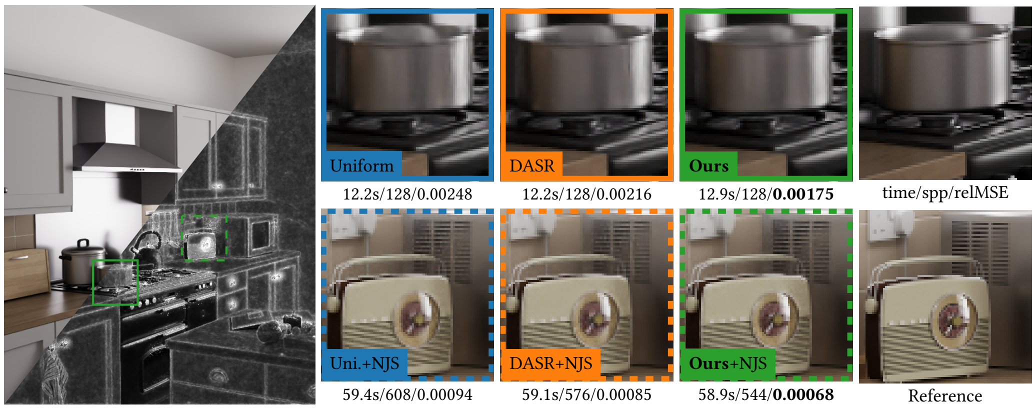
Denoising-Aware Adaptive Sampling for Monte Carlo Ray Tracing

Arthur Firmino
Luxion, Technical University of Denmark
Denmark
arthur.firmino@luxion.com

Jeppe Revall Frisvad
Technical University of Denmark
Denmark
jerf@dtu.dk

Henrik Wann Jensen
Luxion
USA
henrik.jensen@luxion.com



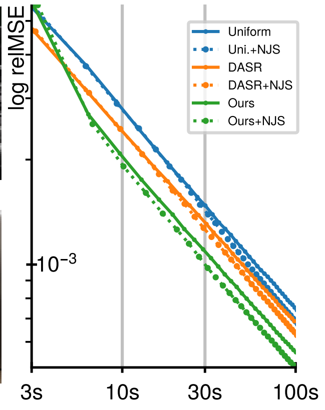


Figure 1: We propose a novel variance estimate for neural networks and apply it to a denoiser (here *Intel Open Image Denoise*) to obtain a denoising-aware sampling map (left). The close-ups show comparisons of our adaptive sampling method with uniform sampling and deep adaptive sampling (DASR) [Kuznetsov et al. 2018] (top row) and with additional post-correction using the neural James-Stein (NJS) combiner [Gu et al. 2022] (bottom row). In all cases, our adaptive sampling method leads to faster convergence (right). Scene: Country Kitchen by Jay-Artist (CC-BY).

ABSTRACT

Monte Carlo rendering is a computationally intensive task, but combined with recent deep-learning based advances in image denoising it is possible to achieve high quality images in a shorter amount of time. We present a novel adaptive sampling technique that further improves the efficiency of Monte Carlo rendering combined with deep-learning based denoising. Our proposed technique is general, can be combined with existing pre-trained denoisers, and, in contrast with previous techniques, does not itself require any additional neural networks or learning. A key contribution of our work is a general method for estimating the variance of the outputs of a neural network whose inputs are random variables. Our method iteratively renders additional samples and uses this novel variance estimate to compute the sample distribution for each subsequent iteration. Compared to uniform sampling and previous adaptive sampling techniques, our method achieves better equal-time error in all scenes tested, and when combined with a recent denoising post-correction technique, significantly faster error convergence is realized.

SIGGRAPH '23 Conference Proceedings, August 06–10, 2023, Los Angeles, CA, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0159-7/23/08.
https://doi.org/10.1145/3588432.3591537

CCS CONCEPTS

• **Computing methodologies → Ray tracing.**

KEYWORDS

adaptive sampling, denoising, neural network, progressive rendering, variance

ACM Reference Format:
Arthur Firmino, Jeppe Revall Frisvad, and Henrik Wann Jensen. 2023. Denoising-Aware Adaptive Sampling for Monte Carlo Ray Tracing. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 06–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3588432.3591537

1 INTRODUCTION

Monte Carlo ray tracing methods, and in particular light transport algorithms such as path tracing [Kajiya 1986], have become the de facto standard in offline production rendering and have recently also been gaining attention in interactive and real-time applications. The stochastic nature of such algorithms means that many samples per pixel (spp) are often required to achieve noise-free images, and despite faster and dedicated ray tracing hardware, doing so can still be prohibitively expensive.

Advances in image denoising methods have enabled the increased adoption of Monte Carlo ray tracing, as these enable one to remove noise from a rendered image whilst sufficiently preserving image details. Substantial progress was made even prior to the wider adoption of deep learning, an overview is available from Zwicker et al. [2015]. Some of these methods in particular relied on the use of auxiliary features, sample statistics, and error estimates to optimize various filter parameters. As deep learning gained popularity and ease of use, research produced new denoising techniques, the majority based on convolutional neural networks and also relying on auxiliary features such as surface albedo and normals.

Sample variance is generally non-uniform over image space. Adaptive sampling has thus seen wide usage in production renderers [Kulla et al. 2018; Christensen et al. 2018; Fascione et al. 2018; Burley et al. 2018], often simply guided by variance since the higher sample counts used in production allow for better variance estimates. Earlier non-learning based denoising methods have proposed combining iterative adaptive sampling with denoising [Li et al. 2012], as the use of filtering enables computing a variance or error estimate of the denoised image which may be of more interest than the noisy rendered image. More recently, methods have been proposed based on deep learning where a neural network, trained in tandem with a denoising network, is used to predict the sample distribution for the renderer [Kuznetsov et al. 2018; Salehi et al. 2022]. One drawback of such an approach is that the resulting prediction may be more a function of the network's training and might have less to do with the variance of the denoised image.

In this paper, we bridge the gap between previous adaptive sampling work and modern deep-learning based denoisers by introducing a novel variance estimate for denoised outputs. We compute this in a single pass while simultaneously denoising the image. Our estimate is based on a first-order Taylor series expansion of the denoiser, and is computed using forward mode automatic differentiation and the Jacobian-vector product. The estimated variance is then used to compute the sample distribution for the subsequent rendering iteration, similar to previous methods. The advantage is that our sample distribution places additional samples where they will benefit the denoised image rather than the noisy input image.

In a progressive rendering, we would prefer to always look at the denoised image. Maybe even the tone-mapped version. Modern denoisers use a neural network, so our method enables a direct way of achieving denoising-aware adaptive sampling. We can let our method adapt to the tone mapping as well. If we run the denoiser for every progressive update to always look at the denoised image, the overhead of our method is modest. Even if not, the cost is not more than using a separate neural network for adaptive sampling. Recent work suggests adaptive progressive pixelwise blending of the denoised image with the input image to ensure consistency (convergence to the ground truth) [Firmino et al. 2022; Gu et al. 2022]. By correspondingly blending the variance estimates, our method enables denoising-aware adaptive sampling for this type of consistent denoised progressive rendering too.

2 RELATED WORK

2.1 Monte Carlo Denoising

Image denoising is a vital component of modern rendering pipelines, necessary in production rendering for the practical application of Monte Carlo light transport algorithms such as path tracing. Pioneering approaches for denoising Monte Carlo rendered images sought to reduce stochastic noise by locally blurring the image using a variety of non-linear filters [Mitchell 1987; Lee and Redner 1990; Rushmeier and Ward 1994; Xu and Pattanaik 2005; Overbeck et al. 2009]. Later work locally adapt these filters by incorporating knowledge on auxiliary image features such as surface normals and albedo, pixel variance, and estimates of reconstructed error [McCool 1999; Li et al. 2012; Rousselle et al. 2013; Bitterli et al. 2016].

