DOI: 10.1111/j.1467-8659.2011.02078.x

COMPUTER GRAPHICS *forum*
Volume 31 (2012), number 1 pp. 19–32

A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain

Francesco Banterle, Massimiliano Corsini, Paolo Cignoni and Roberto Scopigno

Visual Computing Laboratory, ISTI-CNR, Italy
{francesco.banterle, massimiliano.corsini, paolo.cignoni, roberto.scopigno}@isti.cnr.it

Abstract

In this work we present a new algorithm for accelerating the colour bilateral filter based on a subsampling strategy working in the spatial domain. The base idea is to use a suitable subset of samples of the entire kernel in order to obtain a good estimation of the exact filter values. The main advantages of the proposed approach are that it has an excellent trade-off between visual quality and speed-up, a very low memory overhead is required and it is straightforward to implement on the GPU allowing real-time filtering. We show different applications of the proposed filter, in particular efficient cross-bilateral filtering, real-time edge-aware image editing and fast video denoising. We compare our method against the state of the art in terms of image quality, time performance and memory usage.

Keywords: bilateral filter, cross-bilateral filter, real-time filtering, video denoising, edge-aware painting, GPU techniques

ACM CCS: I.4.1 [Image Processing and Computer Vision]: Enhancement Filtering I.3.3 [Computer Graphics]: Picture/Image Generation Bitmap and framebuffer operations

1. Introduction

Non-linear filters are widely used in computer graphics, imaging and vision for many different applications. In particular, the bilateral filter proposed by Tomasi and Manduchi [TM98] is one of the most popular non-linear filter, because it filters areas of an image while keeping strong edges. Moreover, its straightforward formulation and flexibility make it applicable to different problems such as: removing noise from images [TM98] and 3D meshes [FDCO03, AGDL09], images/videos stylization [WOG06], high dynamic range (HDR) tone mapping [DD02], flash photography [PSA*04, ED04], edge aware upsampling [KCLU07, DBPT10] and many others. According to the Tomasi and Manduchi formulation the bilateral filter is defined as

$$B(I,\mathbf{x}) = \sum_{\mathbf{x}_i \in \Omega} I(\mathbf{x}_i) f_r(\|I(\mathbf{x}_i) - I(\mathbf{x})\|) g_s(\|\mathbf{x}_i - \mathbf{x}\|), \quad (1)$$

where I is a k-dimensional image, Ω is the set of pixels $\mathbf{x}_i$ in an n^k window, f_r and g_s are, respectively, the *range attenuation* and *spatial attenuation* functions. These functions are weight functions, and they are typically Gaussian functions. Recently, a generalized and elegant formulation of Equation (1) was proposed by Adams *et al*. [AGDL09, ABD10];

$$\mathbf{v}_j = \sum_{\mathbf{p}_i \in \Omega} \mathbf{p}_i e^{-\frac{1}{2}\|\mathbf{p}_i - \mathbf{p}_j\|^2}, \quad (2)$$

$\mathbf{p}$ is a multi-dimensional pixel [e.g. $\mathbf{p}_i = (r_i, g_i, b_i, x_i, y_i)^T$, where $(r_i, g_i, b_i)^T$ is an RGB colour and $(x_i, y_i)^T$ are the spatial coordinates of the pixel], which is pre-multiplied by standard deviations of the Gaussian function [e.g. $(\sigma_r, \sigma_g, \sigma_b, \sigma_x, \sigma_y)^{-T}$]. However, Equation (2) assumes that weight functions are Gaussian functions. In our approach, we do not make such an assumption, keeping the original formulation. The complexity of the plain evaluation of Equation (1) is very high, i.e. $\mathcal{O}(N^k)$ for each pixel of the image. During the last years, this has motivated a substantial research effort in the development of algorithms to accelerate its computation. Some of the developed techniques can be applied also

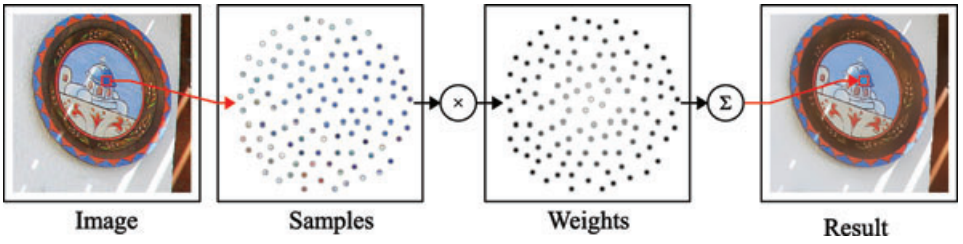

Figure 1: *The main idea of the proposed algorithm: instead of evaluating all samples around a window (left), Ω, a subset of them, Λ, is used (centre), and their contributions with their respective weights are summed up obtaining the final value (right).*

to accelerate high-dimensional non-linear filters such as non-local means [AGDL09, ABD10].

The contribution of this paper is an acceleration technique for bilateral filtering that is straightforward to implement on a GPU, computationally efficient, memory inexpensive, does not require pre-processing and it is capable to provide high-quality results for colour bilateral filtering. The main idea of our technique is to use a suitable subsampling scheme to accelerate the computation of Equation (1). In other words, the filter kernel is not evaluated on the entire spatial domain Ω but on a subset of it in order to provide an estimation of the exact filter value. For each pixel a different sampling pattern is used in order to avoid structured noise on the filtered image. This approach cannot be used to accelerate high-dimensional filtering like other recent approaches [AGDL09, ABD10]. However, it is straightforward to implement on CPU and GPU, and capable to provide high-quality results with respect to the full implementation. Despite the limitation mentioned, the proposed approach can also be applied to accelerate the cross bilateral filter and it can be easily generalized for video processing as shown in the Applications Section. The proposed algorithm has a time computational complexity of $\mathcal{O}(mNk)$, where N is the radius of the kernel, k is the number of dimensions of the image and m is the number of pixels of the image. This improves the brute force evaluation of Equation (1), which has a time computational complexity of $\mathcal{O}(mN^k)$. Moreover, our proposed method maintains the no pre-processing feature and the space computational complexity of the naïve brute force approach, which is $\mathcal{O}(1)$ in memory space. These characteristics allow to apply bilateral filters with medium-large kernel to high-res images (more than 2 Megapixel) at interactive time without memory limitations. Another advantage of our method is that it does not make any assumption about the shape of f_r and g_s while other methods assume to deal with Gaussian functions [PD06, AGDL09, ABD10] as in the formulation in Equation (2).

2. Related Work

Several methods have been recently developed to accelerate different types of bilateral filters, such as the greyscale bilateral filter, the colour bilateral filter or the cross bilateral filter. Some of these methods are able to compute efficiently also high-dimensional non-linear filters like non-local means. The state-of-the-art methods are reviewed here.

One of the most straightforward ways to accelerate the computation of the bilateral filter is to separate the filter into two 1D filters in the case of images, and three filters in the case of videos. This allows to turn the computational complexity into $\mathcal{O}(Nk)$ per-pixel instead of $\mathcal{O}(N^k)$. This is a rough approximation of the bilateral filter, but it has demonstrated to provide good results in video denoising applications [PvV05]. The main disadvantage of this technique is that for medium-large kernel radius (over 20 pixels) the method produces inaccurate results. Moreover, the space complexity is increased from $\mathcal{O}(1)$ to $\mathcal{O}(m)$ because a temporary image has to be stored for evaluating the filter a second time.

