, r_i, g_i, b_i)$ with samples that are nearby in space and color. This is motivated by the supposition that objects with similar color often have similar depths in a scene. The continuous nature of bilateral filters makes it possible to compute the depth at locations $\vec{P}_j$ for which no estimate $\vec{V}_j$ is provided. This produces a depth map at a resolution equal to that of the queried points $\vec{P}_j$.

As we have previously discussed, however, the bilateral filter implicitly assumes a scene with piecewise constant value. This assumption does not hold true for many scenes and can result in artifacts as shown in Figure 9(c). A trilateral upsampling approach, however, assumes that scene depths are piecewise planar, and thus can generate higher quality depth maps.

In this section we discuss how to use our kernel sampling technique for the joint trilateral upsampling of unstructured range data.

Spatial Variation

In this application our goal is to upsample sparse range data with respect to a high resolution color image. We observe that similarly colored pixels often belong to the same locally planar object. Thus, we should vary our Gaussian to align with these local planes to best make use of the information provided by the depth samples.

Our data is sparse and unstructured, however, so we must find an analog to local gradients at each sample point. The most logical approach is to fit a plane in the neighborhood of each sample point (x_i, y_i, z_i). The best-fit plane $z = A_i(x - x_i) + B_i(y - y_i) + C_i$ minimizes the following error function:

$$\sum_j w_j \left(z_j - (A_i(x_j - x_i) + B_i(y_j - y_i) + C_i)\right)^2,$$

where w_j is the weight of the data point (x_j, y_j, z_j). Differentiating with respect to A_i, B_i, C_i and then setting the expressions equal to zero yields a system of linear equations that can be solved easily. If we let the weights w_j be bilateral, then a single pass of the bilateral filter can be used to calculate the coefficients of this linear system for all i simultaneously. Details of our bilateral plane-fitting algorithm are provided in supplement Section C.

In order to use our kernel sampling method, however, we require a dense map of plane coefficients. We generate this map with a joint bilateral upsample of plane coefficient values $\vec{V}_i = (A_i, B_i, 1)$ using position vectors $\vec{P}_i = (x_i, y_i, r_i, g_i, b_i)$. Note that the bilateral filter's piecewise constant assumption is valid here because we are blurring plane coefficients rather than depth values.

Finally, we apply a confidence thresholding and inpainting step to clean up the plane coefficients as we did for HDR image gradients.

Implementation

With the data's anisotropy extracted, fitting the application into our kernel sampling algorithm is mostly a matter of again defining the quantities D_i, $\mathcal{D}$, and k.

The ideal anisotropic kernel D_i for a pixel can be generated in a manner similar to that used in our tone mapping application. D_i for a pixel with plane coefficients A_i, B_i is equivalent to an isotropic bilateral filter on the tilted range image $\hat{Z}_i(x, y) = Z(x, y) - A_i(x - x_i) - B_i(y - y_i)$.

The choice of sampled kernels $\mathcal{D}$ is more complicated than in the tone mapping case, however. The distribution of plane orientations in range data typically exhibits strong asymmetries due to the presence of large planar features like the ground plane. Accordingly, one might suspect that using the most common plane orientations would be more efficient than uniformly sampling the space of orientations as we did for tone mapping. A simple uniform sampling of orientations does indeed "waste" many bins on uncommon plane orientations, but the cost of filtering a segment is linear in the number of data points it contains; thus the total time complexity of the filtering step remains linear in the size of the dataset, regardless of $|\mathcal{D}|$. Having many bins does add overhead, but its effect on runtime is additive, not multiplicative. As a result, we can use a denser sampling of plane orientations without incurring any significant performance penalties. For the depth maps shown in the following section, we use 8 tilt direction bins and 12 tilt magnitude bins.

As in the tone mapping case, the number of spatial dilation samples k determines the visibility of seams between segments. We find $k = 16$ to again be an adequate choice.