Advances in the training of neural networks, coupled with the wider availability of capable hardware led to the use of deep learning in new denoising algorithms. Approaches have been diverse and range from prediction of filter parameters [Kalantari et al. 2015], or entire filter kernels [Bako et al. 2017; Vogels et al. 2018], to direct prediction of radiance values and coupling denoising with supersampling [Xiao et al. 2020; Thomas et al. 2022]. In terms of image quality, particularly at lower sample counts [Chaitanya et al. 2017; Gharbi et al. 2019; Hasselgren et al. 2020], learning-based denoisers have outclassed previous classical approaches, however difficulties relating to non-zero bias at high sample counts has led to the recent development of post-correction techniques [Back et al. 2020]. In this area, Zheng et al. [2021] proposed an ensemble method for combining multiple denoised images, Firmino et al. [2022] proposed training a network on error estimates to infer blending parameters between rendered and denoised images, Back et al. [2022] proposed a self-supervised loss for correcting network parameters during denoising, and most recently Gu et al. [2022] proposed a method based on James-Stein theory and using a network to help estimate the variance of the unbiased input. Our method enables adaptive sampling that adapts to such denoising techniques.

2.2 Adaptive Sampling

Adaptive sampling can improve the quality of Monte Carlo rendered images by, for example, allocating samples proportionally to sample variance. Many of the classical denoising works applied this to image reconstruction by developing error or variance estimates for the reconstructed image and using those estimates to guide adaptive sampling, thus achieving better image quality in equal time [Dippé and Wold 1985; Lee et al. 1985; Painter and Sloan 1989]. In the development of adaptive sampling [Zwicker et al. 2015], the *a posteriori* techniques are the ones more closely related to our work. Rousselle et al. [2012], for example, leverage denoising based on non-local means filtering and estimate output variance using the difference of two independent filtered images. Local, unbiased error estimates of cross-bilateral filters was also found beneficial for adaptive sampling [Li et al. 2012; Rousselle et al. 2013]. We transfer this type of adaptive sampling to a neural network setting.

With denoising based on neural networks, methods for adaptive sampling have been proposed that also rely on recent advances in deep learning. Vogels et al. [2018] investigated training an error-predicting network to be used for adaptive sampling and evaluated the quality of different guiding metrics. Kuznetsov et al. [2018] tackled the problem of adaptive sampling at very low sample counts by training a sampling map predictor in tandem with the denoising network by propagating gradients through the renderer. This idea was later combined with temporal optimization by Hasselgren et al. [2020] to improve quality at interactive rendering rates. Salehi et al. [2022] made the training of such networks practical for higher sample count rendering, eschewing the expensive rendering of samples during training, and instead generating them from analytical distributions. One commonality between these methods is the quality of the predicted sampling distribution, which relies on the quality of the network training and may therefore be unable to adapt to new scenes or sample counts outside the training dataset's distribution. We contrast these previous approaches by instead basing our method on statistical estimates of the denoiser's output.

2.3 Statistical Estimates for Denoised Images

We briefly review estimates of two statistics, variance and error, applicable to images denoised using deep-learning based approaches and cite their appearance in rendering literature.

2.3.1 Double-Buffer Variance Estimate. A common method for estimating the variance involves rendering two independent images, $\mathbf{x}_a$ and $\mathbf{x}_b$, such that the denoised images $f(\mathbf{x}_a)$ and $f(\mathbf{x}_b)$ are independent from each other. Their variance can then be estimated by simply calculating the squared difference:

$$\text{Var}\left[f_i(\mathbf{x})\right] = \tfrac{1}{2}\mathbb{E}\left[(f_i(\mathbf{x}_a) - f_i(\mathbf{x}_b))^2\right].$$

It should be noted that this variance estimate is not of the combined and denoised samples, $f(\frac{1}{2}(\mathbf{x}_a + \mathbf{x}_b))$, as one might wish. For $f(\cdot)$ as a learning-based denoiser, this estimate appears in the method of Gu et al. [2022]. Earlier it appeared in the context of adaptive sampling [Rousselle et al. 2012] and denoising [Bitterli et al. 2016].

2.3.2 Double-Buffer Error Estimate. For estimating the squared error of a denoised image, a method in the same vein as Section 2.3.1, involves rendering two independent images, $\mathbf{x}_a$ and $\mathbf{x}_b$, such that the denoised image $f(\mathbf{x}_a)$ is independent of $\mathbf{x}_b$, and likewise for $f(\mathbf{x}_b)$ and $\mathbf{x}_a$. The corresponding error estimate for the ith pixel, $\mathbb{E}\left[(f_i(\mathbf{x}) - \mu_i)^2\right]$, is equal to

$$\mathbb{E}\left[\frac{1}{2}((f_i(\mathbf{x}_a) - \mathbf{x}_{b,i})^2 - \hat{\sigma}^2_{b,i} + (f_i(\mathbf{x}_b) - \mathbf{x}_{a,i})^2 - \hat{\sigma}^2_{a,i})\right],$$

where $\hat{\sigma}^2_a$ and $\hat{\sigma}^2_b$ are unbiased estimates of the variance of $\mathbf{x}_a$ and $\mathbf{x}_b$, respectively. However, just as the variance estimate before, this error estimate is not of the combined and denoised samples, $f(\frac{1}{2}(\mathbf{x}_a + \mathbf{x}_b))$. This error estimate forms the basis of the self-supervised loss function recently proposed by Back et al. [2022].

2.3.3 Stein's Unbiased Risk Estimate (SURE). Another possibility is using SURE [Stein 1981], which has previously been used for adaptive sampling by Li et al. [2012]. The expression for SURE, in the general multivariate normal case, is [Liu 1994]

$$\mathbb{E}\left[(f_i(\mathbf{x}) - \mu_i)^2\right] = \mathbb{E}\left[(f_i(\mathbf{x}) - \mathbf{x}_i)^2 + 2(\mathbf{J}_f(\mathbf{x})\,\hat{\Sigma})_{i,i} - \hat{\sigma}^2_i\right],$$

where $\mathbf{J}_f(\mathbf{x})$ is the Jacobian matrix of f, and $\hat{\Sigma}$ is the estimated covariance matrix of $\mathbf{x}$. This estimate is valid for normally-distributed $\mathbf{x}$, a condition which may be approximately satisfied given a sufficiently high sample count, as per the Central-Limit Theorem.

3 DENOISING-AWARE ADAPTIVE SAMPLING

We consider possibilities for *a posteriori* methods of performing adaptive sampling in tandem with neural network based denoisers. As pointed out by Zwicker et al. [2015], this will require at least estimating some statistic about the denoised image, such as error or variance. Methods prior to learning-based denoisers had the advantage of being able to derive closed-form solutions for such estimates, as the denoising filter was often also expressed in closed-form. In the following section we seek to bridge this gap with our variance estimate for deep neural networks, which forms the backbone of our denoising-aware adaptive sampling proposal.

3.1 Deep Neural Network Variance Estimate

To avoid the need for a second denoised image (double-buffer) and improve on the noise in previous denoiser variance estimates, we introduce a novel Monte Carlo estimate of a deep neural network's output variance given variance estimates of its input. We find a method for its practical computation using forward mode auto-differentiation available in popular deep learning frameworks.