Paris and Durand [PD06] proposed a novel approach to compute bilateral filtering by reformulating it as a convolution in a higher dimension, i.e. in a spatio-tonal space (more details in [PD09]). The first step of this approach, called *splatting*, consists in converting the data into the high-dimensional spatio-tonal space. At this point the data are blurred with standard convolution, which can be computed efficiently. Finally, the filtered values are transformed back into their original domain through interpolation. This last phase of the algorithm is called *slicing*. Chen *et al.* [CPD07] proposed the *Bilateral Grid*, a GPU-friendly data structure for the implementation of this method on modern graphics hardware taking advantage of parallelism. This approach can handle also higher order filters but at the cost of a huge amount of memory consumption. Moreover, the Bilateral Grid is not translation-invariant, i.e. the filtering result depends on the phase of the grid [Por08]. The Gaussian kD-tree [AGDL09] relies on the same paradigm as the Bilateral Grid but employs a different data structure to perform the splatting, blurring and slicing operations. This technique can accelerate filters of very high dimensions in an efficient way due to its low memory overhead. The most recent acceleration techniques based on this paradigm is the one based on the Permutohedral Lattice by Adams *et al.* [ABD10]. This data structure has properties from both, the Bilateral Grid and the Gaussian kD-tree. It is

less sparse then the Gaussian kD-tree, but at the same time is less memory consuming than the Bilateral Grid. Furthermore, this technique can efficiently compute high-dimensional filters.

A different approach is to accelerate bilateral filtering reducing redundant operations exploiting the windows' sequential overlaps. This approach is well known for median filtering. Ben Weiss [Wei06], taking inspiration from the histogram-based acceleration algorithm for median filter by Huang [Hua81], developed the fastest algorithm for median filter [$\mathcal{O}(\log N)$ runtime per pixel]. The same approach has been used by Weiss to accelerate the bilateral filter assuming that the spatial function is an average across a square kernel. Histograms and their properties were also used by Porikli [Por08] to develop the fastest algorithm for greyscale bilateral filter (one channel of the image at a time is processed). This algorithm achieves constant time evaluation, $\mathcal{O}(1)$, by reformulating the bilateral filter in the case of constant spatial filtering $f_s(\mathbf{x}) = c$ as an exact sum of linear filters. This sum can be computed in constant time independently on the kernel size using the *integral histogram*. In the same work, Porikli extended this exact formulation for the case of arbitrary spatial filtering using a Taylor expansion. In this case the resulting filter is a good approximation of the exact bilateral filter. An improvement of this work has been recently proposed by Yang *et al*. [YTA09], where quantization is moved into a range function instead of image intensities obtaining a battery of linear filters. The filtered images are obtained by linear interpolation of the output of such filters. The method of Yang *et al*. has been further optimized by Yu *et al*. [YFH*10] introducing a trade-off optimization criteria. Gunturk [Gun10] proposed another improvement of the algorithm of Porikli, which employs a box kernel by using a weighted sum of multiple box kernels (combined optimally) in order to approximate an arbitrary domain kernel. Moreover, Igarashi *et al*. [IIS*10] proposed to use hierarchical histograms in an algorithm similar to Porikli [Por08] and Yang *et al*. [YTA09] to reduce greatly the memory requirements of this kind of filter (from a multiple of the image size down to a multiple of the image's row). All these methods based on histograms work for eight-bit images and are not suitable to be applied to HDR images.

Yoshizawa *et al*. [YBY10] proposed an acceleration method based on a multi-pole transformation called Fast Gauss Transform (FGT). FGT [GS91] is a technique for fast and error-controlled computation of a weighted sum of Gaussians. The main idea is to reformulate the bilateral filter as a weighted sum of Gaussians in a high-dimension domain, and evaluating the resulting summation using the FGT. This algorithm has linear complexity in the image elements.

Fattal *et al*. [FAR07] provided an acceleration scheme for the application of the bilateral filter at multiple scales for detail enhancement purposes. This scheme reuses computations at the j-th scale level for computing the filter at the $(j+1)$-th scale level. In a more general context of image filtering they proposed another multi-resolution approach based on second generation wavelets named *edge-avoiding wavelets* [Fat09], which can be applied to obtain edge-preserving smoothing at multiple scales at a very high speed.

Table 1: *A summary of the different algorithms approximating bilateral filters. GPU means the description of a graphics HW implementation by the authors. Memory means the memory requirement (beside the image itself) of the filter where: Image is the same amount of the input image, Low/Medium/High are with respect to the Image itself (*note that, for HDR images, histogram based methods can require a high amount of memory). HDR denotes the ability of the technique to deal with HDR content.*

Technique	GPU	Memory	HDR
Separable bilateral [PvV05]	Yes	Image	Yes
Bilateral grid [PD09, CPD07]	Yes	High	Yes
Gaussian kD-tree [AGDL09]	Yes	Medium	Yes
Permutohedral lattice [ABD10]	Yes	High	Yes
Fast Gauss transform [YBY10]	No	Low	Yes
Histograms [Por08, YTA09]	Yes	Low/High[a]	No
Proposed method	Yes	Very Low	Yes

Note: [a]For HDR images, histogram-based methods can require a high amount of memory. HDR denotes the ability of the technique to deal with HDR content.

Our algorithm differs from the aforementioned methods since it is entirely based on a subsampling strategy and working directly in the spatial domain. The idea to use subsampling to accelerate processing is not new, from the first attempts to evaluate the rendering equation for ray tracing using different sampling strategies [Coo86] (including Poisson-disk distributed samples) to the work of Thevenaz *et al*. [TBU08] where a subsampling strategy based on Halton sampling is used to accelerate the evaluation of mutual information between two images. Also, the Bilateral Grid employs downsampling during its construction and the Gaussian kD-tree uses Monte Carlo sampling during the node construction. Furthermore, Chen *et al*. [CPD07] proposed a Poisson-disk subsampling optimization for videos which is applied in the spatial-tonal domain. However, our subsampling strategy is performed in the spatial domain (the same domain where the image is defined), without the need to project samples in other domains (splatting) and without the need to unproject them (slicing). This allows a straightforward implementation on CPU and graphics hardware. Despite its straightforwardness, it is capable to provide high-quality results for images and videos. As just stated, the main limit is that it is specific for a certain class of filters and cannot be extended to higher order filters such as the Bilateral Grid, the Gaussian kD-tree, the Permutohedral Lattice and the FGT. However, we do not make any assumptions on the dynamic range of the image, spatial and range functions. Table 1 summarizes the properties of the other approaches with respect to our proposal.

3. Algorithm

The main idea of our algorithm is to evaluate the bilateral filter Equation (1) only for a subset Λ of an N^k kernel window Ω in a k-dimensional image (e.g. $k = 2$ for an image and $k = 3$ for a video sequence, considered as a 3D image), see Figure 1. The pseudo-code of the proposed fast bilateral filtering is shown in Algorithm 2, the implementation of the full bilateral filter is reported in Algorithm 1 for comparison.

Algorithm 1 The brute force bilateral filter algorithm. I is a k-dimensional image, f_r and g_s are attenuation functions, Ω is an N^k window.

Data I, $\mathbf{x}$
Result: B;
$B \leftarrow 0$;
$K \leftarrow 0$;
ForEach $\mathbf{x}_i \in \Omega$
$\omega \leftarrow f_r(\|I(\mathbf{x}_i) - I(\mathbf{x})\|)g_s(\|\mathbf{x}_i - \mathbf{x}\|)$;
$K \leftarrow K + \omega$;
$B \leftarrow B + I(\mathbf{x}_i)\omega$;
$B \leftarrow \frac{B}{K}$;

Algorithm 2 The proposed fast bilateral filter algorithm. I is a k-dimensional image, f_r and g_s are attenuation functions, Ω is a N^k window. *getSample* is a function which gets a sample from Ω. All the N samples are defined as Λ.

