FAST: Filter-Adapted Spatio-Temporal Sampling for Real-Time Rendering

WILLIAM DONNELLY, SEED - Electronic Arts, Canada
ALAN WOLFE, SEED - Electronic Arts, USA
JUDITH BÜTEPAGE, SEED - Electronic Arts, Sweden
JON VALDÉS, Frostbite - Electronic Arts, Sweden

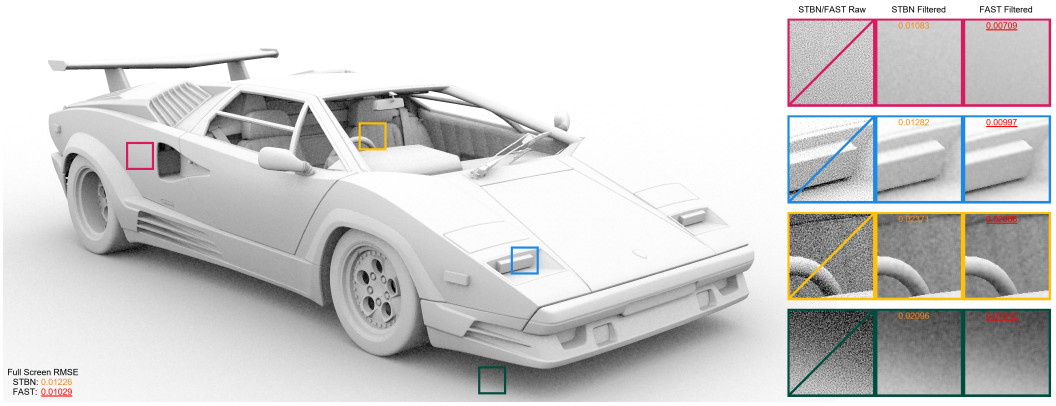


Fig. 1. Ray-traced ambient occlusion using screen space tiled noise textures for ray directions. One cosine-weighted hemispherical sample per frame was temporally filtered over twelve frames using an exponential moving average (EMA) with an alpha of 0.1. The final render undergoes a depth-aware 5x5 box blur. Spatiotemporal blue noise (STBN) [Wolfe et al. 2022] is compared to a member of our FAST noise family, which is optimized towards a 5x5 box blur over space and EMA over time. The inset boxes show the noise before spatial filtering (STBN upper left, FAST lower right), then the final render using STBN, and the final render using FAST. RMSE is shown in the final render boxes. Notice how the blue noise has noticeably blocky artifacts and a higher RMSE than the noise optimized for the spatiotemporal filtering — this is a consequence of a mismatch between the box filter used in denoising and the gaussian filter for which blue noise is optimal. Full-sized render uses FAST noise.

Stochastic sampling techniques are ubiquitous in real-time rendering, where performance constraints force the use of low sample counts, leading to noisy intermediate results. To remove this noise, the post-processing step of temporal and spatial denoising is an integral part of the real-time graphics pipeline. The main insight presented in this paper is that we can optimize the samples used in stochastic sampling such that the post-processing error is minimized. The core of our method is an analytical loss function which measures post-filtering error for a class of integrands — multidimensional Heaviside functions. These integrands are an approximation of the discontinuous functions commonly found in rendering. Our analysis applies to arbitrary spatial and spatiotemporal filters, scalar and vector sample values, and uniform and non-uniform

Authors' Contact Information: William Donnelly, SEED - Electronic Arts, Canada, wdonnelly@ea.com; Alan Wolfe, SEED - Electronic Arts, USA, awolfe@ea.com; Judith Bütepage, SEED - Electronic Arts, Sweden; Jon Valdés, Frostbite - Electronic Arts, Sweden.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, https://doi.org/10.1145/3651283.

probability distributions. We show that the spectrum of Monte Carlo noise resulting from our sampling method is adapted to the shape of the filter, resulting in less noisy final images. We demonstrate improvements over state-of-the-art sampling methods in three representative rendering tasks: ambient occlusion, volumetric ray-marching, and color image dithering. Common use noise textures, and noise generation code is available at https://github.com/electronicarts/fastnoise.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: rendering, noise, sampling

ACM Reference Format:
William Donnelly, Alan Wolfe, Judith Bütepage, and Jon Valdés. 2024. FAST: Filter-Adapted Spatio-Temporal Sampling for Real-Time Rendering. *Proc. ACM Comput. Graph. Interact. Tech.* 7, 1 (May 2024), 16 pages. https://doi.org/10.1145/3651283

1 INTRODUCTION

Modern rendering algorithms frequently rely on Monte Carlo methods to evaluate complicated high-dimensional numerical integrals. In offline rendering one typically averages over n samples per pixel, with the goal of converging toward the ground truth solution when n is large. Quasi-Monte Carlo methods such as low-discrepancy sequences can give an asymptotic improvement in convergence, reducing the root mean square error in the image from $O(1/\sqrt{n})$ to almost $O(1/n)$ as $n \to \infty$ [Niederreiter 1992].

In real-time rendering, the number of samples is limited by performance considerations — often to a single sample per pixel or less. With so few samples, the asymptotic improvement of Quasi-Monte Carlo methods is not enough and some form of denoising is necessary. A typical denoising setup is a combination of edge-aware spatial filtering, and exponential moving average with temporal reprojection [Schied et al. 2017]. However, more complicated and sometimes proprietary denoising can be found in real-time rendering [Spitzer 2021]. Denoisers importantly allow the cost of sampling to be amortized across multiple pixels in the image, and across multiple frames. Although these filters may have an infinite response in the time dimension, the number of effective samples typically remains bounded, so improvements in image quality have to come from an improved distribution of samples.

Just as low-discrepancy sequences can improve convergence by more fairly distributing samples as a function of sequence index, image quality under denoising can be improved by distributing samples more fairly within small regions of the image plane. Samples with this property are generally referred to as *blue noise* [Mitchell 1987; Ulichney 1987, 1988], and are characterized by a Fourier spectrum whose support is concentrated in high ("blue") frequencies. By reducing the power in low frequencies, blue noise sampling improves both the perceptual quality of images [Mitchell 1987] and performance under low-pass denoising filters [Panteleev and Schied 2019].

The benefits of blue noise sampling can also be realized in the time domain. Samples with blue noise properties over both space and time improve convergence and temporal stability when used with a temporal filter, while maintaining the desirable properties of blue noise in image space [Wolfe et al. 2022]. This raises a question: what are the optimal properties of samples for use in conjunction with temporal and spatial filtering?

To answer this question we construct a loss function that measures the quality of samples. We incorporate a number of insights about the sampling problem. Specifically:

- **General sample spaces** - Our algorithm generates D-dimensional samples [Georgiev and Fajardo 2016] or samples on a sphere with uniform or cosine-weighted distribution [Wolfe et al. 2022]. While it is possible to warp scalar samples from $[0, 1]$ to a target distribution,

such transformations degrade the blue noise properties of the samples [Keller et al. 2019]. Our algorithm can optimize samples drawn from any specified distribution.