Given a function $f_i : \mathbb{R}^N \rightarrow \mathbb{R}$ of independent random variables $X_1, ..., X_N$ with variances $\text{Var}[X_j] = \sigma^2_j$, a first-order approximation of the variance of f_i, based on its Taylor series expansion, is (the law of propagation of errors) [Mandel 1964; Wolter 2007]

$$\text{Var}[f_i(X_1, ..., X_N)] \approx \sum_{j=1}^{N} \left|\frac{\partial f_i}{\partial x_j}\right|^2 \sigma^2_j\,. \tag{1}$$

The non-constant part of a first-order Taylor expansion is a directional derivative. For a neural network function f, we leverage forward mode auto-differentiation to find a practical method for computing the directional derivative of the network's output. This directional derivative is synonymous with the Jacobian-vector product [Grosse 2021]. For a neural network function $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$, its Jacobian-vector product (JVP) of $\mathbf{v} \in \mathbb{R}^N$ at $\mathbf{x} \in \mathbb{R}^N$ is equal to $\mathbf{J}_f(\mathbf{x})\mathbf{v} \in \mathbb{R}^M$, where $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{M\times N}$ with $\mathbf{J}_f(\mathbf{x})_{i,j} = \frac{\partial f_i}{\partial x_j}(\mathbf{x})$.

We select $\mathbf{v} \in \mathbb{R}^N$ as a vector of random variables for which each element $v_j = \{+\sqrt{\hat{\sigma}^2_j}, -\sqrt{\hat{\sigma}^2_j}\}$ with equal probability, and where $\hat{\sigma}^2_j$ is an unbiased estimate of σ^2_j. Then $\mathbb{E}[v_j] = 0$, $\mathbb{E}[v^2_j] = \sigma^2_j$, and $\mathbb{E}[v_j v_k] = \mathbb{E}[v_j]\mathbb{E}[v_k] = 0$ for $j \neq k$. Considering observed $\mathbf{x}$, the expectation of the square of the ith element of the JVP at $\mathbf{x}$ is then

$$\begin{aligned}
&\mathbb{E}\left[\left(\left(\mathbf{J}_f(\mathbf{x})\mathbf{v}\right)_i\right)^2\right] = \mathbb{E}\left[\left(\textstyle\sum_{j=1}^N \frac{\partial f_i}{\partial x_j}(\mathbf{x})v_j\right)^2\right] \\
&= \mathbb{E}\left[\textstyle\sum_{j=1}^N \left|\frac{\partial f_i}{\partial x_j}(\mathbf{x})\right|^2 v^2_j + 2\sum_{j=1}^N \sum_{k=1}^{j-1}\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x})\frac{\partial f_i}{\partial x_k}(\mathbf{x})\right)v_j v_k\right] \\
&= \textstyle\sum_{j=1}^N \left|\frac{\partial f_i}{\partial x_j}(\mathbf{x})\right|^2 \mathbb{E}[v^2_j] + 2\sum_{j=1}^N \sum_{k=1}^{j-1}\left(\frac{\partial f_i}{\partial x_j}(\mathbf{x})\frac{\partial f_i}{\partial x_k}(\mathbf{x})\right)\mathbb{E}[v_j v_k] \\
&= \textstyle\sum_{j=1}^N \left|\frac{\partial f_i}{\partial x_j}(\mathbf{x})\right|^2 \sigma^2_j \approx \text{Var}[f_i(X_1, ..., X_N)]\,.
\end{aligned} \tag{2}$$

In practice, the common approach for computing the Jacobian-vector product is to evaluate the directional derivative at each of the neural network's layers with respect to that computed in the previous layer, and this can be computed in a single pass while simultaneously evaluating the network. An even more efficient method for computing the JVP was recently proposed [Balestriero and Baraniuk 2021]. Our approach can directly reap the benefits once this method becomes commonly available. We found our proposed estimate to resemble an older method described by Christianson and Cox [2006], with the key difference being their method requiring the propagation of every coefficient of a multivariate first-order Taylor series to each output, an approach intractable for neural networks. In this manner, our suggestion of using the

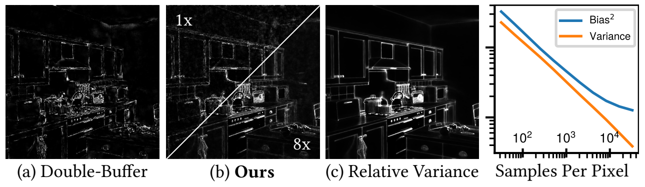


Figure 2: Qualitative comparison of two different estimates of the relative variance of a denoised image using: (a) two independently denoised 16spp inputs (Section 2.3.1) or (b) one or eight v-vector samples (1x or 8x) for the estimate in Eq. 2 on a single denoised 32spp input. We also show (c) ground truth values, computed using many independently rendered and denoised images. The double-buffer estimate exhibits more low-frequency error making it a poor guide for adaptive sampling, as large regions may erroneously receive too few samples. The rightmost plot shows the relative variance and relative squared bias of the denoised image up to 32k samples per pixel, revealing that variance is a relevant component of its error up to at least 10k samples for this scene.

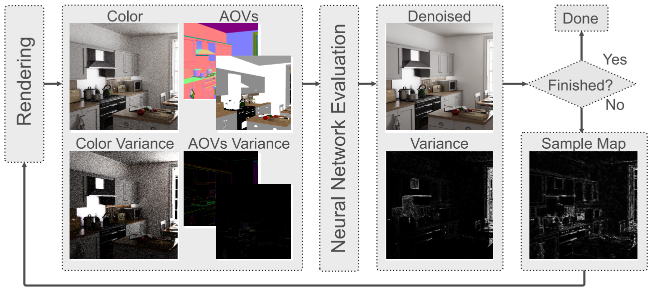


Figure 3: General overview of the proposed iterative adaptive sampling algorithm. Forward auto-differentiation is used to calculate a variance estimate of the denoised image, as described in Eq. 1. The sample distribution for the subsequent iteration is calculated according to Eq. 3.

Jacobian-vector product which requires the propagation of only a single scalar to each output is the advancement required for making the estimate applicable to neural networks.

In the context of uncertainty quantification in deep learning [Abdar et al. 2021], our estimate is related to the quantifying of aleatoric uncertainty that arises from noise in the network's input. In contrast to the variance propagation method of Postels et al. [2019], used to estimate the epistemic uncertainty of a model using Monte Carlo dropout, our method does not ignore the covariance between features in the network's hidden layers, stated as a practical limitation in their work and in one experiment resulting in a magnitude underestimation of 93.7%.

For the case of Monte Carlo denoising, where the neural network's output is local as it is effectively a weighted sum of many individual pixel estimates, its output has significantly less (although correlated) variance than its inputs. This is also reflected in our estimate of its variance, which exhibits little to no pixel-wise independent noise, as exemplified in Figure 2.

3.2 Denoising-Aware Adaptive Sampling

We present an algorithm for use with iterative adaptive sampling (see Figure 3): given a pre-existing neural network based denoiser, we distribute samples according to the variance of the current denoised image. The initial iteration uses samples uniformly distributed over the image plane. After the rendering phase of each iteration, we denoise the accumulated samples while simultaneously computing the variance of the denoised image. The variance of each denoised pixel is then divided by the square of the denoised radiance value of that same pixel. We also divide by the number of accumulated samples so that sampling is proportional to the expected relative variance decrease from one additional sample. The resulting image has values

$$\frac{\mathrm{Var}[f_i(\mathbf{x})]}{(N_i+1)(f_i(\mathbf{x})^2+\epsilon)}\,, \tag{3}$$

where we use $\epsilon = 10^{-2}$ to avoid dividing by 0. This image is clipped to a minimum of 0, blurred using a gaussian filter with a standard deviation of 0.5 in a 5×5 window, and subsequently normalized. The blurring is done to avoid large discontinuities in the sample distribution, as the denoiser used may not always be robust to such inputs. For unbounded images, we opted to use relative variance to guide our adaptive sampling to avoid over-emphasizing brighter areas of the image, similar to previous work [Li et al. 2012].