Data I, $\mathbf{x}$
Result: B
$B \leftarrow 0$;
$K \leftarrow 0$;
for $i \leftarrow 1$ **To** N_{samples}
$\mathbf{x}_i \leftarrow getSample(\Omega)$;
$\omega \leftarrow f_r(\|I(\mathbf{x}_i) - I(\mathbf{x})\|)g_s(\|\mathbf{x}_i - \mathbf{x}\|)$;
$K \leftarrow K + \omega$;
$B \leftarrow B + I(\mathbf{x}_i)\omega$;
$B \leftarrow \frac{B}{K}$;

While the brute force algorithm evaluates Equation (1) for all pixels in Ω, our technique is quite straightforward: it evaluates the equation only for a well-distributed subset Λ with $|\Lambda| = N_{\text{samples}}$. The key part of the algorithm is the function *getSample*(.) which returns the (integer) coordinates of the sample to use. This function has to carefully choose the samples in order to achieve an accurate approximation of the exact value of the bilateral filter. We tested different sampling strategies for *getSample*(.) such as: regular sampling (RPS), Monte Carlo sampling (MCS), stratified Montet Carlo sampling (SMS), i.e. jittering, and Poisson-disk sampling (PDS). In our test we use Bridson's algorithm [Bri07] to generate Poisson-disk distribution of samples. In order to avoid structured noise and improve randomness a set of different patterns is pre-computed. Then, for each pixel, a random pattern from the pre-computed ones is applied. See Figure 2 for some examples of the patterns' sets. In the case of PDS, 64 tiles are sufficient to obtain good results in terms of randomness [Lag07]. The storage of the set of patterns produces a small overhead in memory, for example a 256×256 kernel needs only 64 Kb of memory independently of the size of the original image. Moreover, these sampling patterns are quite fast to compute, the most computationally expensive tiles are generated from the PDS which requires only a few milliseconds using Bridson's algorithm.

A visual comparison between the different sampling strategies is shown in Figure 3. A straightforward strategy such as RPS produces structured noise (Figure 3c). Therefore, a randomization is needed to remove these kind of artefacts. However, a completely random selection of samples such as in MCS leads to noise (Figure 3d), the same happens for the SMS strategy (Figure 3e). Due to these problems, we chose to get samples that are generated using a Poisson-disk distribution which trades aliasing for noise (Figure 3b), but this noise is masked due to the properties of the human visual system [Coo86].

From a theoretical point of view we could make this sampling process totally unbiased by using an unbiased Maximal Poisson Disk sampling procedure like the recent one of Ebeida *et al*. [EPM*11]. In order to ensure that the corresponding estimator remains unbiased, we could randomly shift the set using the Cranley–Patterson rotation [KK02, SHD11]. Using this shift scheme the storage of a point set is further reduced; for example a 256×256 kernel needs only 1 Kb of memory. Regarding the error bound of our technique, this is related to the variance of the image. While the error is expected to be low in smooth regions, it could increase depending on the variance of a region (i.e. corners and edges), because only a few samples are taken (kN) over the full region (N^2).

The proposed algorithms can work for videos as well. The only difference between the 2D images and videos (or 3D images) is that Poisson-disk samples are computed, respectively, in 2D and in 3D. This cannot be extended trivially to k-dimensions like other approaches since it is not guaranteed that the Poisson-disk sampling strategies could work well for any dimension. This depends heavily on the nature of the dimension we subsample, for example subsampling the support of a non-local means filter, whose components are formed by PCA components, give unpredictable results in terms of quality of the approximation.

Due the straightforwardness of our approach, we implemented the proposed algorithm directly with a GLSL shader on an OpenGL framework [Khr10]. This allows to have efficient implementations even on low-end graphics hardware.

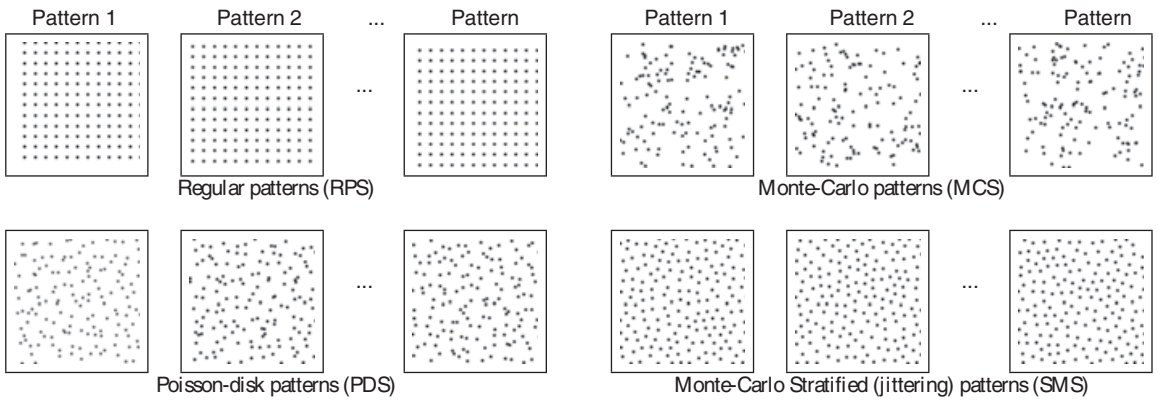

Figure 2: *An example of pre-computed patterns for each of the tested sampling strategies.*

3.1. Analysis of the number of samples required

Since the number of samples of the patterns used plays a fundamental role in the quality of the approximation of the proposed algorithm, we report here an analysis of the impact of the variation of this parameter with respect to the quality obtained using Gaussian functions as attenuation functions (g_s with variance σ_s, f_r with variance σ_r). In particular, we tested the accuracy of our approximation by using different sample densities. For a kernel of $N \times N$ pixels, kN samples for $k = 1, 2, \ldots, 7$ are used and the accuracy evaluated. (The mPSNR is computed as described in Section 6.) Six different HDR images of our data set (Section 6) are used in this test varying either σ_r or σ_s. As it can be noticed from the graphs in Figure 4, using N or $2N$ samples for a kernel of size $N \times N$ is reasonable in terms of quality. Higher accuracy can be achieved for $N_{\text{samples}} = 3N$ samples at a cost of increased computational time. However, it can be noted that from this point ($N_{\text{samples}} = 3N$) graphs starts to increase slowly in terms of quality, where the difference between $3N$ and $4N$ samples is only 1 dB.

Observing the bottom graphs (large kernels) in Figure 4 it is interesting to note that for a fixed mPSNR value, for example 44 dB, the needed samples decrease. This reduction follows the shape of a rectangular hyperbola with the coordinate axes parallel to their asymptotes. From fitting data of these graphs into a hyperbola, it can be elicited that small kernels ($\sigma_s < 20$ pixels) need $2.5N$ samples, medium kernels ($20 \leq \sigma_s \leq 40$ pixels) need between $0.75N$ and $2.5N$ samples, and large kernels ($\sigma_s > 40$ pixels) need $0.75N$ samples.

