Adaptive Rendering with Non-Local Means Filtering

Fabrice Rousselle* University of Bern
Claude Knaus University of Bern
Matthias Zwicker University of Bern

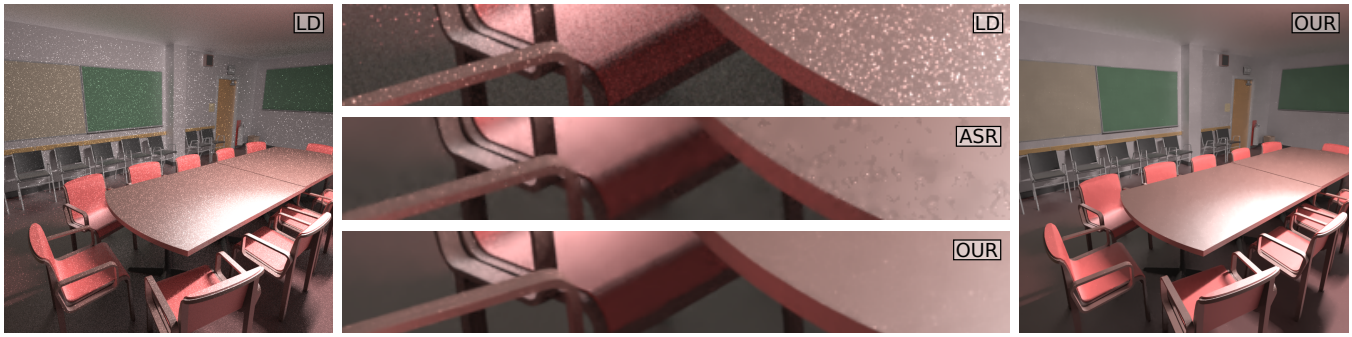

Figure 1: *Our adaptive rendering method handles arbitrary light transport effects, employs a state of the art denoising filter, and incurs little computational overhead. The "conference" scene includes indirect illumination and moderately glossy surfaces. We compare path tracing using low-discrepancy sampling (far left) to our approach (far right). The close-ups in the middle show low-discrepancy sampling (LD), adaptive sampling and reconstruction (ASR) by Rousselle et al. [2011], and our approach (OUR), all at equal rendering time.*

Abstract

We propose a novel approach for image space adaptive sampling and filtering in Monte Carlo rendering. We use an iterative scheme composed of three steps. First, we adaptively distribute samples in the image plane. Second, we denoise the image using a non-linear filter. Third, we estimate the residual per-pixel error of the filtered rendering, and the error estimate guides the sample distribution in the next iteration. The effectiveness of our approach hinges on the use of a state of the art image denoising technique, which we extend to an adaptive rendering framework. A key idea is to split the Monte Carlo samples into two buffers. This improves denoising performance and facilitates variance and error estimation. Our method relies only on the Monte Carlo samples, allowing us to handle arbitrary light transport and lens effects. In addition, it is robust to high noise levels and complex image content. We compare our approach to a state of the art adaptive rendering technique based on adaptive bandwidth selection and demonstrate substantial improvements in terms of both numerical error and visual quality. Our framework is easy to implement on top of standard Monte Carlo renderers and it incurs little computational overhead.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing;

Keywords: adaptive sampling and reconstruction

Links: DL PDF

*e-mail: roussell@iam.unibe.ch

1 Introduction

In this paper we present an algorithm to reduce numerical error and visual artifacts in Monte Carlo rendering. Our approach relies on image space adaptive sampling and filtering, which is attractive because it makes few assumptions about the specific light transport or lens effects being rendered. Recent techniques that follow this strategy [Overbeck et al. 2009; Rousselle et al. 2011] have been shown to effectively improve the visual quality over standard sampling and filtering for a wide range of effects that are prone to noise artifacts, such as indirect illumination, soft shadows, participating media, depth of field, or motion blur. We observe, however, that previous work in adaptive rendering is based on image denoising techniques that are not competitive with the state of the art in image processing. For example, Overbeck et al.'s approach [Overbeck et al. 2009] uses straightforward wavelet shrinkage, and Rousselle et al.'s algorithm [Rousselle et al. 2011] is based on adaptive bandwidth selection with isotropic Gaussian filters. Both techniques leave ample room for improvement, as we show in this paper.

The main contribution of our approach is to extend an image denoising filter that is competitive with the state of the art in image processing and employ it in an adaptive rendering framework. We show that our approach is more robust under severe noise than previous techniques, significantly reducing numerical error and visual artifacts. In addition, our approach is more effective at removing noise even in complex image regions while minimizing the smoothing of image features. Finally, our technique is compatible with efficient low discrepancy sampling while previous techniques assumed random sampling, which affords additional improvements in output quality. Our technique shares the advantages of other image based approaches. It can deal with arbitrary light transport and lens effects, and it only requires the Monte Carlo samples as its input such that it is straightforward to implement it on top of existing renderers.

Our framework builds on an iterative strategy consisting of three components in each iteration step. First, we distribute a given budget of Monte Carlo samples over the image. We sample the image uniformly in the initial iteration step, while consecutive iterations employ adaptive sampling. Second, we filter the image to reduce

noise. Finally, we estimate the error remaining in the filtered image to drive adaptive sampling in the next iteration step. The effectiveness of this scheme largely hinges on the denoising filter. A core idea in our approach is to extend and adapt the non-local means filter [Buades et al. 2005], which is competitive with the state of the art in image denoising, to our adaptive rendering framework. A challenge in applying such a filter for adaptive rendering is the need to obtain an error estimate to drive adaptive sampling. We address this issue using a dual-buffer strategy: we simply split the samples into two sets that we render and filter in two separate image buffers. The difference between the filtered buffers serves as an effective error estimate. Hence we demonstrate that it is possible to employ sophisticated denoising filters for adaptive rendering, and our results show significant improvements over previous work. In summary we make the following contributions:

- An adaptive rendering framework based on non-local means filtering. We demonstrate significant improvements in numerical error and visual quality compared to previous work.
- A dual-buffer strategy for non-local means filtering. This allows us to obtain error estimates to drive adaptive sampling, and to extend our method to low-discrepancy sampling.
- Extensions of the non-local means filter to adapt it to denoise images produced by Monte Carlo rendering.

2 Previous Work

Adaptive Rendering. Image space adaptive sampling and filtering has been pioneered by Mitchell [1987], and many subsequent techniques have been inspired by his approach. Bolin and Meyer [1998] use a perceptual metric to drive the sample distribution, and leverage a Haar wavelet representation of the data to estimate values that have yet to be sampled. Recent techniques include the work of Bala et al. [2003], which uses edge detection to constrain the interpolation of sparsely sampled data points. This approach is prone to artifacts when edge detection fails. Chen et al. [2011] propose a method specifically designed to handle depth of field effects, which uses depth maps to select the appropriate filter from a filterbank on a per-pixel basis. Adaptive wavelet rendering by Overbeck et al. [2009] introduces an iterative framework alternating between filtering and sampling steps, similar to our method. They perform a wavelet decomposition of the rendered image and denoise it using wavelet shrinkage [Donoho and Johnstone 1994]. Rousselle et al. [2011] build on the same iterative framework. Their work relies on adaptive bandwidth selection for denoising, and they report consistently higher quality results compared to adaptive wavelet rendering. Their approach relies on isotropic filters, however, which cannot properly resolve noise along edges. In addition, errors in adaptive bandwidth selection can produce jarring artifacts, particularly in the presence of spike noise. Our method uses a filtering approach that is much more robust to noise and can accommodate arbitrary anisotropy in the signal, which leads to significantly lower error and improved quality at equal render time. Sen and Darabi [Sen and Darabi 2012] propose an adaptive filtering method based on a joint bilateral filter which can leverage known scene features. A major conceptual difference to our approach is that we develop an error estimation technique that enables adaptive sampling, while their work focuses only on filtering. A practical difference is that the computational complexity of our filter is proportional to the number of pixels, while theirs is proportional to the number of samples. As a consequence, their filter typically accounts for more than 50% of rendering time and takes on the order of minutes, while the overhead of our combined adaptive sampling and filtering approach is usually only a few seconds.

