Pacific Graphics (2012)
C. Bregler, P. Sander, and M. Wimmer (Editors)

Point-wise Adaptive Filtering for Fast Monte Carlo Noise Reduction

Jie Guo[1] and Jingui Pan[1]

[1]State Key Lab for Novel Software Technology, Nanjing, China

Abstract

Monte Carlo based photorealistic image synthesis has proven to be one of the most flexible and powerful rendering techniques, but is plagued with undesirable artifacts known as Monte Carlo noise. We present an adaptive filtering method designed for Monte Carlo rendering systems that counteracts noise while respecting sharp features. The filter operates as a post-process on a noisy image augmented with three screen-space geometric attribute buffers, and by using a point-wise adaptive (varying window size) filtering kernel, this method is able to reinforce the preservation of important scene reflected edges, in less time. Comparative results demonstrate the simplicity and efficiency of our method, which makes it a feasible and robust solution for smoothing noisy images generated by Monte Carlo rendering techniques. CUDA implementation also makes the algorithm potentially practical for interactive Monte Carlo rendering in the near future.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—

1. Introduction

Monte Carlo techniques are widely used in global illumination, which calculate the image colors via randomly or quasi-randomly sampling an integration domain covering all light paths from a luminaire to the camera. Currently, only these approaches can handle the wide range of surface geometries, reflection models, and lighting effects that occur in the real world. While conceptually simple, these techniques are computationally expensive since a great number of samples are required in order to create stunning visual phenomena without noticeable noise. Monte Carlo noise is inevitable when insufficient samples are provided. One effective way to tackle this problem is image denoising that uses a fairly low sampling density to get a coarse image and then smooth out noise by applying suitable filters in a post-processing stage.

Traditional image noise reduction algorithms, such as box filters or Gaussian filters, can effectively average away undesired noise values, however they also blur the important geometric details or illuminating discontinuity (e.g. light spots on glossy surface or hard shadow boundaries) [BCM05, PKTD08]. To preserve edges, bilateral filters [TM98], anisotropic diffusion [McC99], nonlocal means filter [BCM05] and other algorithms have been introduced. Despite their impressive results, they are either poor in retaining subtle geometric and shading details or computationally expensive due to their iterative nature.

In this paper, we present an adaptive smoothing approach for noise removal and feature preservation of path traced 3D scenes, especially the scenes containing mirror-reflected objects. This approach has the following characteristics:

- **Feature-preserving**: Similar to bilateral filter, our filter is also a nonlinear filter with adaptive kernels. However, we adopt second-bounce geometric information as the guidance to achieve better filtering results, especially for reflected edges. This auxiliary information can efficiently identify the subtle geometric details such as tiny holes and shape corners on reflected surfaces.
- **Adaptive kernel**: The filter's window size is made to adapt to the local characteristics of a noisy image. This can efficiently reduce the filtering time when most parts of the image only need small window size.
- **Non-iterative and GPU acceleration**: Unlike most of the denoising approaches which update the images iteratively until the noise has been diffused sufficiently, we have concentrated on a single-pass technique which makes it more efficient than iterative schemes. In addition, since we have

DOI: 10.2312/PE/PG/PG2012short/017-022

implemented the whole filtering procedure on the GPU using CUDA, the performance has been greatly improved compared with CPU version approaches.

2. Related Work

Almost all of the filters used in the early years of stochastic rendering methods are linear filters [CPC84] that either lack adequate samples to eliminate noise or produce excessive samples to smear boundaries in the images. Lee and Redner [LR90] firstly introduced the use of nonlinear filters such as alpha trimmed filters to reduce speckled noise in stochastically rendered images. This method only discards statistical outliers and averages the remaining samples, making this a non-energy-preserving filter. DeCoro et al. [DWR10] also removed low-probability yet high-energy outliers when filtering noise in Monte Carlo rendering. Rushmeier and Ward [RW94] proposed an energy-preserving nonlinear filter that spreads out the contribution of input sample values into the output image depending on a variance estimate to suppress the outliers and reduce the noise. Considering the noise is mostly caused by the indirect illumination, filters can be applied only to the low-frequency indirect illumination separated from the rest of the image in order not to blur the important scene features caused by the other part of light transport [JC95, BEM11].