3.2.1 Denoising with Post-Correction. Our variance estimate is easily applied to the recently proposed neural James-Stein (NJS) combiner [Gu et al. 2022], yielding an effective adaptive sampling technique. In their work, two biased inputs, denoted here as $f(\mathbf{x}_a)$ and $f(\mathbf{x}_b)$, produced by denoising independent unbiased estimates x_a and x_b, are blended by predicted per-pixel parameters α_i, and then combined with the unbiased input $\mathbf{x} = (\mathbf{x}_a + \mathbf{x}_b)/2$ using parameters ρ_i whose calculation is based on James-Stein theory. The variance of the two denoised inputs, $\mathrm{Var}[f(\mathbf{x}_a)]$ and $\mathrm{Var}[f(\mathbf{x}_b)]$, can be computed using our estimate from Section 3.1, with which, along with the variance estimate of the unbiased input $\mathrm{Var}[\mathbf{x}]$, the variance of the final combined output is estimated by

$$\rho_i^2(\alpha_i^2\mathrm{Var}[f_i(\mathbf{x}_a)] + (1-\alpha_i)^2\mathrm{Var}[f_i(\mathbf{x}_b)]) + (1-\rho_i)^2\mathrm{Var}[\mathbf{x}]\,, \tag{4}$$

which we directly substitute for the numerator in Eq. 3 to get the adaptive sampling technique. We overlook the fact that α_i and ρ_i are themselves random variables, and this simplification seems reasonable given that we expect these parameters to have little to no variance, possibly except when the post-correction is transitioning from biased to unbiased inputs. We also omitted the regression-based optimization used by Gu et al. [2022], which improves the quality of the biased inputs at lower sample counts, as enabling it would have required deriving the variance of the regressed images.

3.2.2 Tone Mapping. It is useful to consider that the final output image may be tone mapped to values in $[0, 1]$, and that we may wish to incorporate this in our adaptive sampling. We can easily incorporate a tone-mapping operator $\mathcal{T} : \mathbb{R}^N \rightarrow \mathbb{R}^N$ into our method by redefining the denoising function, $f_{\mathcal{T}}(\mathbf{x}) = \mathcal{T}(f(\mathbf{x}))$, using auto-differentiation to compute

$$\frac{\partial}{\partial x_j} f_{\mathcal{T}i}(\mathbf{x}) = \sum_{k=1}^{N}\left(\frac{\partial}{\partial f_k}\mathcal{T}_i(f(\mathbf{x}))\frac{\partial}{\partial x_j}f_k(\mathbf{x})\right) \tag{5}$$

and in this way include the tone mapping in our variance estimate. When computing the sample distribution, it is no longer necessary or desirable to use relative variance as in Eq. 3 since the output is now bounded, so we simply use $\mathrm{Var}[f_{\mathcal{T}i}(\mathbf{x})]/(N_i + 1)$.

4 IMPLEMENTATION

We implemented our method in PyTorch [Paszke et al. 2019] using its forward auto-differentiation (autodiff) capabilities to compute the Jacobian-vector product (JVP) and neural network variance as outlined in Section 3.1. For the denoising network, we opted for the U-Net architecture [Ronneberger et al. 2015] and used the pre-trained weights from version 1.4.3 of Intel's Open Image Denoise [Áfra 2019], as it is already of high quality and doing so saved us from the trouble of training a denoising network from scratch. An additional advantage of the U-Net architecture is that it is relatively simple and standard, meaning we could apply forward autodiff and compute the JVP out of the box. Evaluating the network while simultaneously computing the JVP takes approximately three times longer than just evaluating the network.

While we could have picked a different denoising architecture, such as KPCN [Bako et al. 2017] and used pre-trained weights from existing work, doing so would have required implementing forward differentiation for the custom operations of that work. Writing the corresponding forward autodiff operations should not be more challenging than implementing the backward autodiff operations typically required for training. Unlike backpropagation, forward propagation does not require storing additional state. The time and space complexity of computing the forward autodiff steps is the same as the forward evaluation steps, unless the operation involves difficult to compute derivatives which is rarely the case.

Of note when combining our method with the post-correction method of Gu et al. [2022], as their method is implemented in a machine learning framework other than PyTorch, we omit the high time overhead of switching between contexts, approximately 130 milliseconds, such that our reported results are representative to that of a regular implementation.

For our experiments, we used Mitsuba 3 [Jakob et al. 2022] with a block size of 2×2 pixels to better facilitate thread parallel adaptive sampling. For certain scenes, adaptive sampling takes longer to render an overall equal amount of samples. This is due to the non-uniform cost per sample for each pixel and our method often placing samples in regions with higher cost. We briefly experimented with measuring this cost and taking it into account when distributing samples, akin to recent work considering the cost of parameter decisions [Grittmann et al. 2022; Rath et al. 2022]. While this resulted in slightly quicker rendering iterations, it did not improve the overall speed-up with regard to the error metrics used.

We also experimented with changing the granularity of the rendering iterations, from sample counts as low as 4 to as high as 128. The benefit of finer iterations is the time to first image and subsequent benefits from adaptive sampling. However, after sufficient samples there was little benefit in terms of equal-time quality. One penalty from using finer iterations is the increased overhead of switching context between rendering and denoising, and while our method proved robust to finer iterations, we ultimately settled on 32 samples per pixel as providing a good balance.

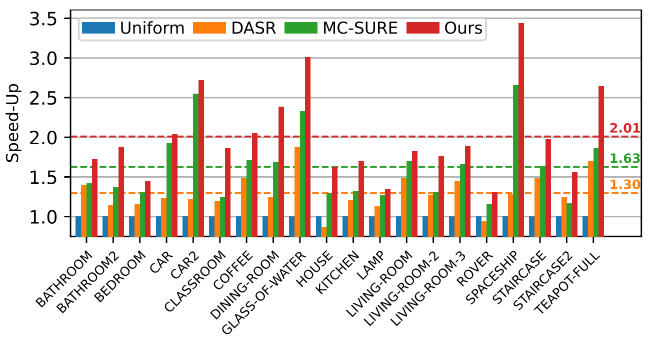


Figure 4: Speed-up over uniform sampling, in equal-error terms using relMSE, after 30s of rendering for the different adaptive sampling methods. Our main proposed method consistently outperforms all others. Horizontal lines indicate average speed-up over the 20 scenes.

4.1 Adaptive Sampling with MC-SURE

In their adaptive sampling work, Li et al. [2012] find the derivatives of the cross-bilateral filters they use. With these, they can obtain the middle term of the SURE expression (Section 2.3.3) in its entirety. To compute SURE for a neural network, we follow the approach of Firmino et al. [2022] and compute the middle term using the Monte Carlo SURE estimate put forward by Ramani et al. [2008]:

$$(\mathrm{J}_f(\mathbf{x}) \cdot \hat{\Sigma})_{i,i} \approx \frac{1}{\epsilon K} \sum_{k=1}^{K} \left(b_k^T (f(\mathbf{x} + \epsilon \mathbf{b_k}) - f(\mathbf{x})) \right)_i .$$

In this expression, b_k are random vectors distributed according to $\mathcal{N}(0, \hat{\Sigma})$, $\epsilon = 10^{-4}$, and $K = 4$ is the number of samples for the estimate. Computing SURE in this manner, using the denoiser in question, stands in contrast to computing SURE for cross-bilateral filters despite denoising with a neural network. The latter approach was used in the adaptive sampling comparisons of Kuznetsov et al. [2018], where they found their approach superior to that of Li et al. [2012]. When performing adaptive sampling using SURE, we simply substitute the error estimate for the numerator of Eq. 3 and use a standard deviation of 4.0 with a 17×17 window for the Gaussian filter due to the higher variance of the estimate. As adaptive sampling using SURE in this manner has not previously been carried out, we consider it part of our contribution and include it in our comparisons, denoted MC-SURE.