4. Applications

We now apply our algorithm to some applications in computer graphics and image processing in order to show the effectiveness of our proposal. These applications are image

Figure 3: *An example of different sampling strategies for implementing function getSample applied to the Greek Dome image with $\sigma_s = 60$ pixels and $\sigma_r = 0.15$: (a) all samples (standard bilateral filter); (b) 60 samples using Poisson-disk sampling; (c) 60 samples using a regular pattern sampling; (d) 60 samples using pure Monte Carlo sampling; (e) 60 samples using stratified Monte Carlo sampling (jittering). Note that the use of Poisson-disk sampling produces the closest approximation to the full bilateral filter. The other sampling strategies create visual artefacts and pattern-like artefacts, particularly when a regular sampling pattern is used. (Please, refer to the electronic version for the proper readability of the images.)*

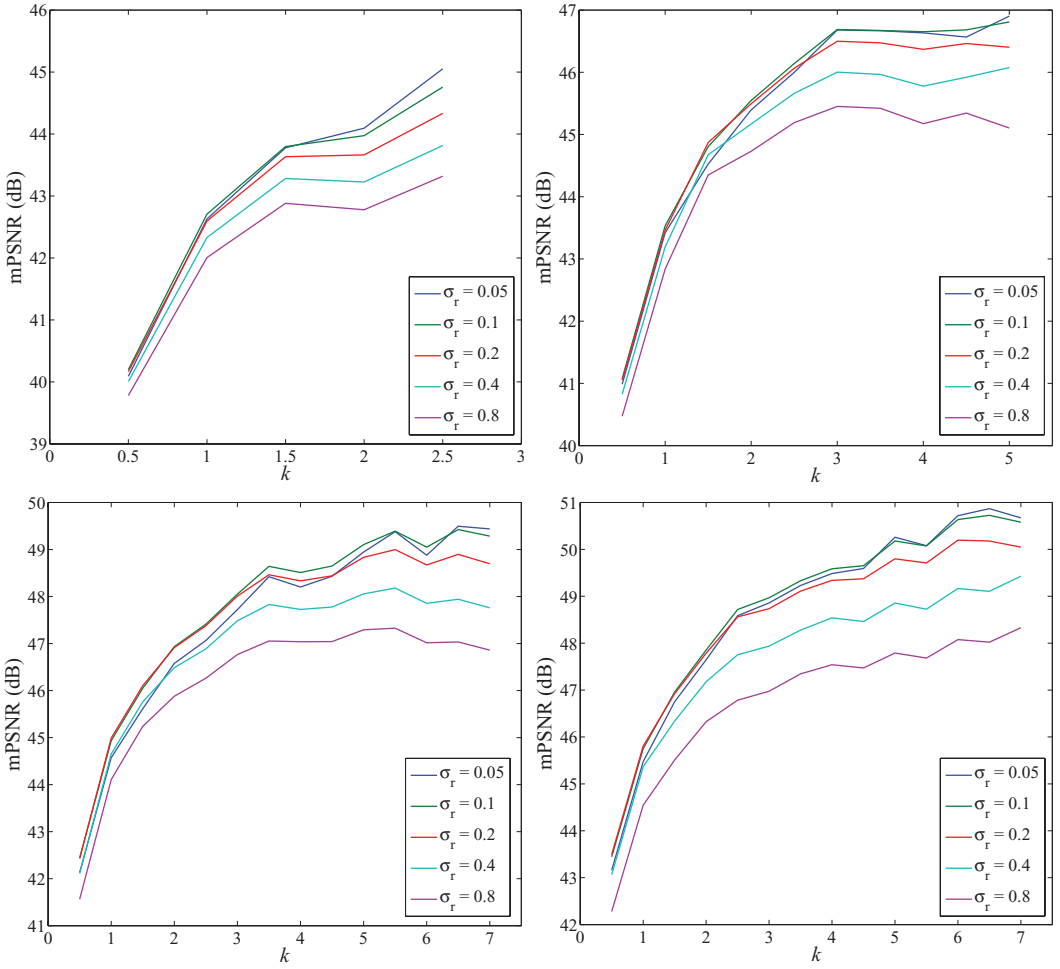

Figure 4: *mPSNR error obtained varying the number of samples used ($N_{samples} = kN$) applied to six HDR images of our data set (Section 6) varying σ_r. (Top left panel) $\sigma_s = 10$. (Top-right panel) $\sigma_s = 20$. (Bottom left panel) $\sigma_s = 40$. (Bottom-right panel) $\sigma_s = 80$. We used the PDS strategy.*

and video filtering, iterative filtering, cross/joint bilateral filtering, edge-aware image editing and tone mapping.

4.1. Video denoising

A typical application of the bilateral filter is image and video denoising, because the filter removes Gaussian noise while keeping strong edges. We show an application of video denosing in Figure 5. Our approach is very suitable for video denoising, because it requires a small amount of memory and it can achieve real-time frame rate for HD (1920 × 1080) resolution videos (around 12 ms per frame).

4.2. Abstraction

Our fast bilateral filter is very suitable for iterative filtering, since it converges very closely to the reference result due to the fact that samples change in spatial position for each pixel at each iteration. Iterative bilateral filtering can be used for abstraction or stylization of photographs or videos [WOG06], see Figure 6 for a result.

4.3. Flash/no flash photography

A straightforward extension of our bilateral filter is the cross/joint bilateral filter [PSA*04, ED04]. This filter transfers strong edges from an image to another one. In this case, Algorithm 2 needs to be slightly modified. An extra input image is added, I_t, which is the source of edges to be transferred. Moreover, the input of f_r becomes $\|I_t(\mathbf{x}_i) - I_t(\mathbf{x})\|$ instead of $\|I(\mathbf{x}_i) - I(\mathbf{x})\|$. One of the applications of this filter is flash/no flash photography, for an example see Figure 7.

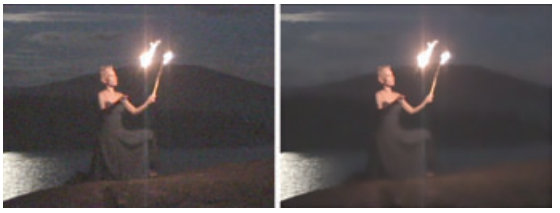

Figure 5: *An example of video denoising applied to a* 640 × 480 *video sequence. (Left-hand side) A frame of the video sequence before the filtering. (Right-hand side) The filtered frame with the application of our filter. A kernel* 40 × 40 × 5 *was used, the processing time for each frame is 1.7067 ms on average.*

Figure 6: *An example of stylization using our bilateral filter with a* 40 × 40 *pixels kernel applied to a* 552 × 574 *image: (top-left panel) the original image; (top-right panel) a stylized image, our bilateral filter was applied five times; (bottom-left panel) the magnitude of the gradients of the top right image; (bottom-right panel) the magnitude of gradients using the full bilateral filter. Note that our filter produces a smooth gradient field as the full bilateral filter due to the fact that different sampling patterns are employed at each iteration.*

4.4. Edge-aware image editing

The cross bilateral filter can also be used for edge-aware image editing. In this application, some parameters of the image need to be modified (i.e. hue, saturation or brightness) in an area selected by a brush stroke avoiding to modify neighbour areas which are separated by edges. We implemented an edge-aware system where the stroke of a brush is

Figure 7: *An example of flash/no flash photography using our cross/joint bilateral filter applied to a* 3072 × 2048 *image: (top-left panel) a photograph of a scene taken with the environment illumination. Note that noise is present in the image due to the use of a high ISO and high shutter speed. (Top-right panel) The same photograph taken using a flash. Note that the original illumination of the environment is lost. (Bottom-left panel) Noise removal using the flash/no flash photography technique with a standard cross bilateral filter. (Bottom-right panel) Noise removal using our algorithm for cross bilateral filter with an* 80 × 80 *kernel. This evaluation took 350 ms.*

Figure 8: *An example of an edge-aware painting application: (left-hand side) the user draws a stroke over an image where hue, brightness or saturation need to be changed. An edge-aware map (green box) is generated applying the cross/joint bilateral filter to the stroke depicted. (Right-hand side) The hue is then shifted using the edge-aware map.*

filtered using our fast cross/joint bilateral filter to preserve edges. The output of the filter is an edge-aware brush which can be used to modify parameters of the image, see Figure 8 for an example.