Anisotropic diffusion [PM90] is a well-established tool in early vision to accomplish edge-preserving smoothing. McCool [McC99] developed an energy-preserving filter based on anisotropic diffusion that averaged out noise using diffusion equations but edges and textures could be maintained. However, the discrete diffusion nature and the iterative solver make it a slow process. Bilateral filtering was developed by Tomasi and Manduchi [TM98] as a fast alternative to anisotropic diffusion. Unlike Gaussian filters, the weight of a pixel depends not only on a spatial kernel, but also on a function in the intensity domain which decreases the weight of pixels with large intensity differences. Later, several fast versions of bilateral filter have been proposed, e.g. [DD02, PD09]. Barash [Bar02] presented a link between anisotropic diffusion and bilateral filtering using an extended definition of intensity that includes spatial coordinates. Although simple, fast and effective for a variety of problems in computer vision and computer graphics, bilateral filtering has certain difficulties, for instance, it performs poor smoothing in high gradient regions. Trilateral filter [CT03] was introduced to smooth signals towards a sharply-bounded, piecewise-linear approximation, which provides stronger noise reduction and better performance. Xu and Pattanaik [XP05] modified bilateral filter and built a novel local adaptive noise reduction kernel in a non-iterative way. To reinforce the preservation of discontinuity, more geometric information is introduced into the bilateral filtering, such as position, normal or depth, forming the discontinuity buffers. These buffers are widely used in real-time global illumination techniques to perform filtering or upsampling of sparsely sampled-data [WKB*02, LSK*07, SGNS07]. For fast filtering, Dammertz et al. [DSHL10] derived an edge-avoiding À-Trous filtering method that incorporates a wavelet transformation into the bilateral filtering, and combines edge-stopping functions from multiple input images including noisy ray traced image, normal buffer and position buffer. This method was further extended [HDL11] by introducing a data-adaptive edge weight, which showed better performance. Recently, Sen and Darabi [SD12] extended cross-(joint-) bilateral filter [ED04, PSA*04] by introducing two terms that reduce the importance of features that are functions of the random parameters when reducing the noise. While this approach has demonstrated high-quality results for a wide range of Monte Carlo effects from low input sample density, it comes at the price of additional complexity, accompanied by higher computational cost.

Our method is also based on bilateral filtering, but it excels in its simplicity both in concept and implementation, and it is better in performance due to non-iterative nature and GPU acceleration. Furthermore, since we employ additional geometry information to guide the filtering, we achieve visually more satisfying results than traditional bilateral filters.

3. Theory

3.1. Feature-preserving Filtering

The principle of image filtering is combining samples of several adjacent pixels into one integral estimate. Given an input image I, the estimator is given by:

$$\hat{I}(X) = \frac{\sum_{\xi \in NS(X)} w(X,\xi) I(\xi)}{\sum_{\xi \in NS(X)} w(X,\xi)} \tag{1}$$

where X is a 2D screen sample in the image, $X = (i, j)$, and $I(\xi)$ is the pixel value of its neighbor ξ in neighboring set $NS(X)$. The weight $w(X,\xi)$ is the key component in this equation that determines the quality and efficiency of the final results.