Multi-dimensional methods exploit the structure of the sampled signal in high dimensional path space, which is not accessible in methods restricted to image space information. Hachisuka et al. [2008] propose a general solution applicable to arbitrary effects. They perform adaptive sampling and anisotropic filtering directly in multi-dimensional path space, but their approach scales poorly to higher dimensional cases. Other methods have proposed specialized solutions for specific cases. Soler et al. [2009] consider depth of field. Egan et al. propose optimal sheared filters based on the frequency analysis of motion blur [2009] and soft shadows [2011]. Lehtinen et al. [2011] similarly exploit shearing in the light field, but handle combinations of motion blur, depth of field, and soft shadows. The main advantage of our work compared to these techniques is that we are not restricted to any particular light transport or lens effect.

Many real-time and interactive methods employ sophisticated denoising filters, but dispense with adaptive sampling for performance reasons. These methods often exploit geometry buffers storing normals and depth to guide filtering. For instance, Ritschel et al. [2009] perform edge-aware cross bilateral filtering on sparsely sampled scenes, Shirley et al. [2011] exploit the depth buffer to define a per-pixel filter radius, Dammertz et al. [2010] propose an edge-avoiding filter using an à-trous wavelet transform, and Bauszat et al. [2011] use guided image filtering for highly efficient filtering. While these methods can produce impressive results, they often fail when high frequency image features are not represented in the geometry buffers.

Image Denoising. Our approach is inspired by powerful denoising methods developed in the image processing community. We provide references to some of the most successful ideas, without any claim of completeness. A fundamental approach in image denoising is to transform the input into a domain where the signal is expected to be sparse, that is, most coefficients are zero. Denoising is then simply achieved by shrinking the coefficients in the transform domain, for example by thresholding small coefficients to zero, followed by an inverse transformation. There is a large number of algorithms based on wavelet shrinkage, initially proposed and thoroughly analyzed by Donoho and Johnstone [1994]. This is also the denoising method employed by Overbeck et al. [2009]. Significant improvements over the basic approach can be achieved by taking into account statistical models of the distribution of the wavelet coefficients [Portilla et al. 2003], learning dictionaries of small image patches to construct sparse transform domains [Mairal et al. 2008], or using collaborative filtering of groups of image regions [Dabov et al. 2007], to name a few. Our denoising approach is based on non-local means (NL-Means) filtering [Buades et al. 2005], which relies on a fundamentally different strategy than transform domain methods. The main idea is simply to estimate each pixel as a weighted average of other input pixels, where the weight for each pair of pixels is determined by the similarity of small image patches centered at the two pixels. This is essentially a generalization of the bilateral filter [Tomasi and Manduchi 1998], where the weights are determined by the similarities of the pixels themselves, rather than small patches centered around them. This small extension, however, leads to greatly improved denoising performance, and NL-Means filtering is competitive with the state of the art in image denoising. Ji et al. [2009] modify the NL-Means distance computation to operate with a rotationally invariant metric, but since they cannot estimate the parameters of their filter from the noisy input data, their method can only be applied to inputs with uniform noise of a known distribution. Kervrann and Boulanger [2006] adapt the NL-Means filter window size on a per-pixel basis. While this approach may lead to slight improvements, we chose to use the basic NL-Means formulation for simplicity and computational efficiency. While Xu and Pattanaik [2005] use a modified bilateral filter to denoise renderings produced with a Monte Carlo raytracer,

they do not include adaptive sampling. In addition, our approach based on NL-Means filtering provides significantly improved denoising performance.

3 Algorithm Overview

The objective of our algorithm is to optimize rendering quality given a user specified sampling budget. Figure 2 gives a visual overview of our method. We build on image space adaptive sampling and filtering using an iterative scheme, where each iteration is composed of three steps: (1) sampling, (2) filtering, and (3) estimating the residual error. In each subsequent iteration we use the error estimate from the previous one to drive adaptive sampling, where we distribute a predetermined fraction of the sample budget each time. We bootstrap the algorithm in the first iteration by sampling the image uniformly. A key component of our approach is the denoising filter in step two. We use a variant of the NL-Means filter tailored to our adaptive rendering framework. During the iteration we keep track of per pixel statistics including sample count and empirical variance, in addition to the usual pixel value. We use these statistics to adapt the NL-Means filter to local image characteristics. In addition, we operate on two image buffers (indicated in red and green in Figure 2), instead of one, each receiving half of the samples. This is a key idea of our method that allows us to improve filtering quality and obtain simple but effective error estimates to drive adaptive sampling.

We next describe the main components of our algorithm. We review the original NL-Means formulation and introduce our extensions in Section 3.1. Section 3.2 covers our error estimation technique, and Section 3.3 describes our adaptive sampling scheme.

3.1 The NL-Means Filter

The NL-Means filter [Buades et al. 2005] is a non-linear, edge-preserving filter that computes each output pixel as a weighted sum of input pixels. The set of input pixels contributing to one output pixel may stem from a large region in the input image, hence the term *non-local*. A key feature of the NL-Means filter is that the weights are determined by the distance between small image patches, as illustrated in Figure 3. The NL-Means filter is a generalization of the bilateral filter [Tomasi and Manduchi 1998], which considers distances between pairs of pixel values, instead of small patches, to compute filter weights. This extension leads to much improved denoising performance, and NL-means and its variants are among the most successful denoising algorithms.

We propose several key extensions to the original filter formulation that allow us to use it effectively in an adaptive rendering framework:

- dual-buffered filtering (Section 3.1.1),
- support for non-uniform variance (Section 3.1.2),
- symmetric distance computation to better handle gradients (Section 3.1.3).

We start by giving a definition of the original NL-Means filter, and then detail our extensions and their impact on the filtering process.

The NL-Means filter computes the filtered value $\hat{u}(p)$ of a pixel p in a color image $u = (u_1, u_2, u_3)$ as a weighted average of pixels in the square neighborhood of size $2r+1 \times 2r+1$ centered on p, as illustrated in Figure 3,

$$\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \in N(p)} u_i(q) w(p,q), \tag{1}$$

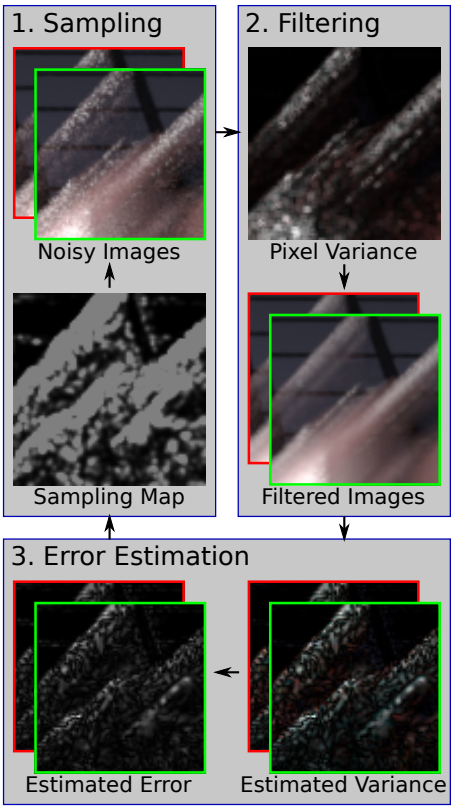

Figure 2: *Overview of our dual-buffer framework. We indicate the two buffers with red and green borders. We iterate over three steps: sampling, filtering, and error estimation. In the filtering step, we first estimate the pixel variances using the noisy buffers. Then we denoise the buffers with our variant of NL-Means filtering. In the error estimation step, we first estimate the residual variance in the filtered buffers, and then derive the potential error reduction afforded by placing an additional sample per pixel. In the sampling step, we distribute a set of samples according to the estimated errors in each buffer. We iterate until the sample budget is exhausted.*

where $N(p)$ is the square neighborhood centered on p, $w(p,q)$ is the weight of the contribution of q to p, i is the index of the color channel, and $C(p)$ is a normalization factor,

$$C(p) = \sum_{q \in N(p)} w(p,q).$$

The weight $w(p,q)$ of a neighbor q is based on the distance between a pair of small patches of size $2f+1 \times 2f+1$ centered at p and q. The patch distance $d^2(P(p), P(q))$ is the average of the per-pixel and per color channel squared distances $d_i^2(p,q)$ over the patches,

$$d_i^2(p,q) = \left(u_i(p) - u_i(q)\right)^2, \tag{2}$$

$$d^2(P(p), P(q)) = \frac{1}{3(2f+1)^2} \sum_{i=1}^{3} \sum_{n \in P(0)} d_i^2(p+n, q+n), \tag{3}$$

where $P(p)$ and $P(q)$ are the patches centered on p and q, $P(0)$ represents the offsets to each pixel within a patch.

