Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Adaptive Sampling for Geometry-aware Reconstruction Filters

Pablo Bauszat[1] and Martin Eisemann[1] and Marcus Magnor[1]

[1]Computer Graphics Lab, TU Braunschweig, Germany

Abstract

We present an adaptive sampling scheme for Monte-Carlo-based renderers with the aim to support geometry-aware filtering techniques for interactive computation of global illumination. While sophisticated filtering techniques for homogeneous areas can already produce high-quality results with as few as one sample per pixel, these approaches lack the ability to filter sufficiently in the vicinity of complex geometric structures. The result are visible artifacts in the final rendering result. Our sampling scheme distributes the samples for the indirect illumination in the image plane according to the necessity of a geometry-aware filtering. We show how to implement our scheme efficiently on current graphics hardware and how to combine it with a sophisticated filtering in order to achieve high-quality interactive frame rates for global illumination simulations. The resulting computational overhead is only in the range of a few milliseconds, making our approach suitable for real-time implementations.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Raytracing—Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Computer Graphics [I.3.3]: Picture/Image Generation—

1. Introduction

Physically-motivated renderers like path tracing solve the integral of the rendering equation [Kaj86] by Monte-Carlo techniques, i.e. the result of several samples is accumulated to approximate the integral [Kel97]. Image fidelity can be heavily affected by how many samples are used and the way they are distributed. As the necessary amount of samples per pixel can be in the range of hundreds or thousands, it is not possible with current commodity hardware to compute the final image in real-time. Adaptive sampling strategies [Whi80, Mit87, BM98, HJW*08] usually strive to focus effort only in areas where it is likely to be needed, which is often based on the convergence or variance. The benefit of these techniques is the ability to rapidly decrease the number of necessary samples and that they eventually converge to the correct result. A current drawback is that even the most sophisticated sampling schemes are still far away from being efficient enough to allow for real-time rendering on commodity hardware. A different approach are filtering techniques that increase image quality by averaging over several neighbouring pixels [WKB*02, SIMP06, LSK*07, DSHL10]. The weighted average of the samples is based on the geometric similarity of neighbouring samples. Classic adaptive sampling strategies are not designed to support these filtering strategies as their main goal is the variance reduction in the approximation of the rendering equation without any knowledge of the later filtering. Some approaches exist, which do incorporate this knowledge, e.g. in the field of virtual point lights (VPLs). By splitting the VPLs into disjoint sets for neighbouring pixels and recombining their contribution afterwards in a geometry-aware manner the correct rendering integral can be reconstructed in many cases [WKB*02,LSK*07,RGK*08]. Unfortunately, they are are limited to certain constraints, e.g. static scenes [WKB*02, LSK*07] or precise parameter adjustment for varying scenes [RGK*08]. Bias compensation and redistribution for VPLs also hinder their usage and increase code complexity. In addition, current interactive approaches working in image space still suffer from the inability to filter correctly in regions of high geometric variance [DSHL10]. Visually disturbing outliers or ringing artifacts are the result.

In our approach, we design an adaptive sampling strategy that directly supports image-based filtering techniques to prevent these artifacts. Therefore, we evaluate scene com-

DOI: 10.2312/PE/VMV/VMV11/183-190

plexity for each pixel before sampling. In contrast to other approaches, we do not have the necessity to base the distribution of samples or VPLs on the change of variance in the image as it is commonly the case. Our approach computes the distribution of the allowed ray budget, i.e. the number of rays that can be traced per frame, before any shading ray is cast which turns out to be extremely beneficial for GPU implementation and provides simple means for different levels of rendering quality based on the capabilities of the underlying hardware. By computing more samples in geometrically complex areas, we show that this approach can directly support state-of-the-art filtering techniques and fits well onto current graphics hardware, allowing for high-quality interactive path tracing of arbitrary, even dynamic, scenes.

The rest of the paper is organized as follows. After reviewing relevant previous work in Section 2, we give a short introduction into geometry-aware filtering and specifically into the edge-avoiding À-Trous wavelet transform for fast global illumination filtering [DSHL10] which we use exemplarily to show the benefit of our approach, Section 3. In Section 4, we describe our approach as an efficient way to support geometry-aware filtering of the incident indirect illumination for high-quality interactive path tracing. Experimental evaluation results for a variety of different test scenes are presented in Section 5, before we discuss limitations and conclude with Section 6.