Traditionally, the Gaussian low-pass filter is a linear filter, since the filter action is independent of the image content and only the distances between positions matter. Although it can be implemented efficiently using fast Fourier transforms, it fails at edges. Bilateral filtering can tackle this problem by taking into consideration the variation of intensities as well. The weight function of bilateral filtering can be defined in Gaussian form:

$$w(X,\xi) = G_p(X,\xi) G_c(I(X), I(\xi)) \tag{2}$$

with

$$G_p(X,\xi) = \exp\{-\frac{1}{2}(\frac{\|X-\xi\|}{\sigma_p})^2\} \tag{3}$$

$$G_c(I(X), I(\xi)) = \exp\{-\frac{1}{2}(\frac{\|I(X)-I(\xi)\|}{\sigma_c})^2\} \tag{4}$$

Here, G_p is the traditional domain Gaussian function measuring the geometric closeness between X and ξ, while G_c is the range function measuring the similarity in photometric range. Both functions are controlled by the standard deviation parameters (σ_p and σ_c). Unfortunately, the traditional bilateral filter fails to deal with complex geometry details and is also sensitive to outliers.

The cross-(joint-) bilateral filter [ED04, PSA*04] extends the basic idea of traditional bilateral filter by decoupling the weight function from the input image. It is based on arbitrary input images to compute the weighting terms. E.g. the edge-avoiding À-Trous filter [DSHL10] uses normal and position buffers to denoise a path traced image.

We draw our inspiration mainly from this approach, but we incorporate three additional Gaussian functions into Eq. 2 to reinforce the preservation of subtle geometry details and important lighting effects. The first function G_n identifies the normal's similarity between X and ξ:

$$G_n(n(X), n(\xi)) = \exp\{-\frac{1}{2}(\frac{\|n(X) - n(\xi)\|}{\sigma_n})^2\} \quad (5)$$

which is similarly defined in [DSHL10]. In this function, n returns the surface normal of the first intersection point in the scene when shooting a ray from camera. Obviously, a smooth surface can be described as one having smoothly varying normals, whereas features such as corners and edges appear as discontinuities in the normals. Thus, the normal variations offer an intuitive geometric meaning in determining smoothness.

Instead of using position buffer, we adopt the information of the second hitpoints of the paths. This is because when specular reflections or glossy reflections occur in the scene, the positional distribution and normal distribution of the second intersection points are particularly useful. The second position Gaussian function G_{sp} measures the positional distribution of the second hitpoints of the paths in a path tracer, which is defined as:

$$G_{sp}(sp(X), sp(\xi)) = \exp\{-\frac{1}{2}(\frac{\|sp(X) - sp(\xi)\|}{\sigma_{sp}})^2\} \quad (6)$$

The second normal Gaussian function G_{sn} is defined similarly:

$$G_{sn}(sn(X), sn(\xi)) = \exp\{-\frac{1}{2}(\frac{\|sn(X) - sn(\xi)\|}{\sigma_{sn}})^2\} \quad (7)$$

In these two functions, sp and sn return the world position and the normal of the second hitpoint of a traced path, respectively. The geometric information of the second hitpoint is utilized to detect high-frequency features on specular or glossy surface and prevent filtering across reflected discontinuities.

Finally, the total weight of our filtering method is the integration of all the aforementioned Gaussian functions:

$$w(X, \xi) = G_p G_c G_n G_{sp} G_{sn} \quad (8)$$

3.2. Adaptive Size Selection

The filter's window size influences the results, and it is usually hard to select one size that fits to all regions of an image. The choice of a small size leads to the jaggy effect, whereas a too large one increases filtering time, leading to performance degradation. For a more efficient reconstruction, it is better to compute a per-pixel size. We make use of adaptive varying size for the filter according to the color distribution of pixels. The color variance can be used to indicate diffuse or specular reflected surfaces. Diffuse surface usually has high variance since it is characterized by light being reflected in all directions and the reflected color varies widely, while for glossy surface, light is reflected in a small cone around the mirror-reflection direction, therefore, most of the samples will have similar reflected colors and the variance will be low. In our setting, the blur size of the pixel is made linear with its color variance. If the sampling rate is low, clustering the samples from the nearby pixels might be necessary to make the variance more smooth and reliable. Compared to the fixed sized filter, variable sized filter shows better performance, since the run time of the algorithm is directly related to the size of the filter.