Before we can use kernel sampling on this data, we must make a small modification to Algorithm 3 in order to accommodate joint

upsampling. For each depth sample i in a segment, we add it to a permutohedral lattice using a position $\vec{P}_i = (x_i, y_i, r_i, g_i, b_i)$ and tilted depth value $\vec{V}_i = (\hat{Z}(x, y), 1)$. The remaining pixels j in the segment have no depth value, however, so we indicate this by adding values $\vec{V}_j = (0, 0)$ that have zero homogeneous weight corresponding to their positions $\vec{P}_j = (x_j, y_j, r_j, g_j, b_j)$. Filtering each segment propagates depth information from the depth samples to nearby pixels in space and color. We then interpolate depth values across segments to generate a final high resolution depth map.

Results

We first apply this method to synthetic range data in order to analyze its accuracy against ground truth. We also show that our technique generates reasonable depth maps for real-world datasets.

Figure 9 shows our results for the `synthetic1` dataset. To simulate the sparseness of a laser range finder, we randomly select 1% of the pixels from the ground truth depth map to represent our depth samples. Figure 10 shows our results for the real-world `highway2` dataset. The depth maps produced by our algorithm are smoother and more accurate than those produced by simple bilateral upsampling. The bilateral filter flattens the depth values of constant color regions (see the building on the right edge of the scene and the white lane markers), while our trilateral method does not. See the supplemental video for a better illustration.

Table 3 compares the runtimes and relative errors of our upsampling algorithms, along with a naïve version of our kernel sampling technique that does not segment the dataset as in Figure 5(b). As in tone mapping, the bilateral filter is faster but produces lower quality results than kernel sampling. Note that sampling without segmentation amplifies runtime without reducing error significantly.

	`Synthetic1`	`Highway2`
Bilateral	0.59s (2.41%)	1.37s
Kernel sampling	3.71s (0.95%)	9.18s
Kernel sampling (No segment.)	57.90s (0.85%)	131.68s

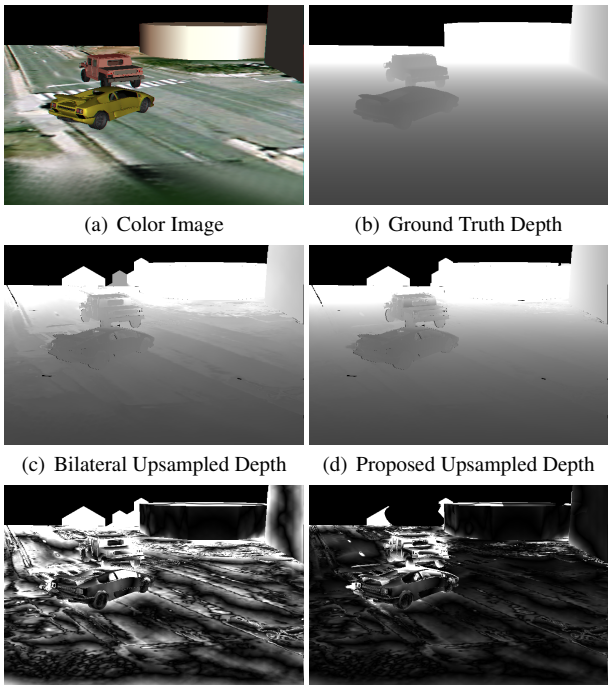
Table 3: *Depth map upsampling runtimes, with average relative error in parentheses.*

4.3 Performance and Parallelization

The applications presented above were implemented for a single serial 3.0 GHz CPU. Parallel GPU implementations for bilateral filtering algorithms exist [Chen et al. 2007; Adams et al. 2009; Adams et al. 2010] and could easily be integrated into our workflow to increase performance. Additionally, because the kernel sampling and segmentation algorithm requires many independent filtering operations, it could be easily integrated into a parallel architecture to yield a yet still greater increase in performance.

5 Limitations and Future Work

The kernel sampling method, while its time complexity grows only polynomially with d_p and linearly with the dataset size, relies on the additional assumptions that the kernel is locally constant (or at the very least, that its spatial variance is graceful). The number of sampled kernels $|\mathcal{D}|$ also affects the resulting quality of the output: while we argue that the runtime is largely unaffected, having too few samples causes the method to degenerate to a spatially invariant Gaussian filter, whereas having too many samples may create segments with too few points each and the dilation to be less effective. Lastly, while our spatially varying technique is more powerful than its spatially invariant counterpart, it remains necessarily slower.