4.5. Tone mapping

Another application shown is tone mapping of HDR images [DD02]. In this case, we separated an HDR image into a base and a detail image using our bilateral filter. Then, the

Figure 9: *An example of Tone Mapping using our bilateral filter applied to a 0.6 Megapixel image. The kernel is* 30×30 *pixels, and the image took 14 ms to be rendered.*

Figure 10: *An example of the density map with a* 20×20 *pixels kernel. (Left-hand side) The original image. (Right-hand side) The corresponding density map with the number of samples bar.*

base layer is tone mapped using the Reinhard *et al.*'s global tone function [RSSF02]. Finally, the tone mapped base and the detail are recombined obtaining the final tone mapped image (Figure 9).

5. An Adaptive Extension

Sacrificing a bit the simplicity of the proposed approach, our algorithm can be easily extended by making the sampling process *adaptive* with respect to the image signal. Taking into account that low-frequency areas require a very small number of samples (e.g. 3–5), while more samples are needed in high-frequency areas. In order to make the sampling process adaptive, we re-arrange the pre-computed tiles in a multi-resolution way, with denser and denser Poisson-disk distributions, so that it is possible to use a different density of samples according to the image signal. In this experimental version of the algorithm, the selection of the number of samples is guided by a density map based on the gradients' magnitude of the luminance channel of the image to be filtered (see Figure 10).

In this case, the memory requirements increase from $\mathcal{O}(1)$ to $\mathcal{O}(m)$, where m is the number of pixels of the image, because we need to store the density map. Note that a simple evaluation of gradients at the pixel level leads to artefacts. The density map needs to be smoothed to avoid artefacts such as banding between high- and low-sampling areas. After some tests, we can state that the proposed adaptive version improves the uniform algorithm; the speed-up is around 20–27%. This is limited due to the time spent for building the density map. Moreover, the adaptive variant improves the overall quality of around 1.5 dB. A different approach to make the proposed algorithm really adaptive and hopefully more effective with respect to the tested one, is to formulate a heuristic to correlate the local gradient energy with N_{samples} for a certain kernel size and employ a real-time method to generate the Poisson-disk pattern with the estimated number of samples required like the one by Li-Yi Wei [Wei08]. This should guarantee both very high speed rate and quality. We leave thorough evaluation of this improvement as an interesting matter of future research.

6. Analysis and Results

In this section, we evaluate our bilateral filter to determine how close it is to the brute force filtering. Moreover, we compare our algorithm with other state-of-the-art techniques. In our evaluation and comparison, we assessed the visual quality, the numerical accuracy and the time performance of the proposed algorithm. Although we showed in Section 3.1 that less samples are needed for large kernels, in order to be conservative as much as possible, we fixed the number of samples in our algorithm to $2N$ samples, where N is the radius in pixels of the kernel.

6.1. Visual quality

The techniques that we chose for the comparison are the Separable Bilateral Filter [PvV05], the Bilateral Grid [PD06], the Gaussian kD-tree [AGDL09] and the Permutohedral Lattice [ABD10]. This comparison does not pretend to be exhaustive, many other techniques could be used in the comparison such as the constant time bilateral filtering [Por08] and the FGT [YBY10]. However, these tests are sufficient to provide a general idea of the visual quality that our method can achieve. In this comparison, we indicate with I the original image, with I^* the image filtered with the full bilateral filter and with I^f the image filtered with one of the tested techniques. The results are summarized in Figure 11 (see supporting information for more comparisons). The difference image, which is useful to assess the visual differences, is computed using the absolute difference of values for each channel. To increase readability, these differences are multiplied by 10. As it can be noted, the subsampling bilateral filter exhibits good visual quality, even for complex images.

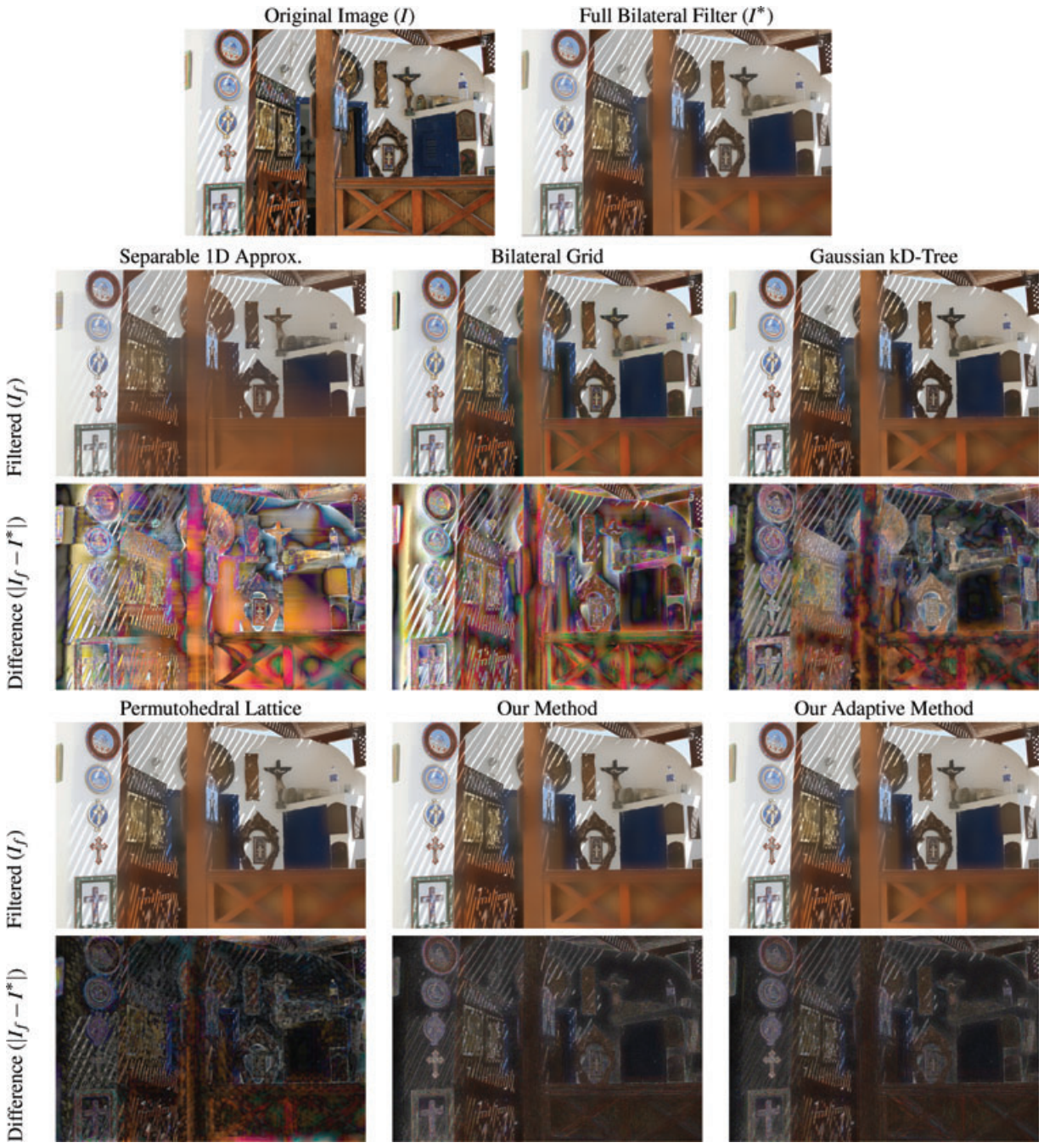

Figure 11: *Visual comparison of several acceleration methods for colour bilateral filtering using $\sigma_s = 100$ pixels and $\sigma_r = 0.19$. The difference images are amplified by 10 to improve error visibility.*