3.3. Outlier Suppression

When high dynamic range images are used, some samples can be several orders of magnitude larger than their neighbors, and this spiked noise cannot be easily eliminated due to improper blending. To address this, we employ two modifications: One is an outlier replacing operator to detect and replace "abnormal" data with the average, and the other is based on [XP05] that we use an estimated range value of the current pixel instead of the true value when determining the range kernel.

3.4. Filtering Framework

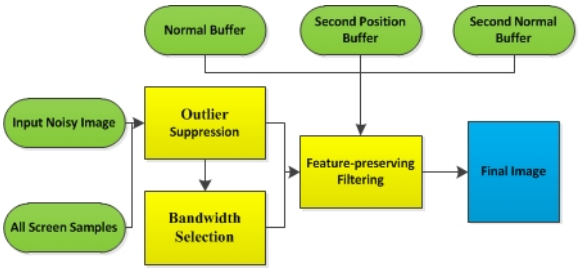

Figure 1: *A diagram showing our main building blocks.*

A diagram showing our main building blocks is given in Figure 1. Our method takes a noisy Monte Carlo path traced image and all the screen samples, augmented with three auxiliary input buffers: normal buffer, second position buffer and second normal buffer, and generates a smooth output image without disturbing noise. We store all data in linear device

memory allocated via CUDA. We firstly perform outlier suppression on the original noisy image and all screen samples to eliminate spiked noise, and select window size for each pixel using outlier suppressed data. All these procedures are accelerated by GPU using CUDA. In our framework, we simply filter the final path traced images, instead of filtering direct and indirect illumination separately.

4. Implementation and Results

We have implemented our filtering method using nVidia CUDA programming language, and integrated it as a post-processing step into a CPU version path tracer. Currently, the standard deviation parameters of the Gaussian functions are set manually. All results shown here are computed on a PC with Intel Core 2 Quad at 2.5 GHz, 2G RAM, and a GeForce GTS 250 graphics card supporting CUDA 3.2. All images in this paper are generated using a final output resolution of 512^2 except where explicitly noted.

To validate our algorithm, we have run it on a variety of complex scenes, and compared it with the state-of-the-art filtering algorithms. Figure 2 compares our adaptive filter with edge-avoiding À-Trous filter for a scene consisting of a dragon model facing a mirror-reflected suface. The reference image using 10000 spp (samples per pixel) is shown in Figure 2(d), and takes more than 5 hours to render. Although our solution is several orders of magnitude faster, this is close to the reference image. Note that our approach can effectively capture the mirror-reflected scene features without noticeable noise, while edge-avoiding À-Trous filter fails to do so.

Figure 3 shows the effects of adaptive varying blur size and fixed size. This scene is illuminated by the high-frequency Eucalyptus Grove environment map. Obviously, for a narrow filter, the image is not sufficiently smoothed; hence distracting light blotches are still evident. On the other hand, a wide filter tends to take a lot of filtering time, as shown in table 1, since the filtering time grows with the size of filter. Our point-wise adaptive filter can generate lower costs without losing image quality by diminishing size in low variance regions.

Table 1: *The illustrations and comparisons of timing performance for Figure 3. The filtering time grows with the window size of the filters, while the variable window size can lower the filtering time without losing image quality. The reference path traced image at 40000 spp takes more than one hour to render.*

Input	Filtering time (ms)				Ref.
	4	8	16	vary	
7s	145.6	255.2	654.7	187.0	>1h

In Figure 4, we show a Cornell Box scene containing a mirror ball and a glossy bunny. The benefit of including OS can be seen when comparing Figure 4(b) against Figure 4(c). Note that visually disturbing noise still exists in Figure 4(b). We also compare against À-Trous filter, as well as against the reference image rendered at 10000 spp in 7 hours. Observe that edge-avoiding À-Trous filter incorrectly blurs reflected scene features on the mirror ball, while ours retains well since we consider information of both the first and second hit-points. Close-ups presented in Figure 4 clearly demonstrate the effectiveness of our method.