2. Related Work

Fast ray tracing In order to realize any interactive path tracing, the basic necessity is a fast ray tracing kernel. Exploiting fine-grained parallelism and coherency in the rays, the pioneering work of Wald *et al.* [Wal04] showed how interactive ray tracing is possible on standard PCs. Unfortunately, ray coherence is not given in global illumination simulations which is needed for such approaches to work efficiently. Recently Aila *et al.* [AL09] described how to exploit the architecture of current GPUs for fast ray casting. To accelerate the general light transport, Boulos *et al.* [BWB08] propose to reintroduce coherency by reordering of the ray packets. Another approach is to introduce acceleration data structures specifically designed for incoherent rays [DHK08, EG08, Tsa09]. Even with these sophisticated rendering techniques, current hardware is not fast enough to generate noise-free path traced images at interactive frame rates, but they offer a good basis to build upon.

Adaptive Sampling There is a vast amount of literature dealing with sampling and reconstruction and we will only describe the most relevant work here. Excellent surveys can also be found in [DBB06, PH10].

In his seminal work on recursive ray tracing, Whitted [Whi80] proposed an adaptive sampling strategy which first samples the image plane on a regular grid and then subdivides the squares based on the difference between the sampled values on the corners of the square. Unfortunately, such regular subdivisions result in structured aliasing patterns visible in the rendered image. Adaptive stochastic approaches create a higher density of samples in areas where it is most needed, e.g. high variance or contrast [Mit87, BM98]. As the samples are stochastically distributed, aliasing patterns are less visible. Early on, Kajiya [Kaj86] proposed multi-dimensional sampling but was unsatisfied with his results. It took more than twenty years before a succesful approach was finally presented by Hachisuka *et al.* [HJW*08]. Unfortunately, these adaptive sampling patterns usually assume that rendering time is not restricted in any way. For interactive applications a limited sampling budget of only a few million samples per frame is currently available. These are too few to reconstruct noise-free results with these approaches for standard resolutions of one megapixel and more.

Filtering Techniques for Global Illumination The unifying idea behind edge-avoiding, or geometry-aware filtering, in global illumination simulations is to take weighted averages of nearby samples based on their geometric properties, such as normals or position in space. Wald *et al.* [WKB*02] made use of the discontinuity buffer [Kel97] to prevent filtering across edges in the scene. Similarly, Laine *et al.* [LSK*07] use interleaved sampling and edge-aware boxfiltering for a $n \times m$ pixel region to filter the image. Just recently, Dammertz *et al.* [DSHL10] proposed the edge-avoiding À-Trous wavelet transform for fast edge-aware filtering of the incident illumination. The filter is fast to compute, fits well to graphics hardware and produces very good results, given the few milliseconds it takes to compute the filter. One caveat to using this approximation technique is that it introduces bias in the final rendering result [KA91]. Artifacts such as blurry patches, outliers or ringing replace the noise seen in pure Monte Carlo ray tracing based approaches. A theoretical generalization of these approaches is the *joint* or *cross-bilateral* filter introduced in [PSA*04, ED04]. Unfortunately, to our best knowledge, no technique exists, which can compute the exact cross-bilateral filter for larger filter kernels in real-time. Just recently, Bauszat *et al.* [BEM11] proposed to use a guided image filter to estimate the incident indirect illumination. While their results are even better than a full cross-bilateral filter, it seems currently not possible to incorporate an adaptive sampling scheme into this technique without loosing real time capability.

3. Geometry-aware filtering

The idea behind geometry-aware filtering is the assumption that samples of the incident lighting captured at neigboring pixels may contribute to the pixel under consideration to virtually increase the number of samples. This assumption holds in most cases as long as the surfaces show near-lambertian properties and the geometric characteristics of

neighboring pixels are similar. The similarity is computed by a cost function based on at least the normal and depth deviation [WKB*02, LSK*07, DSHL10] to prevent filtering across edges or depth discontinuities.

The general concept behind this kind of filtering is described by the so-called *joint* or *cross bilateral filter* [ED04, PSA*04]. In the classic bilateral filter algorithm [TM98, SB97] similarity between neighbouring pixels is based on two components, a spatial, i.e., pixel distance, and a perceptional similarity, i.e., intensity value. The first weighting function G_{σ_s} is the spatial weighting, equal to a standard gaussian filter. The second weighting function G_{σ_r} is also called range weight, where the coefficient at each position q in the filter kernel is based on the difference between the intensity values at the pixel under consideration p and the offset pixel at position q respectively. Combining both results in an edge-aware filter kernel.