An important observation is that, because the input signal is noisy, the measured squared distances are biased. Therefore, the original NL-Means filter [Buades et al. 2005] subtracts the variance of

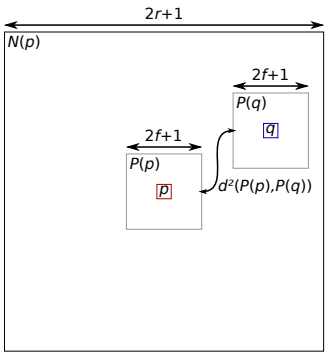

Figure 3: *NL-Means computes the filtered value $\hat{u}(p)$ of pixel p as a weighted average of all pixels q in a square neighborhood of size $2r+1 \times 2r+1$. The weight between p and q is based on the squared distance $d^2(P(p), P(q))$ between the pair of patches $P(p)$ and $P(q)$ of size $2f+1 \times 2f+1$ centered on p and q.*

the measured squared distances to cancel out the noise contribution from the patch distance. Assuming uniform pixel noise with variance σ^2 and uncorrelated pixels p and q, the modified patch distance is

$$max(0, d^2(P(p), P(q)) - 2\sigma^2).$$

The weight $w(p,q)$ of the contribution of pixel q to p is then obtained using an exponential kernel,

$$w(p,q) = \exp^{-\frac{max(0, d^2(P(p),P(q)) - 2\sigma^2)}{k^2 2\sigma^2}},$$

where k is a user specified damping factor that controls the strength of the filter. A lower k value yields a more conservative filter.

We also use the patchwise extension proposed by Buades et al. [Buades et al. 2005], which produces slightly smoother outputs. Instead of weighting only the pixel p at the center of the patch with the weight $w(p,q)$, we weight all pixels in the patch centered at p with $w(p,q)$. Each pair of pixels occurs in $2f+1 \times 2f+1$ patches, each time with a distinct weight $w(p+n, q+n)$, where n is the offset of p and q in the patch. In the patchwise implementation the final weight $W(p,q)$ for a pair of pixels is simply the average over all weights that involve these two pixels,

$$W(p,q) = \frac{1}{(2f+1)^2} \sum_{n \in P(0)} w(p+n, q+n).$$

3.1.1 Dual-Buffer Filtering

Dual-buffered filtering is our key modification to NL-Means that enables most of our other extensions and allows us to employ it in an adaptive rendering framework. We observe that in the original NL-Means approach the filter weights and the input signal are correlated. This makes it challenging to analyze the error introduced by the filter. In addition, since the filter weights not only adapt to the signal but also to the noise, some noise tends to be preserved in the output. We address both issues with our dual-buffered implementation. We maintain two image buffers, A and B, that receive half the samples in each iteration of our framework (Figure 2). We also store pixel statistics including the number of samples and the empirical sample variance separately in each buffer. We then use the two buffers to cross filter each other, that is, we use the filter weights computed from buffer A to filter buffer B, and vice versa. The final output is the average of the two filtered buffers. Our approach eliminates the correlation between the filter weights and the noise, and we get much improved filtering as shown in Figure 4. In addition, the per-pixel differences between the buffers serve as a basis for our variance and error estimation components that we describe next.

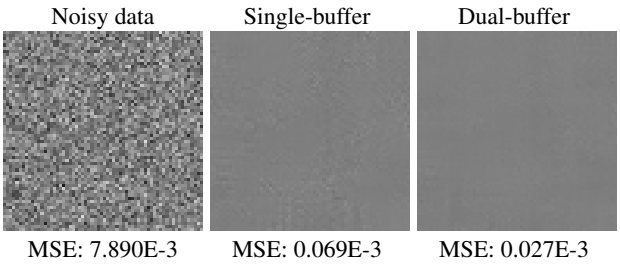

Figure 4: *Filtering a noisy uniform input (left), using the single-buffer approach (center), or our dual-buffer approach (right). Using a single buffer tends to preserve structures from the noisy input because of the correlation between filter weights and input. The filter parameters are $r = 10$, $f = 3$ pixels, and $k = 0.45$. We list the mean squared error (*MSE*) under each image.*

3.1.2 Non-uniform Variance

While the original NL-Means formulation assumes uniform noise over input images, Monte Carlo rendering generally leads to highly non-uniform noise patterns. The type of noise and its magnitude depend on the scene geometry, object materials, and light transport and lens effects. Therefore, let us denote per pixel variance of color channel i at pixel p by $\mathrm{Var}_i[p]$, replacing the uniform variance σ from Section 3.1. We modify the previous per-pixel squared distance computation by performing variance cancellation and normalization on a per-pixel basis,

$$d_i^2(p,q) = \frac{(u_i(p) - u_i(q))^2 - \alpha(\mathrm{Var}_i[p] + \mathrm{Var}_i[q,p])}{\epsilon + k^2(\mathrm{Var}_i[p] + \mathrm{Var}_i[q])}, \quad (4)$$

where α controls the strength of variance cancellation. In addition, we define $\mathrm{Var}_i[q,p] = min(\mathrm{Var}_i[q], \mathrm{Var}_i[p])$ to clamp the variance at position q to the variance at p. This ensures that the potentially large variance of brighter neighbors does not cancel out the measured squared difference, and it prevents bright regions from blurring into dark ones. Note that with $\mathrm{Var}_i[p] = \mathrm{Var}_i[q] = \sigma^2$ we fall back to the original definition. We set $\epsilon = 10^{-10}$ to prevent divisions by 0. The value of ϵ must be very small in order to preserve features in dark areas. Finally, the patch distance is computed as in Equation 3, and the weight $w(p,q)$ of the contribution of pixel q to p is now simply

$$w(p,q) = \exp^{-max(0, d^2(P(p),P(q)))}. \quad (5)$$

Additionally, we set to zero weights $W(p,q)$ below a threshold of 0.05, to help preserve fine features in noisy areas where non-feature neighbors can greatly outnumber feature neighbors, dominating the result even with very low weights.

