EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

Volume 32 (2013), Number 2

Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms

Nima Khademi Kalantari and Pradeep Sen

University of California, Santa Barbara

Abstract

Monte Carlo rendering systems can produce important visual effects such as depth of field, motion blur, and area lighting, but the rendered images suffer from objectionable noise at low sampling rates. Although years of research in image processing has produced powerful denoising algorithms, most of them assume that the noise is spatially-invariant over the entire image and cannot be directly applied to denoise Monte Carlo rendering. In this paper, we propose a new approach that enables the use of any spatially-invariant image denoising technique to remove the noise in Monte Carlo renderings. Our key insight is to use a noise estimation metric to locally identify the amount of noise in different parts of the image, coupled with a multilevel algorithm that denoises the image in a spatially-varying manner using a standard denoising technique. We also propose a new way to perform adaptive sampling that uses the noise estimation metric to identify the noisy regions in which to place more samples. We show that our framework runs in a few seconds with modern denoising algorithms and produces results that outperform state-of-the-art techniques in Monte Carlo rendering.

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

1. Introduction

Monte Carlo (MC) rendering systems produce photorealistic images by evaluating multidimensional integrals that model the physical process of light transport in a complex scene. They estimate these integrals through random point-sample calculations that converge to the exact value of the integral with a variance that is reduced as $O(1/N)$ with the number of samples. This means that when the number of samples is small, we get error in the approximation of the integral which appears in the final image as noise. However, many samples need to be computed in order to reduce the variance to acceptable levels, which affects the applicability of Monte Carlo rendering systems for modern production environments.

The problem of reducing noise in Monte Carlo rendering has been the subject of extensive research since Cook et al. [CPC84] introduced the method to the rendering community. Recently, there has been renewed interest in tackling the problem with denoising/filtering approaches [ODR09, DSHL10, BEM11, RKZ11, SD12]. Curiously, most of these techniques use fairly simple filters (e.g., Gaussian kernels, cross-bilateral filters) for denoising. Perhaps the most sophisticated denoising algorithm is that of adaptive wavelet rendering (AWR) [ODR09] which modifies a simple, wavelet-based denoising method called soft-thresholding [Don95] to make it spatially-varying for the rendering application. However, this has several shortcomings, such as still requiring a sizable number of samples to produce reasonable results (at least 32 samples/pixel).

In the image processing community, on the other hand, there have been many powerful denoising algorithms proposed over the past few years, many of them inspired by soft thresholding. However, most of these algorithms assume that the noise is spatially-invariant and does not change globally across the image. This is not true in our application since the noise due to the random parameters in the MC rendering system is often local and spatially-varying [SD12]. Therefore, conventional wisdom in the rendering community is that these powerful methods cannot be applied to the problem of denoising MC rendering.

The goal of this work is to develop a framework to harness the power of standard, spatially-invariant denoising algorithms to address the problem of noise in Monte Carlo rendering. Rather than modifying a specific denoising algorithm for our rendering application as in the AWR method, we want to be able to use *any* spatially-invariant method for noise reduction. Our key insight is that we can do this by first applying a metric that accurately estimates the noise level's standard deviation at every local region in the image. Once this is known, we can then use a multilevel algorithm to efficiently apply a spatially-invariant denoising method to the entire image with a small set of different noise parameters, and combine the results to produce the final noise-free image. Our algorithm is mostly intended to be used as a post-

process filtering step (i.e., we first render our samples with standard Monte Carlo techniques and then denoise the result [ODR09, DSHL10, SD12]). However, the noise estimation metric can also be used to guide an adaptive sampling step. Our final algorithm is simple and fast, taking on the order of seconds to denoise a rendered image. Our results are also visibly better than those produced with state-of-the-art Monte Carlo approaches.

2. Previous Work in Monte Carlo Rendering

The use of Monte Carlo algorithms for rendering was introduced by Cook et al. [CPC84, Coo86] in their seminal work that extended standard Whitted ray-tracing [Whi80] to produce a variety of interesting effects, such as depth of field and motion blur. Since then, researchers have been exploring a variety of algorithms to reduce the MC noise and produce high-quality images in less time.

2.1. Adaptive and Reprojection Techniques

Several approaches use adaptive sampling to place more samples in the noisy regions and reduce noise. These techniques usually use a local measure of color variance to determine where to place more samples [Whi80, Mit87, RFS03]. However, it can often be difficult to tell if the high variance comes from the Monte Carlo noise or from scene detail [SD12]. Other adaptive algorithms place samples at the vertices of a grid or use a hierarchical data structure that increases resolution in areas that require more samples (e.g., [Kaj86]). These approaches interpolate between samples and have been extended to observe edge boundaries (e.g., [Guo98, BWG03]) to improve the rendered results.

More recently, the multidimensional adaptive sampling algorithm (MDAS) of Hachisuka et al. [HJW*08], adaptively samples the space by looking for rapidly changing sample values in all dimensions of integration. Although it can handle general Monte Carlo effects, it suffers from the curse of dimensionality as the number of dimensions increases, so the algorithm performs best when the number of parameters is low.

Soler et al. [SSD*09] proposed to leverage sparsity in the Fourier domain to place samples efficiently for scenes with depth of field, while Egan et al. [ETH*09, EDR11, EHDR11] addressed the noise of a single Monte Carlo effect with a specialized adaptive sampling scheme in conjunction with a sheared reconstruction filter. All of these approaches focus on specific Monte Carlo effects and are not general.

Finally, Lehtinen et al. [LAC*11] exploited the anisotropy of the integrand in MC algorithms by efficiently reusing (or reprojecting) an initial sparse set of samples to get a denser sampling. Their method can handle depth of field, motion blur and soft shadow effects. More recently, Lehtinen et al. [LALD12] extended the idea of reprojection to handle global illumination.

2.2. Filtering Approaches to Noise Removal

There have been a variety of MC noise filtering methods proposed in the past (e.g., [LR90, RW94, JC95]). In order to preserve edges and detail in the scene, some previous filtering methods have used anisotropic diffusion [McC99], bilateral filters [XP05, DSHL10], or guided image filters [BEM11]. Parametric methods for noise reduction have also been explored, such as the work of Meyer and Anderson [MA06] that removes indirect illumination noise from animated sequences by projecting the noisy images into a compressed PCA basis. Chen et al. [CWW*11] combined depth map with pixel variance map to guide sampling and proposed a multiscale reconstruction method for depth of field effect. These approaches often focus on specific effects and cannot handle general Monte Carlo effects.