6.2. Accuracy evaluation

Our tests evaluate the accuracy of the proposed technique using several metrics: the maximum error (E_{MAX}), the mean error ($\bar{E}$), the mean relative error ($\bar{E}_r$) and the root mean square error (*RMSE*). According to the previously used notation [YBY10], the described metrics are computed as

$$E_{MAX} = \max |I^* - I^f|, \tag{3}$$

$$\bar{E} = \frac{1}{m} \sum_{i=1}^{m} |I^*(\mathbf{x}_i) - I^f(\mathbf{x}_i)|, \tag{4}$$

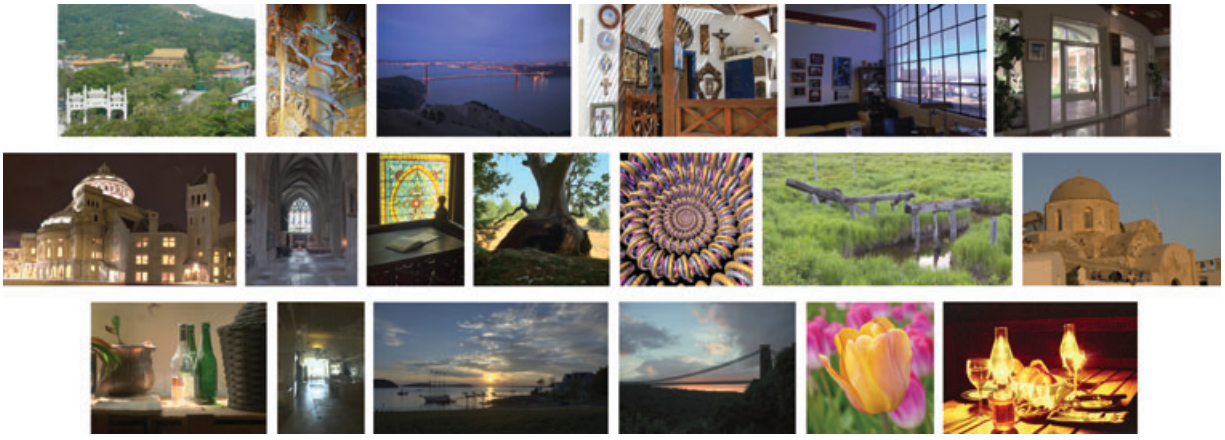

Figure 12: *The data set of 19 HDR and LDR images used in our tests.*

$$\bar{E}_r = \frac{1}{m}\sum_{i=1}^{m}\frac{|I^*(\mathbf{x}_i) - I^f(\mathbf{x}_i)|}{I^*(\mathbf{x}_i)}, \tag{5}$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(I^*(\mathbf{x}_i) - I^f(\mathbf{x}_i))^2}, \tag{6}$$

where m is the number of pixels considered. Since we tested the accuracy of the filter on many HDR images, we used the multi-exposure Peak-Signal-to-Noise Ratio (mPSNR) [MCHAM06]. The mPSNR is a popular quality metric for testing HDR compression methods [BADC11]. This metric takes a series of exposures which are tone mapped using a simple gamma curve defined as

$$T(v,c) = \left[255(2^c v)^{\frac{1}{\gamma}}\right]_0^{255}, \tag{7}$$

where c is the current f-stop, v is a colour value and $\gamma = 2.2$. Then, the mPSNR is computed as

$$\mathrm{MSE}(I^f, I^*) = \frac{1}{m \times p}\sum_{c=1}^{p}\sum_{i=1}^{m}\left(\Delta R_{i,c}^2 + \Delta G_{i,c}^2 + \Delta B_{i,c}^2\right), \tag{8}$$

$$\mathrm{mPSNR}\,(I^f, I^*) = 10\log_{10}\left(\frac{3\times 255^2}{\mathrm{MSE}\,(I^f, I^*)}\right), \tag{9}$$

where p is the number of exposures (sampling uniformly the dynamic range of the image), and $\Delta R_{i,c} = T(R^f(\mathbf{x}_i), c) - T(R^*(\mathbf{x}_i), c)$ for the red colour channel, and so on for the green and blue channels.

In our tests, we filtered 19 images, shown in Figure 12, with several parameters of $\sigma_s \in [1, 320]$ pixels and $\sigma_r \in [0.05, 1.2]$ for a total of about 1000 filtered images. All LDR images were normalized in the interval [0, 1]. The accuracy results are summarized in Table 2. Several considerations can be drawn from this evaluation. As expected, our method can have a high maximum error since, in some cases, the samples do not give a correct estimate of the exact filter value. However, the accuracy is good taking into account the mean error. It is interesting to point out that the mPSNR shows that our approach can compete with the state-of-the-art methods. This is one of the most important results presented here.

6.3. Timing comparisons

We evaluated the time performance of our algorithm and compared with the other solutions presented in Section 6.1 on graphics hardware. The GPU versions of these methods are available on the website of the respective authors. In these tests, we varied the kernel size of the filter (Figure 13 left panel), the size of the image (Figure 13 middle panel) and the size of the temporal width of the kernel for videos (Figure 13 right panel). The machine, that we used in these tests, is an Intel Core i7 2.8 Ghz equipped with 4 GB of main memory and a NVIDIA GeForce GTX 480 with 2.5 GB of memory under a 64-bit Linux distribution.

From the graphs in Figure 13, it can be seen that our algorithm has linear performance, following its linear complexity in the radius kernel size. It can be noted that it is slower than the separable bilateral filter. While the separable bilateral filter accesses texels coherently minimizing cache misses, our method fetches GPU texture in a random order which is not ideal for caches. The separable bilateral filter has very poor performance in terms of quality as reported in Table 2. It is important to note that our methods typically have interactive frame rates (between 7 Hz and 20 Hz) if not real time. In these tests, we omitted the results of the Gaussian kD-tree. We excluded them because they were on the order of 10–20 s. The Gaussian kD-tree performs better in dimensions higher

Table 2: *Numerical accuracy of the proposed technique against other acceleration methods based on available CPU versions.*

	Maximum error	Mean error	Relative error	RMSE	mPSNR
Separable bilateral	2.32	0.03	0.84	0.05	26.27
Bilateral grid (3D)	0.98	0.02	0.18	0.15	28.94
Bilateral grid (5D)	0.38	0.01	0.08	0.02	35.85
Gaussian kD-tree	0.68	0.08	0.06	0.06	35.08
Permutohedral lattice	1.59	0.003	0.02	0.01	39.80
Our method	2.64	0.02	0.03	0.12	39.66
Our method adaptive	1.91	0.01	0.02	0.08	39.75