The cross bilateral filter extends this idea by decoupling the weighting functions from the input image. Instead of basing the filter function on relations in the input image I, it is based on arbitrary input images to compute the weighting terms. E.g., the edge-avoiding geometry-aware filter proposed by Dammertz *et al.* [DSHL10] uses the following formulation to denoise a path traced image I at pixel position p:

$$F(I_p) = \frac{1}{W_p} \sum_{q \in NH_p} I_q \cdot h(q) \cdot G(p,q) \;, \tag{1}$$

with

$$G(p,q) = G_{\sigma_r}(p,q) \cdot G_{\sigma_n}(p,q) \cdot G_{\sigma_x}(p,q) \tag{2}$$

$$G_{\sigma_r}(p,q) = e^{\left(-\frac{||I_p - I_q||^2}{\sigma_r^2}\right)} \tag{3}$$

$$G_{\sigma_n}(p,q) = e^{\left(-\frac{||N_p - N_q||^2}{\sigma_n^2}\right)} \tag{4}$$

$$G_{\sigma_x}(p,q) = e^{\left(-\frac{||D_p - D_q||^2}{\sigma_x^2}\right)} \tag{5}$$

In this filter kernel the spatial component $h(q)$ is defined by a B_3 spline interpolation $(\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16})$. $G_{\sigma_r}(p,q)$ is the range kernel for the color similarity used to retain high-frequency informations such as shadows that cannot be detected by geometric information. $G_{\sigma_n}(p,q)$, the normal similarity and $G_{\sigma_x}(p,q)$, the similarity of the hitpoint in euclidian space. The normalization factor W_p is defined as the sum of all weights in the neighbourhood

$$W_p = \sum_{q \in NH_p} h(q) \cdot G_{\sigma_s}(p,q) \cdot G_{\sigma_r}(p,q) \cdot G_{\sigma_n}(p,q) \tag{6}$$

3.1. Edge-Avoiding À-Trous wavelet transform

The previously defined geometry-aware filter is computationally very intensive. If low sampling rates are used, a radius of up to 48 pixels might be necessary to remove noise sufficiently from the path traced image. In order to speed up the computation Dammertz *et al.* [DSHL10] proposed to use an iterative *algorithme À-Trous* (with holes) [HKMMT89]. The idea is to repeatedly convolve the image with a kernel whose number of non-zero-coefficients stays constant but the initial filter is spread in size by a factor of 2^{level} and filled with zeros. The computational complexity stays constant for each level. An explanatory example is given in Figure 1. More details are given in [DSHL10].

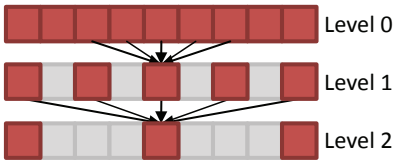


Figure 1: Visualization of the À-Trous wavelet transform for three levels. Arrows indicate non-zero entries in the filter kernel. Gray pixels are those not taken into account by the À-Trous wavelet transform.

This approximation works fast and produces good results in general. In regions of complex geometry, however, the weights for neighboring samples will be very low, lowering the noise reduction capability of the filter and resulting in visible outliers in the final rendering result.

4. Adaptive Sampling Scheme

To supplement the image-space filtering, we propose an adaptive sampling scheme for the incoming, indirect illumination that distributes more samples to pixels in variant neighbourhoods, which cannot fully benefit from the filtering. The proposed adaptive sampling scheme specifically targets interactive applications and is designed for execution on parallel architectures such as the GPU. While in our case the algorithm is used for the indirect illumination only, it is defined generally and, thus, can easily be adopted for other applications e.g. adaptive sampling of pixel domains on the image plane.

In contrary to common adaptive sampling schemes where the sampling rate for each pixel is computed locally, we compute the sampling rate for each pixel by distributing a fixed sample budget over the complete image. This is preferable for interactive applications which prefer constant frame times, because using only the local importance information for each pixel can lead to overall sample counts that vary heavily depending on the scene view point.