Variance Estimation. Estimating the pixel variance $\mathrm{Var}[p]$ is a key component of our algorithm (we omit the index i of the color channel for better readability from now on). Assuming that samples are drawn from a random distribution, we could estimate the pixel variance using the empirical variance of the samples contributing to the pixel, which we denote by $\Sigma[p]$, as proposed by Rousselle et al. [2011]. Random sampling, however, may lead to significantly higher variance than more sophisticated sampling strategies such as low-discrepancy sequences [Kollig and Keller 2002]. Therefore,

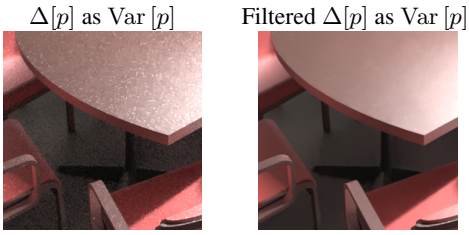

Figure 5: *Filtering with low-discrepancy samples. In the left image, we directly used the squared difference $\Delta[p]$ between our two buffers as an estimate of the pixel variance* Var $[p]$ *in the NL-Means filter. In the right image we cross-filtered $\Delta[p]$ using the sample variance $\Sigma[p]$ to obtain a smoother estimate for* Var $[p]$*. The unfiltered buffer variance $\Delta[p]$ is too noisy to reliably estimate pixel distances, whereas our filtered variance yields a smooth output.*

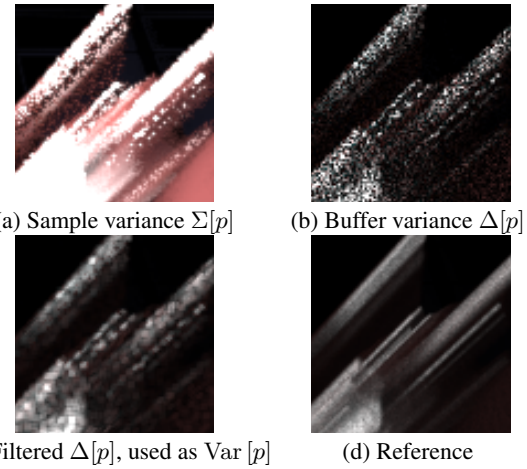

Figure 6: *Estimating* Var $[p]$*. We use the sample variance $\Sigma[p]$ (a) to cross-filter the inter-buffer variance $\Delta[p]$ (b), an unbiased but very noisy estimate of the true variance. The resulting filtered variance (c) is our estimate for* Var $[p]$*. It is a reasonable approximation of the reference variance (d) as computed over 100 images.*

we develop a simple approach to support low-discrepancy sampling in our framework, taking advantage of our two buffers. The same idea was developed by Sijbers et al. [1998] to estimate the variance in magnetic resonance images, but assuming uniform variance across the image.

We obtain an initial estimate $\Delta[p]$ of the pixel variance Var $[p]$ using the squared difference between the two buffers, $\Delta[p] = (A(p) - B(p))^2/2$. This is an unbiased estimate, but it is rather noisy because it is based on just two samples, that is, the two buffers. We found that we can improve our results, as illustrated in Figure 5, by applying an additional smoothing step to the initial variance estimates $\Delta[p]$. The intuition behind our smoothing approach is that the empirical sample variances $\Sigma[p]$ have a structure similar to the actual pixel variances, but $\Sigma[p]$ may be strongly biased. Therefore, we smooth the initial estimates $\Delta[p]$ by cross filtering them with the empirical sample variances $\Sigma[p]$, that is, we compute NL-Means filter weights using $\Sigma[p]$ and apply them to filter $\Delta[p]$. We illustrate the process in Figure 6.

To obtain the filter weights we also need the variance of $\Sigma[p]$, which we compute as the difference between $\Sigma[p]$ from buffers A and B. We compute pixel distances as in Equation 4, but using Σ and its variance estimate. Because the variance estimate for Σ is noisy, we set $\alpha = 4$ in Equation 4 to discard all differences lower than two standard deviations. We use a small neighborhood with $r = 1$, a patch size of $f = 3$, and $k = 0.45$. The small neighborhood ensures that the variance peaks, which are aligned with the outliers pixel values, are preserved. Since the filtering tends to increase the variance estimate of pixels neighboring noisy regions, we also clamp the filtered value of $\Delta[p]$ so that it does not exceed $\Sigma[p]$.

3.1.3 Symmetric Distance

Our symmetric distance computation is designed to improve the filtering of smooth gradients. Because the original NL-Means formulation tends to constrain the filter to neighbors orthogonal to the gradient direction, as we illustrate in Figure 7, it often results in distracting artifacts in otherwise smooth gradients. Our extension builds on the observation that, in a smooth gradient, all pixel differences are radially symmetric with respect to the center pixel. The error introduced by the contribution of a given neighbor is canceled by the contribution of the radially symmetric neighbor. We exploit radial symmetry in the signal by defining a modified distance to a pair of symmetric neighbors q_1 and q_2,

$$d_i^2(p,\bar{q}) = \frac{(u_i(p) - u_i(\bar{q}))^2 - (\mathrm{Var}_i[p] + \mathrm{Var}_i[p,\bar{q}])}{\mathrm{Var}_i[p] + \mathrm{Var}_i[\bar{q}]}, \quad (6)$$

where

$$u_i(\bar{q}) = (u_i(q_1) + u_i(q_2))/2,$$
$$\mathrm{Var}_i[\bar{q}] = (\mathrm{Var}_i[q_1] + \mathrm{Var}_i[q_2])/4,$$
$$\mathrm{Var}_i[p,\bar{q}] = (\mathrm{Var}_i[p,q_1] + \mathrm{Var}_i[p,q_2])/4.$$

Note that $\mathrm{Var}_i[\bar{q}]$ is half the mean variance of pixels q_1 and q_2, since the effective sampling rate doubles when we compute $u_i(\bar{q})$. Using these squared symmetric distances we compute the corresponding symmetric weight $w(p,\bar{q})$ as before.

In practice we always compute the asymmetric distances to both q_1 and q_2 as in Equation 4 and the corresponding weights $w(p,q_1)$ and $w(p,q_2)$, in addition to the symmetric weight $w(p,\bar{q})$. Only if the symmetric weight is larger the sum of both asymmetric weights, we set both $w(p,q_1)$ and $w(p,q_2)$ to $w(p,\bar{q})$. Otherwise we keep the asymmetric weights. This prevents the use of the symmetric weights if we do not have enough confidence in the symmetry of the data.

3.2 Error Estimation

The error in the filtered image consists of residual variance and bias introduced by the filtering. An analytical error analysis of the NL-Means filtered output, however, is rather complicated because the filter weights are both noisy and correlated with one another. Instead, we directly estimate the error as the squared relative difference between the two filtered buffers $\hat{A}$ and $\hat{B}$. We use the relative difference to prevent overweighting differences in bright image regions. We estimate the error E separately for each buffer (for simplicity we omit the buffer index from the notation). The error of the filtered buffer $\hat{A}$ is

$$E = \frac{(\hat{A} - \hat{B})^2}{\epsilon + \hat{A}^2},$$

where $\epsilon = 10^{-3}$ is an offset to prevent divisions by 0. Note that our estimate accounts for variance but not bias in the error. We do not attempt to estimate bias because the NL-Means filter tries to avoid bias by design. In Figure 8 we compare our estimated error to the ground truth error, that is, the relative squared distance to the reference image. Our estimate captures the main features of the

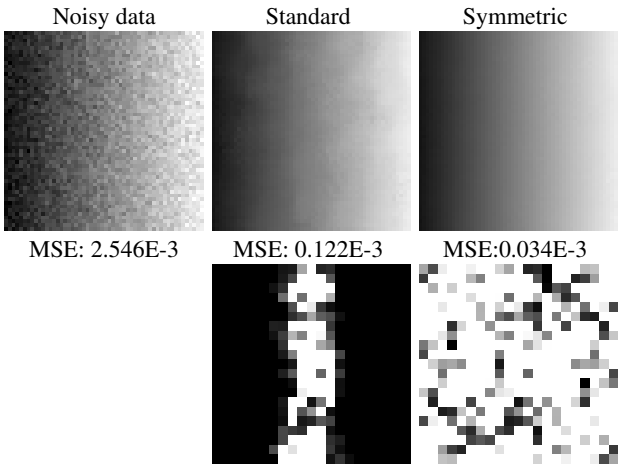

Figure 7: *A noisy ramp (top left), filtered using the standard NL-Means filter (middle column) and our extended filter including symmetric distances (right column). The corresponding filters for the center pixel are shown in the bottom row. The standard filter is constrained to neighbors orthogonal to the gradient direction, while our extended variant has a much larger effective support. The filter parameters are $r = 10$, $f = 3$, and $k = 0.45$.*

ground truth error, but may miss some small image details that are smoothed out in both buffers. These details can only be resolved with a larger number of samples per pixel.