Table 2 provides the performance breakdown for different scenes with different sampling rate (SPP). Each row contains timings for: adaptive size selection (SS), outlier suppression (OS), and feature-preserving filtering (FF). The final column (Total) is the sum of each step's timing. Note that the running time of the outlier replacing operator is included in SS. From the data, we can see that FF usually takes a significant ratio of the running time. The running time of SS depends on the sampling rate since we have to traverse all of the samples, while the running time of FF depends mostly on the scene's structure. For instance, the scene in Figure 3 contains only a small portion of diffuse surfaces which need large window size, and therefore the running time of FF is comparatively low. The variation of OS's running time is very small since we have used a fixed Gaussian filter in this step.

Table 2: *Performance (in ms) breakdown for different scenes using our filtering method.*

Scene	SPP	SS	OS	FF	Total
Figure 2	16	9.5	50.9	275.8	336.2
Figure 3	64	45.9	50.9	90.2	187.0
Figure 4	16	9.4	51.0	203.8	264.2

5. Conclusion and Future Work

In this paper, we have described a general framework for performing feature-preserving filtering on Monte Carlo path traced images from a given set of samples. One key point of our method is the choice of three additional geometric properties as the guidance to retain high-frequency details in the scene when smoothing out undesirable noise. A per-pixel filter size selection scheme is used to reduce the running time. Moreover, our method can be easily integrated in any Monte Carlo based rendering systems as a post-processing stage, and it may also be useful for real-time global illumination techniques to smooth out artifacts [RDGK12].

Several aspects of the proposed method can be improved in future research. We would like to enrich our system by supporting more lighting effects, such as caustics and participating media. We also hope to find a suitable solution to automatically set the standard deviation parameters in the bilateral filter, without introducing too much computational overhead.

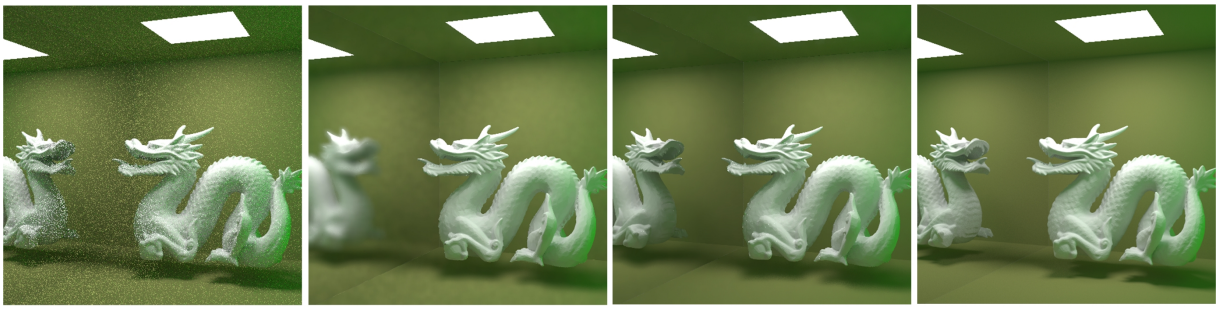

Figure 2: *Visual comparison between edge-avoiding À-Trous filter and ours. This path traced scene contains a dragon model facing a mirror-reflected surface. From left to right: input noisy image rendering at 16 spp (a), filtering results of edge-avoiding À-Trous filter (b) and ours (c). The reference image is rendered at 10000 spp, and takes more than 5 hours (d).*

