Volume 29 (2010), Number 1 pp. 60–73 COMPUTER GRAPHICS *forum*

Fast Gauss Bilateral Filtering

Shin Yoshizawa[1], Alexander Belyaev[2], and Hideo Yokota[1]

shin@riken.jp, a.belyaev@hw.ac.uk, and hyokota@riken.jp
[1] RIKEN, 2-1, Hirosawa, Wako, Saitama, 351-0198, Japan
[2] Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, Scotland, United Kingdom

Abstract

In spite of high computational complexity, the bilateral filter and its modifications and extensions have recently become very popular image and shape processing tools. In this paper, we propose a fast and accurate approximation of the bilateral filter. Our approach combines a dimension elevation trick with a Fast Gauss Transform. First we represent the bilateral filter as a convolution in a high dimensional space. Then the convolution is efficiently approximated by using space partitioning and Gaussian function expansions. Advantages of our approach include linear computational complexity, user-specified precision, and an ability to process high dimensional and non-uniformly sampled data. We demonstrate capabilities of the approach by considering its applications to the image and volume denoising and HDR tone mapping problems.

Keywords: bilateral filter, fast Gauss transform (FGT), Yaroslavsky filter, fast image filtering

ACM CCS: Numerical Analysis [G.1.2]: Approximation; Computer Graphics [I.3.3]: Picture/Image Generation; Image Processing and Computer Vision [I.4.3]: Enhancement.

1. Introduction

The bilateral filter presents a natural combination of the Gaussian spatial and tonal filters. The bilateral filter was first proposed by [AW95] and then rediscovered in [SB97] within the so-called SUSAN approach and in [TM98] where its current name was coined. The bilateral filter can be also considered as a very powerful modification of the Yaroslavsky filter [Yar85] in which the spatial convolution with a box function is changed to the spatial convolution with a Gaussian.

Because of its edge-preserving ability and conceptual simplicity, the bilateral filter has become a popular and powerful image and shape processing tool. It has been extended and generalized in several ways [CT03], [BCM05b, BCM05a], [TSP07] and successively used for video enhancement [BM05, MS05], mesh denoising [FDCO03], high-dynamic-range (HDR) image compression [DD02] purposes, and many other image processing and computer graphics applications. See [PKTD07] for a nice overview of properties, extensions, and applications of the bilateral filtering approach.

Mathematically, the classical bilateral filter in its continuous setting is defined as a spatial-tonal normalized convolution

$$I^{\text{new}}(\boldsymbol{x}) = \frac{\int g_1(\boldsymbol{x}-\boldsymbol{y})g_2(I(\boldsymbol{x})-I(\boldsymbol{y}))I(\boldsymbol{y})d\boldsymbol{y}}{\int g_1(\boldsymbol{x}-\boldsymbol{y})g_2(I(\boldsymbol{x})-I(\boldsymbol{y}))d\boldsymbol{y}}, \quad \boldsymbol{y}\in\mathbb{R}^n, \tag{1}$$

applied to an image $I(\boldsymbol{x})$, $\boldsymbol{x}\in\mathbb{R}^n$. Here $g_1(\cdot)$ and $g_2(\cdot)$ are Gaussian kernels used for spatial and tonal filtering, respectively. Often other types of decaying kernels are used in (1) instead of Gaussians.

Previous work on fast bilateral filtering. The nonlinear and nonlocal nature of (1) makes its straightforward implementation computationally expensive: a naive discrete implementation of (1) has $O(N^2)$ computational complexity where N is a number of image elements (pixels/voxels). So various approaches have been proposed for accelerating the bilateral filter.

Noted that a separable convolution with a Gaussian is

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

fast, Pham and Vliet [PvV05] suggested to apply a one-dimensional bilateral filter subsequently to each spatial direction. The resulting filter scheme mimics (1) and is quiet fast. However it often generates undesired artifacts at image regions where image edges are aligned with coordinate axes.

Weiss [Wei06] introduced a 3D histogram based approach for accelerating the 2D bilateral filter with a square box spatial kernel. In a very recent paper [Por08], Porikli has also employed a 3D histogram technique and achieved a remarkably low computational complexity when a box spatial kernel is used. Another achievement of the approach of [Por08] consists of a constant complexity algorithm for a modification of the Yaroslavsky filter with constant spatial kernel (i.e., $g_1 = \text{const}$ in (1)).

Durand and Dorsey [DD02] proposed a fast approximation to the 2D bilateral filter by interpreting (1) as a Gaussian convolution with $g_2(\eta - I(\mathbf{y}))I(\mathbf{y})$ where η is a parameter, computing the convolutions for a sparse set of values of η via 2D Fast Fourier Transforms (FFT), and then linearly interpolating the results. It turns out that the algorithm of [DD02] can be considered as a special case of a more general convolution-based scheme introduced by Paris and Durand in [PD06] where 3D FFT is combined with a downsampling strategy. In [PD09] Paris and Durand extended their approach to color images (bilateral filtering in 5D space). In their bilateral grid approach, Chen et al. [CPD07] combined the convolution-based representation of (1) [PD06] with a GPU parallelization scheme.

Of course, the above-mentioned approximations of the bilateral filter are not free of drawbacks. As noted in [PD09], convolution and 3D grid based schemes may lead to memory overhead when dealing with multidimensional data. The histogram-based approaches of Weiss and Porikli [Wei06, Por08] are also sensitive to image dimensionality and, in addition, do not allow for controlling the approximation error.

Our approach. In this paper, we propose a fast and accurate approximation of the bilateral filter. The method is conceptually simple and combines the dimension elevation trick of [PD06] with Fast Gauss Transform (FGT), a technique for fast and error-controlled computation of a weighted sum of Gaussians. FGT belongs to the family of fast multipole methods which combine clustering and manipulations with truncated series expansions to achieve computationally efficient and accurate approximations of sums of radial basis functions. First we apply the dimension elevation trick of [PD06] and convert the discrete bilateral filter to a normalized weighted sum of Gaussians in a spatial-tonal domain. Then FGT is used for a fast and error-controlled numerical evaluation of the sum. To demonstrate the computational power of our approach we consider a number of applications including image denoising and HDR image tone mapping. The contributions and benefits of our approach can be summarized as follows.

- It has linear computational complexity w.r.t image elements, such as pixels, voxels, etc.
- It allows the user to control speed/accuracy trade-off.
- It is applicable to non-uniformly sampled data.
- It leads to an extremely fast implementation of a version of the Yaroslavsky filter with a spatially constant kernel.
- It turns out to be very efficient for volumetric filtering and HDR image tone mapping tasks.

2. Bilateral Filtering as Convolution

It looks now obvious that bilateral filter (1) can be represented as a spatial-tonal normalized convolution and, therefore, may share many nice properties with linear filtering schemes. To the best of our knowledge, it was first noted by Boomgaard and de Weijer [vdBvdW02] and then extended by Paris and Durand [PD09] who revealed the great power of this seemingly simple observation. As shown below, it is straightforward to extend the Paris-Durand approach to a general multidimensional case.

Observation 1 (nD bilateral filter $\rightarrow$ $(n+1)$D convolution) *The n dimensional bilateral filter is given as a ratio of two $(n+1)$ dimensional convolutions:*

$$J(\boldsymbol{x},u) = \frac{\int_{\mathbb{R}}\int_{\mathbb{R}^n} f(\boldsymbol{y},v)g_1(\boldsymbol{x}-\boldsymbol{y})g_2(u-v)d\boldsymbol{y}dv}{\int_{\mathbb{R}}\int_{\mathbb{R}^n} h(\boldsymbol{y},v)g_1(\boldsymbol{x}-\boldsymbol{y})g_2(u-v)d\boldsymbol{y}dv}, \qquad (2)$$

where $f(\boldsymbol{y},v) = h(\boldsymbol{y},v)I(\boldsymbol{y})$, $h(\boldsymbol{y},v) = \delta(v - I(\boldsymbol{y}))$, $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^n$ are spatial variables and $u,v \in \mathbb{R}$ are tonal ones. Here $\delta(\cdot)$ is the Dirac delta-function.

Substituting $f(\boldsymbol{y},v) = h(\boldsymbol{y},v)I(\boldsymbol{y})$ and $h(\boldsymbol{y},v) = \delta(v - I(\boldsymbol{y}))$ to (2) yields (1) with $I^{\text{new}}(\boldsymbol{x}) = J(\boldsymbol{x},I(\boldsymbol{x}))$.

Although (2) opens an avenue for fast approximations of the bilateral filtering scheme, constructing such approximations is far from being straightforward. At the first glance, a straightforward use of FFT could be an computationally efficient way to evaluate (2). However FFT can not be applied to (2) without a modification because $(\boldsymbol{x},u)$ and $(\boldsymbol{y},v)$ may not be equally spaced in $\mathbb{R}^{n+1}$. In addition, FFT has $O(nN\log N)$ computational complexity in $\mathbb{R}^n$ [FJ05] and, therefore, is not very efficient when dealing with multidimensional data. These difficulties were partially overcome in [PD06] where a FFT-based approach to bilateral filtering was combined with uniform resampling techniques coupled with linear interpolation and downsampling. The bilateral grid [CPD07] also employs uniform resampling for approximating the bilateral filter on GPU.

In contrast to the previous approaches, we use the full power of (2) and obtain a fast and accurate approximation of the nD bilateral filtering.

3. Bilateral Filtering via Gauss Transforms

In this paper we focus on the situation when $g_1(\cdot)$ and $g_2(\cdot)$ of (1) are Gaussian kernels, since it is commonly used in applications. Let $G(\mathbf{x}) = g_1(x)g_2(x) \equiv \exp(-\|\mathbf{x}/\sigma\|^2)$ with

$\mathbf{x} = \{x_1, x_2, ..., x_{n+1}\}$, $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_{n+1}\} \in \mathbb{R}^{n+1}$ and $\|\mathbf{x}/\sigma\|^2 = \sum_{i=1}^{n+1}(x_i/\sigma_i)^2$. The discrete Gauss transform we need for approximating (2) is now defined as follows.