Recently, Rousselle et al. [RKZ11] proposed an iterative, adaptive filtering method with two steps. In the first step, they select a Gaussian filter scale for each pixel after computing some samples that minimizes its reconstruction error. In the second step, they adaptively place new samples based on the filter scale selected and go back to the first step. Because of using a simple Gaussian filter in the reconstruction stage, they need enough samples/pixel to produce relatively noise-free results (32 samples/pixel in their cases).

In their paper, Rouselle et al. mention that they did not use more advanced image denoising techniques because of the commonly-held belief that these would not work for rendering since they assume a global noise model. Our framework, on the other hand, estimates the noise level at every pixel of the rendered image and therefore can leverage more advanced image denoising techniques. Thus, we can get better results for images with a high amount of noise.

Finally, Sen and Darabi [SD12] recently proposed a new filtering approach called random parameter filtering (RPF) where the functional dependency between the scene features and the random parameters are used to guide a cross-bilateral filter to remove noise but preserve detail. This algorithm can also handle general Monte Carlo effects, but the simple cross-bilateral filter used in RPF can often overblur scene detail or leave noise behind. Our method works better, as we will show in the results section.

2.3. Wavelet-based Rendering Methods

Our algorithm uses wavelets to estimate the noise level at every pixel, which have also been used in the past to address problems in Monte Carlo rendering. For example, Bolin and Meyer [BM98] proposed to use a wavelet hierarchy to model the visual system and adaptively place samples. Clarberg et al. [CJAMJ05] proposed to use a Haar wavelet basis to represent the environment map and surface BRDF, and demonstrated how to evaluate their product efficiently in the wavelet domain in order to perform efficient importance sampling. More recently, Sen and Darabi [SD11] used

compressed sensing to reconstruct the final image assuming that it was sparse in the wavelet basis.

Perhaps the most integrated use of wavelets in Monte Carlo rendering is the adaptive wavelet rendering (AWR) algorithm of Overbeck et al. [ODR09]. AWR uses a new adaptive sampling metric computed by subtracting the wavelet transform of sample variances from the wavelet magnitudes. For reconstruction (denoising), it uses a modification of a standard soft-thresholding denoising method [Don95] described in the next section. However, as pointed out by the authors, this creates a tradeoff between noise and wavelet artifacts, so images produced with less than 32 samples/pixel have visible wavelet ringing. Our algorithm, on the other hand, can use any modern denoising method and therefore does not have the artifacts of a simple soft-thresholding.

3. Background in Image Denoising

Image denoising is an important problem in the image processing community and many powerful methods have been developed to remove noise but preserve detail. These approaches typically formulate the problem by assuming the noise is Additive White Gaussian Noise (AWGN) added to the ground truth image $\mathbf{x}$:

$$y_p = x_p + n, \tag{1}$$

where subscript p is the pixel coordinate and n is zero-mean Gaussian noise, with standard deviation σ constant for all pixels in the image. The goal of denoising is therefore to estimate $\mathbf{x}$ given the noisy input image $\mathbf{y}$ and σ. This formulation inherently assumes stationary noise (constant over the entire frame), which is a reasonable assumption in image processing (where the noise typically comes from the sensor), but not in Monte Carlo rendering where the noise comes from the MC process itself.

One powerful tool for image denoising is the wavelet transform. The basic idea behind wavelet image denoising is to transform the noisy input image to the wavelet domain, apply a denoising step, and then transform it back to the spatial domain:

$$\tilde{\mathbf{x}} = \mathcal{W}^{-1}(\mathcal{D}(\mathcal{W}(\mathbf{y}), \sigma)),$$

where $\tilde{\mathbf{x}}$ is the estimate of the noise-free image and $\mathcal{D}$ is the denoising step that takes the noisy wavelet coefficients and the noise standard deviation σ as input. Two of the oldest wavelet denoising methods are hard/soft thresholding [DJ94, Don95]. In hard thresholding, the detail wavelet coefficients below a certain threshold are simply set to zero, i.e.,

$$\hat{\hat{x}}_k = \begin{cases} \hat{y}_k & \text{if} \quad \hat{y}_k > t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\hat{y}_k$ is the detail wavelet coefficient of the noisy image and $t = \sigma\sqrt{2\log(N)}$, where N is the number of pixels. In soft thresholding, also known as wavelet shrinkage, the detail coefficients are all subtracted (or "shrunk") by this fixed threshold value t:

$$\hat{\hat{x}}_k = \text{sign}(\hat{y}_k) \cdot \max(0, |\hat{y}_k| - t).$$

Note that both hard/soft thresholding set t using the standard deviation of the noise σ, which is constant for the entire image. In adaptive wavelet rendering, Overbeck et al. [ODR09] modify soft thresholding to use a spatially-varying threshold:

$$\hat{\hat{x}}_k = \text{sign}(\hat{y}_k) \cdot \max(0, |\hat{y}_k| - c_s\Delta_k),$$

where Δ_k is the standard deviation of coefficient k and c_s is a user-defined constant that trades off between smoothness and wavelet artifacts. Effectively, the Δ_k term tries to account for the fact that σ is spatially-varying in MC rendered images. Although hard/soft thresholding methods try to minimize the Mean Squared Error (MSE) between $\mathbf{x}$ and $\tilde{\mathbf{x}}$, the denoised image $\tilde{\mathbf{x}}$ can have objectionable wavelet artifacts because this minimization is done in the spatial domain. Furthermore, AWR uses the orthonormal wavelet basis Cohen-Daubechies-Feauveau (CDF) 9/7 which is not overcomplete. Previous work has shown that denoising with bases that are not overcomplete can exacerbate ringing artifacts [PSWS03]. The combination of soft thresholding and an orthonormal basis is why AWR typically needs at least 32 samples/pixel to produce reasonable images.

To avoid these problems we need to leverage more advanced image denoising techniques. In this paper, we examine two such algorithms, briefly described below. Readers interested in seeking a complete explanation of these techniques are referred to the original papers.

Bayes Least Squares-Gaussian Scale Mixtures (BLS-GSM) [PSWS03] – In this method, the noisy image is decomposed into pyramid subbands at different scales using an overcomplete basis. The noise-free coefficients are then estimated from the noisy ones using a Bayes least-square estimator. There are two major differences between this method and soft-thresholding in AWR which make it more robust: (1) an overcomplete basis is used to decompose the input image, and (2) the noise-free wavelet coefficients are estimated using information from neighbors, instead of simple subtraction of a value from the noisy coefficient.

Block-matching and 3D Filtering (BM3D) [DFKE06] - In this method, the noisy input image is first divided into overlapping blocks of fixed size. For each block, the most similar blocks in the image are found and stacked into a 3D array. The wavelet transform of this 3D array is taken and the noise is attenuated by hard thresholding (Eq. 2). Finally, all the denoised blocks are returned to their original positions and averaged together to produce the denoised image. This process is repeated to further suppress the noise.