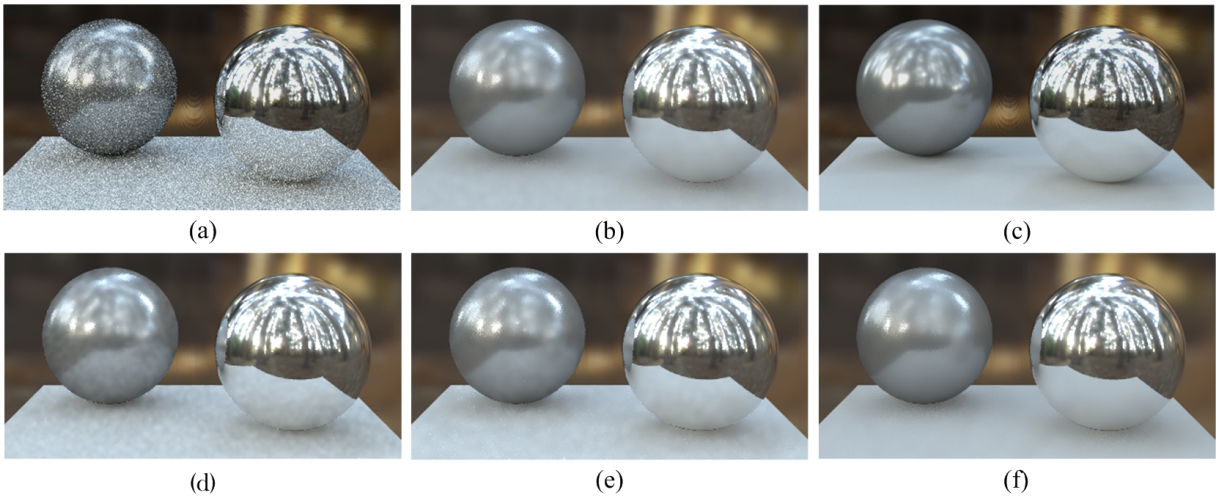

Figure 3: *Comparison between filters with fixed window size and with variable window size. The first row shows the noisy image rendered at 64 spp (a), the filtering results of our variable window sized filter (b), and the reference image rendered at 40000 spp (> 1 hours) (c). The second row gives results of fixed window sized filters with size 4 (d), 8 (e), and 16 (f), respectively.*

6. Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback. This work was supported by Natural Science Foundation of Jiangsu Province of China (BK2010373) and Postgraduate Cultivation and Innovation Foundation of Jiangsu Province (CXZZ11_0045). The Dragon and Bunny models were provided by the Stanford Computer Graphics Laboratory, and the Eucalyptus Grove environment map was provided by Paul Debevec.

References

[Bar02] BARASH D.: A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 6 (June 2002), 844–847. 2

[BCM05] BUADES A., COLL B., MOREL J.: A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation 4*, 2 (2005), 490–530. 1

[BEM11] BAUSZAT P., EISEMANN M., MAGNOR M.: Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering (EGSR)) 30*, 4 (June 2011), 1361–1368. 2

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (1984), SIGGRAPH '84, ACM, pp. 137–145. 2

[CT03] CHOUDHURY P., TUMBLIN J.: The trilateral filter for high contrast images and meshes. In *Proceedings of the 14th Eurographics workshop on Rendering* (2003), EGRW '03, Eurographics Association, pp. 186–196. 2