- **Spatial filtering** - Blue noise is optimized for a Gaussian filter in space, but other filter shapes and sizes are possible [Chizhov et al. 2022; Salaün et al. 2022]. Our loss function optimizes for minimal mean-squared error *after* filtering.
- **Temporal filtering** - When temporal filtering is employed, optimizing for temporal properties of the noise can accelerate convergence [Wolfe et al. 2022]. Temporal filters are often quite different from spatial filters, being typically one-sided and unbounded in time. We show how to optimize samples in time for a prototypical temporal filter based on a randomly truncated exponential moving average [Korać et al. 2023].
- **Discontinuous integrands** - Rendering integrands are typically discontinuous, primarily due to jumps in visibility. These discontinuities are a chief source of noise in rendering. We model such discontinuities by treating the rendering integrand as a random Heaviside function [Heitz and Belcour 2019; Ramamoorthi et al. 2012].

Our main contribution is incorporating all of these features into a single loss function that takes an extremely simple form suitable for numerical optimization. The main source of this simplification is an analytic evaluation of the two-point correlation for random Heaviside functions, which we present in the supplemental work.

We find significant benefits from adapting the noise to match the denoising filter. This difference is most significant under spatiotemporal filtering, where we typically use a mix of finite impulse response filter in the spatial domain with infinite impulse response filter in the temporal domain. This leads to a filter that is highly anisotropic between space and time dimensions and which is not well approximated by an isotropic Gaussian. When samples are optimized to make the noise complementary to such a filter, we obtain results that are distinct from both 3D blue noise and spatiotemporal blue noise — though our algorithm can recover both as special cases.

2 RELATED WORK

The problem of optimizing sampling patterns for optimal spectral properties goes back at least to early work on dithering [Bayer 1973]. Foundational work in this area introduced blue noise criterion for samples [Mitchell 1987; Ulichney 1987, 1988] and efficient algorithms to generate blue noise dither masks [Ulichney 1993].

The relevance of blue noise sampling to real-time computer graphics was demonstrated by Gjoel and Svendsen [2016] who used concepts from signal processing [Christou 2008] to improve the perceptual quality of rendering noise. Blue-Noise Dithered Sampling [Georgiev and Fajardo 2016] extended this concept to vector-valued samples, extending the range of sampling problems that could benefit from blue noise sampling. Rather than generating single samples, Heitz and Belcour [2019] generate seeds for entire low-discrepancy sequences with blue noise properties. The quality of their sequences is evaluated using a large number of randomly-generated Heaviside functions, which were shown to be a good measure even when used for more continuous integrands [Belcour and Heitz 2021]. To overcome these methods only working well for low dimensions, Heitz et al. [Heitz and Belcour 2019] rearrange per-pixel seeds such that the rendering results have blue noise properties; however this can have the side effect of correlating the noise to the signal so that high frequency features of the image can also be converted to blue noise. [Ahmed and Wonka 2020] used space-filling curves to map desirable properties of low-discrepancy sequences into good spectral properties of two-dimension grids of samples, though some of the grid-like structure of the space-filling curve is imprinted into the noise. [Wronski 2020] directly optimized samples using gradient descent, using a loss function based on the Fourier transform of the samples. Wolfe et al.

[2022] introduced Spatiotemporal Blue Noise (STBN) which improves sampling convergence under temporal anti-aliasing and otherwise, while maintaining blue noise properties over space.

While the choice of samples influences the perceptual quality of rendering noise, it also affects the performance of denoising. Temporal antialiasing is an integral part of many real-time rendering pipelines, often used in conjunction with spatial denoising as in SVGF [Schied et al. 2017]; see Yang et. al. [2020] for a review of these methods. Interleaved gradient noise [Jimenez 2014] was designed to take advantage of the non linear neighborhood clamping of temporal anti-aliasing history rejection, by ensuring that every 3×3 block of pixels has a flat histogram, demonstrating the benefit of adapting the sampling patterns to the specific algorithm being used.

The loss function for samples which we will derive in section 3 bears some similarity to loss functions which have appeared in the literature before. Georgiev and Fajardo [2016] use a heuristic loss function that is a product of a Gaussian filter in image space with a gaussian-like kernel $e^{||x-y||^{d/2}/\sigma^2}$ in sample space. Much of the work in this area uses a Gaussian kernel in both image space [Belcour and Heitz 2021] and sample space [Ahmed et al. 2022]. Chizhov et. al. [2022] incorporated more general filters into the optimization problem by minimizing post-filtering L^2 error. Salaün et. al. [Salaün et al. 2022] show how to optimize both blue noise point sets and textures in a single framework, formulating the problem in terms of optimal transport. Their objective is based on a sliced approximation of the Wasserstein distance, under the assumption of Lipschitz-continuous rendering integrands. Korać et. al. [Korać et al. 2023] generalize this work to spatiotemporal sampling by including exponential filtering over the time domain.

3 A LOSS FUNCTION FOR SAMPLING

We consider a simplified model of the rendering pipeline, in which samples are drawn from a fixed array, used for numerical integration, and the result is denoised with a simple linear filter. The goal of this model is to be simple enough to be analytically solvable, while capturing the properties of samples which make them useful for real rendering tasks.

We will formulate rendering as a numerical integral of a function ϕ over a sample space S with measure $d\mu(x)$:

$$\bar{\phi} = \int_S d\mu(s)\phi(s). \tag{1}$$

To approximate this integral we assume that we have a given set of samples $s_i \in S$ distributed across image pixels $i \in I$. These allow us to generate a noisy image ϕ_i and filtered version Φ_i:

$$\phi_i = \{\phi(s_i) | i \in I\}, \qquad \Phi_i = \sum_{j \in I} f_{ij}\phi_j, \tag{2}$$

where f_{ij} is the filter. The image space I is either a 2D index if filtering a single frame, or a 3D index if temporal filtering over multiple frames.

Our goal is to minimize the expected squared error of the filtered image to the ground truth:

$$L = \left\langle \sum_i (\Phi_i - \bar{\phi})^2 \right\rangle. \tag{3}$$

Here angular brackets denote an average over different test integrands ϕ. This is a natural loss function for this problem, appearing widely in literature on halftoning and in the context of Monte Carlo rendering — see [Chizhov et al. 2022] and reference therein.

Before proceeding we should note two key assumptions that go into this choice of loss function. First, the integrand ϕ is assumed to be the same across all samples. The assumption that nearby points in image space are correlated is what allows filtering methods to work, but this correlation is not perfect as the model assumes. Second, we assume linear filtering. Real filtering algorithms

typically perform edge-aware blurring, and temporal filters have mechanisms to reject history when changes in the scene are detected. Both of these effects mean that in practice the effective filter size is smaller than in the idealization considered by the loss function: we compensate them by optimizing for a filter smaller than the one to be used for denoising.

The loss function can be rewritten as:

$$L = \sum_{j,k} F_{jk} K_2(s_j, s_k) + 2 \sum_i K_1(s_i) + \sum_i K_0. \tag{4}$$

where $F_{jk} := \sum_i f_{ij} f_{ik}$ is the overlap between filter kernels, and the functions $K_{0,1,2}$ depend on the two-point correlation function of ϕ:

$$K_2(x, y) := \langle \phi(x)\phi(y) \rangle, \qquad K_1(x) := -\int d\mu(y) K_2(x, y), \qquad K_0 := -\int d\mu(x) K_1(x). \tag{5}$$

The terms proportional to K_1 and K_0 are invariant under permutation of the sample points s_i.