As mentioned earlier, these powerful methods assume a fixed noise level σ and cannot be used directly for denoising MC rendering. Fig. 1 shows what happens when one tries to naïvely apply BM3D to this problem.

4. Our Multilevel Denoising Algorithm

In order to leverage powerful image denoising methods such as BLS-GSM and BM3D to denoise MC rendering, the noise

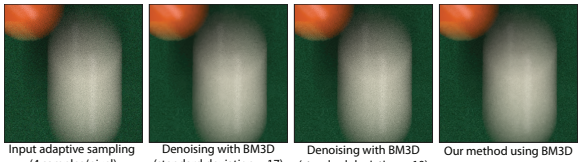


Figure 1: *We used BM3D, a state-of-the-art denoising method, to denoise the* POOLBALL *scene, an inset which is shown here. In this case, we used two different noise standard deviations (10 and 17). When the noise standard deviation is set to 17, the details on the green surface are removed but there is still noise on the balls. The result with a noise standard deviation of 10 keeps the detail on the green surface, but the balls are also noisy. Our complete approach can keep the details on the surface but removes much of the noise from the balls.*

standard deviation σ locally around every pixel in the image must be first accurately estimated. We must then find a way to use these spatially-invariant algorithms in a spatially-varying manner. In the sections that follow, we describe the core of our multilevel denoising algorithm.

4.1. Noise Estimation Metric

The goal of this section is to find a noise map $\mathcal{N}$ that estimates the noise standard deviation for every pixel $\tilde{\sigma}_p$. To do this, we leverage the median absolute deviation (MAD) introduced by Donoho and Johnstone [DJ94], which assumes that the noise is stationary inside a small window of pixels w around the pixel in question:

$$\tilde{\sigma}_{w(p)} = \frac{\text{median}(|\mathcal{D}_0^1|)}{0.6745}, \tag{3}$$

where $\mathcal{D}_0^1$ represents the diagonal detail coefficients of the finest level of the wavelet transform, 0.6745 is a scaling constant which Donoho and Johnstone found to work best in practice, and $\tilde{\sigma}_{w(p)}$ is the estimated standard deviation of the noise for the window around pixel p. This metric works reasonably well when the noise is locally stationary (see Fig. 2), because in practice these wavelet coefficients represent pure noise [DJ94]. The upward bias due to the presence of a signal at this finest level seen by $\mathcal{D}_0^1$ is addressed by taking the median, instead of taking the standard deviation of the coefficients directly.

However, the MAD metric is not enough to accurately estimate the per-pixel noise level because its assumption of local noise stationarity in a small window is violated when there are overlapping regions with different Monte Carlo effects. For example, there might be an in-focus object that overlaps the noisy blur of an out of focus object. This mixing of statistics [SD12] causes problems, as shown in Fig. 3 where artifacts occur around the edges of out of focus chess pieces if we only use $\tilde{\sigma}_{w(p)}$ as our noise map. To avoid this problem, the noise level for each pixel should be localized by also taking into account the standard deviation of the samples at each pixel $\tilde{\sigma}_{s(p)}$. As in previous approaches

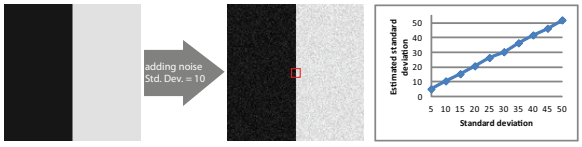
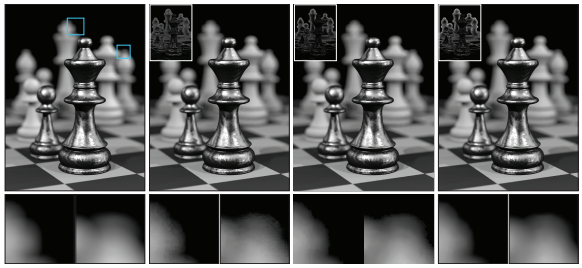


Figure 2: *To show the effectiveness of the MAD metric for locally stationary noise, we created the test image on the left with half the pixels set to 25 and the others set to 225. We then added Gaussian noise with different σ values (from 5 to 50) to the entire image, an example of which is shown in the middle with $\sigma = 10$. The metric in Eq. 3 is computed for the pixels in the box shown in red and used to estimate $\tilde{\sigma}_{w(p)}$. The plot shown on the right shows that $\tilde{\sigma}_{w(p)}$ accurately tracks the actual σ after averaging over 100 runs. On the other hand, other metrics such as variance and contrast used in the graphics community to estimate noise [Mit87, HJW*08] would fail in this case because the step edge here affects the results.*



(a) Reference (b) $\tilde{\sigma}_p = \tilde{\sigma}_{w(p)}$ (c) Ours, w/o dilation (d) Our metric

Figure 3: *The* CHESS *scene rendered with 8 samples/pixel is denoised when (b) $\tilde{\sigma}_{w(p)}$, (c) our metric without the dilation process, and (d) our final metric are used as the noise map. As seen, $\tilde{\sigma}_{w(p)}$ is more global and has problems when there is a mixture of statistics like the edges of out of focus chess pieces. However, the combination of $\tilde{\sigma}_{w(p)}$ with $\tilde{\sigma}_{s(p)}/\mu_{s(p)}$ in our metric works well everywhere. Our metric without dilation produces noisy results which shows that the dilation process helps to fill in the holes in the noise map.*

(e.g., [HJW*08]), we normalize the pixel samples' standard deviation by the average value of the samples in the pixel: $\tilde{\sigma}_{s(p)}/\mu_{s(p)}$. When we weight $\tilde{\sigma}_{w(p)}$ with this per-pixel standard deviation, we get our estimate of the standard deviation of the noise in each pixel:

$$\tilde{\sigma}_p = \left(\frac{\tilde{\sigma}_{s(p)}}{\mu_{s(p)}}\right)^{1/4} \tilde{\sigma}_{w(p)}, \tag{4}$$

where we used a weighted product to combine the two terms with weights equal to 1/4 and 1, giving slightly more weight to the $\tilde{\sigma}_{w(p)}$ term. The overall map might have "noisy" $\tilde{\sigma}_p$ estimates when using a small number of samples. Thus to make the final noise map smoother, we perform a 3-pixel dilation process on the $\tilde{\sigma}_{s(p)}$ and $\tilde{\sigma}_{w(p)}$ terms separately before combining them to form $\tilde{\sigma}_p$ to smooth out the map and fill holes. This dilation simply replaces each value with the maximum value in its 3×3 neighborhood around it, an example of which is shown in Fig. 5. Before we use this noise map with our multilevel algorithm, we should normalize it

by its maximum value and then scale it by $g \cdot \max(\tilde{\sigma}_{w(p)})$ where g is a global parameter that defines the smoothness of the results (larger values result in more smoothness), and the $\max(\tilde{\sigma}_{w(p)})$ term allows our denoising to adapt to the maximum noise level in the image. This allows us to keep the g value constant for all scenes.