Definition. Consider two point sets: targets $\{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_N\}$ and sources $\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_M\}$ in $\mathbb{R}^{n+1}$. Assume that each source point $\mathbf{s}_j$ is equipped with a positive scalar weight α_j where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_M\}$. The discrete *Gauss transform* of target point $\mathbf{t}_i$ w.r.t. all source points is defined by

$$\mathrm{GT}(\alpha, \mathbf{t}_i, \mathbf{s}, M) \equiv \sum_{j=1}^{M} \alpha_j G(\mathbf{t}_i - \mathbf{s}_j). \tag{3}$$

Now Observation 1 implies that

Observation 2 (*n*D Bilateral to $(n+1)$D Gauss Transform) *The n dimensional bilateral filter with Gaussian kernels is given as a ratio of two $(n+1)$-dimensional Gauss transforms*

$$J(\mathbf{u}) = \frac{\int G(\mathbf{u}-\mathbf{v}) f(\mathbf{v}) d\mathbf{v}}{\int G(\mathbf{u}-\mathbf{v}) d\mathbf{v}}, \quad \mathbf{u}, \mathbf{v} \in \mathbb{R}^{n+1}, \tag{4}$$

where $J(\mathbf{u}) = I^{new}(\mathbf{x})$ is the image resulting after bilateral filtering, $\mathbf{u} = (\mathbf{x}, I(\mathbf{x}))$, and $\mathbf{v} = (\mathbf{y}, I(\mathbf{y}))$.

Let us consider two sets of points in the image space $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$ and $\boldsymbol{y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M\}$. The corresponding spatial-tonal points are given by $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}$ and $\mathbf{v} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_M\}$ where $\mathbf{u}_i = (\boldsymbol{x}_i, I(\boldsymbol{x}_i))$ and $\mathbf{v}_j = (\boldsymbol{y}_j, I(\boldsymbol{y}_j))$. Denote $I(\boldsymbol{y}_j)$ by q_j where $\mathbf{q} = \{q_1, q_2, ..., q_M\}$. We have

$$I^{\mathrm{new}}(\boldsymbol{x}_i) = \frac{\sum_{j=1}^{M} q_j G(\mathbf{u}_i - \mathbf{v}_j)}{\sum_{j=1}^{M} G(\mathbf{u}_i - \mathbf{v}_j)} = \frac{\mathrm{GT}(\mathbf{q}, \mathbf{u}_i, \mathbf{v}, M)}{\mathrm{GT}(1, \mathbf{u}_i, \mathbf{v}, M)} \tag{5}$$

which delivers a discrete approximation of (4).

The Gauss transform and its fast implementations are often used as kernel density estimators in statistics. Relations between kernel density estimators and bilateral filtering was recently studied in [TSP07]. It is interesting to observe that (5) can be also considered as a variant of mean-shift filtering [CM02], see also [vdBvdW02] for relationships between normalized convolution filters, local-mode finding, robust estimators, and mean-shift analysis.

Straightforward computing of (5) for all $\mathbf{x}_i$ requires $O(NM)$ operations. Thus a naive implementation of bilateral filtering of an image with N elements (pixels, voxels, etc.) has $O(N^2)$ complexity.

Setting $g_1(\boldsymbol{x} - \boldsymbol{y}) \equiv 1$ in (1) yields a modification of the Yaroslavsky filter [Yar85] which can be thought as a linear diffusion in the tonal space. The filter was thoroughly studied in [SSN09]. In our notations, it is given by

$$I_Y^{\mathrm{new}}(\boldsymbol{x}_i) = \frac{\mathrm{GT}(\{I(\boldsymbol{y}_j)\}, I(\boldsymbol{x}_i), \{I(\boldsymbol{y}_j)\}, M))}{\mathrm{GT}(1, I(\boldsymbol{x}_i), \{I(\boldsymbol{y}_j)\}, M))}, \tag{6}$$

where $\{I(\boldsymbol{y}_j)\} = \{I(\boldsymbol{y}_1), I(\boldsymbol{y}_2), ..., I(\boldsymbol{y}_M)\}$. We call it SC-Yaroslavsky filter since it has a spatially-constant kernel. For any dimensionality n, (6) consists of only one-dimensional Gauss transforms and, therefore, allows for an accurate and extremely fast implementation, see Section 6 for details. In addition we will demonstrate that (6) is very useful for denoising and segmenting applications in image and volume processing.

4. Fast Gauss Transform

The Fast Gauss Transform (FGT) method we employ is derived from a more general fast multipole method [GR87] adapted for dealing with Gaussian kernels. FGT was introduced in [GS91] for rapid evaluations of sums of Gaussians. Since then, FGT has been improved in terms of accuracy [GS98, BR02, WK06] and memory efficiency [GM00, YDGD03, LGM06]. Besides numerous applications in physics and computational mathematics, FGT has been used in the fields of image processing and computer vision. In particular FGT was applied for image segmentation [YDGD03] and object tracking [EDD03] purposes.

Algorithm 1 Fast Gauss Transform

Subroutine: $\mathrm{FGT}(n, \sigma, \{\alpha\}, \{\mathbf{t}\}, \{\mathbf{s}\}, N, M, \varepsilon)$
Inputs: error ε, dimension n, bandwidth parameters $\sigma \in \mathbb{R}^n$, targets $\{\mathbf{t}\}$, sources $\{\mathbf{s}\}$, number of targets N, number of sources M, and scalar weights $\{\alpha\}$.
Inside Parameters: box length w and interaction region radius r.
Outputs: a scalar set $\{\beta\}$.
Require: $\varepsilon > 0$ and $\alpha_j \in \{\alpha\} \geq 0$.

```
Set up box partitions with side length w(σ) for s and t.
Compute the number of kept terms p from ε via an error
    estimator (we use that derived in [WK06]).
for all source boxes. do
    for k = 0 to p do
        Compute the Hermite expansion coefficients A_k.
    end for
end for
for all target boxes i . do
    for all interaction region within (2r+1)^n nearest
    boxes. do
        for k = 0 to p do
            Compute the Taylor expansion coefficients B_i
            with A_k.
        end for
    end for
end for
for all sources do
    Evaluate the Taylor expansion.
    for k = 0 to p do
        Summing up the Taylor series with B_k.
    end for
    Store the sum to a scalar set {β}.
end for
Return {β}.
```

A full description of FGT is beyond the scope of this paper

and we give only a brief overview of how FGT works. The Fast Multipole Method strategy for evaluating the sum of Gaussians (3) includes using far-field and near-field asymptotic expansions for $G(\mathbf{t}-\mathbf{s})$. For the sake of simplicity, we consider the one-dimensional case since multi-index notations allow for treating the multidimensional case in a very similar way. A far-field expansion centered at s_0 for 1D Gaussian $G(t-s)$ has the form

$$\sum_{k=0}^{\infty}[\frac{1}{k!}(\frac{s-s_0}{\sigma})^k]h_k(\frac{t-s_0}{\sigma}) = \sum_{k=0}^{\infty}A_k h_k(\frac{t-s_0}{\sigma}),$$

where $h_k(x) = (-1)^k \frac{d^k}{dx^k}\exp(-x^2)$ are Hermite functions. Interchanging t and s treats this formula as the Taylor series expansion centered at a nearby target t_0 (near-field expansion)

$$G(t-s) = \sum_{l=0}^{\infty}[\frac{1}{l!}h_l(\frac{s-t_0}{\sigma})](\frac{t-t_0}{\sigma})^l = \sum_{l=0}^{\infty}B_l(\frac{t-t_0}{\sigma})^l,$$

$$B_l = \frac{(-1)^l}{l!}\sum_{k=0}^{\infty}A_k h_{k+l}(\frac{s_0-t_0}{\sigma}).$$

Since the Gaussian falls off quickly, only a limited number of terms, say first p terms, in the above expansions are needed for evaluating the sum of Gaussians (3) with a given precision ε.

The original FGT method starts from constructing a space partition consisting of regular boxes parallel to the coordinate axes in $\mathbb{R}^n$. Then the targets and sources are assigned to these boxes. For each source box, it requires $O(p^n N)$ operations to compute the coefficients A_k corresponding to p-truncated expansions. For each target box, all the Hermite expansions in the source boxes within an interaction region are transformed into a Taylor series expansion to be used inside the target box. The sources within $(2r+1)^n$ nearest boxes are sufficient to obtain single ($r=4$) and double ($r=6$) precisions [GS98]. Computing B_l involves $O(np^{n+1})$ operations. This leads to $O((2r+1)^n np^{n+1})$ complexity per each target box if $(2r+1)^n$ interaction regions are involved. Finally, for each target, evaluating the Taylor series expansion takes $O(p^n M)$ operations.

The FGT approximation error consists of two origins: errors due ignoring sources that are far form a given target and truncation errors. The first component of the approximation error is easy to control by choosing a proper value for parameter r, as discussed above. Controlling the second component is based on appropriate error bounds for truncated expansions. In this paper, we use a new and sharp error estimate derived in [WK06].

Algorithm 1 describes a FGT pseudo-code which reduces the computational complexity of evaluating (3) for all targets $\mathbf{t}_i$ from $O(NM)$ to $O(M+N)$. Given a specified precision ε, the algorithm starts from determining the number of kept terms p. We refer to [GS91, GS98] and rest of this paper for further technical details.

The final FGT approximation accuracy is given by

$$\varepsilon\sum|\alpha_j|, \tag{7}$$

where $\{\alpha_j\}$ are the coefficients in (3). The upper bound (7) corresponds to the worst-case scenario. In our experiments with the FGT-based approximation of the bilateral filter, we have found out that the real error is much smaller.

Writing a good implementation of FGT is a difficult task. Several implementations of FGT and its modifications and improvements [BR02, GM00, YDGD03, MSR*08] are available on the web [IM03, MRY*08]. However after testing some of these free-available codes we decided to write our own implementation of FGT in order to have a full control over our FGT-based image filtering schemes. In our implementation we follow [GS91, GS98] with a corrected and improved error estimate as in [WK06].