For the two-point function $K_2(x, y)$ we assume a model based on randomly-oriented Heaviside functions [Heitz et al. 2019; Ramamoorthi et al. 2012]. This model has been widely adopted as it models one of the main sources of noise in rendering, the visibility function. As we prove in the supplemental work, this model can be analytically evaluated in the two cases of most interest: for vectors in D-dimensional Euclidean space $\mathbb{R}^D$ or on the surface of a sphere S^D. In both of those cases the correlation function takes the following form with positive constants α, β:

$$K_2(x, y) = \alpha - \beta d(x, y) \tag{6}$$

where $d(x, y)$ is the Euclidean distance in $\mathbb{R}^D$ or the geodesic distance in S^D.

Two further simplifications are possible. First, we note that the term proportional to α drops out of the loss function, while β simply results in an overall scaling. So we can simply set $\alpha = 0$ and $\beta = 1$ and solve an equivalent minimization problem. The term involving K_0 is simply a constant and does not affect the minimization problem.

Finally, the term in L involving K_1 only depends on the overall distribution of sample points s_i. Typically we want the distribution of the samples s_i to be held fixed proportional to the measure $d\mu$ in order to ensure unbiased sampling. In section 4 we will desribe an optimization algorithm based on permuting the samples s_i; since the K_1 term of the loss function L is invariant under permutation of samples, it does not affect the result of the optimization.

Combining all of these assumptions, we arrive at the following simple loss function:

$$\tilde{L} = -\sum_{j,k} F_{jk} d(s_j, s_k). \tag{7}$$

Minimizing $\tilde{L}$ is equivalent to minimizing the post-filtering mean squared error for samples in any dimension, under the assumption of linear filtering and random Heaviside functions.

4 ALGORITHM

We use an algorithm based on simulated annealing to generate an array of samples s_i minimizing the loss function L (7). The algorithm consists of an initialization step and N iteration steps.

Initialization consists of filling an array with stratified samples of the sampling space. Stratification ensures a high-quality histogram while preventing structured patterns that could appear if using regular sampling or low-discrepancy sequences. For spatiotemporal samples, each two-dimensional XY slice of I is stratified independently to ensure each frame has a high quality histogram.

Each optimization step chooses two points i, j in the sample array and calculates how the loss function (4) would change when swapping sample i with sample j:

$$\Delta\tilde{L}_{ij} = 2\sum_k (F_{ik} - F_{jk})[d(s_j, s_k) - d(s_i, s_k)] + 2(F_{ij} + F_{ji} - F_{ii} - F_{jj})d(s_i, s_j), \tag{8}$$

which can be calculated with two loops over the footprint of the filter, centered at i and j respectively.

If we find that the loss function would decrease ($\Delta L < 0$), then we simply interchange sample i with sample j. More sophisticated annealing methods are possible, where the probability of swapping depends on the value of ΔL. We experimented with different annealing schedules and did not find any significant improvement over the naive algorithm of always swapping if the loss function would decrease. Swapping candidates are generated from a random pairing of points in the array; for spatiotemporal samples we only consider pairing candidates from the same XY slice in order to maintain the histogram of each slice.

4.1 Implementation

Our implementation runs in parallel on a GPU using compute shaders. For simplicity and efficiency we require spatial dimensions to be powers of two. To choose swapping candidates in parallel, we generate a random involution of the form:

$$\rho(i) = \sigma(\tau(\sigma^{-1}(i))). \tag{9}$$

where σ is a random permutation generated with a three-round Feistel network, τ is XOR with a pseudorandom bit string. This function is an involution — $\rho(\rho(i)) = i$ — so by pairing each pixel i with $\rho(i)$ we generate a perfect pairing of the input data. Pseudorandom inputs for τ and σ are generated with a Wang hash [Reed 2013] seeded with the index of the temporal slice and iteration number. Having defined a pairing between samples, each sample i finds its partner $j = \rho(i)$ and computes half of the sum in Eq. (8) in parallel. In a subsequent pass each sample i then adds its result to that of its partner to find $\Delta\tilde{L}_{ij}$ to determine whether a swap occurs.

The advantage of this method is that multiple swaps can be carried out in parallel, but this can result in the loss function increasing. This effect was observed by Belcour and Heitz [2021] who interpreted it as an "effective annealing". In early iterations this effect can be very strong and prevent the algorithm from converging. To avoid this, we only perform a fraction γ of swaps in each iteration. We initially set $\gamma = 1/8$, and double γ whenever the fraction of swaps drops below $\gamma/4$ until $\gamma = 1$. This prevents the algorithm from getting stuck in the initial few iterations, while taking full advantage of parallelism in later iterations.

To give a sense of execution times, a 128×128×32 texture that has a cosine weighted hemispherical unit vector per pixel, and is optimized over space for a 5×5 box filter, and over time for EMA, as a product filter, is generated over 10,000 iterations in 3 minutes on an NVIDIA RTX 3090 GPU.

5 ANALYSIS

In this section we carry out a spectral analysis of the results from our optimizer. We begin by introducing a quantity $\tilde{K}_m$ that measures the Fourier spectrum of the error from sampling a random function, and show how it relates to our loss function. We then plot this quantity across a range of different results from our optimizer, including for different sample spaces and filters, showing how the spectra depend on the choice of spatial and temporal filters.

Quality of sample textures can be assessed using the discrete Fourier transform [Georgiev and Fajardo 2016], where blue noise textures show a characteristic suppression of the spectrum at low frequencies. While the Fourier transform measures spectral properties of the sample points, this provides only an indirect measure of the spectral properties of the rendering results. To better

measure the impact of the sample points on noise of the final image we instead consider the spectrum of the rendering error, averaged over different test integrands. This measure extends to general sample spaces, and we will see that it is closely related to the loss function L.

For a given rendering integrand ϕ, we define $\tilde{\phi}_m$ to be the Fourier transform of the difference between the sampled result and the expected signal:

$$\tilde{\phi}_m = \sum_j e^{-2\pi i m \cdot j} (\phi(s_j) - \bar{\phi}). \tag{10}$$

Averaging the square of the Fourier transform over random rendering integrands, we obtain:

$$\tilde{K}_m := \langle |\tilde{\phi}_m|^2 \rangle = \sum_{j,k} e^{-2\pi i m \cdot (j-k)} \left[K_2(s_j, s_k) + K_1(s_j) + K_1(s_k) + K_0 \right]. \tag{11}$$

This gives a natural measure of the spectral distribution of the rendering error, and defines a positive scalar function for general sample spaces.

To see how $\tilde{K}_m$ relates to our loss function, we assume that the filter f is translation-invariant so that f_{ij} is a function of $(i - j)$. This allows us to write f_{ij} in terms of its Fourier transform $\tilde{f}_m$:

$$f_{jk} = \frac{1}{|I|} \sum_m e^{2\pi i m \cdot (j-k)} \tilde{f}_m, \tag{12}$$

where $|I|$ denotes the number of pixels in the image. In frequency space, the loss function (4) can be expanded in terms of $\tilde{K}_m$ and $\tilde{f}_m$ as:

$$L = \frac{1}{|I|} \sum_m |\tilde{f}_m|^2 \tilde{K}_m. \tag{13}$$

Therefore by minimizing L, the optimizer is minimizing $\tilde{K}_m$ wherever the spectrum of the filter $|\tilde{f}_m|$ is largest. In the following subsections we show how the optimizer is able to shape the noise spectrum to avoid the filter.