5 RESULTS AND DISCUSSION

Our results are presented in equal-time terms when comparing to uniform sampling and other adaptive sampling methods. This allows us to account for the fact that iteratively computing the JVP incurs higher overhead than simply denoising, and that rendering samples that are adaptively distributed often takes longer due to the non-uniform sample cost between different pixels. We compared our main proposed method, and MC-SURE, to deep adaptive sampling and reconstruction (DASR) [Kuznetsov et al. 2018]. For our implementation of their method, we used the same sampling map estimator network trained in their work. As in their work, the sampling map is computed only once after one sample per pixel and is used for the rest of the rendering. The main discrepancy in

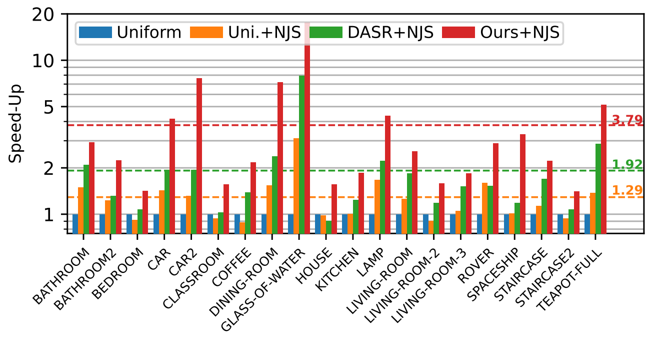


Figure 5: Comparison of different adaptive sampling techniques when combined with the post-correction method by Gu et al. [2022]. Speed-up is over uniform sampling (without post-correction to enable comparison with the factors in Figure 4) in equal-error terms using relMSE after 30 seconds.

our implementation is not using a denoiser trained in tandem with the sampling network, although the network architecture remains similar. Another caveat of our comparison to DASR, is that it is targeted at very low sample count rendering which is not the target domain of our work, and a better comparison would be to the method of Salehi et al. [2022].

The images in our experiments were rendered at 1280×720 or comparable resolution. On average, denoising took 33 milliseconds, or 110 milliseconds when simultaneously estimating the variance of the output. The duration of our rendering iterations, all 32 samples per pixel, varied depending on scene but on average lasted 2.60 seconds for uniform sampling, and 2.91 seconds for adaptive sampling. In all of our experiments, rendering was performed using a 32-core CPU, and denoising done using an NVIDIA RTX 3090 GPU. In our results, we report whole image errors using relative mean squared error (relMSE), $\frac{1}{|P|}\sum_{i=1}^{|P|}(t_i - r_i)^2/(\epsilon + r_i^2)$ where $\epsilon = 10^{-2}$, with the exception of the tone mapping experiment which uses root mean squared error (RMSE), $(\frac{1}{|P|}\sum_{i=1}^{|P|}(t_i - r_i)^2)^{1/2}$, with t_i and r_i being the ith pixel of the test and reference images, respectively, and $|P|$ the number of pixels times channels (three).

Path Tracing with Denoising. For comparing the various adaptive sampling techniques on path-traced inputs, we used a dataset of 20 publicly available scenes [Bitterli 2016]. The rendered images were denoised and their resulting error computed for every 32 samples. This error was then used to calculated the speed-up over uniform sampling. The results, after 30 seconds of rendering, are shown in Figure 4, with detailed results for specific scenes in Figure 7.

Our experiments indicate that adaptive sampling using our proposed variance estimate resulted in the best improvement, compared to MC-SURE, or the learned method. We attribute the better performance of our main contribution to two factors: the variance estimate has itself significantly less variance than the MC-SURE estimate, and therefore does not need to be as aggressively filtered; adaptive sampling proportionally to variance should provide greater benefit than sampling proportionally to error (variance plus squared bias), as there is a direct relationship between the input variance and output variance (Eq. 1) while the relationship between input variance and output bias for neural network denoisers is less clear. We verify this benefit further below in our comparisons to ground-truth sampling. The learned method from Kuznetsov et al. [2018], while it did improve over uniform sampling, was on average worse than both of our methods after our methods' initial iterations. Unlike our methods however, DASR is capable of predicting its sampling map after just one sample per pixel, and at very low sampling rates achieves lower error, while ours would lack reliable statistical estimates.

Table 1: The impact of incorporating the tone-mapping operator in the variance estimate, $\mathrm{Var}[\mathcal{T}(f)]$, used for adaptive sampling, versus not, $\mathrm{Var}[f]$. Speed-up over uniform sampling, in equal error terms using RMSE on tone-mapped images, after 30 seconds.

Speed-up	Average	Worst	Best
$\mathrm{Var}[f]$ / $\mathrm{Var}[\mathcal{T}(f)]$	1.49 / 1.92	0.81 / 1.30	2.77 / 3.35

Post-Correction Denoising. We combined our proposed adaptive sampling technique to the recent post-correction denoiser from Gu et al. [2022] as described in Section 3.2.1. Our results shown in Figures 5 and 11 indicate improved error convergence when combining post-correction denoising with adaptive sampling.

As our method samples according to the variance of the denoised image, it ignores its bias. Post-correction methods will however trade the bias for variance when the squared error of the denoised image begins to exceed the variance of the unbiased image. This trade may increase the variance of the combined image, meaning our method will increase its sampling in such areas. Whether this possible feedback loop can be detrimental or not is an open question, however as our results show substantial improvement over uniform sampling we suggest it is likely an overall positive.

Comparison to Ground-Truth Sampling. We compare our methods to adaptive sampling in proportion to ground-truth relative variance and error, which are estimated after each iteration by denoising 32 independently rendered images using the current iteration's cumulative pixel-wise sample count. As computing these estimates is costly and impractical for actual usage, we report our results, Figures 8 and 9, in equal sample count terms. This experiment's results indicate that adaptive sampling in proportion to denoised variance leads to faster convergence for overall error versus sampling in proportion to denoised error, even when post-correction denoising is used. Furthermore, these results suggest that our method's performance is not significantly worse than ground-truth variance sampling, this despite the noise inherent in our variance estimate of Section 3.1.

Tone Mapping. Using the ACES tone mapping curve [Arrighetti 2017], which is popular in modern game engines, we investigated the impact of including tone mapping in the variance estimate using auto-differentiation. As the tone-mapped values are bounded, we opted to use root mean squared error (RMSE) rather than relative mean squared error (relMSE) for this experiment. The results, see Table 1, demonstrate a large benefit to incorporating tone mapping when performing adaptive sampling, and this benefit is exemplified in Figure 10.

When the desired final output are tone-mapped values, our results in Table 1 and Figure 10, show it is useful to incorporate it with adaptive sampling. Tone-mapping curves often exhibit a toe and shoulder, at their lower and upper bounds respectively, where the curve flattens significantly. This has the effect of compressing the range of radiance values in those areas, as well as their variance, which is easily incorporated into our method. The result is that additional samples are placed wherever else the visible benefit is larger. We likewise found that different choices of guiding metric (i.e. the relative variance used in Eq. 1), optimize for different error metrics. For example, if using (root) mean squared error as the error metric, variance as the guiding metric, perhaps unsurprisingly, performs significantly better than relative variance. Therefore, choosing a good metric is essential to achieving the best perceptual results, as was the conclusion of prior adaptive sampling works [Bolin and Meyer 1998; Hasselgren et al. 2020].

6 LIMITATIONS AND FUTURE WORK

One limitation of our method, perhaps inherent to all adaptive sampling techniques, is the undersampling of small details as illustrated in Figure 6. While this limitation did not particularly stand out in most scenes, it is nevertheless present in some. Our adaptive sampling guide ignores error due to bias, and although this can be addressed by post-correction denoising which trades bias for variance, it first requires sufficient samples in the region of interest, which may be missed due to allocating samples elsewhere.

In our experiments, we utilized an existing pre-trained denoiser as its standard architecture eased our implementation while its relative popularity serves to demonstrate the immediate applicability of our method. This network has possibly been trained solely on uniformly sampled images, and while we did not notice any obvious artifacts, we note that using a network trained on adaptively sampled images may improve quality. Applying our variance estimate to denoisers of a different network architecture remains future work, as does an experimental comparison to the recent work of Salehi et al. [2022].