4.2. Multilevel Denoising Framework

The noise map obtained in the previous section tells us how much each pixel should be denoised by giving us an estimate of the standard deviation of the noise level at every pixel, $\tilde{\sigma}_p$. We now discuss how to use spatially-invariant denoising algorithms to remove this varying noise. The naïve way to do this is to repeatedly denoise the entire image using all the different $\tilde{\sigma}_p$ values, and then select the pixels for the final image from each denoised result that had the appropriate noise level. This is expensive because there are many different $\tilde{\sigma}_p$ values so this would require invoking the denoising algorithm hundreds of thousands of times. Furthermore, trying to accelerate this process by selecting only a small region around each pixel to denoise does not work, since many denoising algorithms like BM3D [DFKE06] use information from other parts of the noisy image to denoise a particular block. Therefore we would still need to process a large region in the image for each $\tilde{\sigma}_p$ level to denoise the pixels.

Instead, we propose to accelerate this process by selecting a small subset of standard deviations to denoise which best represent the varying noise levels in the entire image. Our basic idea is to apply our denoising filter a small number of times using these representative parameters, and then select the pixels from these results that had similar noise for the final image. The idea of accelerating filters by repeatedly applying them to the entire image using a small discrete set of filter parameters and then combining the results back together is not new [DD02, RKZ11]. However, previous methods that do this have selected the filter parameters for the different levels independently of the input data.

Instead, we propose to find the subset of noise levels that best represent the varying $\tilde{\sigma}_p$ values in our noise map $\mathcal{N}$ by dedicating more denoising steps to the noise levels with more pixels in the noisy image. To do this, we do something similar to what is done in importance sampling when we want to select a set of samples from a specific probability density function (pdf). We first compute the cumulative distribution function (cdf) of the noise map $F(\mathcal{N})$ and then invert the cdf using a uniform sampling to get a discrete set of levels. An illustration of this is shown in Fig. 4. For a desired number of levels L, we compute the standard deviation for each discrete level i as:

$$\sigma(i) = F^{-1}(\frac{i-1}{L-1}),\ i = 1,\ldots,L. \tag{5}$$

The number of discrete levels for each scene is best selected according to the noise in the rendered image. We found empirically that $L = \lceil \max(\tilde{\sigma}_{w(p)})/10 \rceil$ gives the best results for our scenes.

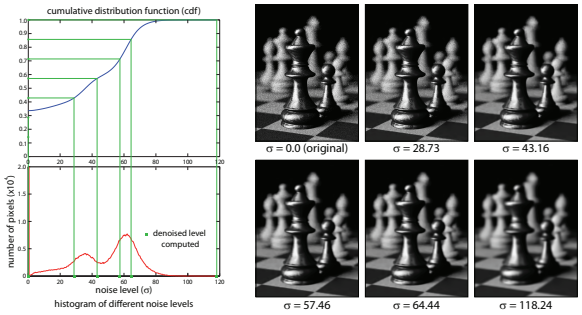


Figure 4: *On the left, we show how our algorithm decides which noise levels to denoise ($L = 8$). As in standard importance sampling, we uniformly sample the y-axis of the CDF with L samples and determine where the levels lie in terms of σ. This gives us a set of discrete levels that has been selected based on the number of pixels that require that amount of denoising. On the right, we show the denoised images at each of those levels. Note that 3 levels give the same standard deviation of 0 which makes the actual number of levels 6. The individual denoised layers are noisy or blurry which shows that applying a denoising method with a single global σ does not work. Our multilevel approach produces better results as shown in Fig. 6.*

Once we have the standard deviation of the noise for each of the L levels, we run the denoising algorithm with each parameter to produce L denoised images: $D_1,\ldots,D_L$. Our last step is then to combine these denoised images to compute the final image. To do this, we simply interpolate the pixel value for each final pixel using the corresponding value from the two levels that are closest in standard deviation to the given pixel. For example, for the case in Fig. 4, if the a pixel has $\tilde{\sigma}_p = 50$, the denoised images with $\sigma = 43.16$ and $\sigma = 57.46$ would be used to compute its final value. The simple alpha-blending process to compute each pixel in the final image $I(p)$ can be written as:

$$I(p) = \alpha \cdot D_k(p) + (1-\alpha) \cdot D_{k-1}(p),$$

where k is the minimum i that satisfies $\sigma(i) > \tilde{\sigma}_p$ and $\alpha = (\tilde{\sigma}_p - \sigma(k-1))/(\sigma(k) - \sigma(k-1))$.

4.3. Adaptive Sampling Using Noise Estimation

Given the accurate noise metric introduced in Sec. 4.1, an obvious extension is to use it to adaptively place samples for reducing the noise. Initially, one might want to use our noise map $\mathcal{N}$ as an importance map, but the two serve different purposes that make it a less-than-ideal choice. On the one hand, the noise map needs to be accurate in estimating the noise at every pixel since inaccuracy will show up as pixelated artifacts in the final result. In the case of the importance map, however, some inaccuracy is desirable to spread the samples around a bit more in the noisy regions in an effort to capture missing detail. For example, if we use $\mathcal{N}$ as the importance map to sample the CHESS scene, we would not be able to capture the boundaries of the out of focus chess

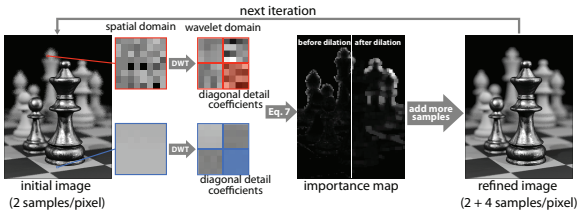


Figure 5: *Overview of our adaptive sampling algorithm.*

pieces correctly. Therefore, instead of using the variance of the samples in each pixel $\tilde{\sigma}^2_{s(p)}$ as we did earlier, we use the contrast metric proposed by Hachisuka et al. [HJW*08]:

$$\gamma_{s(w(p))} = \frac{1}{|s(w(p))|} \sum_{i \in w(p)} \frac{|s_i - \mu_{s(w(p))}|}{\mu_{s(w(p))}}, \tag{6}$$

where $s(w(p))$ contains all of the colors of the samples within window $w(p)$, $\mu_{s(w(p))}$ is the average of these samples, and $|s(w(p))|$ is the number of samples. Therefore the final importance map is:

$$\mathcal{M}_p = \sqrt{\gamma_{s(w(p))} \tilde{\sigma}^2_{w(p)}}. \tag{7}$$

As in the noise map calculation, we perform a dilation process on each of the $\gamma_{s(w(p))}$ and $\tilde{\sigma}^2_{w(p)}$ terms in Eq. 7 to smooth them out before combining them to compute our importance map. Furthermore, for acceleration we do not compute our importance map at every pixel, but rather divide the image into non-overlapping blocks and compute it once per block using the same importance value for the entire block.