5.1 Spatial noise spectrum

In Fig. 2 we plot some representative sample images and their corresponding noise spectra $\tilde{K}_m$ (11) alongside the filters they were optimized for. We show three representative spatial filters: a box filter, a truncated gaussian, and a binomial filter each applied separably in x and y. These filters are applied to optimizing samples in three sample spaces: the interval $[0, 1]$ with uniform measure, the unit sphere with cosine-weighted measure, and the unit sphere with uniform measure. The figure shows that optimizer is able to successfully shape the noise spectrum to avoid the specified filter across different sample spaces.

5.2 Spatiotemporal noise spectrum

In real-time applications it is common to use temporal filtering as a form of denoising. In its simplest form, one chooses a parameter $\alpha \in (0, 1]$ and interpolates between the current frame with weight α and the previous frame with weight $1 - \alpha$. This generates an exponential moving average filter with $I = \mathbb{Z}$ indexing the frame:

$$f_{ij} = \begin{cases} \alpha(1-\alpha)^{i-j} & i \geq j \\ 0 & i < j \end{cases} \tag{14}$$

The overlap between filters can be calculated by summing a geometric series, and is given by

$$F_{ij} = \frac{\alpha(1-\alpha)^{|i-j|}}{2-\alpha}. \tag{15}$$

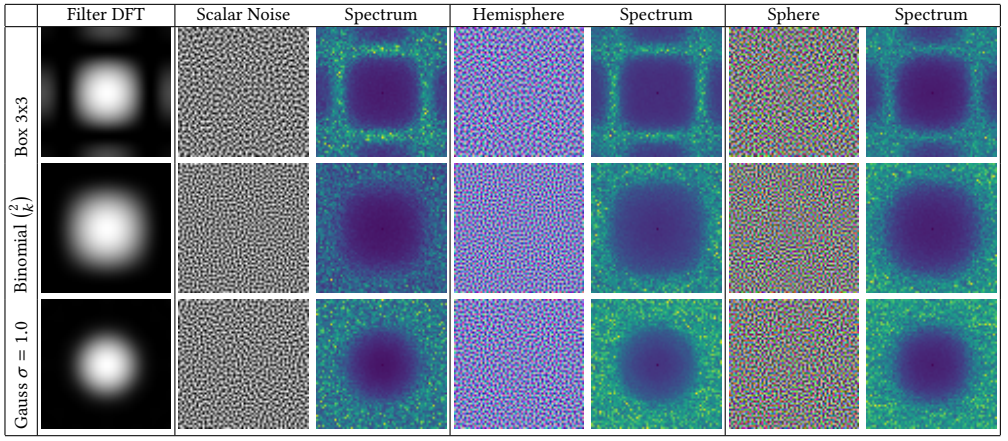


Fig. 2. Sample textures optimized for various spatial filters and sample spaces. The DFT of the filter optimized against is provided in the first column.

Note that when using the filter (15), we are minimizing the long-time limit of the rendering error, after the temporal filter has converged. In practical real-time applications the filter does not always have time to converge. When there is motion of scene or of the camera, the pixels of the current frame are reprojected onto the previous frame, so that a sample at pixel (i, j) will be averaged with a result coming from a different pixel (i', j'). Moreover, rendering results typically undergo nonlinear *neighbourhood clamping* to mitigate rendering artifacts where temporal reprojection causes multiple pixels to reference the same pixel in the previous frame, known as "ghosting" [Karis 2014]. In these cases a pixel may reject its history completely, effectively disabling temporal filtering for a subset of the frame.

To simulate these effects, we can adjust the temporal filter as follows. If we assume that the exponential moving average runs for only m frames, the result is a truncated filter

$$f_{ij}^{(m)} = \begin{cases} \alpha(1-\alpha)^{i-j} & i = j, \dots, j+m-2 \\ (1-\alpha)^{m-1} & i = j+m-1 \\ 0 & i < j \text{ or } i \geq j+m \end{cases} \tag{16}$$

Here the filter is unmodified for the first $m-1$ samples, while the m^{th} sample (with index $i = j+m-1$) is equal to the sum of f_{ij} for $j \geq i+m-1$.

At sample generation time we cannot predict when history will be rejected, but we can adopt a simple statistical model. Let $\beta \in (0, 1)$ be the probability that history will be rejected in any given pixel. We can then take a weighted average of the doubled filters over each cutoff value m:

$$F_{ij} = \beta \sum_{m=0}^{\infty} (1-\beta)^m \sum_k f_{ik}^{(m)} f_{jk}^{(m)}. \tag{17}$$

While the temporal filter has in principle an infinite extent, in practice we store only a finite number n of temporal samples and truncate the sum (17) to $m < n$.

By increasing the parameter β we are reducing the extent of the temporal filter, giving more importance to regions in the image which have less history. This may be desirable, as pixels which reject history average over fewer samples and tend to have the most noise. We therefore leave β as

an adjustable parameter, allowing the user to trade off between rapid convergence of the filter and quality of the converged results. In the examples that follow we use $\alpha = 0.1$, $\beta = 0.1$.

We consider two methods of combining the spatial and temporal filter. If we assumed that the result of sampling will undergo both temporal and spatial filtering, then the optimal way to combine spatial and temporal filters is via a product. An alternative, as advocated by Wolfe et. al. [2022] is to use a weighted sum of the spatial and temporal filter. Such a filter tries to simultaneously minimize the amount of noise in the final frame and its spectral properties in image space. The two different modes — which we call *product* and *separate* — have distinct spectral properties and therefore different use cases.

To see the effect of different choices of temporal filtering on the noise spectrum, we compare four combinations of temporal filter settings. Specifically we compare a Gaussian filter over time with a weighted exponential average (17) and combining temporal filters with a product versus separate. In all examples, we use uniform scalar noise optimized for a Gaussian filter over space, though similar results apply to vector-valued samples.

To show how these affect properties of the noise, in Fig. 3 we plot four different quantities. Let (x, y) denote coordinates on the image plane, and t denote time. The noise spectrum is a function of the frequencies $(\omega_x, \omega_y, \omega_t)$. In the top row, we plot the $\omega_y = 0$ slice through the three-dimensional noise spectrum as a function of ω_x and ω_t — this shows the extent to which the noise is anticorrelated in both space and time dimensions. Next, we plot the slice at $\omega_t = 0$ as a function of ω_x and ω_y. This shows the expected spectral distribution of noise after averaging over all 64 frames of the samples, and so captures the spectral properties of the long-time average. Finally, we plot the noise spectrum of a single time slice of the samples, along with the samples themselves, to see how expected spectral properties of individual frames of noise.

When using a product with a Gaussian filter over time, we are effectively creating three-dimensional blue noise. The (ω_x, ω_t) slice through the spectrum resembles that of two-dimensional blue noise.[1] In fact, the noise is approximately isotropic and so the (ω_x, ω_y) slice has the same shape as the (ω_x, ω_t) slice. As pointed out by Peters [2017], individual slices of three-dimensional blue noise do not have good spectral properties as is evident from the spectrum of a single slice, which differs only slightly from white noise.