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the*

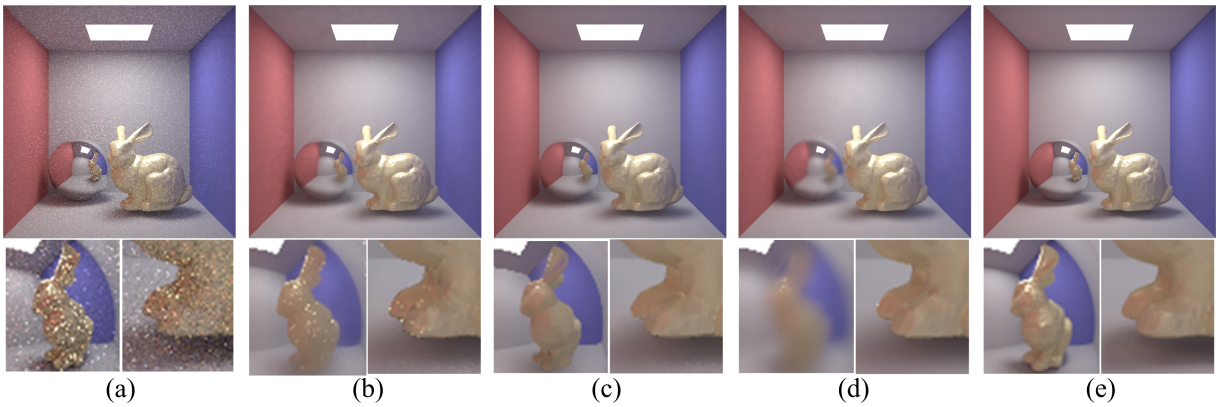

Figure 4: *Results of denoising the Cornell Box scene containing a mirror ball and a glossy bunny. The input image is rendered at 16 spp (a). Here we compare results of our filter without OS (outlier suppression) (b) and with OS (c), edge-avoiding À-Trous filter (d), and the reference image rendered at 10000 spp in 7 hours (e). Close-ups demonstrate the effectiveness of our method.*

29th annual conference on Computer graphics and interactive techniques (2002), SIGGRAPH '02, ACM, pp. 257–266. 2

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P. A.: Edge-avoiding à-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (2010), HPG '10, Eurographics Association, pp. 67–75. 2, 3

[DWR10] DECORO C., WEYRICH T., RUSINKIEWICZ S.: Density-based outlier rejection in Monte Carlo rendering. *Computer Graphics Forum (Proc. Pacific Graphics) 29*, 7 (Sept. 2010). 2

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 673–678. 2, 3

[HDL11] HANIKA J., DAMMERTZ H., LENSCH H.: Edge-optimized à-trous wavelets for local contrast enhancement with robust denoising. *Computer Graphics Forum 30*, 7 (2011), 1879–1886. 2

[JC95] JENSEN H. W., CHRISTENSEN N. J.: Optimizing path tracing using noise reduction filters. In *Proceedings of WSCG* (1995), vol. 1995. 2

[LR90] LEE M. E., REDNER R. A.: Filtering: A note on the use of nonlinear filtering in computer graphics. *IEEE Comput. Graph. Appl. 10*, 3 (May 1990), 23–29. 2

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proc. Eurographics Symposium on Rendering* (2007). 2

[McC99] MCCOOL M. D.: Anisotropic diffusion for monte carlo noise reduction. *ACM Trans. Graph. 18*, 2 (Apr. 1999), 171–194. 1, 2

[PD09] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. *Int. J. Comput. Vision 81*, 1 (Jan. 2009), 24–52. 2

[PKTD08] PARIS S., KORNPROBST P., TUMBLIN J., DURAND F.: A gentle introduction to bilateral filtering and its applications. In *ACM SIGGRAPH 2008 classes* (2008), SIGGRAPH '08, ACM, pp. 1:1–1:50. 1

[PM90] PERONA P., MALIK J.: Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell. 12*, 7 (jul 1990), 629 –639. 2

[PSA*04] PETSCHNIGG G., SZELISKI R., AGRAWALA M., COHEN M., HOPPE H., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 664–672. 2, 3

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The state of the art in interactive global illumination. *Comput. Graph. Forum 31*, 1 (2012), 160–188. 4

[RW94] RUSHMEIER H. E., WARD G. J.: Energy preserving non-linear filters. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), SIGGRAPH '94, ACM, pp. 131–138. 2

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (June 2012), 18:1–18:15. 2

[SGNS07] SLOAN P.-P., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (2007), PG '07, IEEE Computer Society, pp. 97–105. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision* (1998), ICCV '98, IEEE Computer Society, pp. 839–845. 1, 2

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive global illumination using fast ray tracing. In *Proceedings of the 13th Eurographics workshop on Rendering* (2002), EGRW '02, Eurographics Association. 2

[XP05] XU R., PATTANAIK S. N.: A novel monte carlo noise reduction operator. *IEEE Comput. Graph. Appl. 25*, 2 (Mar. 2005), 31–35. 2, 3