Our adaptive sampling algorithm then uses the final importance map $\mathcal{M}$ to decide where to allocate its budget of samples. To do this, we use an iterative approach that uses part of the budget in each iteration and recomputes the importance map each time for the next iteration. In the first step, we sample the whole scene with 2 samples/pixel to initialize our importance map. We kept this number small to do not waste our sample budget by throwing them in regions that do not need it. After this initialization step, we spend the remaining sample budget in 3 more iterations. In each of these, we dedicate $1/7$, $2/7$ and $4/7$ of the number of samples remaining after the initialization step, respectively. We also adjust the block size for calculating the importance map to 8, 4 and 2 for each iteration, respectively. An illustration of our adaptive sampling process is shown in Fig. 5.

Our full algorithm, which uses both adaptive sampling during rendering and multilevel denoising as a postprocess, is called adaptive multilevel denoising (AMLD).

5. Results

We implemented the proposed framework in MATLAB and leveraged its parallel computing toolbox to accelerate the running time. The adaptive sampling method was implemented in C++ and integrated into PBRT2 [PH10] which was then run directly from the MATLAB program. To test a couple of image denoising algorithms with our framework, we ran our experiments on the BLS-GSM [PSWS03] and BM3D [DFKE06] denoising algorithms mentioned earlier. We implemented the BLS-GSM method in C++ and used the code provided by Dabov et al. for BM3D. For both algorithms, we used the standard default parameters provided by the respective authors.

To handle color images, we compute the terms in Eqs. 4 (for the noise map) and 7 (for the importance map) for each channel separately and then average them before computing the final noise/importance map value. We also tried applying weights based on perceptual importance of each color channel (more weight to the green and less weight to the blue channel). However, the difference in the results was not noticeable. Furthermore, we tonemap the sample values to the range 0 to 1 before computing these terms because most denoising algorithms work on standard images. As for the parameters of our algorithm, we use a window size $w = 8$ and a smoothness parameter (described at the end of Sec. 4.1) $g = 1.5$ for BLS-GSM and $g = 3$ for BM3D. This parameter g is kept constant for all scenes and there are no other parameters in our code.

All the results shown here were computed on an Intel dual quad-core Xeon X5570 3.06 GHz machine with 16 GB of memory. We compare our results against four state-of-the-art general Monte Carlo algorithms: Multidimensional Adaptive Sampling (MDAS) [HJW*08], Adaptive Wavelet Rendering (AWR) [ODR09], the method of Rousselle et al. [RKZ11], and Random Parameter Filtering (RPF) [SD12]. We also show comparisons to algorithms designed to handle only one specific MC effect, such as the sheared-filter motion blur (SFMB) method of Egan et al. [ETH*09]. For all these algorithms, we used the implementations provided by the respective authors. Note that AWR uses a different rendering system that is accelerated with a partition traversal algorithm [ORM08], so their rendering times are faster than PBRT and their shading models are a little different.

In Fig. 6, we begin by comparing our full adaptive framework (AMLD) using two denoising algorithms (BM3D and BLS-GSM) against AWR, which, like our method, uses only sample colors and not any higher dimensional feature information. As can be seen, both denoising methods in our framework produce smooth, high quality results. However, we found BM3D to be typically faster and better, and therefore we will use it to produce most of the results in the paper. Furthermore, in comparison with AWR, our approach does not have the objectionable wavelet artifacts, which are even worse when the scenes are animated as can be seen in the supplemental video.

In Fig. 7, we compare our approach with the method of Rousselle et al. [RKZ11], which is also an iterative filtering technique but it uses a simple Gaussian kernel to remove the noise. The KILLEROOS scene has soft shadows on the sharp lines on the floor which cannot be filtered while simultaneously preserving the sharp edges by the Rousselle et

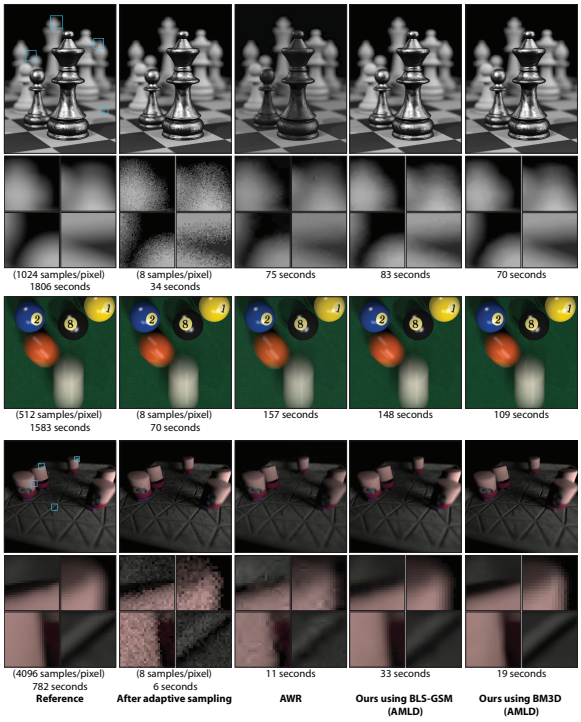


Figure 6: *Comparison between AWR and our approach for the (top)* CHESS *scene with depth of field, (middle)* POOLBALL *scene with motion blur, and (bottom)* TOASTERS *scene with area light sources and depth of field, all rendered at 8 samples/pixel.*