By combining Gaussian over time with a sum rather than a product, we create samples with the spectral properties of spatiotemporal blue noise [Wolfe et al. 2022]. Optimization with the temporal filter ensures that after averaging all frames, the noise is uniformly suppressed, as seen from the $\omega_t = 0$ slice of the noise spectrum. However, this comes at the expense of the single frame spectrum, which exhibits some suppression at low frequencies but not as strongly as two-dimensional scalar samples.

Using the exponential moving average over time, we suppress the spectrum in a cylindrical region of $(\omega_x, \omega_y, \omega_t)$ space, leading to strong blue noise properties of both the average and a single frame. However unlike the separate filter, it does not uniformly suppress rendering noise in the long time limit, rather it simply shapes the noise to have a stronger spectral falloff.

Using an exponential moving average over time combined separately gives a compromise between advantages of the previous two methods. The $\omega_t = 0$ slice of the spectrum is uniformly suppressed, even more so than with a Gaussian filter over time, leading to a less noisy long-time average. The spectral properties of individual frames are also slightly better than are obtained with a Gaussian

[1]The dark vertical line through the middle of the (ω_x, ω_t) slice corresponds to zero spatial frequency - here the noise spectrum is suppressed because each slice is constrained to have a fixed histogram. This is a property of the optimization algorithm and is the same for all noise types.

filter over time, though not as good as when combining the exponential moving average via a product.

In sections 6.2 and 6.3 we will see how these results translate into properties of rendering results, and the relative advantages of the different methods of combining filters.

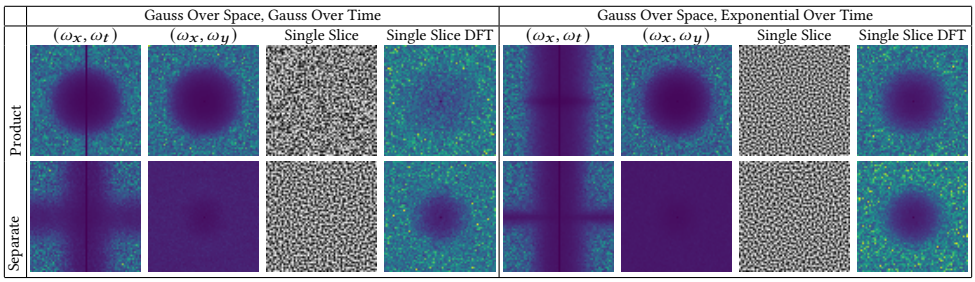
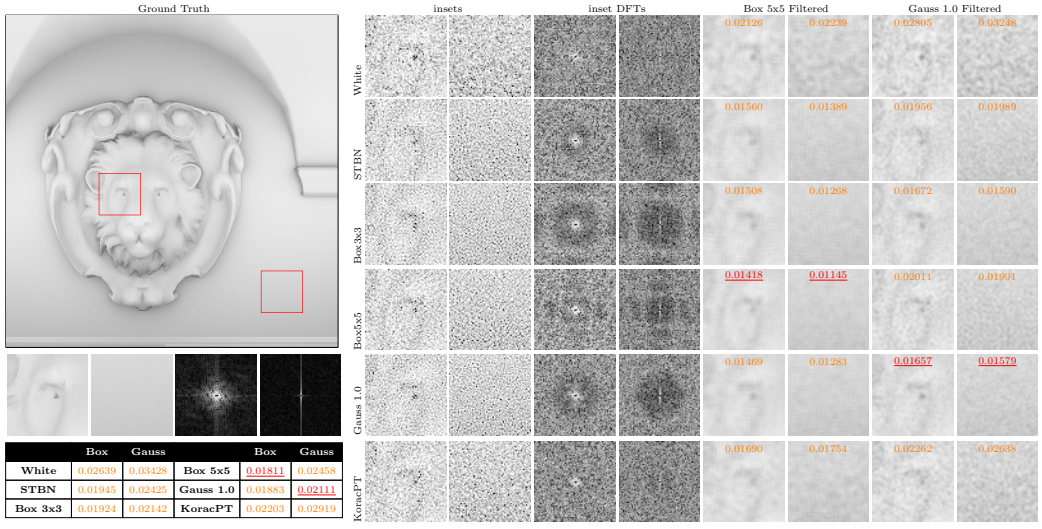
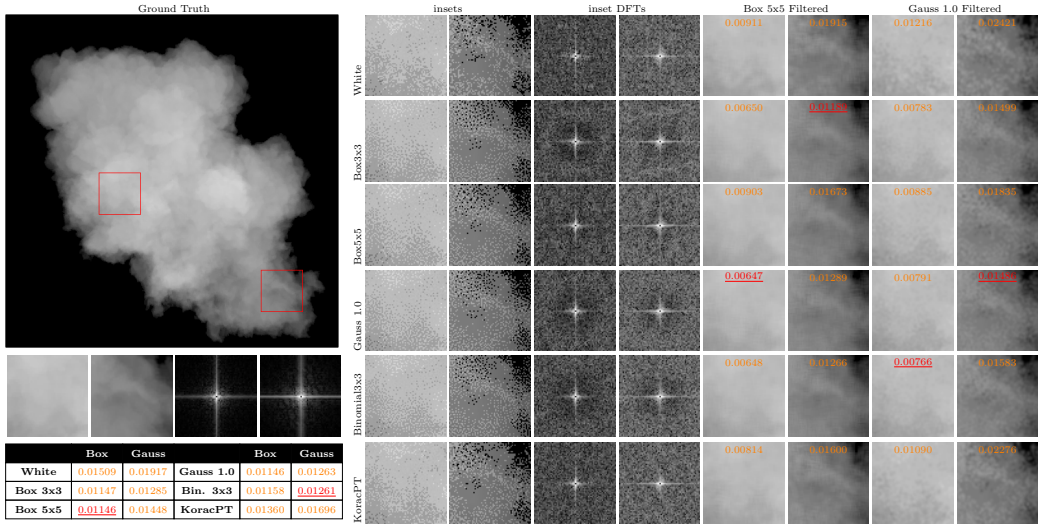


Fig. 3. Spatiotemporal sample textures optimized for different temporal filters and with different modes for combining filters. These choices lead to distinct properties of the noise spectrum which have uses in different rendering scenarios. (ω_x, ω_t) shows the average frequencies over space and time by showing the $(\omega_x, \omega_y = 0, \omega_t)$ slice of the volume DFT. (ω_x, ω_y) shows the average spatial frequencies over time by showing the $(\omega_x, \omega_y, \omega_t = 0)$ slice of the volume DFT. A single slice of each volume texture and resulting DFT are shown as well.



	Box	Gauss		Box	Gauss
White	0.02639	0.03428	Box 5x5	0.01811	0.02458
STBN	0.01945	0.02425	Gauss 1.0	0.01883	0.02111
Box 3x3	0.01924	0.02142	KoracPT	0.02203	0.02919

Fig. 4. Ray-traced ambient occlusion with four samples per pixel using various noise sources, filtered with an edge-aware 5×5 box filter, and a $\sigma = 1.0$ Gaussian filter. Numbers in filtered renders are RMSE, red is the lowest. KoracPT uses the perceptual + TAA optimized noise provided in the supplemental material of [Korać et al. 2023], and gave the best results of the noise types provided. Full screen RMSE in lower left table.