5. $O(N)$ Algorithms

Now Algorithm 1 and Observation 2 lead to $O(N)$ implementations of the bilateral and SC-Yaroslavsky filters, as described by Algorithms 2 and 3, respectively. In contrast to [Por08] and [AGDL09] our FGT-based approximations are error-controllable.

Algorithm 2 $O(N)$ Bilateral Filter

Input: error ε, dimensionality n, spatial bandwidths $\sigma^{g_1} \in \mathbb{R}^n$, tonal bandwidth $\sigma^{g_2} \in \mathbb{R}$, targets $\{\mathbf{x}\}$, sources $\{\mathbf{y}\}$, number of targets N, number of sources M, scalar image intensity $\{I(\mathbf{x})\}$.
Called Functions: FGT$(\cdot)$.
Output: $\{I^{\text{new}}(\mathbf{x})\}$.
Require: $\varepsilon > 0$, $\exists I(\mathbf{x}) \geq 0$, $\exists I(\mathbf{y}) \geq 0$.
1: $\{q\} \leftarrow \{I(\mathbf{y}_1), I(\mathbf{y}_2), ..., I(\mathbf{y}_M)\}, \mathbf{y}_j \in \mathbf{y}$.
2: $\{1\} \leftarrow \{1, 1, ...1\}$.
3: $\{\mathbf{u}\} \leftarrow \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\} : \mathbf{u} \ni \mathbf{u}_i \leftarrow (\mathbf{x}_i, I(\mathbf{x}_i))$.
4: $\{\mathbf{v}\} \leftarrow \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_M\} : \mathbf{v} \ni \mathbf{v}_j \leftarrow (\mathbf{y}_j, I(\mathbf{y}_j))$.
5: $\sigma \leftarrow (\sigma^{g_1}, \sigma^{g_2})$.
6: $\{f_i\} \leftarrow \text{FGT}((n+1), \sigma, \{q\}, \{\mathbf{u}\}, \{\mathbf{v}\}, N, M, \varepsilon)$.
7: $\{g_i\} \leftarrow \text{FGT}((n+1), \sigma, \{1\}, \{\mathbf{u}\}, \{\mathbf{v}\}, N, M, \varepsilon)$.
8: **for** $i = 1$ to N **do**
9: $\quad I^{\text{new}}(\mathbf{x}_i) \leftarrow \frac{f_i}{g_i}$.
10: **end for**
11: **Return** $\{I^{\text{new}}(\mathbf{x})\} \leftarrow \{I^{\text{new}}(\mathbf{x}_1), I^{\text{new}}(\mathbf{x}_2), ...I^{\text{new}}(\mathbf{x}_N)\}$.

6. Results

Our implementation of FGT is based on the original approach of Greengard and Strain [GS91, GS98] and, therefore, is especially efficient in dealing with low dimensional data. For high dimensional data, one should probably use the so-called IFGT, Improved Fast Gauss Transform, which has a number of benefits over FGT [YDGD03, MSR*08]. However our own experiments with IFGT were not very encouraging.

Algorithm 3 $O(N)$ SC-Yaroslavsky Filter

Input: error ε, tonal bandwidth $\sigma^{g_2} \in \mathbb{R}$, number of targets N, number sources M, scalar image intensity $\{I(\mathbf{x})\}$.
Called Functions: FGT$(\cdot)$.
Output: $\{I_Y^{\text{new}}(\mathbf{x})\}$.
Require: $\varepsilon > 0$, $\exists I(\mathbf{x}) \geq 0$, and $\exists I(\mathbf{y}) \geq 0$.

```
{I(x)} ← {I(x_1), I(x_2), ..., I(x_N)}, x_i ∈ x.
{I(y)} ← {I(y_1), I(y_2), ..., I(y_M)}, y_j ∈ y.
{1} ← {1, 1, ...1}.
{f_i} ← FGT(1, σ^{g_2}, {I(y)}, {I(x)}, {I(y)}, N, M, ε))
{g_i} ← FGT(1, σ^{g_2}, {1}, {I(x)}, {I(y)}, N, M, ε))
for i = 1 to N do
    I_Y^new(x_i) ← f_i / g_i.
end for
Return {I_Y^new(x)} ← {I_Y^new(x_1), I_Y^new(x_2), ... I_Y^new(x_N)}.
```

Given an image $y = I(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$, $n = 2$ or 3, let us recall that the Gaussian kernel in (3) is given by $G(\mathbf{x}) = g_1(\boldsymbol{x})g_2(x) \equiv \exp(-\|\mathbf{x}/\sigma\|^2)$ with vector of bandwidth parameters $\sigma = [\sigma^{g_1}, \sigma^{g_2}] = [\sigma_1, \sigma_2, ..., \sigma_{n+1}] \in \mathbb{R}^{n+1}$. Assume that the source points belong to the bounding box $[0, \gamma_1] \times \cdots \times [0, \gamma_{n+1}]$. In our experiments with FGT, we employ the bounding box $[0, w_1] \times \cdots \times [0, w_{n+1}]$ with $w_i = \frac{\sigma_i}{\sqrt{2}\gamma_i}$ and use $\varepsilon = \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$ and interaction region radii $r = \{2, 3, 4, 5, 6\}$. At the first glance, in view of (7) it does not seem correct to use large epsilon values. However, as noted before, the FGT accuracy upper bound (7) represents the worst-case scenario and that scenario is not realized for typical images (e.g., for a typical image, its intensity distribution is not concentrated on a small set of points). In our numerical experiments, even setting $\varepsilon = 10^3$ leads to a reasonably good FGT-based approximation of the bilateral filter.

We will also need the standard deviation s_d of 1D distribution of the intensity values $\{I(\boldsymbol{x})\}$.

Error metrics. Given an original image $I(\boldsymbol{x})$, denote by $I^e(\boldsymbol{x})$ and $I^a(\boldsymbol{x})$ the images obtained after applying an exact filtering scheme (we consider the bilateral filter and its variations) and an approximation of the scheme (e.g., our FGT-based approximations), respectively. Let $d_j = I_j^e - I_j^a$ be the difference between the exact and approximated bilateral filtering results, where the subindex $j = 1, 2, ..., N$ enumerates the image elements (pixels, voxels, etc.). In addition to the method noise $d(\boldsymbol{x})$, the following error metrics are used for evaluating our fast bilateral filter:

$$L^1 = \frac{1}{N}\sum_{j=1}^{N}|d_j|, \quad L^\infty = \max_j(|d_j|), \quad \text{ARE} = \frac{1}{N}\sum_{j=1}^{N}\frac{|d_j|}{I_j^e},$$
$$\text{MRE} = \max_j(|d_j|/I_j^e), \quad \text{MNR} = |\max_j(d_j) - \min_j(d_j)|,$$

where the abbreviations ARE, and MRE stand for average and maximum relative errors, respectively. We also employ PSNR, the peak signal-to-noise ratio (the larger, the better), used for similar purposes in [PD06, Por08]. Here

$$\text{PSNR} = -10\log_{10}\left(\frac{1}{N}\sum_{j=1}^{N}\left(\frac{d_j}{\max_j(I_j^e, I_j^a)}\right)^2\right).$$

Timing and accuracy. The computation time is measured in seconds (s), minutes (**m**), and hours (**h**). All our numerical experiments reported in this paper were performed on a Core2 Extreme X9770 (3.2 GHz quad core, no parallelization was used) PC with 16GB RAM and 64 bit OS. We compare our FGT-based bilateral filter with the recent state-of-the-art CPU-based fast bilateral filtering schemes: the histogram-based approach of [Por08], the FFT-based fast bilateral filter of [PD06], and the separable bilateral filter of [PvV05]. We use the program codes available from the authors of [PD06] and from the web [Era08] for [Por08]. The Gaussian functions used in these codes are appropriately modified to coincide with $G(\mathbf{x})$ defined in Section 3.

Figure 1: *Left: Lena image consisting of* 512×512 *pixels. Right: after applying the exact bilateral filter with* $\sigma^{g_2} = (16, 16)$ *and* $\sigma^{g_1} = 51$*; computational time:* 58 ***m***.

We start our numerical experiments from the famous Lena image: Figure 1 presents the original image and an exact bilateral filtering result. Figure 2 delivers the visual, method noise, and PSNR comparisons of our FGT-based method with those developed in [PvV05], [Por08], and [PD06]. We set the number of bins equal to 256 in our evaluation of the approach of [Por08] and, as a ground truth benchmark, use exact bilateral filtering of the Lena image with a constant spatial kernel (it takes 9.8s to compute) instead of the right image of Figure 1. Following notations adopted in [PD06, PD09], we use s_s and r_s to denote the space (spatial) and range (tonal) sampling rates. The timing and errors measurements corresponding to Figure 2 are given in Table 1. Figures 3, 5, and 6 demonstrate speed and accuracy advantages of our method over the approaches developed in [PvV05], [PD06], and [Por08] for varying bandwidth parameters. The graphs in Figures 3, 5, and 6 correspond to [PvV05] (red), [Por08] (green), [PD06] with (1,1) abbreviating $(s_s, r_s) = (\sigma^{g_1}, \sigma^{g_2})$ (blue) and (0.1,0.1) staying for $(s_s, r_s) = (0.1\sigma^{g_1}, 0.1\sigma^{g_2})$ (pink), and our method where (10,2) means $(\varepsilon, r) = (10, 2)$ (orange), and (1,6) stays for $(\varepsilon, r) = (1, 6)$ (sky-blue). We do not include detailed graphs for the ARE and L^∞ metrics because they are similar to the

graphs for L^1 and MRE error metrics, respectively. Table 2 presents a computational time comparison of the methods for various image sizes (we use the same the enumeration notations and parameter settings as in Figure 2). Table 3 summaries the bilateral filter approximation error results for the Lena image. As demonstrated by the top image of Figure 6, the relation between the computational time and accuracy (we use L^1 metric in this case) for our method differ for different pairs of the bandwidth parameters. This is so because, for given bandwidth parameters, our FGT-based algorithm automatically adapts the necessary computational cost in order to satisfy a given precision.