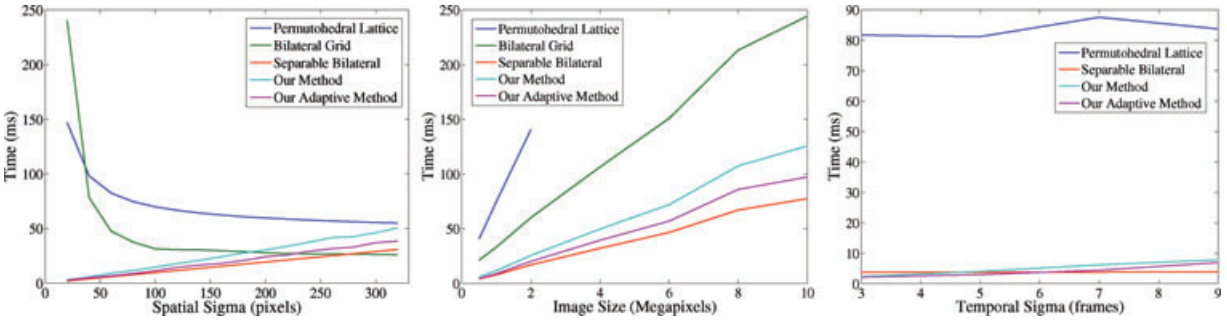

Figure 13: *Timing comparisons of the proposed algorithm with $N_{samples} = 2N$: (left panel) timing performance for RGB images (5D); a 1 Megapixel image was used. Kernels of size from* 20×20 *to* 320×320 *pixels were tested using* $\sigma_r = 0.1$. *(Middle panel) Timing performance when varying the image size from 0.5 to 10 Megapixel, a kernel with an* 80×80 *pixels spatial kernel and* $\sigma_r = 0.1$ *was used in this test. Note that the permutohedral lattice went out of memory for images larger than 4 Megapixel. (Right panel) Timing performance for temporal bilateral filtering (6D). A 1 Megapixel video was used with a* 30×30 *pixels spatial kernel size. Temporal kernels of width 3, 5, 7 and 9 frames were used.*

than three–six [ABD10]. These values are completely out of comparisons compared to the other methods which work in 10–100 ms. From these tests, we can point out that the bilateral grid is the fastest method for medium-large kernels (more than 150 pixels). Our methods perform better only for small-medium kernels (less than 150 pixels). However, the bilateral grid applies an aggressive downsampling for keeping the convolution kernel constant and this can reduce the quality of the filtered image. The main advantages of our technique with respect to this technique is about the memory usage (as shown in the next section) and higher accuracy of the results as shown in Table 2. The bilateral grid was not tested for temporal bilateral filtering because the original paper applies a 3D bilateral grid for each frame [CPD07] and no 4D implementation is available. The permutohedral lattice, which is the method with the best mPSNR, has interactive performance, but it becomes attractive only for very large kernels (more than 300 pixels). Similarly to the bilateral grid, this method consumes a large amount of memory as well. For example, we were not able to perform all timing tests when increasing the image size (Figure 13 middle panel) because this method went out of memory with 4 Megapixel images. Moreover, it went out of memory for filtering HD videos (1920×1080), so the spatial resolution of videos was set to 1 Megapixel.

6.4. Memory consumption

We evaluated the memory consumption performance of our algorithm and compared with the others presented in Section 6.1 using a 1.2 Megapixel HDR image with 4.4 order of magnitudes of dynamic range. The results of these comparisons are shown in Figure 14. From these comparisons, our method consumes less memory than other methods. Indeed, it needs only extra memory for storing samples which depends on the kernel size N and the number of tiles k. One of the key points of our algorithm is that it does not depend on the image size. In practice this memory overhead is negligible: if we store a coordinate in an unsigned short, a 300×300 kernel for a 2D image using 64 tiles requires only 75 Kb of memory. The only drawback is when using its adaptive form which needs to store the magnitude of gradients of the image. However, a downsampled image to one quarter of the original can produce results with a negligible variance, and this suggests that further downsampling could be achieved. Moreover, it is clear from the graphs in

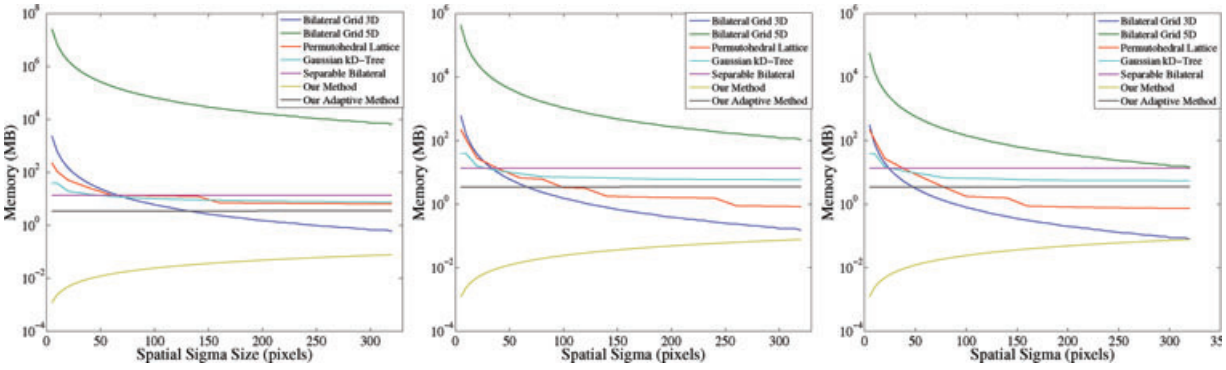

Figure 14: *Memory requirements comparisons of the proposed algorithm with $N_{samples} = 2N$, using an 1.2 Megapixel HDR image (4.4 order of magnitude of dynamic range) and varying kernels of size from 20×20 to 320×320: (left panel) $\sigma_r = 0.05$; (middle panel) $\sigma_r = 0.2$; (right panel) $\sigma_r = 0.4$.*

Figure 14 that other methods are competitive only for very large kernels (more than 350 pixels wide).

6.5. Limitations

The main limitation of the proposed technique is that it cannot be naturally extended to accelerate high-dimensional filters. For example, the Gaussian kD-tree [AGDL09] can be modified in order to accelerate non-local means by filtering in 16 dimensions. Another problem arises when the bilateral filter behaves as a Gaussian filter (when σ_r is very high or near to the maximum value of the image). In this case, some noise could be present on the filtered image. This problem characterizes also other acceleration techniques such as the permutohedral lattice, the Gaussian kD-tree or the bilateral grid, where artefacts appear as square-like or triangle-like patterns, see Figure 11. We also point out that, for images with very high gradients, the estimation made through sparse sampling could be insufficient and high-density sampling could be required to obtain accurate results in terms of visual quality.

7. Conclusions

In this paper we proposed a low-memory, efficient and practical approach to accelerate bilateral filtering based on a subsampling strategy. Despite the simplicity of the proposed algorithm, it is capable to obtain results comparable with the state-of-the-art in terms of approximation of the standard bilateral filter, especially for medium-large kernel size. A very low-memory overhead is necessary (to store the Poisson-disk samples) making the technique applicable to deal with images with very high resolution. The method can also be implemented on graphics hardware in a very straightforward way making it available also for devices with low-end graphics hardware. Moreover, the proposed technique can be applied to other classes of filters such as the cross/joint bilateral filter, and video filtering, and the HDR images can be processed in the same way as LDR images. The experimental results elicited the excellent trade-off between quality and speed of the proposed technique. The many presented applications demonstrate the effectiveness of our approach.

Acknowledgements

We greatly thank Marco Di Benedetto, Andrew Adams and Jiawen Chen for their help with OpenGL code and compiling their filters' implementations. We thank Sylvain Paris, Fredo Durand and Hugues Hoppe for providing their public data and code. We also thank the anonymous reviewers for their feedback and suggestions to improve the paper. This work was funded by the EC IST IP project '3D-COFORM' (IST-2008-231809).