	Box	Gauss		Box	Gauss
White	0.01509	0.01917	Gauss 1.0	0.01146	0.01263
Box 3x3	0.01147	0.01285	Bin. 3x3	0.01158	0.01261
Box 5x5	0.01146	0.01448	KoracPT	0.01360	0.01696

Fig. 5. Ray marching heterogeneous participating media using 4 steps per pixel using various noise sources, filtered with a 5×5 box filter, and a $\sigma = 1.0$ Gaussian filter. Numbers in filtered renders are RMSE, red is the lowest. KoracPT uses the perceptual + TAA optimized noise provided in the supplemental material of [Korać et al. 2023], and gave the best results of the noise types provided. Full screen RMSE in lower left table.

6 RESULTS

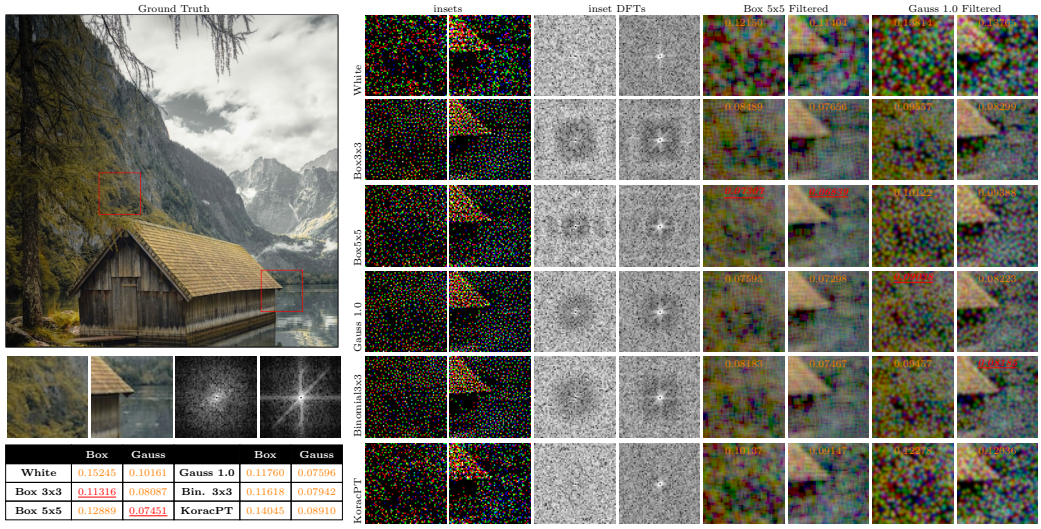
Our work supplies per-pixel random values, so has a wide range of use in real time rendering algorithms. To demonstrate its use we perform tests in three common real-time rendering tasks: ray-traced ambient occlusion, ray-marching participating media, and dithering a color image before quantization.

An essential part of our rendering tests is that the noise textures contain the specific type of random value required by the algorithm. For instance, the ambient occlusion technique could construct a point in disk from two random scalar values and then add a z component to make it a cosine-weighted hemispherical sample. These warping functions distort the samples [Keller et al. 2019], degrading their spectral properties leading to inferior results. Instead, for ambient occlusion we use cosine-weighted hemispherical samples to remove any transformations done to the sample values before the rendering algorithm uses them.

Our render test results are shown under exponential moving average (EMA) to simulate temporal anti-aliasing without temporal reprojection or neighborhood history clamping [Yang et al. 2020]. Our noise textures have a depth of 64 for the time axis, but [Yang et al. 2009] show that with an EMA α value of 0.1, a maximum effective sample count of about 19 can be nearly fully reached with 32 samples. Also, while our noise textures are 128x128 spatially, we have found that the dimensionality required is situational, where the textures are too small if tiling artifacts are noticeable. Our testing shows that 64x64 or 128x128 works well for most usage cases.

In our rendering tests, all noise textures are 128x128x64 in size. A pixel (p_x, p_y) at frame t reads the noise texture at

$$(p_x \pmod{128}, p_y \pmod{128}, t \pmod{64}).$$



	Box	Gauss		Box	Gauss
White	0.15245	0.10161	Gauss 1.0	0.11760	0.07596
Box 3x3	0.11316	0.08087	Bin. 3x3	0.11618	0.07942
Box 5x5	0.12889	0.07451	KoracPT	0.14045	0.08910

Fig. 6. Dithering a color image before quantizing to one bits per color channel (8 colors total), using various noise sources, filtered with a 5×5 box filter, and a $\sigma = 1.0$ Gauss filter. Numbers in filtered renders are RMSE, red is the lowest. KoracPT uses the perceptual + TAA optimized noise provided in the supplemental material of [Korać et al. 2023], and gave the best results of the noise types provided. Full screen RMSE in lower left table.

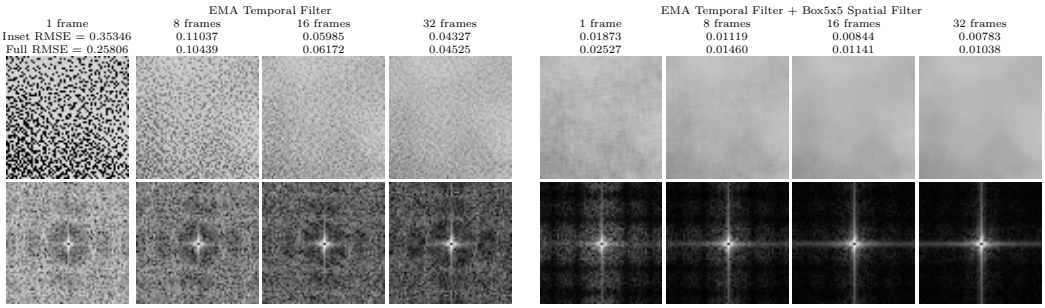


Fig. 7. 1 step ray marching using FAST noise optimized for a 5×5 box filter over space, and EMA over time, filtered with those same filters. Renders and DFTs over time to show how the image and frequencies evolve.

To support k random values per pixel, value $i \in \mathbb{Z}$, $[0, k)$ is read at

$$((p_x + R(i)) (\text{mod } 128), (p_y + R(i)) (\text{mod } 128), t (\text{mod } 64)),$$

where R is the R2 low discrepancy sequence [Roberts 2018]. This simulates having k independent sets of noise textures, as described in [Wolfe et al. 2022].

Our ray-traced ambient occlusion reads cosine-weighted hemispherical unit vectors from the noise textures as directions for occlusion rays. The rays have a maximum distance D, and after finding a hit at distance d, or a miss at distance $d = D$, will give an occlusion value of $\left(\frac{d}{D}\right)^a$ where

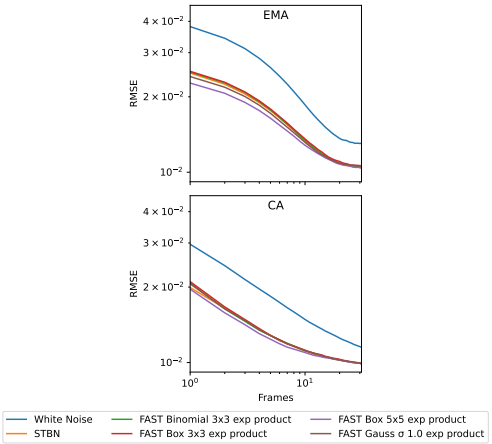


Fig. 8. RMSE of cloud rendering followed by a 5x5 box filter. Noise optimized for the filter (purple) gives lowest error, and outperforms STBN. EMA combines samples using an exponential moving average, CA uses a cumulative average.