An alternative to using the estimate by Ramani et al. [2008] to estimate the middle-term of SURE is to take the product $(\mathbf{J}_f(\mathbf{x})\mathbf{v})_i\mathbf{v}_i$, with the terms defined as in Section 3.1. While the benefits of this alternative estimate are yet to be investigated and although they may improve the SURE method, as the ground-truth experiments indicate that adaptive sampling using variance leads to faster overall error convergence, we think it is unlikely to affect our conclusions.

In production settings, quasi-Monte Carlo (QMC) integration is often preferred due to its faster convergence rate. It remains future work to test if our method could be applied to randomized-QMC, which allows for estimating variance by creating few randomized replications of a deterministic sample sequence [Owen 1998], assuming the denoiser used is robust to correlated sampling.

Temporal denoising is an exciting avenue to explore, for our adaptive sampling technique as well as for our variance estimate. While the denoiser we use does not address temporal denoising, we briefly investigated denoising sequences of independent images and noted a visible decrease in the temporal variance of the output, which was expected as placing more samples in areas of high output variance should subsequently decrease variance. It is possible to back-propagate gradients through the forward-propagated variance estimate, which would allow it to become part of the loss function used in training neural network denoisers. Penalizing output variance might lead to denoisers that are inherently more temporally stable, while the bias-variance trade-offs of doing so present an interesting question.

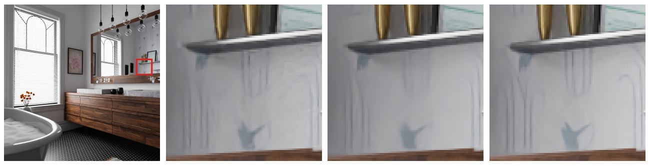


Figure 6: Despite the overall lower error, adaptive sampling can result in under sampling details compared to uniform, due to placing too many samples elsewhere as evidenced in this example after 120s. Given sufficient samples however, and due to the $(N_i + 1)$ term in the denominator of Eq. 3, our method eventually samples the undersampled region (right).

Outside of the rendering application presented here, we believe that the deep neural network variance estimate could have an impact in other fields. In particular, it could be applicable wherever neural networks are employed on data that could be interpreted as random variables, such as on noisy sensor measurements or estimates from other types of Monte Carlo simulations.

7 CONCLUSION

We have proposed an *a posteriori* adaptive sampling method for neural network denoisers, and shown better performance than uniform sampling and adaptive sampling based on error estimates in equal-time comparisons. In the process, we have presented a general method for estimating the variance of the outputs of a deep neural network whose inputs are random variables, and this estimate is computed using the Jacobian-vector product while simultaneously evaluating the network. When combined with a recent post-correction denoising algorithm, even better error convergence is achieved by our method, and we also demonstrate that incorporating tone mapping into our variance estimate yields an improvement. The generality of our proposed adaptive sampling algorithm eases its implementation in applications already using neural network denoisers. Our generic variance estimate for neural networks could be of general interest with potential applications beyond denoising.

ACKNOWLEDGMENTS

We thank the following artists for creating the scenes used in our work: Mareck (BATHROOM), nacimus (BATHROOM2), SlykDrako (BEDROOM), piopis (CAR), thecali (CAR2, SPACESHIP), NovaZeeke (CLASSROOM), cekuhnen (COFFEE), Wig42 (DINING-ROOM, LIVING-ROOM, LIVING-ROOM-3, STAIRCASE), aXel (GLASS-OF-WATER), MrChimp2313 (HOUSE), Jay-Artist (KITCHEN, LIVING-ROOM-2), UP3D (LAMP), vajrablue (ROVER), NewSee2l035 (STAIRCASE2), Benedikt Bitterli (TEAPOT-FULL). This research is a part of PRIME which is funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska Curie grant agreement No 956585.

REFERENCES

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 76 (2021), 243–297. https://doi.org/10.1016/j.inffus.2021.05.008

Attila T Áfra. 2019. Intel® Open Image Denoise. https://www.openimagedenoise.org/.

Walter Arrighetti. 2017. The academy color encoding system (ACES): A professional color-management framework for production, post-production and archival of still and motion pictures. *Journal of Imaging* 3, 4 (2017), 40. https://doi.org/10.3390/jimaging3040040

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep combiner for independent and correlated pixel estimates. *ACM Transactions on Graphics* 39, 6 (2020), 242:1–242:12. https://doi.org/10.1145/3414685.3417847

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2022. Self-supervised post-correction for Monte Carlo denoising. In *SIGGRAPH 2022 Conference Papers*. 18:1–18:8. https://doi.org/10.1145/3528233.3530730

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics* 36, 4 (2017), 97:1–97:14. https://doi.org/10.1145/3072959.3073708

Randall Balestriero and Richard Baraniuk. 2021. Fast Jacobian-Vector Product for Deep Networks. https://doi.org/10.48550/arXiv.2104.00219 arXiv:2104.0021 [cs.LG]

Benedikt Bitterli. 2016. Rendering Resources. https://benedikt-bitterli.me/resources/.

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly weighted first-order regression for denoising Monte Carlo renderings. In *Computer Graphics Forum*, Vol. 35. 107–117. https://doi.org/10.1111/cgf.12954

Mark R. Bolin and Gary W. Meyer. 1998. A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98*. 299–309. https://doi.org/10.1145/280814.280924

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The design and evolution of Disney's Hyperion renderer. *ACM Transactions on Graphics* 37, 3 (July 2018), 33:1–33:22. https://doi.org/10.1145/3182159

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics* 36, 4 (2017), 98:1–98:12. https://doi.org/10.1145/3072959.3073601

Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, Brenton Rayner, Jonathan Brouillat, and Max Liani. 2018. RenderMan: an advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics* 37, 3 (August 2018), 30:1–30:21. https://doi.org/10.1145/3182162

Bruce Christianson and Maurice Cox. 2006. Automatic propagation of uncertainties. In *Automatic Differentiation: Applications, Theory, and Implementations*. Springer, 47–58. https://doi.org/10.1007/3-540-28438-9_4

Mark A. Z. Dippé and Erling Henry Wold. 1985. Antialiasing through stochastic sampling. *Computer Graphics (SIGGRAPH '85)* 19, 3 (July 1985), 69–78. https://doi.org/10.1145/325334.325182

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: a batch-shading architecture for spectral path tracing in movie production. *ACM Transactions on Graphics* 37, 3 (August 2018), 31:1–31:18. https://doi.org/10.1145/3182161

Arthur Firmino, Jeppe Revall Frisvad, and Henrik Wann Jensen. 2022. Progressive denoising of Monte Carlo rendered images. *Computer Graphics Forum* 41, 2 (May 2022), 1–11. https://doi.org/10.1111/cgf.14454

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics* 38, 4 (2019), 125:1–125:12. https://doi.org/10.1145/3306346.3322954

Pascal Grittmann, Ömercan Yazici, Iliyan Georgiev, and Philipp Slusallek. 2022. Efficiency-aware multiple importance sampling for bidirectional rendering algorithms. *ACM Transactions on Graphics* 41, 4 (July 2022), 80:1–80:12. https://doi.org/10.1145/3528223.3530126

Roger Grosse. 2021. Taylor Approximations. In *Neural Network Training Dynamics*. Lecture Notes, University of Toronto, Chapter 2.