(a) Color Image (b) Ground Truth Depth

(c) Bilateral Upsampled Depth (d) Proposed Upsampled Depth

(e) Bilateral Depth Relative Error (f) Proposed Depth Relative Error

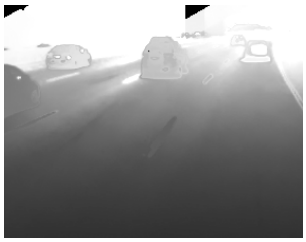
Figure 9: `Synthetic1` *dataset upsampling results.* **Top***: The input image and the corresponding depth map, respectively. In order to simulate the sparse nature of real range data, we discard all but a random 1% of the ground truth depth map.* **Middle***: Bilateral upsampling (left) has a tendency to flatten constant color regions and generates rough results. Our kernel sampling (right) generates smoother depth maps when using equivalent kernel standard deviations.* **Bottom***: The relative error maps show the ratio of absolute depth error to ground truth depth at each pixel (scaled up to show detail). Range data courtesy of Jennifer Dolson, Stanford.*

There are many other domains that could benefit from accelerated spatially varying Gaussian filtering. So long as the underlying signal contains a spatially varying anisotropy or scale component, our technique offers improved filtering quality over the bilateral filter. One such application is reducing photon shot noise in digital imaging. Shot noise varies with signal strength and is particularly prevalent in areas such as astronomy and medicine, so these areas could make use of a fast photon shot denoising filter. Another potential application is video denoising in which we would align blur kernels in the space-time volume with object movement. Light field filtering or upsampling could also be performed by aligning blur kernels with edges in the ray-space hyper-volume.

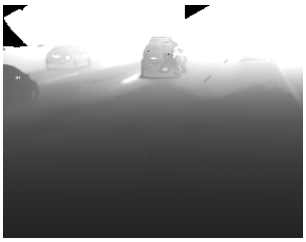
6 Conclusion

In this paper we propose a flexible scheme for accelerating spatially varying high-dimensional Gaussian filters. By segmenting and tilting image data, our algorithm can approximate spatially varying, anisotropic Gaussian kernels while leveraging recent work in isotropic bilateral filter acceleration structures. Our algorithm achieves comparable results to a naïve trilateral filter implementation and is an order of magnitude faster than prior methods. Our algorithm generates significantly better results compared to the bilateral filter for diverse applications, in exchange for a moderate increase in computational complexity. Thus, the primary contribution of this work is to make feasible spatially varying Gaussian filters as an alternative for traditional bilateral filter applications.

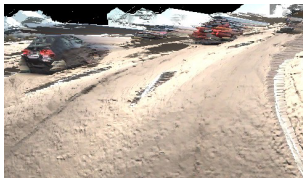
(a) Color Image and Depth Samples



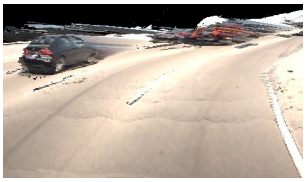
(b) Bilateral Upsampled Depth



(c) Proposed Upsampled Depth



(d) Bilateral Inferred Geometry



(e) Proposed Inferred Geometry

Figure 10: `Highway2` *dataset upsampling results.* **Top***: The input image and range data collected by an instrumented car.* **Middle***: Bilateral upsampling (left) generates a rough depth map that flattens constant color regions such as the white lane markings. Our algorithm (right) produces more reasonable results.* **Bottom***: The geometry inferred from the upsampled depth maps, lit to accentuate surface detail. Range data courtesy of Jennifer Dolson, Stanford.*

Acknowledgements

The authors would like to thank Marc Levoy and the Stanford graphics group for their useful discussions and feedback throughout the course of this work. Jongmin Baek's work on this project was funded in part by the Nokia Research Center in Palo Alto and David E. Jacobs acknowledges support from a Hewlett Packard Fellowship. Finally, we would like to thank the anonymous reviewers for their insightful comments and suggestions for improvement.