Because we use the error estimation in the next sampling step to guide the distribution of samples, we additionally weight it according to the potential error reduction obtained by adding a single sample in each pixel. Given that the pixel variance is inversely proportional to the number of samples, adding one sample to a pixel p will decrease the variance by a factor of $1/(1 + n_p)$, where n_p is the number of samples already contributing to the filtered value of p. The new sample will also contribute to the pixels in the neighborhood of p, whose filter weights include p. Consequently, the error weight of an additional sample in p is

$$W(p) = \frac{\sum_{N(p)} w(p, q)}{1 + n_p},$$

and the weighted error of p is simply $W(p)E(p)$.

3.3 Sampling

We allocate an equal fraction of the total sample budget to each iteration. We split the number of samples in each iteration evenly over the two image buffers and distribute samples according to the same desired per-pixel sample density in each buffer. We represent the desired sample density using a sampling map that specifies for each pixel the number of additional samples to draw.

For the initial iteration we use a uniform sampling density. For the subsequent iterations we obtain the sampling map as follows. We sum the weighted error maps of both buffers, and smooth it using a small Gaussian kernel with $\sigma = 0.8$. This smoothing step allows pixels that missed a rare event (for instance a fast moving object), to still be sampled if their neighbors recorded the event, effectively filling holes in our sampling map. We then normalize the sampling map to sum up to $N/2$, where N is the sample budget per iteration, which gives us the number of samples to be drawn per pixel. Finally, we clamp the per-pixel sample count to a predefined value to

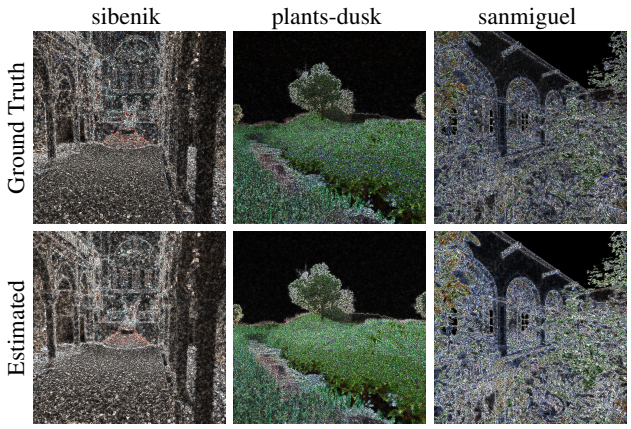

Figure 8: *Comparison of our error estimation using the relative squared difference between the two filtered buffers and the relative squared difference to the reference image, using uniform distributions of 32 samples per pixel.*

prevent pixels with very large errors from draining too many samples. For each pixel, we use the fractional part of the sample count as the probability of drawing an additional sample. For instance, if we need to draw 3.2 samples in a pixel, we have a 80% chance of drawing 3 samples, and a 20% chance of drawing 4 samples. The rounding error is propagated from pixel to pixel, to ensure that the overall average is maintained.

We show the sample density maps obtained with our algorithm for a set of test scenes in Figure 9. This figure also shows ground truth sample density maps obtained using ground truth error maps, that is, the relative squared difference to a reference image. The density maps are similar in both cases, but the MSE is lower when driving the sampling using the ground truth error. We presume this is because the ground truth error captures the bias introduced by the filtering, while our estimation does not.

4 Implementation

We integrated our algorithm in the PBRT [Pharr and Humphreys 2010] framework. We implemented a dual-buffered film interface, along with a new sampler and denoiser. The renderer of PBRT was also modified to perform sampling over multiple iterations. We accelerate NL-Means filtering by exploiting the fact that all operations on pixel patches amount to averaging per-pixel values over a patch. This averaging could be performed in constant time using summed area tables [Liu et al. 2008], or in linear time using separable box filters. We use a box filter for numerical stability. Consequently, the cost of our implementation is mostly driven by the filter window size, while the patch size has a negligible impact. This brings the cost of the filter in line with that of a bilateral filter. Given the embarrassingly parallel nature of the filter, we ported it to CUDA to further improve performance. We suffer some overhead due to data transfer between the CPU and the GPU at each iteration, but this could be eliminated by using the NVIDIA OptiX [Parker et al. 2010] framework instead of PBRT to do the raytracing.

Our weighted error estimation, which we use to drive the adaptive sampling, assumes that the bias introduced by our filter is negligible. To minimize the amount of bias, we use a very conservative filter to drive sampling, and a more aggressive one for final reconstruction. During the iteration we use parameters $r = 7$ and $\alpha = 0.5$ in Equation 4. For final reconstruction we use $r = 10$ and $\alpha = 1$. In both cases we use $f = 3$ and $k = 0.45$. To en-

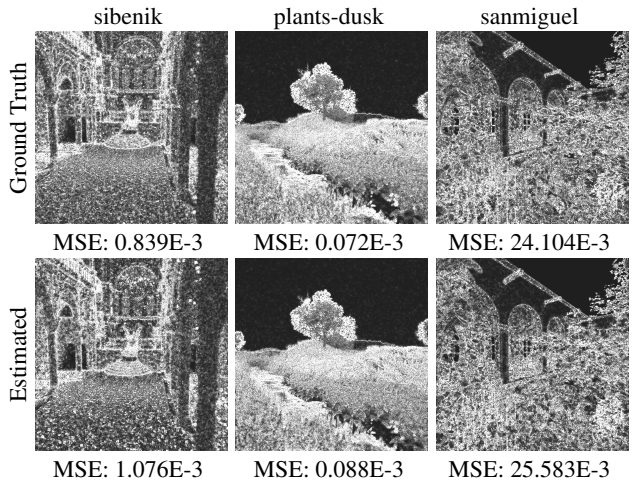

Figure 9: *Sampling density maps with* 32 *samples per pixel on average. The top row shows maps obtained by driving the sampling with the squared difference to the reference image. The bottom row shows maps obtained with our estimated error. We list* MSE *values of the corresponding renderings under each image.*

sure that our reconstruction of pixel values accounts for the sharply varying sample densities across image edges, we use a subpixel grid and simple push-pull filtering to fill subpixel holes, similarly as proposed by Rousselle et al. [2011].

5 Results

We evaluate our algorithm on a set of five test scenes and compare our results to the algorithm proposed by Rousselle et al. [2011] (ASR), and unfiltered uniform low-discrepancy sampling (LD). Our test machine is a dual 4-core XEON system at 2.50GHz, with 8GB of RAM, and a GeForce GTX580 GPU. All three methods are implemented on top of PBRT, using 8 threads. For the ASR algorithm we use the settings suggested in the original paper. For our algorithm we use four iterations, one with uniform sampling followed by three with adaptive sampling, and the parameters mentioned in Section 4. Each iteration is assigned one fourth of the total sample budget. We report the relative mean squared error (MSE), but also provide the perceptually based structural similarity (SSIM) [Wang et al. 2004] with respect to the reference in Table 1. The MSE is computed directly on the high-dynamic range data, while the SSIM is computed on tone-mapped images. Our tone mapping is a simple gamma correction, using $\gamma = 2.2$, followed by clamping to the range $[0, 1]$.

