Eurographics Symposium on Rendering (DL-only Track) (2022)
A. Ghosh and L.-Y. Wei (Editors)

Spatiotemporal Blue Noise Masks

Alan Wolfe[1,2], Nathan Morrical[1,3], Tomas Akenine-Möller[1] and Ravi Ramamoorthi[1,4]

[1]NVIDIA [2]EA SEED [3]University of Utah [4]University of California, San Diego

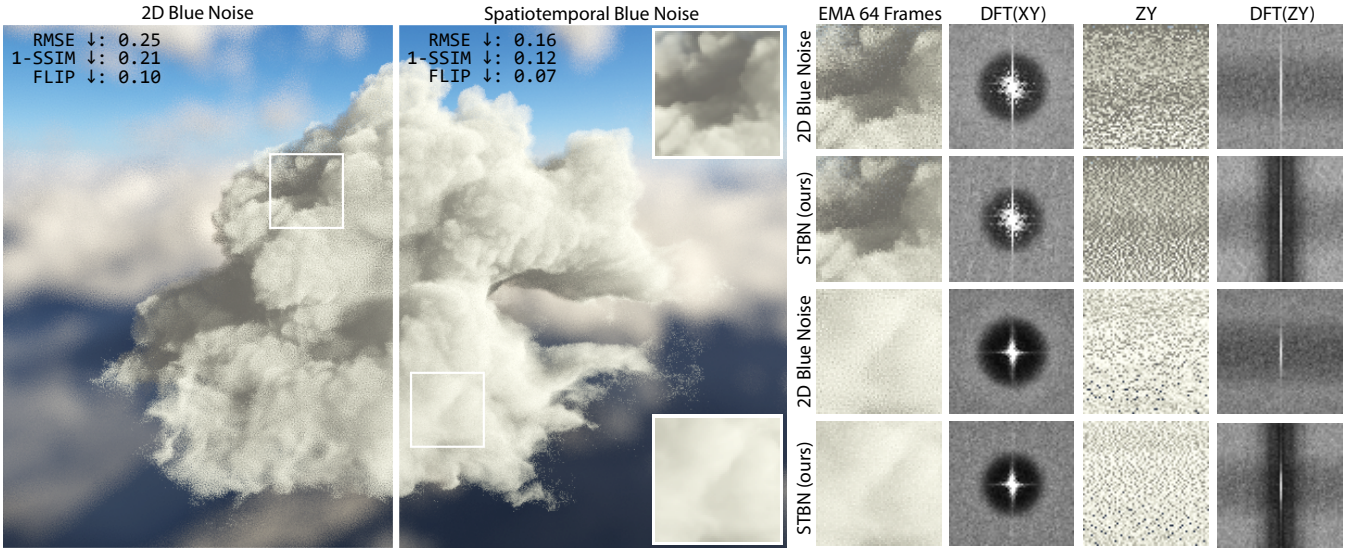


Figure 1: All images rendered using an exponential moving average (EMA) with $\alpha = 0.1$. Left: The Disney Cloud [Dis20] rendered using stochastic single scattering, where free-flight distances are sampled using a series of blue noise masks over time. Traditional 2D blue noise masks (far left) are easy to filter spatially, but exhibit a white noise signal over time, making the underlying signal difficult to filter temporally. Middle: Our spatiotemporal blue noise (STBN) masks also exhibit blue noise in the temporal dimension, resulting in a signal that is easier to filter over time. Right: Crops of the main image and corresponding discrete Fourier transforms over both space (DFT(XY)) and time (DFT(ZY)). The ground truth is shown in the insets in the large image (upper and lower right corners). This rendering uses scalar valued masks but we are also able to create uniform and nonuniform (importance-sampled) vector valued masks.