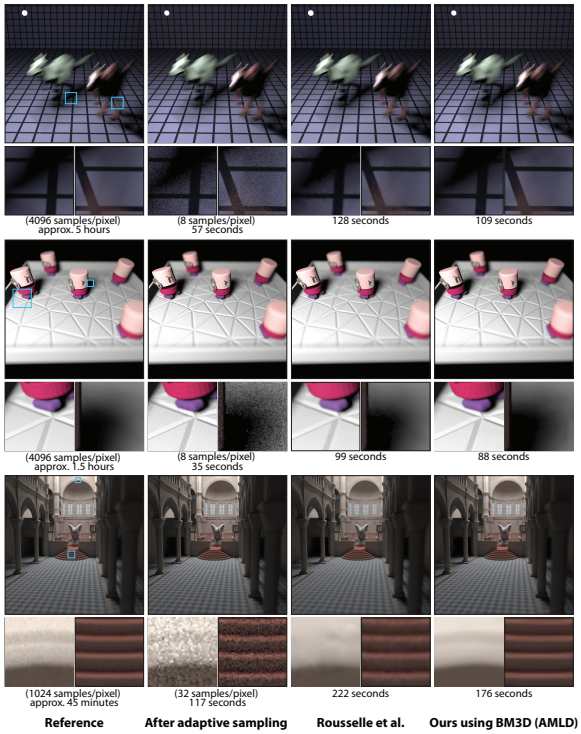


Figure 7: *Comparison between the Rousselle et al. method and our method for the (top)* KILLEROOS*, (middle)* TOASTERS*, and (bottom)* SIBENIK *scenes. Our method is better able to keep the details and remove the noise.*

al.'s method, as pointed out by the authors themselves. Our method, on the other hand, can produce smooth shadows and sharp lines. For the TOASTERS scene, the inset on the left shows detail on the floor that appears sharp in the reference image. Our method preserves this detail while smoothing the noise from the soft shadows, but the Rousselle et al. method blurs the details slightly and produces shadows that are not fully smooth. Finally, the SIBENIK scene is a challenging path-traced scene because it still contains significant amount of noise at 32 samples/pixels. As can be seen, our algorithm is better at removing the unwanted noise while preserving the important scene detail. The difference is expected because it is difficult for the Gaussian filter in Rousselle et al.'s method to compete with a sophisticated denoising algorithm such as BM3D.

Fig. 8 compares our method with MDAS and RPF on the POOLBALL and TOASTERS scenes. These methods use additional feature information and therefore require more memory. Moreover, they have some parameters that we optimized for each scene to get the best results (in our method all the parameters are fixed). Our results are both faster and of higher quality than those produced by MDAS or RPF.

We also compare our adaptive algorithm against the simpler, postprocess multi-level denoising (MLD) version applied on standard, low-discrepancy Monte Carlo samples. As can be seen in Fig. 9, the adaptive AMLD method produces slightly better results (MSE for AMLD is 1.42×10^{-4}, and for MLD is 1.54×10^{-4}). To get the same MSE as our adaptive algorithm, MLD would only need 4 additional more samples per pixel, which suggests that there is still benefit from using our MLD denoising framework separately as a postprocess if desired.

Indeed, the next two comparisons (shown in Figs. 10 and 11) use standard MC samples as the input to our MLD algorithm. Fig. 10 compares MLD against different methods on the CAR scene with motion blur, rendered at 4 samples/pixel. In this case, the sheared-filter motion blur algorithm (SFMB) [ETH*09] is specifically designed for motion blur but our result is comparable to SFMB in many parts of the image and in some cases better, such as in the shadows below the car. RPF can handle the shadow below the car better than SFMB, but overblurs the back wall. Our algorithm is able to preserve the detail in these regions.

For a numerical comparison, we compare the mean-squared error (MSE) of our approach with the RPF on the ROBOTS scene in Fig. 11, which shows our method has lower MSE than the RPF algorithm for varying sampling rates. Moreover, the bottom part of the figure shows an equal time comparison with RPF (we are not able to match the timing exactly because the low discrepancy sampling pattern requires power-of-two samples). As seen, our method with

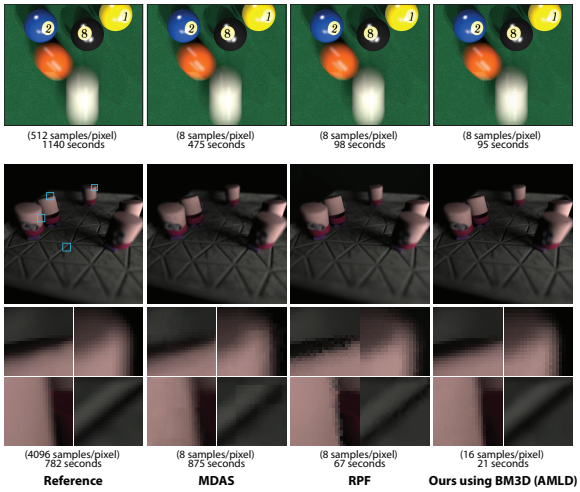


Figure 8: *Comparison between different algorithms for the* POOLBALL *and* TOASTERS *scenes. Our algorithm is both faster and has better quality than MDAS and RPF. Note that the resolution for the* POOLBALL *scene in this figure is different from Fig. 6 and thus the timings are different.*

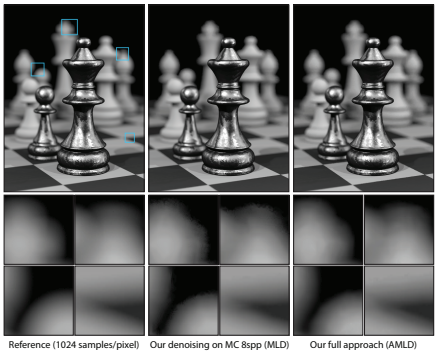


Figure 9: *Results of our approach with and without adaptive sampling. The MLD method is applied to samples computed with standard low-discrepancy MC techniques.*

64 samples/pixel takes approximately as long as RPF with 8 samples/pixel and generates an image that is better visually and is an order of magnitude better in MSE.

Finally, in terms of timing, our algorithm is reasonably fast, taking on the order of seconds to denoise an image. We have included timing numbers for comparison in most of the figures. A thorough timing breakdown for the CHESS scene is shown in Table 1.

6. Discussion, Limitations, and Future Work

The flexibility and simplicity of our framework allows us to use state-of-the-art video denoising methods to remove the noise from Monte Carlo renderings of animated sequences. In the supplemental video, we use our framework with the BM3D video denoising method [DFE07] and compare the results with both AWR [ODR09] and RPF [SD11]. Our method is much faster (less than 10 seconds for an 800×600 frame, excluding the rendering time) and has higher quality.