Figure 7 presents visual and method noise comparisons of our method with that of [PvV05] for a volume filtering task. The corresponding timing results and error measurements are presented in Tables 4 and 6, respectively, for various volume sizes. Table 5 demonstrates how fast our method in processing large size volumetric images (512^3 voxels) for various sets of the bandwidth parameters. According to our estimates, a straightforward and exact bilateral filtering of a volume dataset consisting of 512^3 would take approximately **31 years** of computational time.

When σ^{g_1} is small, our method requires greater computational time to compare with middle and large values of σ^{g_1} for the same approximation accuracy.

As mentioned before, we have examined the fast bilateral filter of [PD06, PD09] with two parameter settings, $(s_s, r_s) = (\sigma^{g_1}, \sigma^{g_2})$ and $(s_s, r_s) = (0.1\sigma^{g_1}, 0.1\sigma^{g_2})$ (the image intensity range is supposed to be rescaled to $[0,1]$). Although the first setting is recommended by the authors, we decided to test the second one since it delivers a much more accurate approximation to the bilateral filter. In our experiments with the histogram-based approach of [Por08], we use the 256 bins, although following numbers of bins: 16, 32, and 64 are recommended by the author. Obviously the more bins are used, the more accurate approximation is achieved. In all numerical experiments presented in this paper, our method with $(\varepsilon, r) = (1,6)$ delivers a more accurate approximation of the bilateral filter w.r.t. L^1, L^∞, MNR, and PSNR error metrics than the methods of [PD06, PD09] and [Por08] with the highest accuracy settings. Setting $(\varepsilon, r) = (10,2)$ gives approximately the same accuracy as the above mentioned methods with the highest accuracy settings w.r.t. L^1 and PSNR metrics but our method is much faster. In addition, those highest accuracy settings may cause memory overflows when dealing with large-size images, as indicated by (b) and (d) of Table 2.

Table 7 presents the timing results for our fast SC-Yaroslavsky image and volume filters. Since the filter is based on 1D FGT which is very accurate [LKdF05], we do not include here a detail error analysis in this case and provide the reader with a typical example: for the Engine volumetric image consisting of 64^3 voxels, our FGT-based SC-Yaroslavsky filtering scheme $(\varepsilon, r) = (1,6)$ takes only 0.05s to compute with high accuracy w.r.t. the error metrics employed ($L^1 = 0.0098$, $L^\infty = 1.27$, ARE $= 0.0016$, and MRE $= 0.02$).

Overall, the numerical experiments show that our approach leads to fast and accurate bilateral and SC-Yaroslavsky image and volume filtering. For example, our fast bilateral filter processes 116K (71K), 474K (541K), 467K (812K), and 873K voxels (pixels) per second in average when (ε, r) is set to (1,6), (10,2), (10^2,2), and (10^4,2), respectively. Our fast SC-Yaroslavsky filter with $(\varepsilon, r) = (1,6)$ is highly accurate and processes images and volumes at high speed of 5.28M pixels/voxels per second.

The plots of Figure 4 illustrate timing and accuracy of our methods w.r.t. the FGT parameters (ε, r) and can serve as a guidance for choosing these parameters. It is well known that the $r = 4$ and $r = 6$ are sufficient to obtain single and double error precisions [GS98]. It is rather unexpected that using $r = 2$ delivers quite satisfactory results as demonstrated by Tables 1-5 and in Figures 2-8 and 10-12. We refer to [LKdF05] for an empirical study of how the computational cost depends on the FGT accuracy ε.

7. Applications

Noise reduction. The bilateral filter is well known as a powerful denoising tool. Thanks to our FGT-based fast bilateral filtering scheme, we can use it iteratively in real time. In our experiments with image denoising by iterative bilateral filtering, we set σ^{g_2} proportional to the standard deviation of the image intensity s_d which is updated after each iteration. Dealing with color images, we process each color channel separately. For volume denoising, the same volume rendering transfer function and pseudo color are used for visualizing noisy and corresponding denoised volumetric images. Figures 13 and 8 demonstrate how well our fast bilateral filter applied iteratively performs image and volume denoising.

Figure 9 demonstrates 3D image processing with our fast SC-Yaroslavsky volumetric filter applied iteratively. It reveals hidden image structures and seems useful for 3D image segmentation and feature extraction purposes.

HDR tone mapping. Our fast bilateral filter is useful for high-dynamic-range (HDR) tone mapping. We have combined the HDR image displaying technique of [DD02] with our FGT-based filter and extended the technique to 3D volumetric images. In our numerical experiments, we have used a part of the HDR image displaying code available from the authors of [DD02]. Figures 10, 11, and 14 demonstrate the results of HDR image tone mapping based on our fast bilateral filtering scheme with $(\varepsilon, r) = (10^2, 2)$. In Figure 12, a 3D hurricane velocity image is used to demonstrate the potential of our FGT-based bilateral filter for HDR processing of volumetric data (in this example, we set $(\varepsilon, r) = (10^4, 2)$ and $c_p = 50$ where the latter is the contrast parameter described in [DD02]).

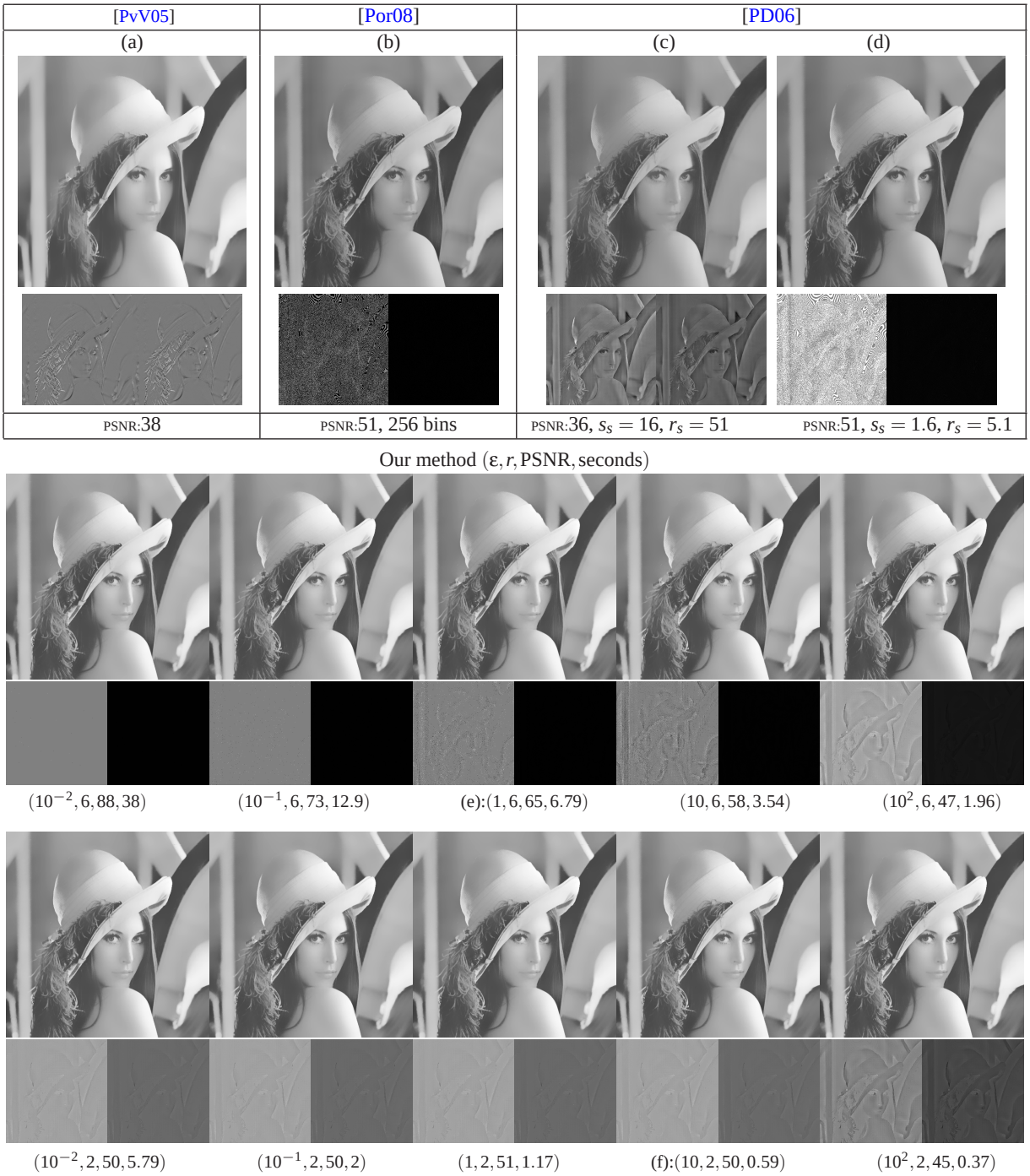

Figure 2: *A visual comparison of the fast bilateral filtering schemes of [PvV05] (a), [Por08] (b), [PD06] (c,d), and our FGT-based method (e,f). Here s_s and r_s are the space and range sampling rates for the fast bilateral filter of [PD06]. The bandwidth parameters are those used in Figure 1. For each experiment, the lower-right image represents the method-noise image, the differences between the exact filtering result and its approximation. The lower-left images are the method-noise images after rescaling in order to fit the standard $0-255$ intensity range. See also Table 1 for the method-noise intensity ranges (MNR) before rescaling, timings, and approximation errors.*