We propose a 3-step algorithm: In the first step, an *importance map* is generated indicating for each pixel its sampling importance. Second, the sampling rate for each pixel is computed using the importance map as input and the results are stored in a resolution-sized integer buffer. After computing the number of samples per pixel, the samples for the whole

(a) (b) (c)

Figure 2: Visualization of importance maps with a filter radius of 2x2. (a) Generated using only the normal weight, (b) using only the position weight and (c) using the normal- and position weight combined.

image are finally generated and processed using a batch system.

4.1. Importance map generation

To compute the importance of each pixel, we consider the geometry-aware filtering kernel given in Equation (1). The factor W_p can be used as an indicator giving information about how much a pixel is influenced by its neighbourhood during the filtering process. The higher the value, the more information is gathered from the neighbouring pixels. In contrary, pixels with a lower factor benefit less from image-spaced filtering. Therefore, we want to distribute more samples to pixels with a lower normalization factor. Because we compute the importance map before any color samples are taken, we do not have any color information and thus, we compute the importance value for each pixel based on the normal and position information only. To compute an importance value in the range $[0,1]$, we divide the factor W_p (without the color weighting) by the maximum weight that could be achieved in a filter kernel, which is simply the number of pixels in the neighbourhood. Figure 2 shows a visualization of importance maps based on a) the normal only, b) the position only and c) the combined information. Notice, that in the images the normalized weighting sums are visualized directly and thus, bright areas represent pixels with a high W_p and can benefit more from filtering. For the final sampling rate computation, we use $1 - W_p$ to compute the actual pixel importance value. The radius of the neighbourhood can be used to adjust the influence range of an edge to its neighbourhood. The larger the radius, the more pixels near an edge are assigned a higher priority. Because computing larger filter neighborhoods is very time consuming, we usually use a filter radius of $2-4$ which is sufficient to detect high variant areas and edges and works well in practice.

4.2. Sampling rate computation

After computing the importance map, the actual number of samples for each pixel is computed. Interestingly, the problem of distributing a fixed integer number over multiple elements using floating-point weights or *quotas* is similar to the problem of apportionment of seats in political elections. For computing the sampling rate n for each pixel, we use the equation

$$n = minSamples + \lfloor \frac{s \cdot q}{t} \rfloor \quad (7)$$

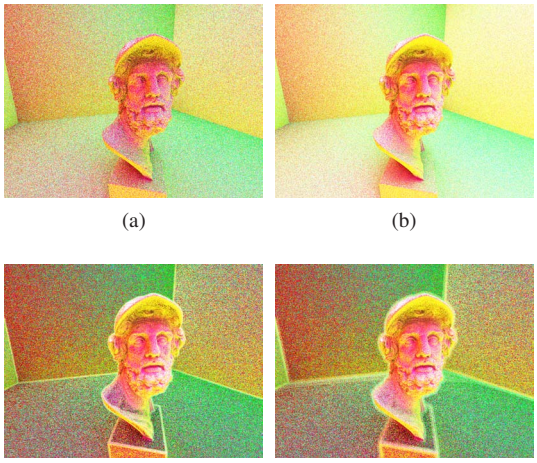
$$s = sampleBudget - (minSamples \cdot numPixels) \quad (8)$$

where s is the total number of distributable samples for the image, q is the weight or quota of the pixel and t is the total sum of all quotas in the importance map. *minSamples* is the minimum number of samples computed per pixel. The fixed sample budget is simply computed by multiplying the uniform sampling rate by the resolution. Notice, that the equation is equal to the first step of the largest remainder method (also known as the Hare-Niemeyer or Hamilton method) [NN08]. Unfortunately, the actual number of distributed samples is not equal but smaller than the total number of distributable samples for the uniform sampling due to rounding errors. To distribute the missing samples, it would be necessary to compute the left fraction for each quota and sort these fractions which is a costly operation on the GPU. Therefore, we decided to avoid these computational overhead and accept the fact that the adaptive sampling scheme does not generate the same but a slightly lower overall sample count as the uniform sampling approach. The computation of the sampling rate for each pixel and the previous computation of the total sum of all importance values can both be performed in parallel. The sum of the importance map is computed using a simple "Reduce" operation, which can be implemented efficiently on the GPU [HSO07].

4.3. Sample generation, processing and storing

After the sampling rate for each pixel is computed, the samples are finally generated and processed using a batch system. Starting at the first pixel, samples are generated until the batch buffer, which contains the samples, is full. In our implementation, we use a resolution-sized buffer which is also used for the primary rays when sampling the image plane. When the batch buffer is full, the samples are processed, the results are stored and the generation process is resumed until all samples have been processed.