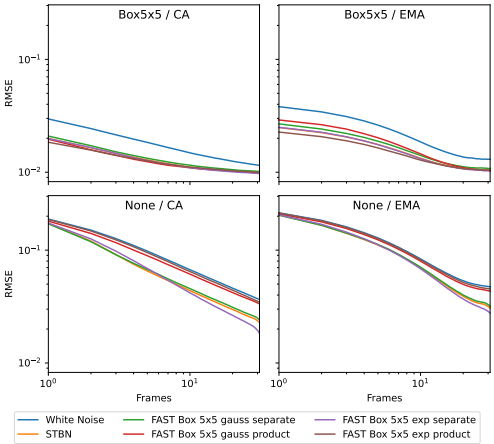


Fig. 9. RMSE of cloud rendering spatially filtered (top) shows that product (brown) is best. When not filtered (bottom) separate (purple) is best. EMA combines samples using an exponential moving average, CA uses a cumulative average.

a and D are user-tuneable parameters to achieve desired artistic results. When multiple samples are taken per pixel, the results are averaged to give the final occlusion value for that pixel.

The ray marching participating media technique takes N evenly spaced steps through a bounding volume, applying lighting and absorption to each step. This technique requires a single scalar per pixel, to advance the starting position of the ray march between 0 and 1 step lengths.

For the dithering technique, three scalar values are read from a noise texture and used to dither each color channel individually before quantizing to a lower bit depth. While this is useful for decreasing the bit depth of color images, it is also helpful in encoding spatially coherent data such as G-buffer fields like albedo or normal. This technique may also have use in exotic displays or sensors, being able to use lower memory bandwidth or work with reduced hardware capabilities while better preserving final image quality.

6.1 Spatial Filtering

When using noise textures as a source of random numbers within rendering algorithms, the characteristics of the noise used tends to show up in the rendering results, as can be seen in the inset DFTs of Figures 4, 5, and 6.

When using noise complementary to a specific filter being used, the noise in the resulting image is optimized to be removed by that filter. This is shown in those same figures where the noise optimized for a 5×5 box filter tends to have the lowest RMSE when box filtered, and the noise optimized for a $\sigma = 1.0$ Gaussian filter tends to have the lowest RMSE when filtered by a $\sigma = 1.0$ Gaussian filter.

The rendering process does not entirely preserve the noise characteristics, since the integrands vary across the image plane. As a result, there are situations where the noise paired with a filter does not have the lowest RMSE. In those situations, a smaller filter performs better, but we find the RMSE of the noise paired with the filter is still close to optimal.

Our render results show that these statements hold when using importance sampled vectors in ambient occlusion, a single scalar to ray march through participating media with four equally spaced steps, and three scalar values to dither a color image before quantizing it to 1 bit per color channel.

6.2 Spatial & Temporal Filtering

Fig. 7 shows that render error takes on the frequencies of the noise textures under EMA, which is the reverse of a box 5×5 filter in that figure. When filtered under a 5×5 box filter, the image has a much lower RMSE as time progresses. This explains how the noise optimized for a 5×5 box filter performs the best in Fig. 8, significantly outperforming the other sample types including spatiotemporal blue noise.

6.3 Only Temporal Filtering

So far we have considered the case where both spatial and temporal filtering are used. We now consider the case where we have only temporal filtering, where it is still desirable to have blue-noise distributed errors for their perceptual properties. As explained in [Peters 2017], three-dimensional blue noise - noise optimized for an isotropic three-dimensional Gaussian filter - is unsuitable for animating two-dimensional slices of blue noise. [Wolfe et al. 2022] addressed this problem by separating the spatial and temporal kernels and summing their energy. We therefore consider the impact of these two choices — exponential moving average versus Gaussian filtering over time, and tensor product versus separate filtering — under scenarios with and without spatial filtering. We have found that separating the kernels gives the best results when no spatial filtering occurs, but combining the kernels via a tensor product provides the best results when spatially filtering. We have also found that optimizing for an exponential moving average filter on the time axis is superior to optimizing for a Gaussian filter on the time axis, which would result in blue noise over time. Both of these points are shown in Fig. 9.

7 FUTURE WORK

We have presented a general framework for optimizing samples toward a given sample space, distribution, filter and noise model. While we have calculated the form of the loss function for some of the most commonly used distributions in rendering, these are not the only possibilities. Other applications of random sampling which could benefit from this work include dithering to low bit depth render targets, anti-aliasing, ray traced glossy reflections, indirect lighting, depth of field and motion blur.

Our algorithm relies on several assumptions about the rendering pipeline that could be better studied. First is our idealized modeling of rendering integrands as random Heaviside functions. It would be possible to directly measure the correlation function $\langle \phi(x)\phi(y) \rangle$ (5) in real rendering scenarios. This function could be used as an input to generate samples targeted toward specific applications. Similarly, we rely on the user to specify a filter f_{ij} to be used in denoising, but real filters are typically edge-aware. This effective filter could also be measured in real scenes, providing a potentially better input to our algorithm. More generally, noise could be optimized for non linear filters and denoisers. One possibility in this direction would be to use a fully differential neural denoising pipeline, and to jointly optimize the sample points alongside the weights of the neural denoiser.

Lastly, we believe our work has applications outside of real-time rendering, as evidenced by spatiotemporal blue noise finding use in graphical machine learning algorithms [Bauer et al. 2023]. Another potential application is in diffusion models, which have emerged as popular generative

models for image synthesis. While currently these models are based on Gaussian white noise, the ability to generate noise tailored toward a particular spectrum may unlock new possibilities.

8 CONCLUSION

Real-time rendering poses a unique challenge for sampling: without the computational budget to reach convergence, real-time applications have to hide sampling error with perceptually favorable blue noise, or remove it with a denoiser. We have presented a theoretical framework to shape the spectrum of rendering noise to optimize samples for perceptual quality and denoising performance.

We have shown that samples can be optimized toward specific spatial and temporal filters, leading to better results with lower error for the same computational costs. In the time domain, we find that adapting samples for an exponential moving average filter gives better convergence than blue noise over time when using temporal antialiasing. In the spatial domain, we also find significant benefits from adapting the noise spectrum to the shape of the denoising filter. This holds particularly for situations like our ambient occlusion example where the signal is slowly varying in screen space. In this case we can clearly see the shadow of the spatial filter in the Fourier spectrum of the noise. For problems with more small-scale structure – such as our volumetric smoke example – we find it is beneficial to optimize toward a smaller spatial filter. For such cases, or in situations without spatial filtering, we found that samples optimized toward a 3×3 binomial filter generally perform well. The way in which filters are combined also makes a difference, depending on the desired application. In cases without spatial filtering we find that additively combining filters yields the best perceptual results, while under spatial filtering it is best to combine spatial and temporal filters via a product.

We have demonstrated improvements over state of the art sampling under various types of spatial and temporal filtering, with scalar and non-uniform vector-valued noise textures, in three different rendering applications. While these specific applications have shown good results, we present them only as a representative sample. Our work can provide per-pixel random values to any stochastic rendering algorithms, so it has a wide range of use cases not explored here.