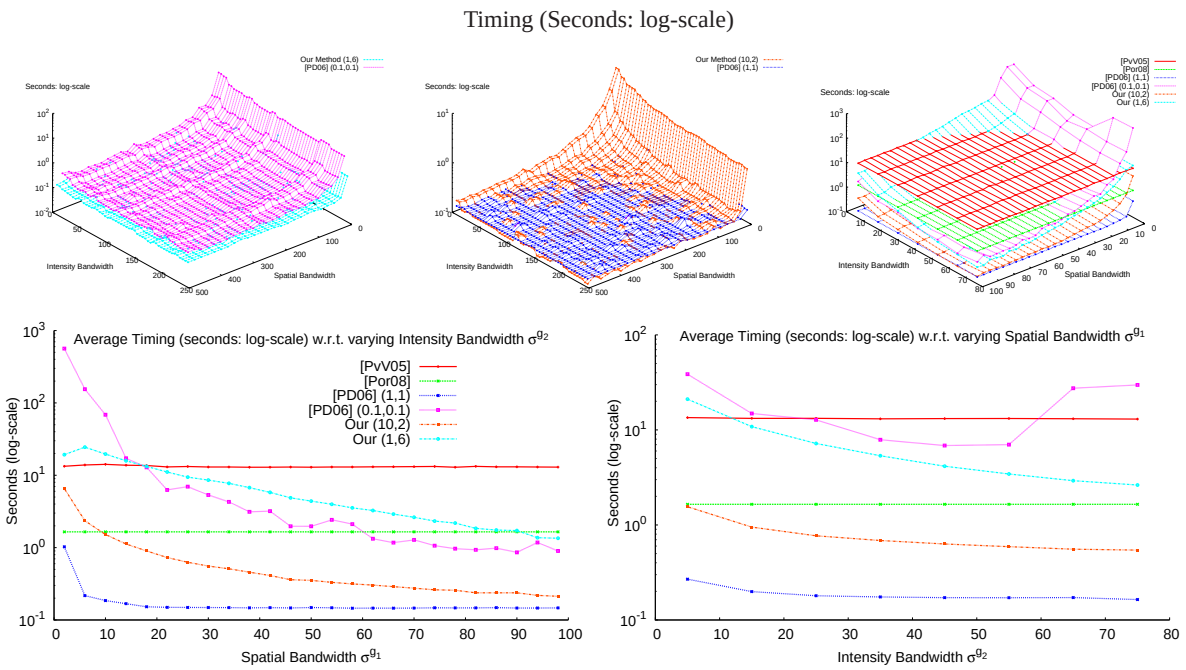

Figure 3: *Timing comparisons of the exact bilateral filter and its approximations applied to the Lena image (*512×512 *pixels). Varying bandwidth parameters are used. The upper images present a timing comparison of our method and with those developed in [PvV05], [PD06], and [Por08] in bandwidth parameter domains. The lower-left and lower-right images are the detailed 2D plots of computational time averaged w.r.t.* σ^{g_2} *against* σ^{g_1} *and vice versa, respectively.*

	Timing	L^1	L^∞	ARE	MRE	MNR
(a)	14.7s	1.47	39	0.0129	0.3659	75
(b)	1.59s	0.56	2	0.0046	0.0178	2
(c)	0.16s	2.98	33	0.025	0.3	60
(d)	8.2s	0.5	35	0.004	0.018	2
(e)	6.79s	0.02	1	0.0001	0.018	2
(f)	0.59s	0.395	31	0.0033	0.215	49

Table 1: *Timing and approximation errors corresponding to Figure 2.*

size	256^2	512^2	1024^2	2048^2	4096^2	8192^2
(a)	1.6	13.2	109	877	7235	69828
(b)	0.34	1.59	6.46	26.98	N/A	N/A
(c)	0.04	0.14	0.6	2.3	9.2	37.5
(d)	2.7	7.95	38.03	134.2	N/A	N/A
(e)	1.92	6.64	17.8	50.9	149.6	679.7
(f)	0.18	0.65	1.8	6	22.9	144.5

Table 2: *Computation times (in seconds) for the Lena image in various resolutions. Labels (a)-(f) refer to the methods and parameter settings used for Figure 2.*

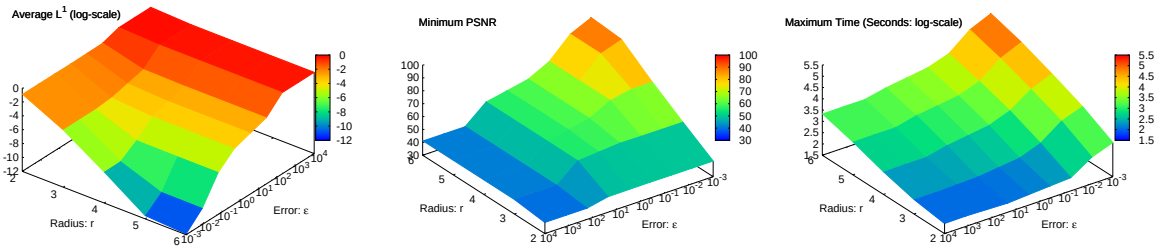

Figure 4: *The plots show the variation of* L^1 *error (left, log-scaling is applied), PSNR (center), and computation time (right) as functions of the FGT parameters* (ε, r)*. We present averaged* L^1 *error, minimal PSNR, and maximal computational time w.r.t. the bandwidth parameters* σ^{g_1} *and* σ^{g_2}*.*

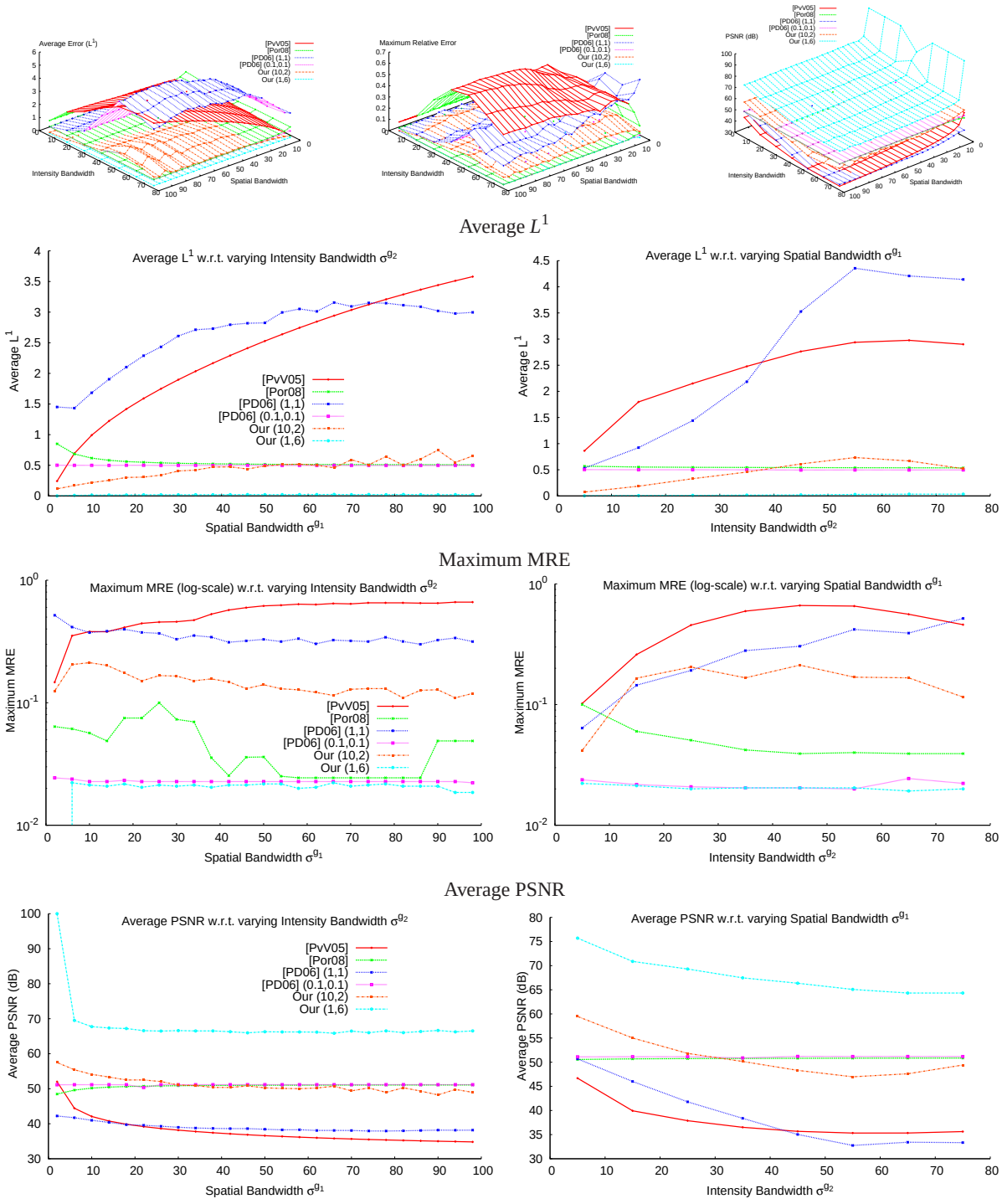

Figure 5: *A comparison of the fast bilateral filtering schemes considered in this paper on the Lena image. The top images present plots of L^1 error (left), MRE error (center) and PSNR (right) against the bandwidth parameters $(\sigma^{g_1}, \sigma^{g_2}) \in [0,100] \times [0,75]$. The remaining six images demonstrate dependence of average L^1 error, maximal MRE error and average PSNR w.r.t one bandwidth parameter against another bandwidth parameter.*