To generate and store samples, a mapping is needed which associates a sample in the batch to a specific pixel. Therefore, we use a *sample-to-pixel buffer*, which contains the index of the associated pixel for each sample in the batch. To construct this buffer for each batch, a second buffer, the *sample-offset buffer*, is used which stores the starting index of the samples for each pixel. The sample-offset buffer is constructed once, after computing the number of samples for each pixel, using an exclusive scan on the buffer which contains the sampling rates. This can be efficiently computed on the GPU (see also [HSO07]). The sample-to-pixel buffer is created for each batch before the samples of the batch are



(a) (b) (c) (d)

Figure 3: Adaptive sampling of the indirect illumination. Top row: Uniform sampling using 4 samples (a) and 16 samples (b). Bottom row: Adaptive sampling using a minimum sample count of 1 sample per pixel and an overall sample budget of four times the resolution. The filter range is set to 4x4 (c) and 10x10 (d). Here the cropped unnormalized values are shown to emphasize the differences between the images.

generated. To fill the sample-to-pixel buffer in parallel, a thread is started for each pixel and by using the sample-offset buffer each thread can decide if its associated pixel has any samples inside the current batch and fill in the information if needed. The routine to store the samples must be implemented carefully and thread-safe to handle the case where multiple samples are stored for the same pixel in parallel.

Figure 3 shows a comparison between the uniform- and the adaptive sampling scheme. Notice, that in the images the adaptive sampling scheme successfully shifts samples from the homogenous areas to the areas near edges, improving the quality of these pixels. The homogenous areas may look undersampled in the adaptive approach compared to the uniform sampling, however, these are exactly the areas which will benefit greatly from filtering.

By modifying the routines that generate and process the samples of a batch, the algorithm can easily be adjusted for other sampling applications. E.g. the generation of hemisphere samples can simply be replaced by pixel domain sampling for adaptive sampling of the image plane.

5. Results and Discussion

In this section, we compare our adaptive scheme with a uniform sampling of the image plane with and without filtering. For all approaches, we use approximately the same number of samples. All measurements were done at a resolution of

(a) Ajaxbust (b) Dwarf (c) Grass

Figure 4: Reference images used for our comparisons.

Window size	Time in ms
2x2	8.3
4x4	26.2
6x6	50.7
8x8	84.9
10x10	128.7

Table 1: The table shows the overhead in milliseconds for computing the number of samples per pixel for the adaptive scheme.

1024×768 pixels. The reference images, showing the incident indirect illumination in Figure 4, were computed using 1024 samples per pixel. For the unfiltered images and the approach of Dammertz *et al.* [DSHL10], we use one sample for sampling the pixel domain and four samples for the indirect illumination. We compute two bounces for the indirect lighting. Due to the different characteristics of the direct and indirect lighting, we filter and compare only the incident indirect illumination. The whole rendering system was implemented on an AMD 5600+ 2.8 GHz Dual core system with 2 GB of RAM. The used GPU is an NVidia 285 GTX supporting CUDA 3.2 and computing capability 1.3.

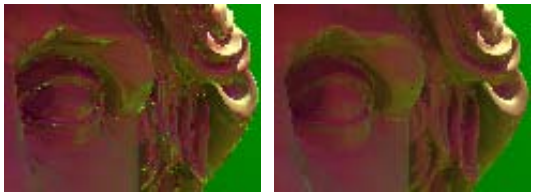
We first benchmarked the computational overhead introduced by the different steps of our adaptive sampling scheme. Table 1 shows the timings to compute the importance map and to redistribute the samples, i.e. computing the number of samples to be traced in our adaptive scheme per pixel. Though the timings increase quadratically with the radius, a small window of size 2×2 or 4×4 is generally enough to estimate a good distribution. We use a 4×4 window for all examples shown.

The required time for generating and storing samples in our adaptive scheme is mainly dependent on the number of samples that are generated and processed. Table 2 shows the additional execution time for the batch generation and storage of the samples for different sampling rates. The overhead of the adaptive sampling scheme is moderate and feasible for real-time applications, provided that a small window is used for computing the importance map.