ACKNOWLEDGMENTS

We thank Christos Loukovikas, the Need for Speed team, and Lamborghini for permission to use the car model in Fig. 1. Cabin photo in Fig. 6 by Philip Jahn. We thank Diede Apers, Tom Hammersley, and Dustin Hulm for being early adopters, and Andrew Hellmer and Martin Mittring for comments on an earlier draft. We thank EA SEED for fostering a research environment conducive to work that is impactful to both academic and industry interests, including the Future Graphics Group and associated individuals: Colin Barré-Brisebois, Vicki Ferguson, Jon Greenberg, Henrik Halén, Chris Lewin, Martin Mittring, Uma Jayaram, Kristina Tomaz-Young, and Jenna Frisk. Thank you!

REFERENCES

Abdalla G. M. Ahmed, Jing Ren, and Peter Wonka. 2022. Gaussian Blue Noise. *ACM Trans. Graph.* 41, 6, Article 260 (nov 2022), 15 pages. https://doi.org/10.1145/3550454.3555519

Abdalla G. M. Ahmed and Peter Wonka. 2020. Screen-Space Blue-Noise Diffusion of Monte Carlo Sampling Error via Hierarchical Ordering of Pixels. *ACM Transactions on Graphics* 39, 6, Article 244 (Nov. 2020).

David Bauer, Qi Wu, and Kwan-Liu Ma. 2023. FoVolNet: Fast Volume Rendering using Foveated Deep Neural Networks. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 515–525. https://doi.org/10.1109/TVCG.2022.3209498

Bryce E. Bayer. 1973. An optimum method for two-level rendition of continuous-tone pictures. *IEEE International Conference on Communications* 1, 11–15.

Laurent Belcour and Eric Heitz. 2021. Lessons Learned and Improvements When Building Screen-Space Samplers with Blue-Noise Error Distribution. In *ACM SIGGRAPH 2021 Talks* (Virtual Event, USA) *(SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 9, 2 pages. https://doi.org/10.1145/3450623.3464645

Vassillen Chizhov, Iliyan Georgiev, Karol Myszkowski, and Gurprit Singh. 2022. Perceptual error optimization for Monte Carlo rendering. *ACM Trans. Graph.* 41, 3 (2022). https://doi.org/10.1145/3504002

Cameron N. Christou. 2008. Optimal Dither and Noise Shaping in Image Processing. MSc thesis, University of Waterloo.

Iliyan Georgiev and Marcos Fajardo. 2016. Blue-Noise Dithered Sampling. In *ACM SIGGRAPH Talks*. Article 35.

Mikkel Gjoel and Mikkel Svendsen. 2016. Low Complexity, High Fidelity: The Rendering of INSIDE. In *Game Developer's Conference*.

Eric Heitz and Laurent Belcour. 2019. Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds Between Frames. *Computer Graphics Forum* 38, 4 (2019), 149–158.

Eric Heitz, Laurent Belcour, Victor Ostromoukhov, David Coeurjolly, and Jean-Claude Iehl. 2019. A Low-Discrepancy Sampler that Distributes Monte Carlo Errors as a Blue Noise in Screen Space. In *SIGGRAPH Talks*.

Jorge Jimenez. 2014. Next generation post processing in call of duty advanced warfare. *SIGGRAPH Advances in Real-Time Rendering in Games* (2014).

Brian Karis. 2014. High-Quality Temporal Supersampling. *SIGGRAPH Advances in Real-Time Rendering in Games* (2014).

Alexander Keller, Iliyan Georgiev, Abdalla Ahmed, Per Christensen, and Matt Pharr. 2019. My Favorite Samples. In *SIGGRAPH Courses*. Article 15, 271 pages.

Miša Korać, Corentin Salaün, Iliyan Georgiev, Pascal Grittmann, Philipp Slusallek, Karol Myszkowski, and Gurprit Singh. 2023. Perceptual Error Optimization for Monte Carlo Animation Rendering. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 89, 10 pages. https://doi.org/10.1145/3610548.3618146

Don P. Mitchell. 1987. Generating Antialiased Images at Low Sampling Densities. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 65–72. https://doi.org/10.1145/37401.37410

Harald Niederreiter. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611970081 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611970081

Alexey Panteleev and Christoph Schied. 2019. Real-Time Path Tracing and Denoising in 'Quake 2' (Presented by NVIDIA). In *Game Developer's Conference*.

Christoph Peters. 2017. The problem with 3D blue noise. https://momentsingraphics.de/3DBlueNoise.html. [Online; accessed 27-March-2023].

Ravi Ramamoorthi, John Anderson, Mark Meyer, and Derek Nowrouzezahrai. 2012. A Theory of Monte Carlo Visibility Sampling. *ACM Trans. Graph.* 31, 5, Article 121 (sep 2012), 16 pages. https://doi.org/10.1145/2231816.2231819

Nathan Reed. 2013. Quick And Easy GPU Random Numbers In D3D11. https://www.reedbeta.com/blog/quick-and-easy-gpu-random-numbers-in-d3d11/. [Online; accessed 03-April-2023].

Martin Roberts. 2018. The Unreasonable Effectiveness of Quasirandom Sequences. http://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/. [Online; accessed 3-August-2022].

Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022. Scalable Multi-Class Sampling via Filtered Sliced Optimal Transport. *ACM Trans. Graph.* 41, 6, Article 261 (nov 2022), 14 pages. https://doi.org/10.1145/3550454.3555484

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal Variance-guided Filtering: Real-time Reconstruction for Path-traced Global Illumination. In *High Performance Graphics*. 2:1–2:12. https://research.nvidia.com/publication/2017-07_Spatiotemporal-Variance-Guided-Filtering%3A

John Spitzer. 2021. Next-Generation Game Development on NVIDIA RTX GPUs (Presented by NVIDIA). In *Game Developer's Conference*.

Robert Ulichney. 1987. *Digital Halftoning*. The MIT Press. https://doi.org/10.7551/mitpress/2421.001.0001

Robert Ulichney. 1988. Dithering with blue noise. *Proc. IEEE* 76, 1 (1988), 56–79. https://doi.org/10.1109/5.3288

Robert Ulichney. 1993. The Void-and-Cluster Method for Generating Dither Arrays. In *SPIE Symposium on Electronic Imaging Science & Technology*. 332–343.

Alan Wolfe, Nathan Morrical, Tomas Akenine-Möller, and Ravi Ramamoorthi. 2022. Spatiotemporal Blue Noise Masks. In *Eurographics Symposium on Rendering*, Li-Yi Ghosh, AbhijeetWei (Ed.). The Eurographics Association, 117–12610 pages. https://doi.org/10.2312/sr.20221161

Bart Wronski. 2020. *"Optimizing" blue noise dithering – backpropagation through Fourier transform and sorting*. Retrieved October 11, 2022 from https://bartwronski.com/2020/04/26/optimizing-blue-noise-dithering-backpropagation-through-fourier-transform-and-sorting/

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621. https://doi.org/10.1111/cgf.14018 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14018

Lei Yang, Diego Nehab, Pedro V. Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. 2009. Amortized Supersampling. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–12. https://doi.org/10.1145/1618452.1618481