Abstract

Blue noise error patterns are well suited to human perception, and when applied to stochastic rendering techniques, blue noise masks can minimize unwanted low-frequency noise in the final image. Current methods of applying different blue noise masks to each rendered frame result in either white noise frequency spectra temporally, and thus poor convergence and stability, or lower quality spatially. We propose novel blue noise masks that retain high quality blue noise spatially, yet when animated produce values at each pixel that are well distributed over time. To do so, we create scalar valued masks by modifying the energy function of the void and cluster *algorithm. To create uniform and nonuniform vector valued masks, we make the same modifications to the* blue-noise dithered sampling *algorithm. These masks exhibit blue noise frequency spectra in* both the spatial and temporal domains*, resulting in visually pleasing error patterns, rapid convergence speeds, and increased stability when filtered temporally. Since masks can be initialized with arbitrary sample sets, these improvements can be used on a large variety of problems, both uniformly and importance sampled. We demonstrate these improvements in volumetric rendering, ambient occlusion, and stochastic convolution. By extending spatial blue noise to spatiotemporal blue noise, we overcome the convergence limitations of prior blue noise works, enabling new applications for blue noise distributions. Usable masks and source code can be found at https://github.com/NVIDIAGameWorks/SpatiotemporalBlueNoiseSDK.*

CCS Concepts

• ***Computing methodologies*** → ***Rendering;***

1. Introduction

In real-time and complex stochastic rendering scenarios, sample counts per frame are constrained. As a result, many modern visual effects depend on amortizing sampling expense over space and time to achieve higher-quality images at an acceptable performance. Blue noise has long been desired over white noise for error patterns in computer graphics [Yel83], but few attempts have been made to extend spatial blue noise along the temporal domain to gain similar benefits.

In addition to spatial filtering, current real-time techniques often use temporal antialiasing (TAA) [YLS20] to filter rendered images over time. Samples that are more evenly distributed over the time axis make the exponential moving average (EMA) within TAA be more accurate and more stable. Techniques also often seek to integrate over multiple samples per pixel, while still maintaining blue noise error properties spatially. Computer displays—and even human perception—can also perform some amount of implicit integration over time, especially at high frame rates [ANS*19]. These situations motivate the need for good sampling patterns over time in addition to space.

There has been considerable effort to generate high quality *spatial* blue noise patterns—made popular in rendering by Mitchell [Mit91]. Patterns can broadly be divided into two categories: sets of blue noise distributed *sample points* [BH08, LNW*10, dGBOD12], and *masks* containing values in an image [Uli93, GF16]. We focus on the latter. These blue noise masks can be used by stochastic rendering algorithms as a source of pseudorandom random numbers that produce perceptually uniform noise in the resulting image.

Our paper extends blue noise masks to account for the *temporal* domain as well. We generate an array of two-dimensional blue noise masks, where each pixel also has blue noise properties over the time dimension. These masks could be used to make renders that when combined with algorithms like TAA [YLS20] would produce higher quality, more temporally stable results for the same rendering costs.

Our methods realize that desire, as shown in Fig. 1, while also supporting arbitrary probability distribution functions (PDFs) of the values within the masks, allowing for importance sampled spatiotemporal blue noise masks. Where other methods focus on good convergence first and blue noise quality second, we focus on blue noise quality first, and convergence second, enabling higher-quality renders at the lowest of sample counts, i.e., targeting real-time rendering. Note that it has already been shown that denoising methods such as SVGF [SKW*17, SPD18] benefit from spatial blue noise, and we show that denoising algorithms can provide further image quality improvements with our spatiotemporal blue noise masks. The renders in this paper use the values in the masks directly as the point sets used for sampling, and do not use the masks to Cranley-Patterson rotate other sampling sequences such as Sobol or Halton.

A limitation of our work is that at one sample per pixel, convergence is only better than white noise for pixels that are not moving. This is a limitation of all work which attempts to combine better convergence (over time) with blue noise error because when a pixel moves, it can either stay with the sequence at the old pixel location - which would destroy the spatial blue noise - or it can start using the sequence at the new pixel location - which would destroy the temporal blue noise.

Our contributions include:

- An algorithm that generates scalar valued spatiotemporal masks in Section 4.1, and uniform or nonuniform vector valued spatiotemporal masks in Section 4.2.
- Practical analysis of frequency spectrum and convergence speeds of our masks in Section 5.
- Evaluations of our masks when applied to rendering techniques in Section 6.
- Source code for generating Blue noise masks of various types at `https://github.com/NVIDIAGameWorks/SpatiotemporalBlueNoiseSDK`.
- Generated masks ready for use at `https://github.com/NVIDIAGameWorks/SpatiotemporalBlueNoiseSDK/blob/main/STBN.zip`.