In Figure 6 to 8, a comparison between the unfiltered, the classic À-Trous filter [DSHL10] and our approach is given using the mean squared error (MSE) metric [WB09]. The size of the filter was empirically set to the optimal value to

Hemisphere samples	Time in ms
4	2.3
16	3.6
64	8.5

Table 2: The table shows the overhead in milliseconds for generating and storing samples in the adaptive scheme.



(a) Uniform 2×2 sampling (b) Our adaptive sampling

Figure 5: (a) Outliers are visually disturbing artifacts that are hard to conceal but often appear if low sampling rates are used. (b) Our adaptive sampling scheme redistributes the samples to effectively mask these errors.

minimize the MSE, but is the same size for the classic À-Trous and our adaptive scheme. Our minimum sampling rate per pixel is set to two for the results in Figure 6 and 7 and one for the *Grass* scene in Figure 8.

As expected the noisy, adaptively sampled image shows a higher overall MSE, as we concentrate the samples on relatively small regions in the image. But the positive effect on the filtered image is substantial. For the *Grass* scene the MSE is reduced to 67% compared to the original À-Trous filter [DSHL10] with the same amount of samples. For the *Dwarf* scene the overall error is only slightly reduced compared to Dammertz *et al.*, but the error is more equally distributed over the image and outliers are drastically reduced as can be seen in the close-up views in Figure 7c and 7f. This outlier reduction is critical for high-quality rendering, as smaller errors might be masked by the direct illumination, but fireflies will still be visible, see Figure 5.

6. Conclusion

In this paper, we presented a novel, adaptive sampling scheme for interactive Monte Carlo global illumination simulations. In contrast to previous adaptive sampling schemes, our approach aims at supporting geometry-aware filtering methods, like the À-Trous wavelet transform filter by Dammertz *et al.* [DSHL10]. To our best knowledge, this is the first sampling scheme that supports the geometry-aware filter directly instead of trying to reduce the overall variance in the image. This is especially important for real-time applications where only a limited sample budget per frame is given. By computing an importance map before sampling the hemispheres for the indirect illumination, we are able to redistribute the available samples into critical areas of complex geometry while using less samples for more homogeneous areas which can be reconstructed faithfully by filtering.

A current limitation is that ringing artifacts which are caused by the À-Trous filter cannot be handled sufficiently by the adaptive sampling. An interesting future research direction would therefore be to find ways to combine the adaptive sampling scheme presented in this paper with the guided image filtering of Bauszat *et al.* [BEM11]. Another drawback is that for very complex and variant scenes the adaptive sampling will necessarily fail and boil down to a standard uniform distribution, as all areas are considered as equally important, e.g. in complex outdoor scenes.

References

[AL09] AILA T., LAINE S.: Understanding the efficiency of ray traversal on GPUs. In *Proceedings of High-Performance Graphics 2009* (2009), pp. 145–149. 2

[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 (2011), 1361–1368. 2, 6

[BM98] BOLIN M. R., MEYER G. W.: A perceptually based adaptive sampling algorithm. In *Annual Conference on Computer Graphics* (1998), pp. 299–309. 1, 2

[BWB08] BOULOS S., WALD I., BENTHIN C.: Adaptive Ray Packet Reordering. In *Proceedings of IEEE Symposium on Interactive Ray Tracing 2008* (2008), pp. 131–138. 2

[DBB06] DUTRE P., BALA K., BEKAERT P.: *Advanced Global Illumination*. A. K. Peters, Ltd., 2006. 2

[DHK08] DAMMERTZ H., HANIKA J., KELLER A.: Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays. In *Computer Graphics Forum (Proc. 19th Eurographics Symposium on Rendering)* (2008), pp. 1225–1234. 2

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P. A.: Edge-avoiding a-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (2010), pp. 67–75. 1, 2, 3, 5, 6, 7, 8

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

[EG08] ERNST M., GREINER G.: Multi bounding volume hierarchies. In *Proceedings of the IEEE/EG Symposium on Interactive Ray Tracing* (2008), pp. 35–40. 2

[HJW*08] HACHISUKA T., JAROSZ W., WEISTROFFER R. P., DALE K., HUMPHREYS G., ZWICKER M., JENSEN H. W.: Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Transactions on Graphics 27* (2008). 1, 2

[HKMMT89] HOLSCHNEIDER M., KRONLAND-MARTINET R., MORLET J., TCHAMITCHIAN P.: *A real-time algorithm for signal analysis with the help of the wavelet transform*. Springer-Verlag, 1989. 3