References

ADAMS, A., GELFAND, N., DOLSON, J., AND LEVOY, M. 2009. Gaussian KD-trees for fast high-dimensional filtering. *ACM Trans. on Graphics 28*, 3 (Aug.), 21:1–21:12.

ADAMS, A., BAEK, J., AND DAVIS, M. A. 2010. Fast high-dimensional filtering using the permutohedral lattice. In *Proceedings of EUROGRAPHICS 2010*, Eurographics, 753–762.

BUADES, A., COLL, B., AND MOREL, J.-M. 2005. A non-local algorithm for image denoising. *IEEE Conference on Computer Vision and Pattern Recognition 2*, 60–65.

CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. on Graphics 26* (July), 103:1–103:9.

CHOUDHURY, P., AND TUMBLIN, J. 2003. The trilateral filter for high contrast images and meshes. *Eurographics Symposium on Rendering*, 186–196.

DOLSON, J., BAEK, J., PLAGEMANN, C., AND THRUN, S. 2010. Upsampling range data in dynamic environments. *IEEE Conference on Computer Vision and Pattern Recognition*, 1141–1148.

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

EISEMANN, E., AND DURAND, F. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. on Graphics 23* (Aug.), 673–678.

FATTAL, R., LISCHINSKI, D., AND WERMAN, M. 2002. Gradient domain high dynamic range compression. In *Proceedings of SIGGRAPH 2002*, ACM SIGGRAPH, ACM, 249–256.

FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Trans. on Graphics 22*, 3 (July), 950–953.

GENZ, A. 2004. Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Statistics and Computing 14*, 151–160.

GEUSEBROEK, J.-M., AND SMEULDERS, A. W. M. 2003. Fast anisotropic gauss filtering. *IEEE Trans. on Image Processing 12*, 8, 938–943.

GRANADOS, M., AJDIN, B., WAND, M., THEOBALT, C., SEIDEL, H.-P., AND LENSCH, H. 2010. Optimal HDR reconstruction with linear digital cameras. *IEEE Conference on Computer Vision and Pattern Recognition*, 215–222.

LAM, S. Y. M., AND SHI, B. E. 2007. Recursive anisotropic 2-d gaussian filtering based on a triple-axis decomposition. *IEEE Trans. on Image Processing 16*, 7, 1925–1930.

LAMPERT, C. H., AND WIRJADI, O. 2006. An optimal nonorthogonal separation of the anisotropic gaussian convolution filter. *IEEE Trans. on Image Processing 15*, 11, 3502–3514.

LARSON, G. W., RUSHMEIER, H., AND PIATKO, C. 1997. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Trans. on Visualization and Computer Graphics 3*, 291–306.

PARIS, S., AND DURAND, F. 2006. A fast approximation of the bilateral filter using a signal processing approach. In *Proceedings of the European Conference on Computer Vision*, 568–580.

PARIS, S., KORNPROBST, P., TUMBLIN, J., AND DURAND, F. 2008. Bilateral filtering: Theory and applications. *Computer Graphics and Vision*, 1, 1–73.

PETSCHNIGG, G., SZELISKI, R., AGRAWALA, M., AND COHEN, M. 2004. Digital photography with flash and no-flash image pairs. *ACM Trans. on Graphics 23* (Aug.), 664–672.

SHEN, J., FANG, S., ZHAO, H., JIN, X., AND SUN, H. 2009. Fast approximation of trilateral filter for tone mapping using a signal processing approach. *Signal Processing 89*, 901–907.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. *IEEE International Conference on Computer Vision*, 836–846.

YANG, Q., YANG, R., DAVIS, J., AND NISTÉR, D. 2007. Spatial-depth super resolution for range images. *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

ZHANG, B., AND ALLEBACH, J. P. 2008. Adaptive bilateral filter for sharpness enhancement and noise removal. *IEEE Trans. on Image Processing 17*, 5, 664–678.