Renderings of our five test scenes are shown in Figure 14. We performed equal rendering time comparisons to account for the overhead of each method. We rendered all scenes at a resolution of 1024×1024 pixels. The "killeroos" scene illustrates the efficiency of our method for relatively low-dimensional cases including motion blur and area lighting. Our use of low-discrepancy samples coupled with the anisotropy of our filter provides a significant improvement over the ASR algorithm. The "plants-dusk" scene features environment lighting and depth of field. In the focused grass region shown in the close-up, the ASR algorithm produces an overly blurry image and increases the MSE compared to uniform low-discrepancy sampling, illustrating the limitation of the isotropic filters used in ASR. Our method, on the other hand, preserves the small anisotropic features while further removing some of the residual noise. The "sibenik" scene illustrates the case of a

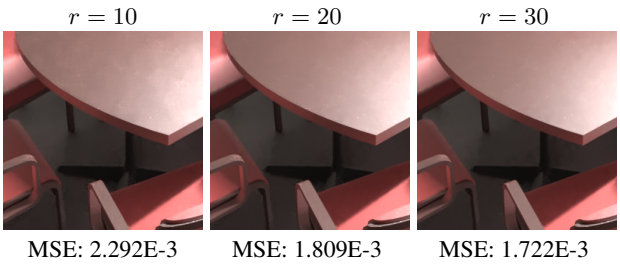

Figure 10: *Filtering of the "conference" scene using varying filter window sizes. With a smaller window size, the spike contributions to their neighborhoods are not sufficiently spread out, yielding a higher* MSE.

moderately noisy scene. It is path-traced and features area lighting, depth of field, and single-bounce indirect illumination. The ASR algorithm gives very good results in smooth regions of the scene, but has difficulties handling noise along the many sharp edges. Our method gives similar results in smooth regions, but is much more effective at resolving sharp edges. The "conference" scene highlights the robustness of our method to severe noise. To this end we rendered the scene using a path-tracer, even though it would be appropriate in practice to use a more sophisticated rendering algorithm that produces less noise in the first place. The scene features single-bounce indirect illumination and moderately glossy materials. The ASR method cannot reliably select the appropriate bandwidth in the presence of such high levels of noise, which leads to jarring filtering artifacts, while our method can still produce a pleasing image with few obvious artifacts. Also, the ASR algorithm tends to suppress noise spikes, producing darker patches on the top of the table. In contrast, our dual-buffered strategy, which decorrelates the filter weights from the noise, maintains the correct luminance by smoothing spikes instead of suppressing them. To ensure that the contribution of spikes was sufficiently distributed, we increased the filter window size to $r = 20$ for this scene (instead of $r = 10$ for the others). We show renderings of this scene using other window sizes in Figure 10. The "sanmiguel" scene illustrates the behavior of our method with highly complex scenes. It is path-traced and features environment lighting, single-bounce indirect illumination, and complex textured geometry. Here again, our algorithm significantly improves upon the results of the ASR algorithm. Noisy features in particular, such as the chandelier in the inset, are much better preserved by our algorithm. We also provide additional results including comparisons to reference images in the supplemental material with this submission.

The computational overhead of our method is largely dominated by the cost of the filtering step. For each iteration, we filter the pixel variance, and we cross filter the two image buffers. The pixel variance is filtered with $r = 1$, while the image buffers are cross filtered using either $r = 7$ (for the intermediate iterations), or $r = 10$ (for the final iteration). The total filtering cost over the four iterations amounts to 8.5s (1.9s per intermediate iteration, and 2.8s for the final iteration). For the "conference" scene we use $r = 20$ to cross-filter the buffers during the final iteration, which increases the total filtering cost to 16.5s, the final iteration taking 10.8s. For our simplest test scene, "sibenik", the overall overhead of our method represented less than 10% of the total rendering time, while reducing the MSE by a factor of 8.6 compared to standard low-discrepancy sampling.

We also provide convergence plots for standard low-discrepancy sampling with no filtering (LD), the ASR algorithm, and our own method in Figure 13. For our method we provide convergence plots

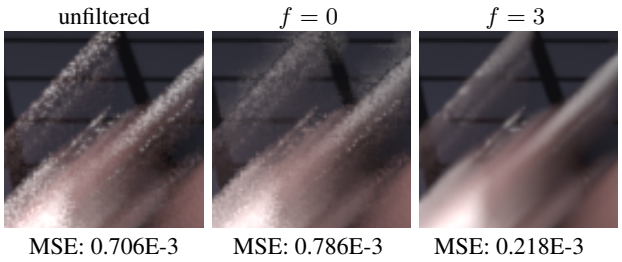

Figure 11: *Filtering of a uniformly sampled image with 16 samples per pixel, using two patch sizes, $f = 0$ and $f = 3$. By setting $f = 0$, we obtain the same behavior as the bilateral filter, minus the spatial component of its distance computation. In both cases, we set $k = 0.45$ and $r = 10$. The smaller patch size is not sufficient to compute robust pixel distances, leading to a significantly degraded image.*

using both adaptive sampling and uniform sampling. Our method consistently yields the lowest MSE value. The gain of adaptive sampling depends on scene complexity. In scenes such as the "conference" scene, where noise can be filtered out effectively over large regions, the adaptive sampling offers a marked improvement. The "killeroos" scene is similar, where noise is restricted to limited regions that the adaptive sampler targets extensively. On complex scenes such as the "sanmiguel" scene, too many features need to be sampled, negating any potential gain of the adaptive sampling.

In Figure 11, we show the impact of the patch-based distance computation of the NL-Means filter. To isolate the impact on the filtering, we used uniform sampling, instead of adaptive sampling. By setting the patch size parameter to $f = 0$, we get the same behavior as the bilateral filter, minus the spatial component of its distance computation. The resulting image has significantly more residual variance and bias because of the unreliable distance estimates than when setting $f = 3$, which is the value we use for all other results.

To illustrate the need for the per-pixel variance estimate used in our formulation of the NL-Means filter, we show in Figure 12 the result of filtering the "sibenik" scene (uniformly sampled using 32 samples per pixels) with a uniform variance estimate for all pixels in the image. In this case, we used the mean pixel variance (computed over the entire image) to guide the filtering. As expected, some regions are then correctly filtered, while others are over- or under-filtered. Some other value may yield a better result, but it would have to be hand-tuned and there would always remain an implicit trade-off between over- and under-filtered regions.

Temporal Coherence Temporal coherence is an important concern when considering video sequences, since inconsistencies between frames will produce flickering artifacts. For unfiltered video sequences these artifacts appear as fine grained temporal noise, but filtered sequences can produce disturbing flickering artifacts because of low-frequency errors. These affect large image patches and are very noticeable. We greatly mitigate these low-frequency artifacts by using a straight-forward space-time extension of the NL-Means filter [Buades et al. 2008], which simply extends a pixel neighborhood to also include pixels from adjacent frames. Inter-pixel distances are still computed over spatial patches centered on the respective pixels, but the extended neighborhood leverages the full spatio-temporal data set and increases temporal coherence. For results showcasing the spatial and space-time filtering approaches, we refer the reader to our supplemental material, which include video sequences of the "conference" and "sanmiguel" scenes. The "conference" sequence in particular illustrates the issue of low-

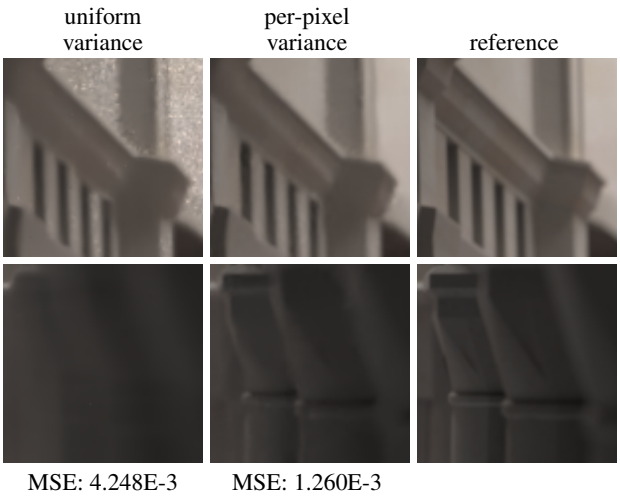

Figure 12: *Filtering of the "sibenik" scene (uniformly sampled using 32 samples per pixels) with the standard NL-Means formulation which assumes a uniform variance across the image (left column) and our own formulation which uses per-pixel variance estimates (middle column). The reference rendering is given in the right column. While filtering using a uniform variance estimate (the mean pixel variance over the image in this case) works for some regions, many end up over- or under-filtered.*

frequency flickering, which is very striking using spatial filtering, and significantly reduced by the space-time filtering extension. However, at the chosen sample rate, the low-frequency flickering is still very noticeable, even using space-time filtering. We believe there is room for further improvement of the temporal filtering, and this would be a fruitful avenue for future work.