References

[ABD10] Adams A., Baek J., Davis M.: Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum 29*, 2 (2010), 753–762.

[AGDL09] Adams A., Gelfand N., Dolson J., Levoy M.: Gaussian KD-trees for fast high-dimensional filtering. *ACM Transaction on Graphics 28*, 3 (2009), 1–12.

[BADC11] Banterle F., Artusi A., Debattista K., Chalmers A.: *Advanced High Dynamic Range Imaging: Theory and Practice* (1st Edition). AK Peters Ltd. Natick, MA (CRC Press), February 2011.

[Bri07] Bridson R.: Fast Poisson disk sampling in arbitrary dimensions. In *SIGGRAPH '07: ACM SIGGRAPH 2007 Sketches* (New York, NY, USA, 2007), ACM, p. 22.

[Coo86] Cook R. L.: Stochastic sampling in computer graphics. *ACM Transaction on Graphics 5* (January 1986), 51–72.

[CPD07] Chen J., Paris S., Durand F.: Real-time edge-aware image processing with the bilateral grid. *ACM Transaction on Graphics 26*, 3 (2007), 103–112.

[DBPT10] Dolson J., Baek J., Plagemann C., Thrun S.: Upsampling range data in dynamic environments. In *CVPR 2010: IEEE International Conference on Computer Vision and Pattern Recognition* (June 2010), IEEE Computer Society, pp. 1141–1148.

[DD02] Durand F., Dorsey J.: Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transaction on Graphics 21*, 3 (2002), 257–266.

[ED04] Eisemann E., Durand F.: Flash photography enhancement via intrinsic relighting. *ACM Transaction on Graphics 23*, 3 (2004), 673–678.

[EPM*11] Ebeida M. S., Patney A., Mitchell S. A., Davidson A., Knupp P. M., Owens J. D.: Efficient maximal Poisson-disk sampling. *ACM Transaction on Computer Graphics 30*, 4 (2011), 49:1–49:12.

[FAR07] Fattal R., Agrawala M., Rusinkiewicz S.: Multiscale shape and detail enhancement from multi-light image collections. *ACM Transaction on Graphics 26* (July 2007), 51:1–51:9.

[Fat09] Fattal R.: Edge-avoiding wavelets and their applications. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), *SIGGRAPH '09, ACM*, pp. 22:1–22:10.

[FDCO03] Fleishman S., Drori I., Cohen-Or D.: Bilateral mesh denoising. *ACM Transaction on Computer Graphics 22* (July 2003), 950–953.

[GS91] Greengard L., Strain J.: The fast Gauss transform. *SIAM Journal on Scientific and Statistical Computing 12*, 1 (1991), 79–94.

[Gun10] Gunturk B.: Fast bilateral filter with arbitrary range and domain kernels. In *IEEE International Conference on Image Processing, 2010* (Hong Kong, China, September 2010), IEEE Computer Society, pp. 3289–3292.

[Hua81] Huang T. S.: *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1981.

[IIS*10] Igarashi M., Ikebe M., Shimoyama S., Yamano K., Motohisa J.: O(1) bilateral filtering with low memory usage. In *IEEE International Conference on Image Processing* (Hong Kong, China, September 2010), IEEE Computer Society, pp. 3301–3304.

[KCLU07] Kopf J., Cohen M. F., Lischinski D., Uyttendaele M.: Joint bilateral upsampling. *ACM Transaction on Graphics 26*, 3 (2007), 96–102.

[Khr10] Khronos OpenGL Working group: *The OpenGL Specification, Version 4.1* (Wiley-Blackwell, Chichester, UK, 26 July 2010).

[KK02] Kollig T., Keller A.: Efficient multidimensional sampling. *Computer Graphics Forum 21*, 3 (2002), 557–563.

[Lag07] Lagae A.: *Tile-Based Methods in Computer Graphics*. PhD thesis, Departement Computerwetenschappen, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium, April 2007.

[MCHAM06] Munkberg J., Clarberg P., Hasselgren J., Akenine-Möller T.: High dynamic range texture compression for graphics hardware. *ACM Transaction on Computer Graphics 25*, 3 (2006), 698–706.

[PD06] Paris S., Durand F.: A fast approximation of the bilateral filter using a signal processing approach. In *ECCV 2006: European Conference on Computer Vision* (Graz, Austria, 2006), pp. 568–580.

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

[Por08] Porikli F.: Constant time O(1) bilateral filtering. In *CVPR 2008: IEEE International Conference on Computer Vision and Pattern Recognition* (Anchorage, AK, USA, June 2008), IEEE Computer Society, pp. 1–8.

[PSA*04] Petschnigg G., Szeliski R., Agrawala M., Cohen M., Hoppe H., Toyama K.: Digital photography with flash and no-flash image pairs. *ACM Transaction on Graphics 23*, 3 (2004), 664–672.

[PvV05] Pham T., van Vliet L.: Separable bilateral filtering for fast video preprocessing. In *IEEE International Conference on Multimedia and Expo* (Genoa, Italy, July 2005), pp. 1–4.

[RSSF02] Reinhard E., Stark M., Shirley P., Ferwerda J.: Photographic tone reproduction for digital images. *ACM Transaction on Computer Graphics 21*, 3 (2002), 267–276.

[SHD11] Schlömer T., Hecky D., Deussen O.: Farthest-point optimized point sets with maximized minimum distance. In *HPG '11: High Performance Graphics Proceedings* (New York, NY, USA, 2011), ACM, pp. 135–142.

[TBU08] Thévenaz P., Bierlaire M., Unser M.: Halton sampling for image registration based on mutual information. *Sampling Theory in Signal and Image Processing 7*, 2 (2008), 141–171.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), IEEE Computer Society, pp. 839–847.

[Wei06] WEISS B.: Fast median and bilateral filtering. *ACM Transaction on Graphics 25*, 3 (2006), 519–526.

[Wei08] WEI L.-Y.: Parallel Poisson disk sampling. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 20:1–20:9.

[WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B.: Real-time video abstraction. *ACM Transaction on Graphics 25*, 3 (2006), 1221–1226.

[YBY10] YOSHIZAWA S., BELYAEV A., YOKOTA H.: Fast Gauss bilateral filtering. *Computer Graphics Forum 29*, 1 (2010), 60–74.

[YFH*10] YU W., FRANCHETTI F., HOE J. C., CHANG Y.-J., CHEN T.: Fast bilateral filtering by adapting block size. In *IEEE International Conference on Image Processing* (Hong Kong, China, September 2010), IEEE Computer Society, pp. 3281–3284.

[YTA09] YANG Q., TAN K.-H., AHUJA N.: Real-time O(1) bilateral filtering. In *CVPR 2009: IEEE International Conference on Computer Vision and Pattern Recognition* (Miami, FL, USA, June 2009), IEEE Computer Society, pp. 557–564.

Supporting Information

Additional supporting information may be found in the online version of this article:

Video S1: This video shows an example of real-time video de-noising using our bilateral filter acceleration method. The video shows: the original footage, the filtered one, and a split-screen comparison between the twos.

Video S2: This video shows an example of edge-aware painting. In particular, this example demonstates how to change the hue in an edge-aware way.

Video S3: This videos demonstrates real-time bilateral filtering on a HDR image; the spatial and intensity kernels are varied.

Please note: Wiley-Blackwell are not responsible for the content or functionality of any supplementary materials supplied by the authors. Any queries (other than missing material) should be directed to the corresponding author for the article.