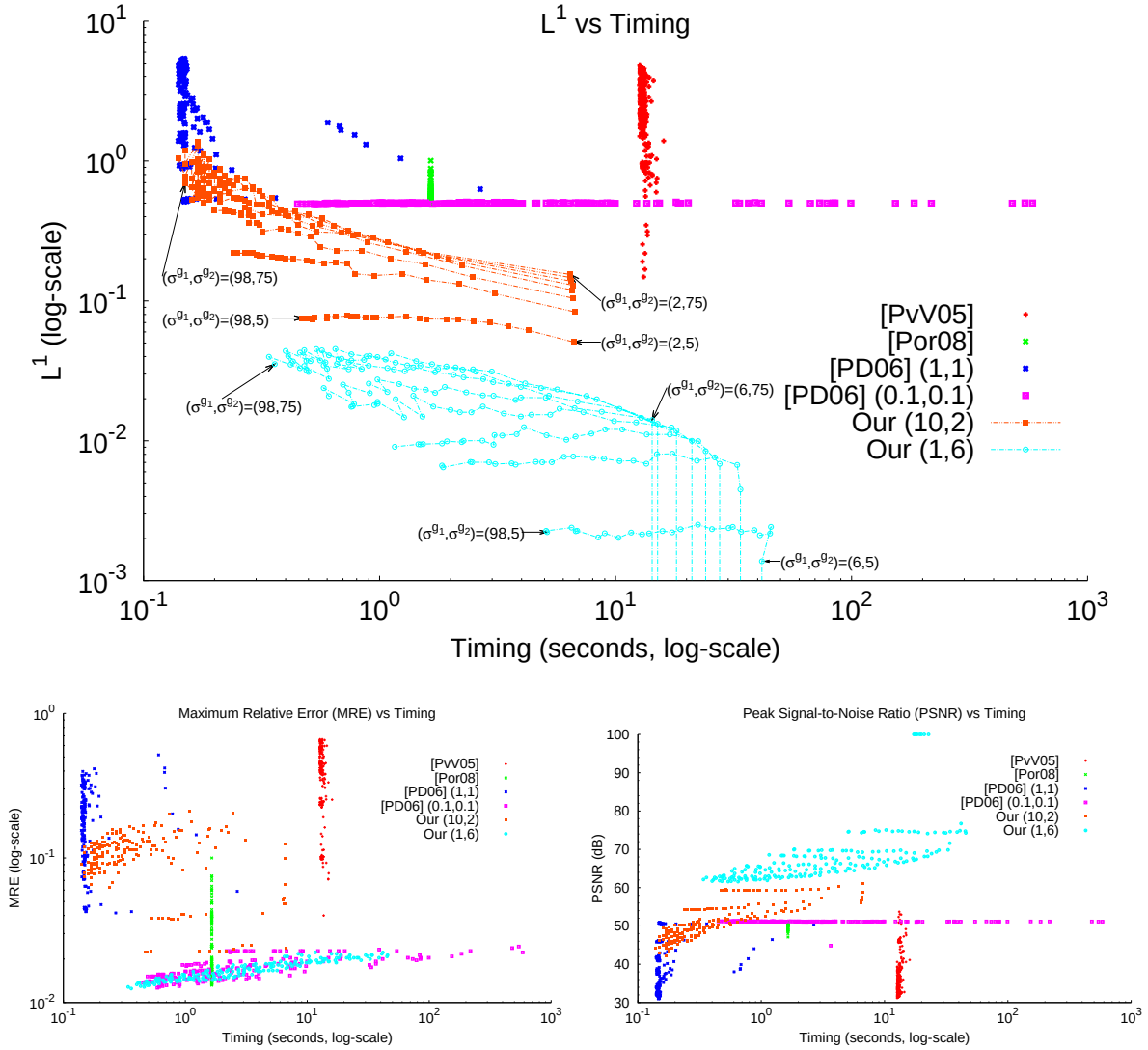

Figure 6: *Error vs timing comparisons (the Lena image is used) for varying bandwidth parameters:* $(\sigma^{g_1},\sigma^{g_2}) \leq (100,75)$. *$L^1$ error (top), MRE error (bottom-left), and PSNR (bottom-right) are considered.*

	[PvV05]					[Por08]					[PD06] (1,1)				
	L^1	L^∞	ARE	MRE	PSNR	L^1	L^∞	ARE	MRE	PSNR	L^1	L^∞	ARE	MRE	PSNR
Max.	4.9	55.0	0.041	0.66	Min. 31	1	8	0.0088	0.1	Min. 47	5.38	46	0.048	0.52	Min. 31
Ave.	2.36	36.8	0.02	0.39	37	0.55	2.24	0.0045	0.024	51	2.66	20	0.022	0.19	38
	[PD06] (0.1,0.1)					Our method (10,2)					Our method (1,6)				
	L^1	L^∞	ARE	MRE	PSNR	L^1	L^∞	ARE	MRE	PSNR	L^1	L^∞	ARE	MRE	PSNR
Max.	0.51	2	0.0044	0.024	Min. 51	1.37	31	0.012	0.21	Min. 42	0.045	1	0.00035	0.022	Min. 62
Ave.	0.5	1	0.004	0.0173	51	0.448	14.5	0.0037	0.1	51	0.019	0.96	0.00015	0.016	68

Table 3: *The average and maximal L^1, L^∞, ARE, MRE errors and the average and minimal PSNR w.r.t. bandwidths* $(\sigma^{g_1},\sigma^{g_2}) \leq (100,75)$ *for the fast bilateral filtering schemes tested in this paper. The Lena image is used for this comparison.*

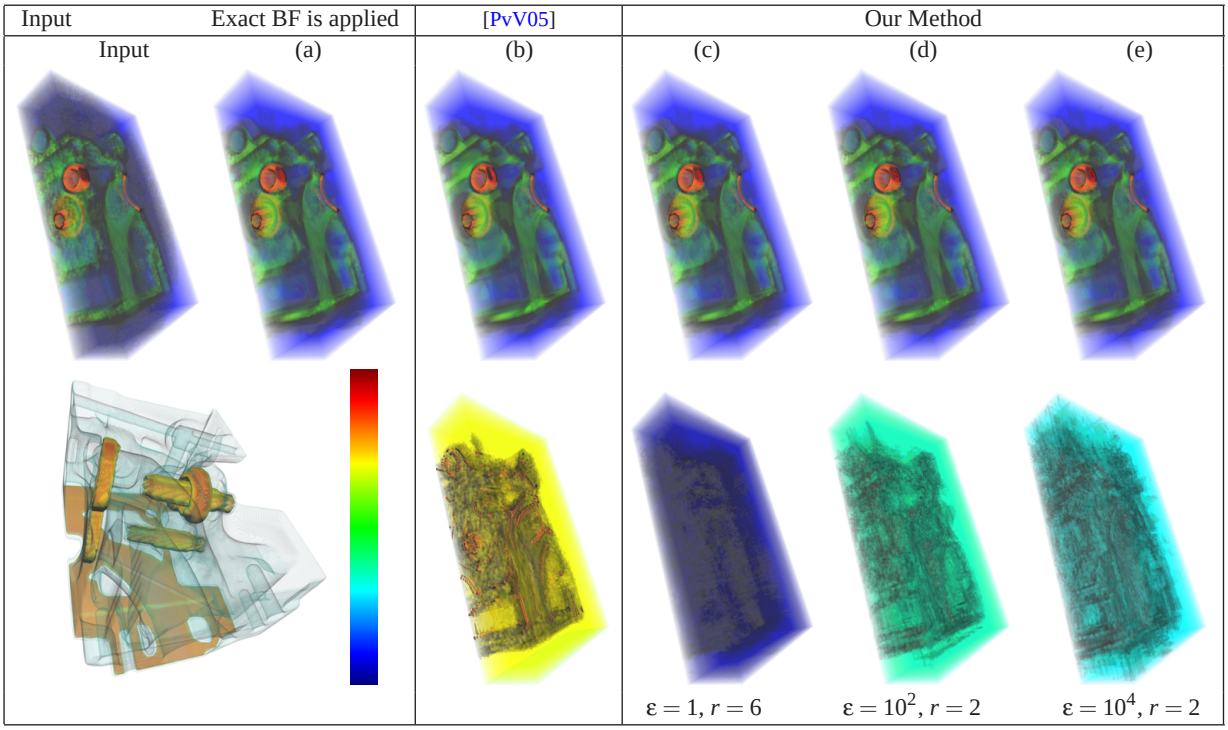

Figure 7: *Left section: an input Engine 3D volumetric image consisting of* 128^3 *voxels is shown with aspect ratio* $1:1:0.5$ *(bottom-left), we use a part of the Engine image (top-left) in this comparison, the same part after exact bilateral filtering with* $\sigma^{g_1}=(5,5,10)$ *and* $\sigma^{g_1}=0.5s_d$ *(top-right), we use a coloring scheme represented by a pseudo color bar (bottom-right) with its lower end (blue) corresponding to* 0 *and its upper (red) corresponding to* 255*. Middle section: approximate bilateral filtering according to [PvV05] is applied (top), method noise (bottom) shows the difference between the exact and approximate bilateral filtering results. Right section: the results our FGT-based method tested for various setting of parameters* (ε,r) *(top), method noise results for the same parameter settings. See Tables 4 and 6 for the corresponding computational times and approximation errors, respectively.*

size	16^3	32^3	64^3	128^3	256^3	512^3
(a)	0.86s	59s	1.03**h**	66.7**h**	N/A	N/A
(b)	0.01s	0.16s	2.7s	46s	13**m**	3.6**h**
(c)	3.3s	9.2s	2.7**m**	1.4**h**	5**h**	6.6**h**
(d)	2.6s	3.4s	14s	1.7**m**	4.4**m**	15.4**m**
(e)	2.5s	3.3s	12.7s	42s	1.4**m**	3**m**

Table 4: *Computation times for different bilateral filtering schemes applied to volumetric datasets of various sizes. The parameter settings and the labels (a)-(e) are those used in Figure 7. The Engine volumetric dataset is used in these experiments.*

φ	2	4	8	16	32	0.2	0.4	0.8
(c)	1.3**h**	15**m**	5.2**m**	4.9**m**	4.8**m**	3.1**h**	1.7**h**	43**m**
(d)	208s	154s	139s	136s	132s	4.8**m**	3.7**m**	184s
(e)	108s	89s	84s	81s	78s	136s	117s	100s