Discussion and Limitations. The efficiency of our adaptive sample distribution is quite dependent on the accuracy of the pixel statistics. For scenes featuring moderate to high noise levels, we found it preferable to use a larger set of samples in the first iteration. This lead us to our current approach of distributing evenly the sampling budget over each iteration. Still, for the "killeroos" and "plants-dusk" scenes, rendered using only 16 samples per pixel, each buffer has only two samples per pixel after the first iteration, illustrating our method's robustness to sparsely sampled data. The main limitation of our method is its image-based nature. While this offers some compelling computational advantages, it limits the adaptivity of both the sampling and the filtering. Overlapping elements with different anisotropy should be processed with different filters, which our method cannot accommodate. Similarly, we rely on brute force sampling, which can be highly inefficient when noise comes from a small subset of the sampling domain. Our variance estimation for low-discrepancy samples offered significant improvements for all of our test scenes compared to random sampling. Nevertheless, it could be made more accurate to better handle scenes with very high frequencies, or sharp spikes of noise. In particular, we drive the cross filtering in part using a very noisy estimate of the variance of the pixel variance computed using only two samples, the two buffers' pixel variance estimates. We believe that an estimation based on the sample distribution kurtosis may lead to a more robust filtering. In the presence of strong noise spikes, we need to use a large filter window to spread out the spike contribution which can lead to an increased bias. Since noise spikes are often due to indirect light, it should be beneficial to filter separately the direct and indirect illuminations, using a larger filter only for

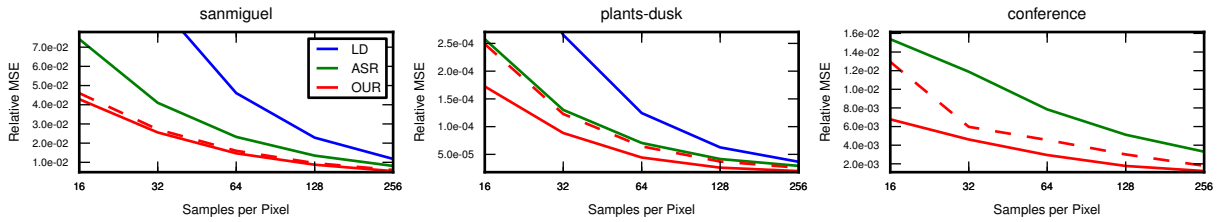

Figure 13: *Convergence plots for some of the scenes of Figure 14. For our method, we show the convergence using adaptive sampling (solid line), and uniform sampling (dashed line). In all scenes, our method consistently improves upon the ASR algorithm. The gain of adaptive sampling with our method is constrained by the scene complexity. For the "conference" scene, the* MSE *of the LD method is too high to appear on the plot.*

Table 1: *Perceptual quality, measured using the* SSIM *metric, for images of Figure 14. Values range from 0 to 1000, where 1000 indicates a perfect match with the reference. The* SSIM *metric is based on tone-mapped images using gamma correction ($\gamma = 2.2$) and clamping to the range $[0, 1]$. The LD column uses uniform low-discrepancy sampling with no filtering. The ASR column uses the method proposed by Rousselle et al. [2011] with adaptive random sampling. The OUR column uses our method with adaptive low-discrepancy sampling.*

Scene	LD		ASR		OUR	
	Full	Inset	Full	Inset	Full	Inset
killines	961	853	993	956	997	987
plants-dusk	971	962	990	912	996	975
sibenik	650	650	935	912	969	956
conference	523	416	914	877	970	963
sanmiguel20	755	532	844	793	891	863

the indirect illumination.

6 Conclusions

We described a robust and versatile adaptive method for Monte Carlo rendering. Our work offers a significant improvement over the previous state of the art in terms of both numerical error and visual quality. The two key characteristics of our method are its powerful non-linear filter, which can adapt to the anisotropy in the image, and its ability to process samples drawn from low-discrepancy sequences, whereas previous work was limited to samples drawn from a random distribution. The main limitation of our method, inherent to its image-space nature, is that it relies on brute force sampling to resolve noise, which is highly inefficient when useful light paths are difficult to find. It would be interesting to experiment with more robust sampling methods, such as Metropolis Light Transport [Veach and Guibas 1997]. Another venue of research would be to extend our adaptive sampling to the time domain in order to directly process animations, instead of individual frames. We could also augment the distance computation by comparing both the pixel mean and the pixel sample variance, which may improve the robustness of the filtering for low-contrast regions where the pixel mean provides insufficient constraints. Similarly, we could augment the distance computation by leveraging each sample's normal, depth, etc. In order to extend our approach to interactive rendering, we could make use of the new filter introduced by Gastal and Oliveira [2012] which can very efficiently approximate the results of the NL-Means filter, and therefore should be possible to integrate within our framework.

Acknowledgements

This project was funded in part by the Swiss National Science Foundation under grant no. 200021_127166. San Miguel model courtesy of Guillermo M. Leal Llaguno of Evolución Visual, Killeroo model courtesy of headus 3D, toasters scene by Andrew Kensler via the Utah 3D Animation Repository, dragon from the Stanford 3D Scanning Repository, Sibenik cathedral courtesy of Marko Dabrovic and Mihovil Odak, gargoyle courtesy of INRIA via the AIM@SHAPE repository, plants ecosystem from Deussen et al. [1998], yeahright model by Keenan Crane, and conference scene by Anat Grynberg and Greg Ward.

References

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

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

BOLIN, M. R., AND MEYER, G. W. 1998. A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98*, 299–309.

BUADES, A., COLL, B., MOREL, J., ET AL. 2005. A review of image denoising algorithms, with a new one. *SIAM Journal on Multiscale Modeling and Simulation 4*, 2, 490–530.

BUADES, A., COLL, B., AND MOREL, J. 2008. Nonlocal image and movie denoising. *International Journal of Computer Vision 76*, 2, 123–139.

CHEN, J., WANG, B., WANG, Y., OVERBECK, R. S., YONG, J.-H., AND WANG, W. 2011. Efficient depth-of-field rendering with adaptive sampling and multiscale reconstruction. *Computer Graphics Forum*.

DABOV, K., FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. 2007. Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on 16*, 8 (aug.), 2080 –2095.

DAMMERTZ, H., SEWTZ, D., HANIKA, J., AND LENSCH, H. P. A. 2010. Edge-avoiding à-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, HPG '10, 67–75.

DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MĚCH, R., PHARR, M., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. In *Proceedings of the*

25th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, SIGGRAPH '98, 275–286.

DONOHO, D. L., AND JOHNSTONE, J. M. 1994. Ideal spatial adaptation by wavelet shrinkage. *Biometrika 81*, 3, 425–455.

EGAN, K., TSENG, Y.-T., HOLZSCHUCH, N., DURAND, F., AND RAMAMOORTHI, R. 2009. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph. 28* (July), 93:1–93:13.

EGAN, K., HECHT, F., DURAND, F., AND RAMAMOORTHI, R. 2011. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Transactions on Graphics 30*, 2 (Apr.), 9:1–9:13.

GASTAL, E. S. L., AND OLIVEIRA, M. M. 2012. Adaptive manifolds for real-time high-dimensional filtering. *ACM Trans. Graph. 31*, 4 (July), 33:1–33:13.

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. W. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph. 27* (August), 33:1–33:10.