Jeongmin Gu, Jose A. Iglesias-Guitian, and Bochang Moon. 2022. Neural James-Stein combiner for unbiased and biased renderings. *ACM Transactions on Graphics* 41, 6 (December 2022), 262:1–262:14. https://doi.org/10.1145/3550454.3555496

Jon Hasselgren, Jacob Munkberg, Marco Salvi, Anjul Patney, and Aaron Lefohn. 2020. Neural temporal adaptive sampling and denoising. 39, 2 (May 2020), 147–155. https://doi.org/10.1111/cgf.13919

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022. *Mitsuba 3 Renderer*. https://mitsuba-renderer.org

James T. Kajiya. 1986. The rendering equation. *Computer Graphics (SIGGRAPH '86)* 20, 4 (August 1986), 143–150. https://doi.org/10.1145/15922.15902

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Transactions on Graphics* 34, 4 (2015), 122:1–122:12. https://doi.org/10.1145/2766977

Christopher Kulla, Alejandro Conty, Clifford Stein, and Larry Gritz. 2018. Sony Pictures Imageworks Arnold. *ACM Transactions on Graphics* 37, 3 (August 2018), 29:1–29:18. https://doi.org/10.1145/3180495

Alexandr Kuznetsov, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2018. Deep adaptive sampling for low sample count rendering. *Computer Graphics Forum* 37, 4 (July 2018), 35–44. https://doi.org/10.1111/cgf.13473

Mark E. Lee and Richard A. Redner. 1990. A note on the use of nonlinear filtering in computer graphics. *IEEE Computer Graphics and Applications* 10, 3 (1990), 23–29. https://doi.org/10.1109/38.55149

Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. 1985. Statistically optimized sampling for distributed ray tracing. *Computer Graphics (SIGGRAPH '85)* 19, 3 (July 1985), 61–68. https://doi.org/10.1145/325334.325179

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Transactions on Graphics* 31, 6 (November 2012), 194:1–194:9. https://doi.org/10.1145/2366145.2366213

Jun S Liu. 1994. Siegel's formula via Stein's identities. *Statistics & Probability Letters* 21, 3 (1994), 247–251. https://doi.org/10.1016/0167-7152(94)90121-X

John Mandel. 1964. *The Statistical Analalysis of Experimental Data*. Interscience Publishers / John Wiley & Sons.

Michael D. McCool. 1999. Anisotropic diffusion for Monte Carlo noise reduction. *ACM Transactions on Graphics* 18, 2 (1999), 171–194. https://doi.org/10.1145/318009.318015

Don P. Mitchell. 1987. Generating antialiased images at low sampling densities. *Computer Graphics (SIGGRAPH '87)* 21, 4 (July 1987), 65–72. https://doi.org/10.1145/37401.37410

Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. 2009. Adaptive wavelet rendering. *ACM Transactions on Graphics* 28, 5 (2009), 140:1–140:12. https://doi.org/10.1145/1618452.1618486

Art B. Owen. 1998. Monte Carlo extension of quasi-Monte Carlo. In *Winter Simulation Conference*, Vol. 1. IEEE, 571–577. https://doi.org/10.1109/WSC.1998.745036

James Painter and Kenneth Sloan. 1989. Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics (SIGGRAPH '89)* 23, 3 (July 1989), 281–288. https://doi.org/10.1145/74333.74362

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS 2019, Vol. 32)*.

Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. 2019. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2931–2940. https://doi.org/10.1109/ICCV.2019.00302

Sathish Ramani, Thierry Blu, and Michael Unser. 2008. Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms. *IEEE Transactions on Image Processing* 17, 9 (September 2008), 1540–1554. https://doi.org/10.1109/TIP.2008.2001404

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Philippe Weier, and Philipp Slusallek. 2022. EARS: efficiency-aware Russian roulette and splitting. *ACM Transactions on Graphics* 41, 4 (July 2022), 81:1–81:14. https://doi.org/10.1145/3528223.3530168

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive rendering with non-local means filtering. *ACM Transactions on Graphics* 31, 6 (November 2012), 195:1–195:11. https://doi.org/10.1145/2366145.2366214

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust denoising using feature and color information. *Computer Graphics Forum* 32, 7 (2013), 121–130. https://doi.org/10.1111/cgf.12219

Holly E. Rushmeier and Gregory J. Ward. 1994. Energy preserving non-linear filters. In *SIGGRAPH '94*. 131–138. https://doi.org/10.1145/192161.192189

Farnood Salehi, Marco Manzi, Gerhard Roethlin, Romann Weber, Christopher Schroers, and Marios Papas. 2022. Deep Adaptive Sampling and Reconstruction using Analytic Distributions. *ACM Transactions on Graphics* 41, 6 (2022), 259:1–259:16. https://doi.org/10.1145/3550454.3555515

Charles M. Stein. 1981. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics* 9, 6 (November 1981), 1135–1151.

Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G. Forbes. 2022. Temporally stable real-time joint neural denoising and supersampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 3 (2022), 21:1–21:22. https://doi.org/10.1145/3543870

Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics* 37, 4 (August 2018), 124:1–124:15. https://doi.org/10.1145/3197517.3201388

Kirk M. Wolter. 2007. *Introduction to Variance Estimation* (second ed.). Springer. https://doi.org/10.1007/978-0-387-35099-8

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural supersampling for real-time rendering. *ACM Transactions on Graphics* 39, 4 (2020), 142:1–142:12. https://doi.org/10.1145/3386569.3392376

Ruifeng Xu and Sumanta N. Pattanaik. 2005. A novel Monte Carlo noise reduction operator. *IEEE Computer Graphics and Applications* 25, 2 (2005), 31–35. https://doi.org/10.1109/MCG.2005.31

Shaokun Zheng, Fengshi Zheng, Kun Xu, and Ling-Qi Yan. 2021. Ensemble denoising for Monte Carlo renderings. *ACM Transactions on Graphics* 40, 6 (December 2021), 274:1–274:17. https://doi.org/10.1145/3478513.3480510

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer graphics forum* 34, 2 (May 2015), 667–681. https://doi.org/10.1111/cgf.12592

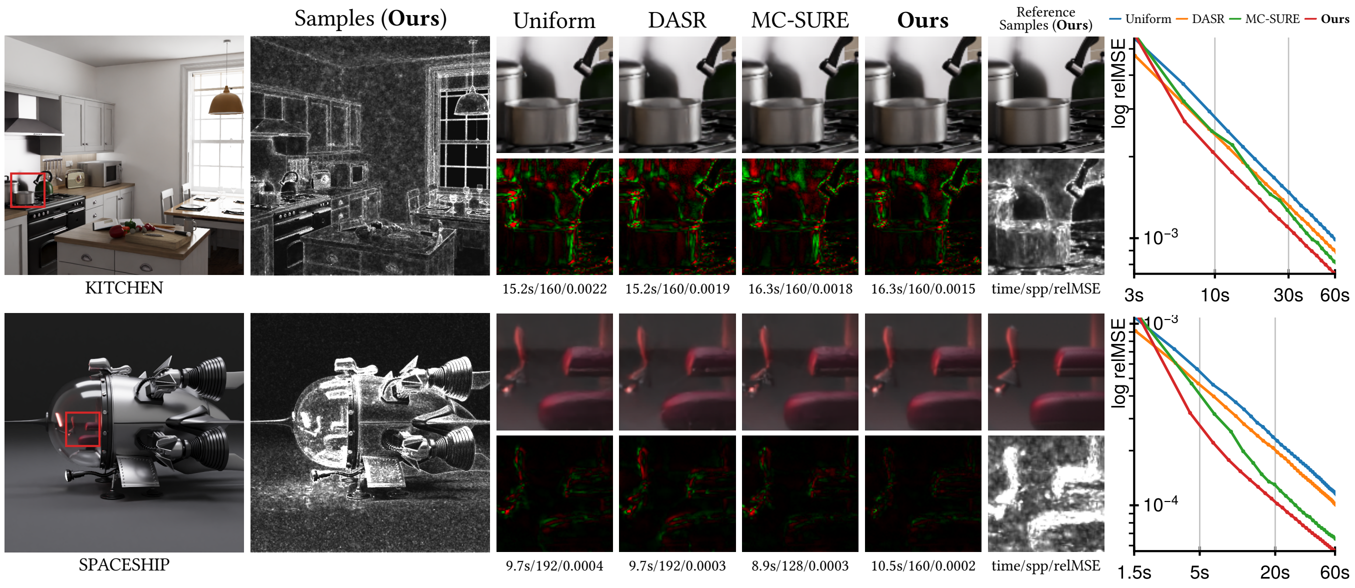


Figure 7: Equal time comparison of the different adaptive sampling methods on two different path-traced scenes (available from Bitterli [2016]), and illustration of the sample distribution from our main proposed method. Directly estimating the variance using the denoising network in question leads to sample distribution that performs better, and leads to less visible denoising artifacts, than the learning based (DASR) and error estimate (MC-SURE) approaches.