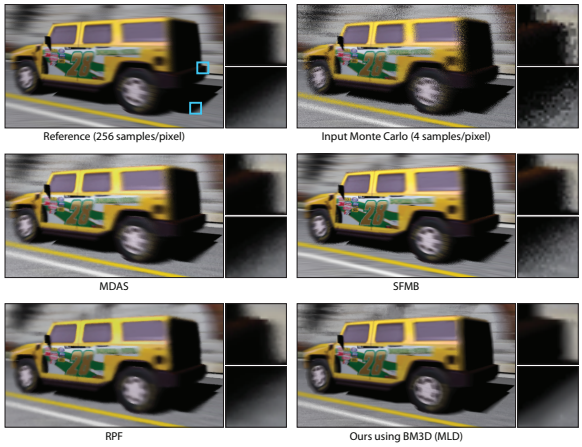


Figure 10: *For this motion-blurred* CAR *scene (rendered at 4 samples/pixel), we compare our non-adaptive MLD method against MDAS, sheared-filter motion blur (SFMB), and Random Parameter Filtering (RPF).*

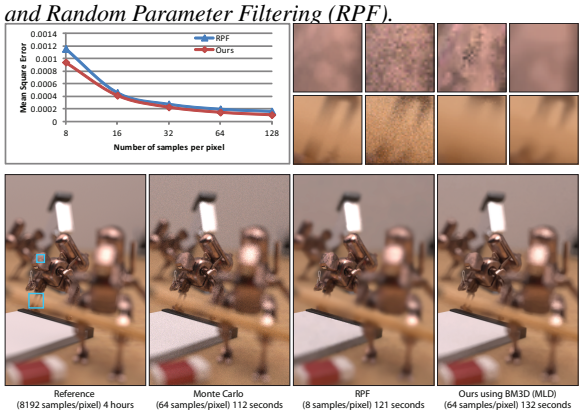


Figure 11: *(top) MSE comparison of our MLD method with RPF for the* ROBOTS *scene with depth of field and path tracing effects. Note that our algorithm is significantly faster than RPF for the same number of samples. (bottom) In an approximate, equal time comparison, RPF at 8 samples/pixel takes about 121 seconds and has an MSE of* 1.15×10^{-3}*, while ours takes 132 seconds to render and reconstruct 64 samples/pixel with an MSE of* 1.46×10^{-4}.

As we discussed in Section 4.1, metrics such as variance and contrast can often incorrectly identify scene details as noise. To show the difference in practice, we compare the sample distribution using the MDAS contrast metric and our proposed importance metric in Fig. 12. As seen, the MDAS metric labels the fixed detail in the green surface of the pool table as noise, wasting many valuable samples in these regions. Our algorithm, on the other hand, correctly focuses the samples on the motion blur alone.

We also experimented with varying the number of noise levels L computed during our multilevel denoising process (see Sec. 4.2). Fig. 13 shows the result of varying this parameter, which provides a tradeoff between quality and speed. In practice, we found that $L = \lceil \max(\tilde{\sigma}_{w(p)})/10 \rceil$ levels were sufficient to produce the results shown in the paper.

Step in the algorithm	Time (secs.)	Percentage
Calculating importance map	0.8	1.1%
Rendering samples	33.8	48.2%
Adaptive sampling:	34.6	49.3%
Calculating noise map	2.4	3.4%
Denoising levels with BM3D	32.8	46.7%
Combining images	0.4	0.6%
Reconstruction:	35.6	50.7%
Total time for AMLD	70.2	100%

Table 1: *Timing breakdown for the complete AMLD algorithm on the* CHESS *scene.*

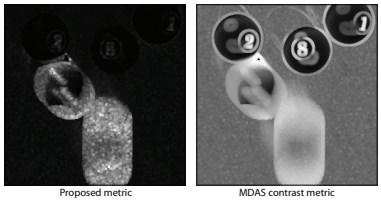


Figure 12: *Comparison of the sample distribution from MDAS and our importance metric (Eq. 7). MDAS's contrast metric tends to sample the green regions of the pool table because the texture detail there results in high contrast.*

Because we are operating on image-space samples using only their color information, we cannot always distinguish noise from scene detail in the general case. For example, there might be a noisy texture on an object that might be considered noise by our estimation metric, so we would blur it. Also, if there is a small noisy region between two smooth regions (like the edges of the checkerboard in the CHESS scene in Fig. 6 or the sharp back edge of the floor in the KILLEROOS scene in Fig. 7), we might consider it scene detail and preserve it. One possible solution is to perform a clustering on the samples [SD12] to exclude the smooth regions when we are calculating the noise metric using Eq. 3. However, this is the subject of future work.

As shown in Figs. 9, 10, and 11, our denoising algorithm can work without the adaptive sampling stage and still produces good results. The adaptive sampler improves the quality of final results with at cost of small additional time for computing the importance map, which is negligible in comparison to the entire rendering time as shown in Table 1. Although other adaptive sampling techniques might be used with our denoising method, the proposed adaptive sampling method uses a similar metric, making it more compatible with our denoising method.

Although for most of the scenes we showed the results with 8 samples/pixel, for some scenes (e.g., SIBENIK) we needed more samples to get a good quality results. Since our algorithm estimates the noise variance at each iteration of its adaptive sampling stage, this value can be used as a stopping criterion for sampling. Therefore it can continue the sampling until the maximum standard deviation of the noise falls below some certain user-defined threshold.

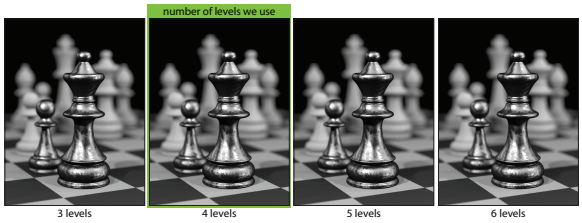


Figure 13: *Here we vary the number of levels L for denoising , as described in Sec. 4.2. There is a significant difference in quality between 3 levels and 4, but afterwards there is not much improvement.*

Finally, we note that this work opens several avenues for future work. First, we only tested a couple of denoising algorithms, since they outperformed state-of-the-art techniques for Monte Carlo rendering which was sufficient for the purposes of this paper. A thorough, comparative study of state-of-the-art denoising algorithms will have to be conducted since there are probably other algorithms that work faster and better. Furthermore, our framework is simple enough where it might be paired up with GPU-friendly denoising algorithms in order to remove the noise from stochastic rasterization algorithms for real time applications.

7. Conclusion

Although powerful denoising algorithms have been developed over the years by the image processing community, they have remained inapplicable for rendering because of their inherent assumption of spatially-invariant noise. In this paper, we have presented a way to use general spatially-invariant denoising algorithms by first estimating the per-pixel noise level followed by a multilevel algorithm that applies the denoising method in a spatially-varying manner. We showed results for various scenes that outperform state-of-the-art methods in MC rendering. In the future, as denoising algorithms improve, our framework will allow us to leverage newly-developed techniques to further improve the quality of the results shown in the paper.