2. Previous Work

The most prevalent method for generating scalar valued blue noise masks is the *void and cluster* algorithm by Ulichney [Uli93]. Georgiev and Fajardo [GF16] are the most prevalent for vector valued masks. These are the algorithms we extend in this work, as neither deals with the axis of time.

Heitz and Belcour [HB19] point out that the improvements of blue noise dithered sampling are quickly lost as sample count and sample dimensionality increase. Their alternative method reorganizes per-pixel random seeds such that the rendered image approximately follows a blue noise pattern. This exhibits white noise temporally but it is meant to be a purely spatially filtered technique.

Ahmed and Wonka [Ahm20] propose a method for spatial blue noise with better convergence. They use a locality-preserving mapping of 2D pixel coordinates into a 1D pixel sequence, and then use 1D low discrepancy sequences across those pixels. This results in a trade-off between spatial and temporal quality.

Heitz et al. [HBO*19] also aim to use a faster converging sequence while retaining spatial blue noise using precomputed permutations of the Sobol sequence. The multiple samples per pixel could be distributed over time, but this comes at the cost of spatial image quality and temporal strobing as shown in Fig. 3.

Gjoel and Svendsen [GS16] present an in depth study of blue noise for real-time rendering, but animate it as white noise over time. 3D blue noise is unfortunately neither blue over space, nor time as can be seen in Fig. 3. A completely different method [BWP*20] for better renders at low sample counts modifies samples at runtime for better results at the same costs, but also does not explicitly address the sampling pattern over the time axis.

While spatial blue noise has been a goal for some time [Uli88], our use of temporal blue noise is to improve convergence rates without damaging the noise spatially. Other work listed above either increases convergence while damaging the noise spatially, or has equal quality spatially, but worse convergence temporally. If one is ever able to generate masks which are blue spatially but have

a faster convergence rate, those masks should likely be prefered - although there is evidence in this paper that temporal blue noise has increased stability when temporally filtered.

3. Background

We briefly provide background on the methods by Ulichney [Uli93] and Georgiev and Fajardo [GF16], on which our algorithm builds.

3.1. Void and Cluster Algorithm

For scalar valued masks, we extend the void and cluster (V&C) method [Uli93] to handle the temporal domain in Section 4.1. To generate a mask M, V&C stores a boolean per pixel specifying if the pixel was turned *on* and an integer index per pixel specifying the order that this pixel was turned on in. This index is used to compute the final output color for that pixel, where index 0 is black and the highest index is white. Every pixel $\mathbf{p}$, that is turned on emits *energy* to every pixel $\mathbf{q}$ in an *energy field* with total energy calculated as

$$E(M) = \sum_{\mathbf{p},q \in M} E(\mathbf{p},\mathbf{q}) = \sum \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma^2}\right), \tag{1}$$

where $\mathbf{p}$ and $\mathbf{q}$ are integer coordinates and distances are computed on wrapped *toroidal* boundaries. σ controls energy falloff over distance—Ulichney [Uli93] recommends $\sigma = 1.5$.

The V&C algorithm uses three ordering phases. In the first phase, V&C generates an initial *binary* pattern where less than half of the pixels are chosen to be turned on randomly. V&C transforms these pixels into an initial blue noise set by repeatedly turning off the tightest cluster pixel—defined as $\text{argmax}_{\mathbf{p}\in M} E(\mathbf{p})$—and turning on the largest void pixel—defined as $\text{argmin}_{\mathbf{p}\in M} E(\mathbf{p})$. This process is repeated until the same pixel is found for both operations. The energy at a pixel $\mathbf{p}$ is defined as

$$E(\mathbf{p}) = \sum_{\mathbf{q}\in M} E(\mathbf{p},\mathbf{q}). \tag{2}$$

Next, V&C converts the blue noise set into a progressive sequence. It iteratively turns off the tightest *cluster* pixel and assigns an index equal to the number of pixels that remain on. This process is repeated until all pixels are turned off, at which point the initial binary pattern is turned back on.

In the second phase, V&C iteratively turns on the largest *void* pixel, assigning the next ordering number, until half of the pixels are on and ordered.