JI, Z., CHEN, Q., SUN, Q., AND XIA, D. 2009. A moment-based nonlocal-means algorithm for image denoising. *Information Processing Letters 109*, 23, 1238–1244.

KERVRANN, C., AND BOULANGER, J. 2006. Optimal spatial adaptation for patch-based image denoising. *Image Processing, IEEE Transactions on 15*, 10, 2866–2878.

KOLLIG, T., AND KELLER, A. 2002. Efficient multidimensional sampling. *Computer Graphics Forum 21*, 3, 557–563.

LEHTINEN, J., AILA, T., CHEN, J., LAINE, S., AND DURAND, F. 2011. Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph. 30* (August), 55:1–55:12.

LIU, Y., WANG, J., CHEN, X., GUO, Y., AND PENG, Q. 2008. A robust and fast non-local means algorithm for image denoising. *Journal of Computer Science and Technology 23*, 2, 270–279.

MAIRAL, J., ELAD, M., AND SAPIRO, G. 2008. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on 17*, 1 (jan.), 53 –69.

MITCHELL, D. P. 1987. Generating antialiased images at low sampling densities. *SIGGRAPH Comput. Graph. 21* (August), 65–72.

OVERBECK, R. S., DONNER, C., AND RAMAMOORTHI, R. 2009. Adaptive wavelet rendering. *ACM Trans. Graph. 28* (December), 140:1–140:12.

PARKER, S. G., BIGLER, J., DIETRICH, A., FRIEDRICH, H., HOBEROCK, J., LUEBKE, D., MCALLISTER, D., MCGUIRE, M., MORLEY, K., ROBISON, A., AND STICH, M. 2010. Optix: a general purpose ray tracing engine. In *ACM SIGGRAPH 2010 papers*, ACM, New York, NY, USA, SIGGRAPH '10, 66:1–66:13.

PHARR, M., AND HUMPHREYS, G. 2010. *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

PORTILLA, J., STRELA, V., WAINWRIGHT, M., AND SIMONCELLI, E. 2003. Image denoising using scale mixtures of gaussians in the wavelet domain. *Image Processing, IEEE Transactions on 12*, 11, 1338–1351.

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J., AND DACHSBACHER, C. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. 28* (December), 132:1–132:8.

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2011. Adaptive sampling and reconstruction using greedy error minimization. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, ACM, New York, NY, USA, SA '11, 159:1–159:12.

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

SHIRLEY, P., AILA, T., COHEN, J., ENDERTON, E., LAINE, S., LUEBKE, D., AND MCGUIRE, M. 2011. A local image reconstruction algorithm for stochastic rendering. In *Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '11, 9–14 PAGE@5.

SIJBERS, J., DEN DEKKER, A., VAN AUDEKERKE, J., VERHOYE, M., AND VAN DYCK, D. 1998. Estimation of the noise in magnitude mr images. *Magnetic Resonance Imaging 16*, 1, 87–90.

SOLER, C., SUBR, K., DURAND, F., HOLZSCHUCH, N., AND SILLION, F. 2009. Fourier depth of field. *ACM Trans. Graph. 28* (May), 18:1–18:12.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, IEEE, 839–846.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '97, 65–76.

WANG, Z., BOVIK, A., SHEIKH, H., AND SIMONCELLI, E. 2004. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on 13*, 4 (april), 600 –612.

XU, R., AND PATTANAIK, S. 2005. Non-iterative, robust monte carlo noise reduction. *IEEE Computer Graphics and Applications 25*, 2, 31–35.

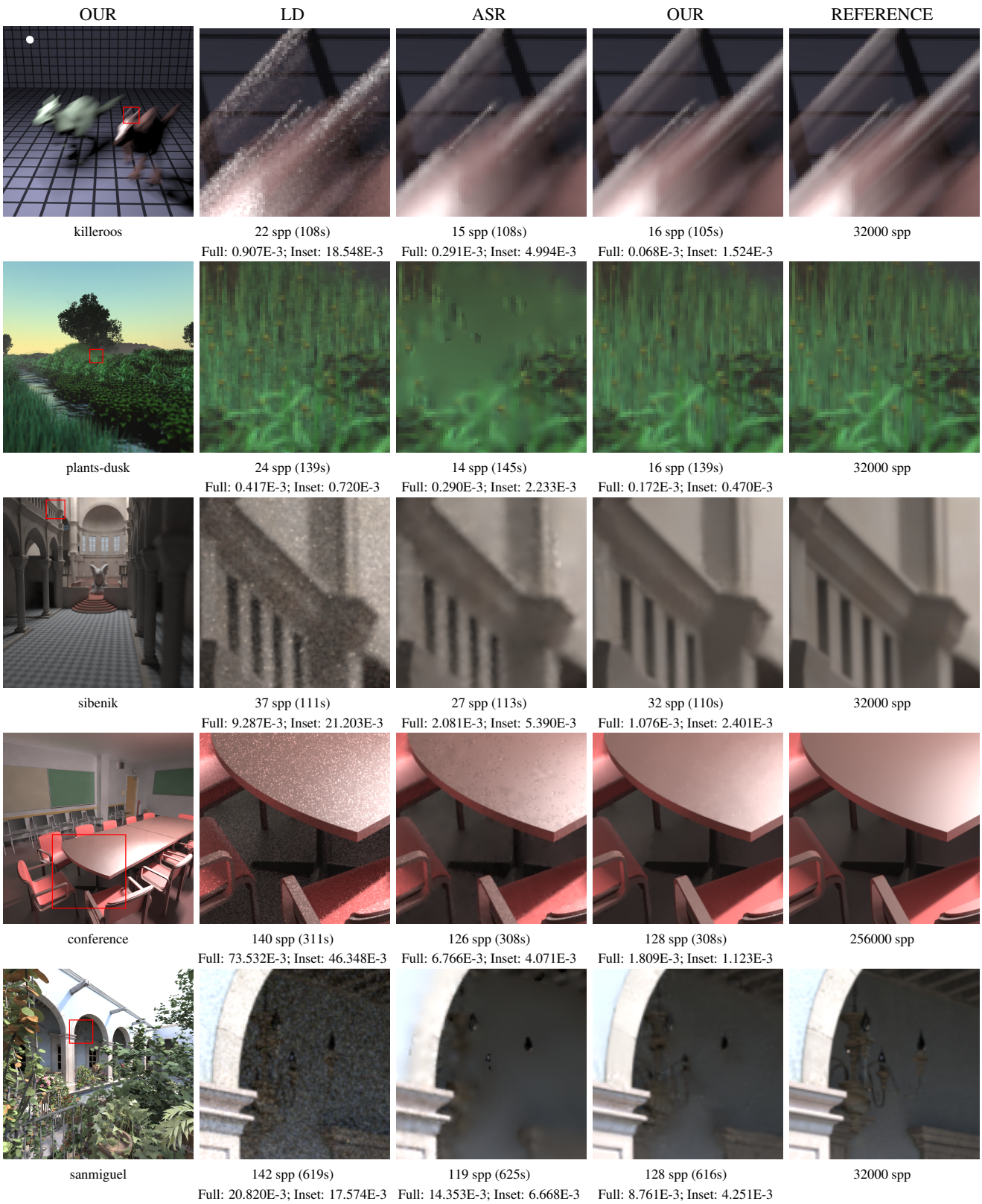

Figure 14: *Renderings using a variety of methods. Non-filtered uniform low-discrepancy sampling (LD); adaptive random sampling and filtering as proposed by Rousselle et al. [2011] (ASR); our method with adaptive low-discrepancy sampling (OUR). The last column shows nearly converged renderings using uniform sampling (REFERENCE). All results for a given scene have an equal rendering time, to account for each method overhead. For the conference scene, we use a larger filtering window ($r = 20$ instead of $r = 10$), to ensure the noise spikes are sufficiently spread out. The* MSE *values are listed under each image and the corresponding* SSIM *values are given in Table 1.*