Acknowledgements

This work was supported by the National Science Foundation under the NSF CAREER award #0845396. We gratefully acknowledge the sources of the scenes in the paper: CHESS – Wojciech Jarosz, POOLBALL – Toshiya Hachisuka, TOASTERS – Andrew Kensler, KILLEROO – headus/Rezard (PBRT2 book), SIBENIK – Marko Dabrovic and Mihovil Odak, CAR – Turbosquid user graphicdoom (car model), Wikipedia user Jongleur100 (wall texture), Kevin Egan (scene), ROBOTS – Jesper Lloyd.

References

[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. 1, 2

[BM98] BOLIN M. R., MEYER G. W.: A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98* (New York, NY, USA, 1998), ACM, pp. 299–309. 2

[BWG03] BALA K., WALTER B., GREENBERG D. P.: Combining edges and points for interactive high-quality rendering. *ACM Trans. Graph. 22* (July 2003), 631–640. 2

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: efficiently evaluating products of complex functions. *ACM Trans. Graph. 24* (July 2005), 1166–1175. 2

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graph. 5* (January 1986), 51–72. 2

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *SIGGRAPH '84* (New York, NY, USA, 1984), ACM, pp. 137–145. 1, 2

[CWW*11] CHEN J., WANG B., WANG Y., OVERBECK R. S., YONG J.-H., WANG W.: Efficient depth-of-field rendering with adaptive sampling and multiscale reconstruction. *Computer Graphics Forum 30*, 6 (2011), 1667–1680. 2

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph. 21*, 3 (July 2002), 257–266. 5

[DFE07] DABOV K., FOI A., EGIAZARIAN K.: Video denoising by sparse 3d transform-domain collaborative filtering. In *Proc. 15th European Signal Processing Conference* (Sept. 2007), pp. 145–149. 8

[DFKE06] DABOV K., FOI A., KATKOVNIK V., EGIAZARIAN K.: Image denoising with block-matching and 3d filtering. In *Electronic Imaging'06* (2006). 3, 5, 6

[DJ94] DONOHO D. L., JOHNSTONE J. M.: Ideal spatial adaptation by wavelet shrinkage. *Biometrika 81*, 3 (1994), 425–455. 3, 4

[Don95] DONOHO D.: De-noising by soft-thresholding. *IEEE Transactions on Information Theory 41*, 3 (may 1995), 613–627. 1, 3

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P.: Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In *Proceedings of High Performance Graphics 2010* (2010), pp. 67–75. 1, 2

[EDR11] EGAN K., DURAND F., RAMAMOORTHI R.: Practical filtering for efficient ray-traced directional occlusion. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 180:1–180:10. 2

[EHDR11] EGAN K., HECHT F., DURAND F., RAMAMOORTHI R.: Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Trans. Graph. 30*, 2 (Apr. 2011), 9:1–9:13. 2

[ETH*09] EGAN K., TSENG Y.-T., HOLZSCHUCH N., DURAND F., RAMAMOORTHI R.: Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph. 28*, 3 (2009), 1–13. 2, 6, 7

[Guo98] GUO B.: Progressive radiance evaluation using directional coherence maps. In *SIGGRAPH '98* (New York, NY, USA, 1998), ACM, pp. 255–266. 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 Trans. Graph. 27* (August 2008), 33:1–33:10. 2, 4, 6

[JC95] JENSEN H. W., CHRISTENSEN N. J.: Optimizing path tracing using noise reduction filters. In *Winter School of Computer Graphics (WSCG) 1995* (1995), pp. 134–142. 2

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86* (New York, NY, USA, 1986), ACM, pp. 143–150. 2

[LAC*11] LEHTINEN J., AILA T., CHEN J., LAINE S., DURAND F.: Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph. 30*, 4 (Aug. 2011), 55:1–55:12. 2

[LALD12] LEHTINEN J., AILA T., LAINE S., DURAND F.: Reconstructing the indirect light field for global illumination. *ACM Trans. Graph. 31*, 4 (July 2012), 51:1–51:10. 2

[LR90] LEE M., REDNER R.: A note on the use of nonlinear filtering in computer graphics. *IEEE Computer Graphics and Applications 10*, 3 (May 1990), 23–29. 2

[MA06] MEYER M., ANDERSON J.: Statistical acceleration for animated global illumination. *ACM Trans. Graph. 25*, 3 (2006), 1075–1080. 2

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

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *SIGGRAPH '87* (1987), pp. 65–72. 2, 4

[ODR09] OVERBECK R. S., DONNER C., RAMAMOORTHI R.: Adaptive Wavelet Rendering. *ACM Trans. Graph. 28*, 5 (2009), 1–12. 1, 2, 3, 6, 8

[ORM08] OVERBECK R., RAMAMOORTHI R., MARK W. R.: Large ray packets for real-time Whitted ray tracing. In *IEEE/EG Symposium on Interactive Ray Tracing (IRT)* (Aug 2008), pp. 41–48. 6

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, second ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010. 6

[PSWS03] PORTILLA J., STRELA V., WAINWRIGHT M., SIMONCELLI E.: Image denoising using scale mixtures of Gaussians in the wavelet domain. *Image Processing, IEEE Transactions on 12*, 11 (Nov. 2003), 1338 – 1351. 3, 6

[RFS03] RIGAU J., FEIXAS M., SBERT M.: Refinement criteria based on f-divergences. In *Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), EGRW '03, Eurographics Association, pp. 260–269. 2

[RKZ11] ROUSSELLE F., KNAUS C., ZWICKER M.: Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 159:1–159:12. 1, 2, 5, 6

[RW94] RUSHMEIER H. E., WARD G. J.: Energy preserving non-linear filters. In *ACM SIGGRAPH '94* (New York, NY, USA, 1994), pp. 131–138. 2

[SD11] SEN P., DARABI S.: Compressive rendering: A rendering application of compressed sensing. *IEEE Transactions on Visualization and Computer Graphics 17* (2011), 487–499. 2, 8

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. on Graph. 31*, 3 (June 2012), 18:1–18:15. 1, 2, 4, 6, 9

[SSD*09] SOLER C., SUBR K., DURAND F., HOLZSCHUCH N., SILLION F.: Fourier depth of field. *ACM Trans. Graph. 28*, 2 (2009), 1–12. 2

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

[XP05] XU R., PATTANAIK S. N.: A novel Monte Carlo noise reduction operator. *IEEE Computer Graphics and Applications 25* (2005), 31–35. 2