Table 5: *Dependence of computation time for our bilateral filter on the bandwidths* σ^{g_1} *and* σ^{g_2} *for Engine volumetric dataset with* 512^3 *voxels. First five columns:* $\sigma^{g_2}=0.5s_d$ *and* $\sigma^{g_1}=\varphi(5,5,10)$*. Last three columns:* $\sigma^{g_2}=\varphi s_d$ *and* $\sigma^{g_1}=(10,10,20)$*. Parameters* (ε,r) *for (c)-(e) are those used in Figure 7.*

size	32^3	64^3	128^3	32^3	64^3	128^3
	L^1			L^∞		
(b)	0.077	0.337	0.532	4.4	29.5	48.8
(c)	2.6×10^{-9}	4.2×10^{-9}	4.9×10^{-3}	1.5×10^{-5}	1.5×10^{-5}	0.73
(d)	0.14	0.1	0.1	6.3	12	24
(e)	0.14	0.15	0.2	6.3	12	21
	ARE			MRE		
(b)	3.8×10^{-3}	0.017	0.036	1.8	173	3838
(c)	3.2×10^{-3}	8.6×10^{-4}	0.13	0.98	0.78	46
(d)	0.11	0.1	0.1	30	9.8	590
(e)	0.11	0.12	0.12	30	14	1216

Table 6: *Approximation errors corresponding to experiments described in Figure 7 and Table 4.*

φ	0.1	0.2	0.4	0.6	0.8	1.0
Image	13.4s	13.4s	13.2s	11.4s	12.8s	12.4s
Volume	27.6s	26.4s	25.8s	24.5s	24.3s	24s

Table 7: *Computation times for our fast* SC*-Yaroslavsky image (*8192^2 *pixels) and volume (*512^3 *voxels) filtering with* $(\varepsilon,r)=(1,6)$*. The bandwidth parameter* σ^{g_2} *is given by* φs_d*.*

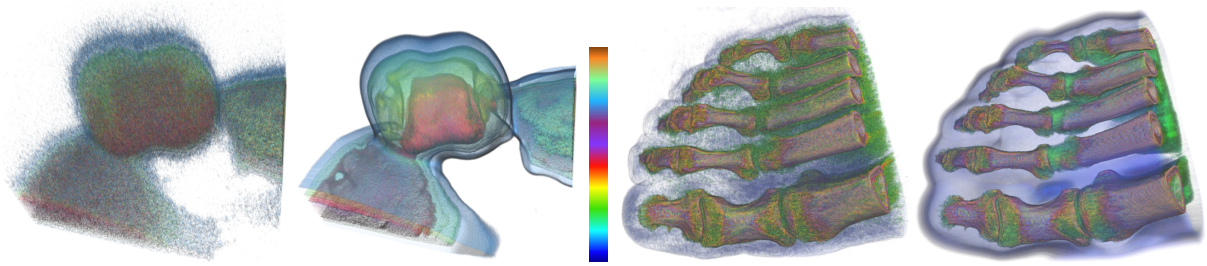

Figure 8: *Examples of volume denoising with our FGT-based fast bilateral filter: two iterations with* $\sigma^{g_2} = s_d$ *and* $(\varepsilon, r) = (10^2, 2)$. *Left-most: original noisy cell-cytokinesis volumetric dataset of size* $256 \times 256 \times 60$ *voxels obtained using a confocal laser microscope. Middle-left: it takes only 9.3 s for our FGT-based bilateral denoising with* $\sigma^{g_1} = (1.6, 1.6, 5)$. *Middle-right: noisy CT-foot volume with* 256^3 *voxels. Right-most: it takes 450 s for our FGT-based bilateral denoising with* $\sigma^{g_1} = (8, 8, 8)$.

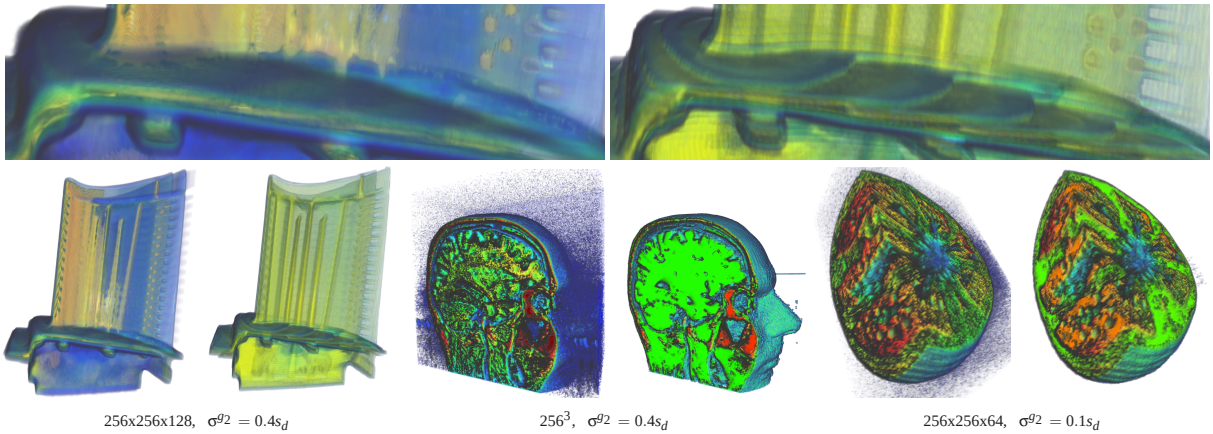

Figure 9: *Volume processing with our fast* SC*-Yaroslavsky filter,* $(\varepsilon, r) = (1, 6)$*: Blade (4 iterations, 5.7 s), MRI-Head (4 iterations, 12.7 s), and Tomato (10 iterations, 8 s). Zoomed parts of the Blade models (two upper images) demonstrate capabilities of the* SC*-Yaroslavsky filter to reveal hidden image structures. The pseudo colors are those used in Figure 7.*

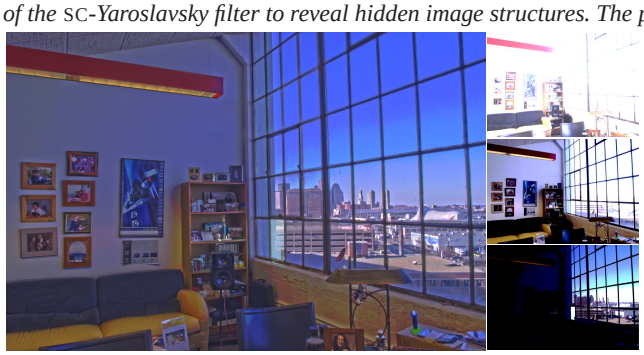

Figure 10: *HDR image tone mapping with gamma correction 1.6 and* $c_p = 15$*: processing* 2000×1312 *pixels takes 2.07 s.*

Figure 11: *HDR image tone mapping with Gamma correction 1.6 and* $c_p = 50$*: processing* 1025×769 *pixels takes 0.98 s.*

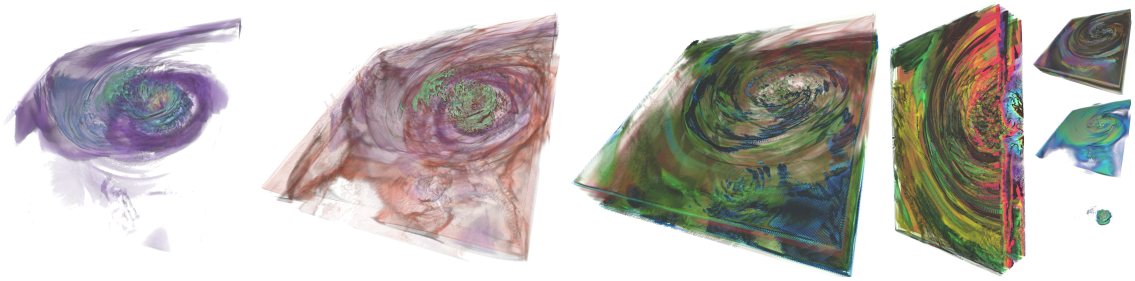

Figure 12: *HDR volumetric tone mapping with* $c_p = 50$*, processing* $512 \times 512 \times 100$ *voxels takes 320s. The pseudo colors are those used in Figure 8. Four left images: different transfer functions are used for visualizing the tone mapped HDR volume.*

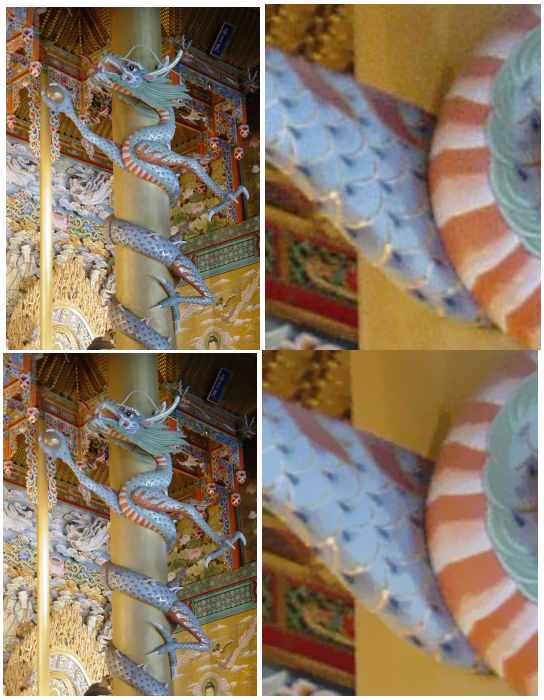

Figure 13: *Color image denoising with our fast FGT-based bilateral filter. Top: original noisy Dragon image consisting of* 1200×1600 *pixels. Bottom: after three iterations of our approximate bilateral filter with* $(\varepsilon, r) = (10^2, 2)$, $\sigma^{g_1} = (10, 10)$ *and* $\sigma^{g_2} = 0.05 s_d$. *Computational time is 240 s.*

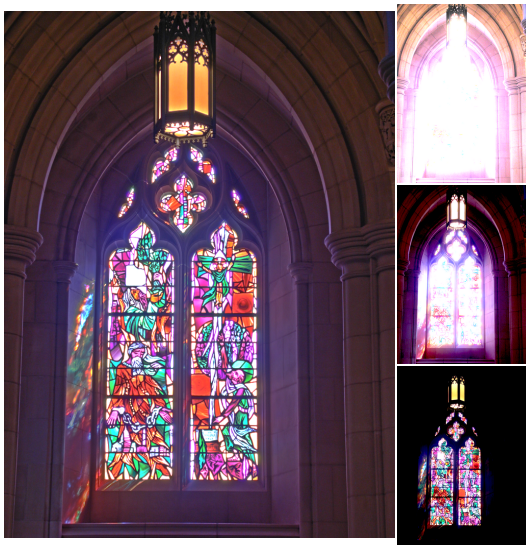

Figure 14: *HDR image tone mapping with gamma correction 2.2 and* $c_p = 50$. *Processing the image with* 767×1023 *pixels takes only 0.8 s.*

8. Conclusion

In this paper, we have developed a FGT-based approach to a fast and accurate approximation of bilateral filtering with Gaussian kernels. Our FGT-based approximation has linear computational complexity and is precision guaranteed. It can be used to process high-dimensional and/or non-uniformly-sampled image data. We have demonstrated that our method outperforms three recent state-of-the-art fast bilateral filtering schemes. We have also presented applications of our fast bilateral filter to real-time image and volume denoising and HDR image displaying problems.