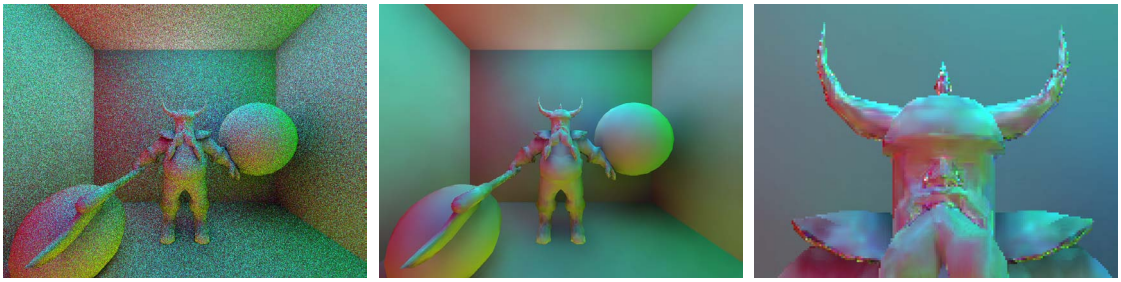
[HSO07] HARRIS M., SENGUPTA S., OWENS J. D.: Parallel prefix sum (scan) with CUDA. In *GPU Gems 3*. Addison Wesley, 2007, pp. 851–876. 4

[KA91] KIRK D., ARVO J.: Unbiased sampling techniques for image synthesis. In *Annual Conference on Computer Graphics* (1991), vol. 25, pp. 153–156. 2

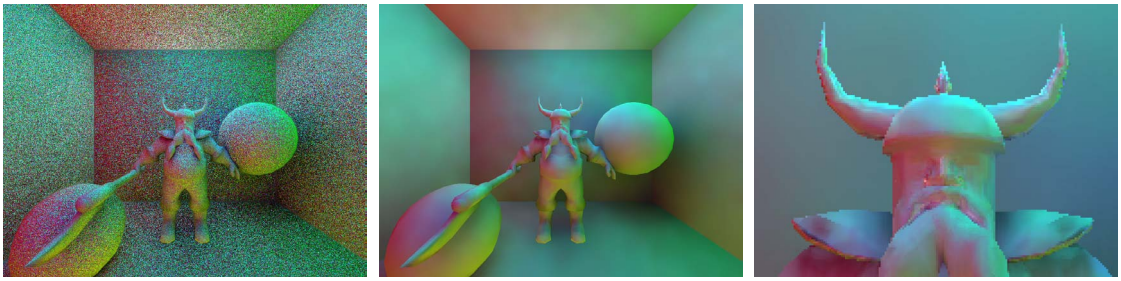
(a) Uniform sampling, MSE = 16219 (b) À-Trous filter, MSE = 329 (c) Close-Up of (b)

(d) Our adaptive sampling, MSE = 21565 (e) Our final result, MSE = 244 (f) Close-Up of (e)

Figure 6: (a) Noisy input image using a uniform 2×2 sampling, (b) the filtered result of (a) using the approach in [DSHL10], (c) Close-up of (b), (d) the noisy input image using our adaptive sampling scheme, (e) our the filtered result using the adaptively sampled image (d) as input for the filtering, (f) Close-up of (e)