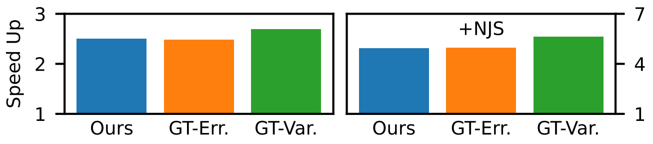


Figure 8: Ground-truth error and variance are costly to obtain. If these were available for adaptive sampling, we could obtain the speed-up shown here (in terms of average sample count) over uniform sampling in equal-error terms using relMSE, after 256 spp on average. Our method compared to adaptive sampling using ground-truth error and variance (left), using post-correction denoising (right). Average of 20 scenes.

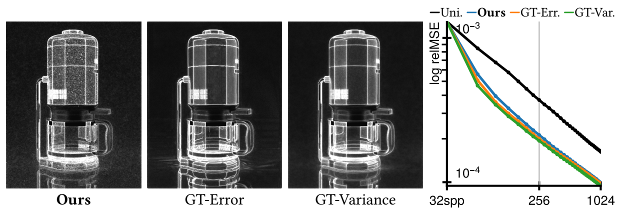


Figure 9: Comparing the cumulative sample distribution after 256 spp of our method with those from adaptively sampling using ground-truth estimates of denoised error and variance. Despite the noise in our variance estimate, we note agreement between our sample distribution and that produced by ground-truth variance. The plot on the right demonstrates that sampling proportional to variance leads to faster convergence versus error, and that our method approaches the ground-truth rates. Scene: Coffee by cekuhnen (CC-BY).

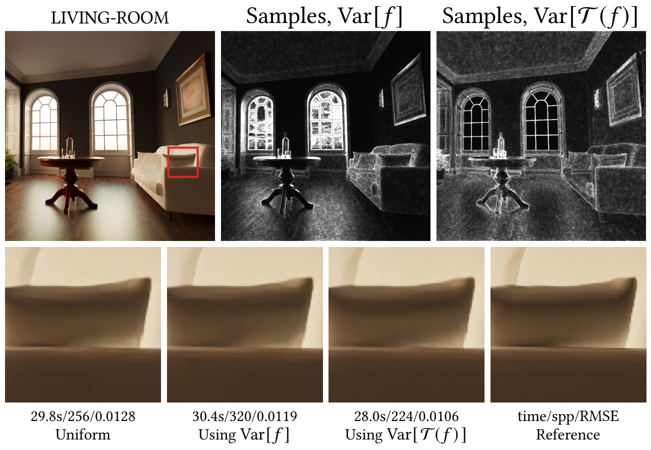


Figure 10: Example of the improved error and sample distribution achieved when incorporating tone mapping in the variance estimate (Section 3.2.2). Tone-mapping the image causes areas in the glossy floor to have their radiance values compressed resulting in the final image having relatively little variance there, as seen in the sample distribution. Scene: The Grey & White Room by Wig42 (CC-BY).

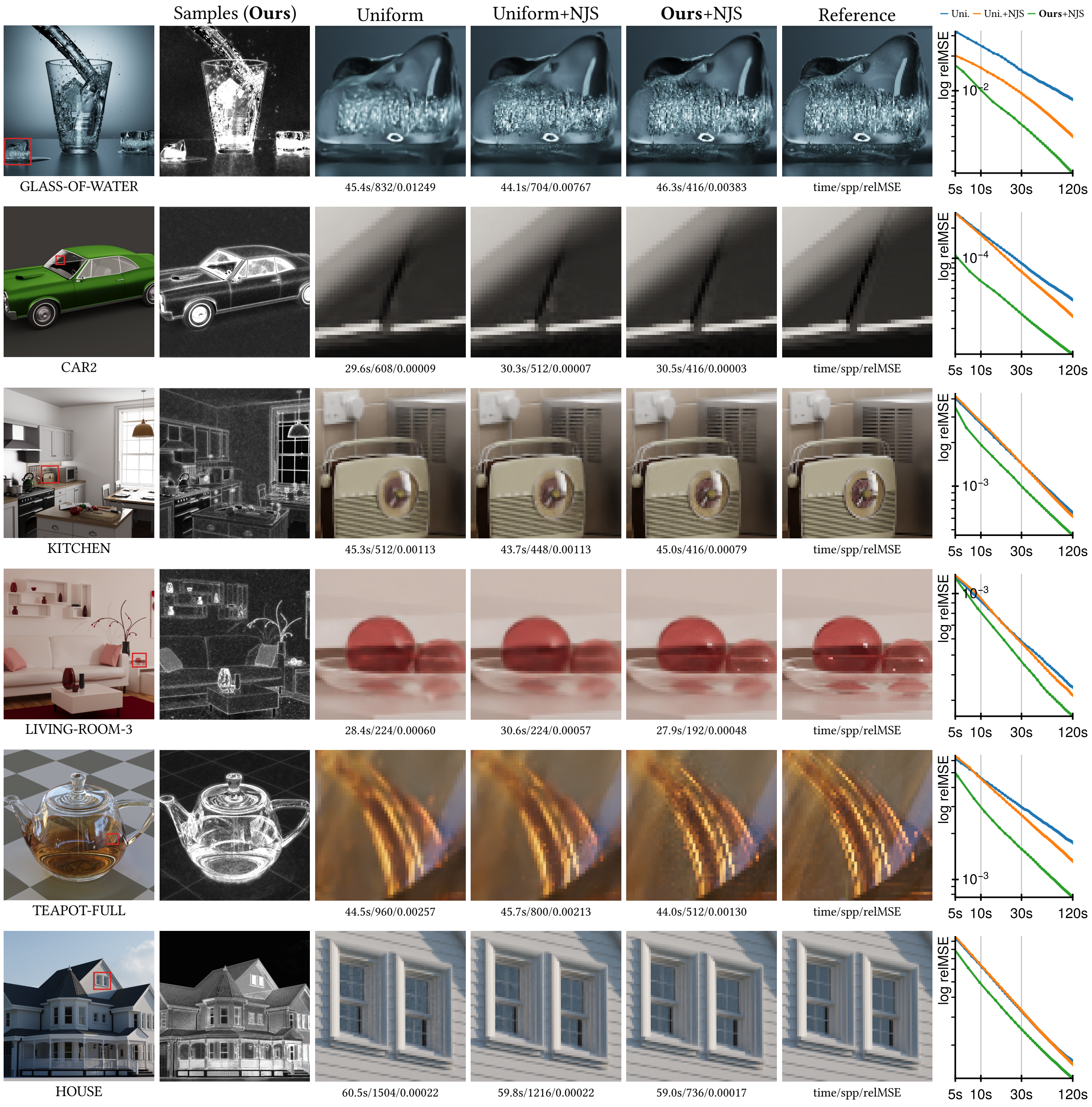


Figure 11: Results from incorporating the recent neural James-Stein (NJS) post-correction denoiser [Gu et al. 2022] with our adaptive sampling algorithm, Section 3.2.1. By estimating the variance of the biased inputs, adapting to the blend of the post-correction with the unbiased input, and distributing samples in proportion to variance, our method achieves substantially faster convergence in all scenes tested, see Figure 5.