Limitations and future work. In our current implementation of the bilateral filter, we deal with the Gaussian kernels only. Dual-tree algorithms of [LGM06] may be useful for extending the FGT approach to other kernels.

The non-local means filter of Buades et al. [BCM05b, BCM05a] is a very powerful image denoising tool and active research is going on to build fast and accurate approximations of the filter [AGDL09]. Since the filter can be described in terms of multi-dimensional Gauss transforms, it seems promising to use our approach for accelerating of the filter. High-dimensional FGT schemes developed in [YDGD03, MSR*08] may be appropriate for this purpose.

Finally, a GPU-based implementation of FGT also constitutes a promising future work.

Acknowledgements. We are grateful to Sylvain Paris, Frédo Durand, Julie Dorsey, and Mark Eramian for making their source codes available. We would like to thank the anonymous reviewers of this paper for their helpful comments and suggestions.

The models are courtesy of Dani Lischinski (Belgium House and National Cathedral), General Electric (Engine), Lawrence Berkeley Laboratory (Tomato), MIT CSAIL (Dragon), NCAR and NSF (Hurricane), Philips Research (CT-Foot), University of Erlangen (MRI-Head), and Satoshi Shimozono (Cell-cytokinesis). The volume rendering software employed to visualize volumes is courtesy of Takehiro Tawara.

This work was supported in part by Grants-in-Aid for Scientific Research of Japan (20700175 and 20113007).

Alexander Belyaev acknowledges the support from the Joint Research Institute for Signal & Image Processing which is a part of the Edinburgh Research Partnership in Engineering and Mathematics (ERPem).

References

[AGDL09] ADAMS A., GELFAND N., DOLSON J., LEVOY M.: Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph. 28*, 3 (2009), 1–12. Proc. of ACM SIGGRAPH.

[AW95] AURICH V., WEULE J.: Non-linear gaussian filters performing edge preserving diffusion. In *Mustererkennung 1995, 17. DAGM-Symposium* (1995), Springer-Verlag, pp. 538–545.

[BCM05a] BUADES A., COLL B., MOREL J. M.: A non-local algorithm for image denoising. In *Proc. of IEEE-CS Conf. on Computer Vision and Pattern Recognition (CVPR'05)* (2005), IEEE Computer Society, pp. 60–65.

[BCM05b] BUADES A., COLL B., MOREL J. M.: A review of image denoising algorithms, with a new one. *SIAM Multiscale Modeling & Simulation 4*, 2 (2005), 490–530.

[BM05] BENNETT E. P., MCMILLAN L.: Video enhancement using per-pixel virtual exposures. *ACM Trans. Graph. 24*, 3 (2005), 845–852. Proc. of ACM SIGGRAPH.

[BR02] BAXTER B. J. C., ROUSSOS G.: A new error estimate of the fast gauss transform. *SIAM J. Sci. Comput. 24*, 1 (2002), 257–259.

[CM02] COMANICIU D., MEER P.: Mean shift: a robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence 24*, 5 (2002), 603–619.

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph. 26*, 3 (2007), 103. Proc. of ACM SIGGRAPH.

[CT03] CHOUDHURY P., TUMBLIN J.: The trilateral filter for high contrast images and meshes. In *EGRW '03: Proc. of the 14th Eurographics workshop on Rendering* (2003), Eurographics Association, pp. 186–196.

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. In *Proc. of ACM SIGGRAPH* (2002), ACM Press, pp. 257–266.

[EDD03] ELGAMMAL A., DURAISWAMI R., DAVIS L. S.: Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. *IEEE Trans. Pattern Anal. Mach. Intell. 25*, 11 (2003), 1499–1504.

[Era08] ERAMIAN M.: Fast bilateral filter (C, Command Line Utility). In *Code of the histogram-based approach [Por08]* (2008). img.cs.usask.ca/img/index.php?page=download.

[FDCO03] FLEISHMAN S., DRORI I., COHEN-OR D.: Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3 (2003), 950–953. Proc. of ACM SIGGRAPH.

[FJ05] FRIGO M., JOHNSON S.: The design and implementation of fftw3. In *Proc. of the IEEE* (2005), IEEE, pp. 216–231.

[GM00] GRAY A. G., MOORE A. W.: 'n-body' problems in statistical learning. In *Advances in Neural Information Processing Systems 13 (NIPS)* (2000), pp. 521–527.

[GR87] GREENGARD L., ROKHLIN V.: A fast algorithm for particle simulations. *J. Comput. Phys. 73* (1987), 325–348.

[GS91] GREENGARD L., STRAIN J.: The fast gauss transform. *SIAM J. Sci. Stat. Comput. 12*, 1 (1991), 79–94.

[GS98] GREENGARD L., SUN X.: A new version of the fast gauss transform. In *Proc. of IEEE Int. Cong. of Mathematicians III* (1998), pp. 575–584.

[IM03] IHLER A., MANDEL M.: Kernel density estimation toolbox for matlab. www.ics.uci.edu/∼ihler/code/.

[LGM06] LEE D., GRAY A., MOORE A.: Dual-tree fast gauss transforms. In *Advances in Neural Information Processing Systems 18*, Weiss Y., Schölkopf B., Platt J., (Eds.). MIT Press, Cambridge, MA, 2006, pp. 747–754.

[LKdF05] LANG D., KLAAS M., DE FREITAS N.: *Empirical Testing of Fast Kernel Density Estimation Algorithms.* Technical Report UBC TR-2005-03, University of British Columbia, 2005.

[MRY*08] MORARIU V. I., RAYKAR V. C., YANG C., DURAISWAMI R., DAIVS L. S.: Figtree: Fast improved gauss transform with tree data structure. www.umiacs.umd.edu/∼morariu/figtree.

[MS05] MAHMOUDI M., SAPIRO G.: Fast image and video denoising via nonlocal means of similar neighborhoods. *Signal Processing Letters 12*, 12 (2005), 839–842.

[MSR*08] MORARIU V. I., SRINIVASAN B. V., RAYKAR V. C., DURAISWAMI R., DAVIS L. S.: Automatic online tuning for fast gaussian summation. In *Advances in Neural Information Processing Systems 21 (NIPS)* (2008), pp. 1113–1120.

[PD06] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. In *ECCV (4)* (2006), pp. 568–580.

[PD09] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. *In Press, International Journal of Computer Vision 81*, 1 (2009), 24–52.

[PKTD07] PARIS S., KORNPROBST P., TUMBLIN J., DURAND F.: A gentle introduction to bilateral filtering and its applications. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), ACM, p. 1.

[Por08] PORIKLI F.: Constant time O(1) bilateral filtering. In *Proc. of IEEE-CS Conf. on Computer Vision and Pattern Recognition (CVPR)* (2008), IEEE Computer Society, pp. 1–8.

[PvV05] PHAM T. Q., VAN VLIET L.: Separable bilateral filtering for fast video preprocessing. In *Proc. of IEEE Int. Conf. on Multimedia & Expo* (2005), IEEE Computer Society. CD1-4.

[SB97] SMITH S. M., BRADY J. M.: SUSAN - A new approach to low level image processing. *Int. J. Comput. Vision 23*, 1 (1997), 45–78.

[SSN09] SINGER A., SHKOLNISKY Y., NADLER B.: Diffusion interpretation of nonlocal neighborhood filters for signal denoising. *SIAM J. Imaging Sciences 2*, 1 (2009), 118–139.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV '98: Proc. of Int. Conf. on Computer Vision* (1998), pp. 839–846.

[TSP07] TAKEDA H., SINA F., PEYMAN M.: Higher order bilateral filters and their properties. In *Proc. of SPIE Conf. on Computational Imaging* (2007), vol. 6498, pp. 64980S.1–64980S.9.

[vdBvdW02] VAN DEN BOOMGAARD R., VAN DE WEIJER J.: On the equivalence of local-mode finding, robust estimation and mean-shift analysis as used in early vision tasks. In *ICPR '02: Proc. of Int. Conf. on Pattern Recognition* (2002), vol. 3, IEEE Computer Society, pp. 30927–30930.

[Wei06] WEISS B.: Fast median and bilateral filtering. *ACM Trans. Graph. 25*, 3 (2006), 519–526. Proc. of ACM SIGGRAPH.

[WK06] WAN X., KARNIADAKIS G. E.: A sharp error estimate for the fast gauss transform. *J. Comput. Phys. 219*, 1 (2006), 7–12.

[Yar85] YAROSLAVSKY L. P.: *Digital Picture Processing - An Introduction*. Springer, 1985.

[YDGD03] YANG C., DURAISWAMI R., GUMEROV N. A., DAVIS L.: Improved fast gauss transform. In *ICCV '03: Proc. of Int. Conf. on Computer Vision* (2003), pp. 464–471.