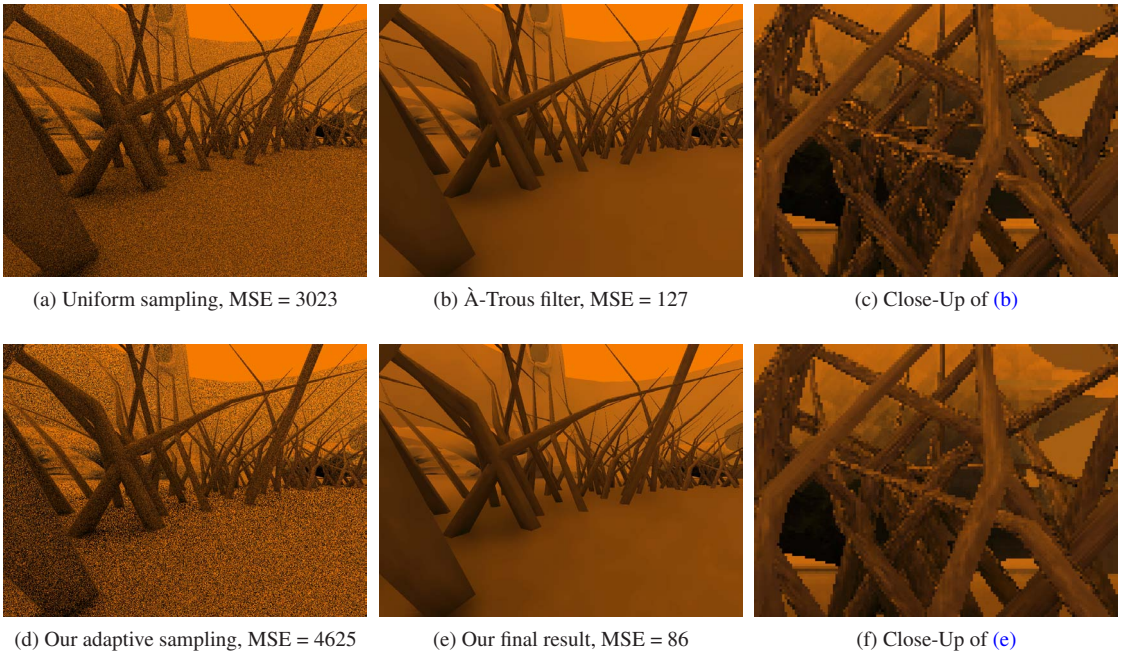


(a) Uniform sampling, MSE = 11356 (b) À-Trous filter, MSE = 275 (c) Close-Up of (b)



(d) Our adaptive sampling, MSE = 16090 (e) Our final result, MSE = 273 (f) Close-Up of (e)

Figure 7: (a) Noisy input image using a uniform 2×2 sampling, (b) the filtered result of (a) using the approach in [DSHL10], (c) Close-up of (b), (d) the noisy input image using our adaptive sampling scheme, (e) our the filtered result using the adaptively sampled image (d) as input for the filtering, (f) Close-up of (e)



(a) Uniform sampling, MSE = 3023 (b) À-Trous filter, MSE = 127 (c) Close-Up of (b)

(d) Our adaptive sampling, MSE = 4625 (e) Our final result, MSE = 86 (f) Close-Up of (e)

Figure 8: (a) Noisy input image using a uniform 2×2 sampling, (b) the filtered result of (a) using the approach in [DSHL10], (c) Close-up of (b), (d) the noisy input image using our adaptive sampling scheme, (e) our the filtered result using the adaptively sampled image (d) as input for the filtering, (f) Close-up of (e)

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 143–150. 1, 2

[Kel97] KELLER A.: *Quasi-Monte Carlo methods for photorealistic image synthesis*. PhD thesis, Department of Computer Science, University of Kaiserslauten, 1997. 1, 2

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007* (2007), Eurographics Association, pp. 277–286. 1, 2, 3

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *Annual Conference on Computer Graphics* (1987), vol. 21, pp. 65–72. 1, 2

[NN08] NIEMEYER H. F., NIEMEYER A. C.: Apportionment methods. *Mathematical Social Sciences 56*, 2 (2008), 240–253. 4

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010. 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 Transactions on Graphics 23* (2004), 664–672. 2, 3

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008) 27*, 5 (2008). 1

[SB97] SMITH S. M., BRADY J. M.: Susan a new approach to low level image processing. *Int. J. Comput. Vision 23* (May 1997), 45–78. 3

[SIMP06] SEGOVIA B., IEHL J. C., MITANCHEY R., PÉROCHE B.: Non-interleaved deferred shading of interleaved sample patterns. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware* (2006), ACM, pp. 53–60. 1

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proceedings of the International Conference on Computer Vision* (1998), pp. 839–846. 3

[Tsa09] TSAKOK J. A.: Faster incoherent rays: Multi-bvh ray stream tracing. In *Proceedings of the Conference on High Performance Graphics* (2009), pp. 151–158. 2

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004. 2

[WB09] WANG Z., BOVIK A.: Mean squared error: Love it or leave it? a new look at signal fidelity measures. *Signal Processing Magazine, IEEE 26*, 1 (2009), 98 –117. 5

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Commun. ACM 23* (June 1980), 343–349. 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), pp. 15–24. 1, 2, 3