In the third phase, V&C reverses the state of all pixels, turning off pixels on and vice versa. V&C iteratively turns off the current tightest *cluster* pixel, giving each pixel the next ordering number until all pixels are off and ordered.

Once all pixels are ordered, this ordering is used to compute the final pixel values in the output image. For floating-point images, per-pixel values can be found by dividing the order of each pixel by the total number of pixels in the image. For k-bit images, these pixel values must be remapped to 0 to $2^k - 1$.

3.2. Swap Algorithm

For vector valued masks, we extend the work of Georgiev and Fajardo [GF16], to handle the temporal domain in Section 4.2. To generate a mask M, with vectors of dimension d, the mask is initialized to uniform white noise, then randomly selected pairs of pixels are repeatedly swapped if doing so decreases the energy of the mask $E(M)$. The total energy is computed as

$$E(M) = \sum E(\mathbf{p},\mathbf{q}) = \sum \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{\sigma_i^2} - \frac{\|\mathbf{V_p}-\mathbf{V_q}\|^{d/2}}{\sigma_s^2}\right), \tag{3}$$

where $\mathbf{p}$ and $\mathbf{q}$ are the integer coordinates, and distances are computed on wrapped *toroidal* boundaries. $\mathbf{V_p}$ and $\mathbf{V_q}$ are the vectors stored at those pixels. σ_i and σ_s control the strength of the energy field over space and between the vector values respectively. The paper recommends $\sigma_i = 2.1$ and $\sigma_s = 1.0$. The $d/2$ exponent is meant to correct for the difference in the average distance between points in the d-dimensional vectors and the 2D mask.

4. Spatiotemporal Blue Noise Masks

We first discuss our algorithm to make scalar valued masks, based on the faster void and cluster algorithm. We then significantly extend the swap algorithm, which is slower, but can enable general vector and importance-sampled masks.

While these scalar and vector valued masks are the main focus of our work, the algorithms can be further customized to make other types of masks. Please see the supplemental material for expositions including masks which are blue over space but stratified over time, as well as a higher dimensional generalization of spatiotemporal blue noise.

The modifications described below can be seen as simple, but the simplicity is beneficial; spatiotemporal blue noise (STBN) is easy to implement and adopt, especially where blue noise masks are already being used.

4.1. Scalar Valued Masks

Scalar masks have a scalar value per pixel and are useful in rendering algorithms that want random scalar values per pixel, such as the volumetric rendering in Section 6.1 and Fig. 1. To generate them, we reformulate the void and cluster algorithm from Section 3.1 such that noise generation is driven by a novel energy function, in the spirit of Equation 2. We distribute energy in three dimensions, but only return nonzero energy between two pixels if they are from the same two-dimensional spatial layer, or if they are the same pixel at different points in time. This is expressed as

$$E(\mathbf{p},\mathbf{q}) = \begin{cases} \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma^2}\right), & \text{if } \mathbf{p}_{xy} = \mathbf{q}_{xy} \text{ or } p_z = q_z \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where a pixel in the three-dimensional spatiotemporal blue noise texture is denoted as $\mathbf{p} = (\mathbf{p}_{xy}, p_z) = (p_x, p_y, p_z)$.

The first condition ensures spatial blue noise, while the second condition ensures that each pixel will exhibit blue noise properties over time. This energy function is illustrated in Fig. 2. We used an initial binary pattern density of 10% of pixels, $\sigma = 1.9$ for all axes.

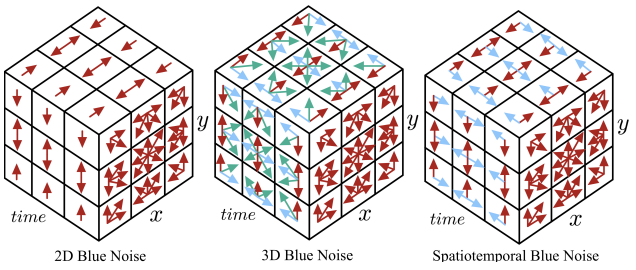


Figure 2: Energy functions of blue noise masks. 2D blue noise is made independently of other slices. 3D blue noise is fully aware of all slices. Our Spatiotemporal Blue Noise adds temporal awareness to each pixel in 2D blue noise.

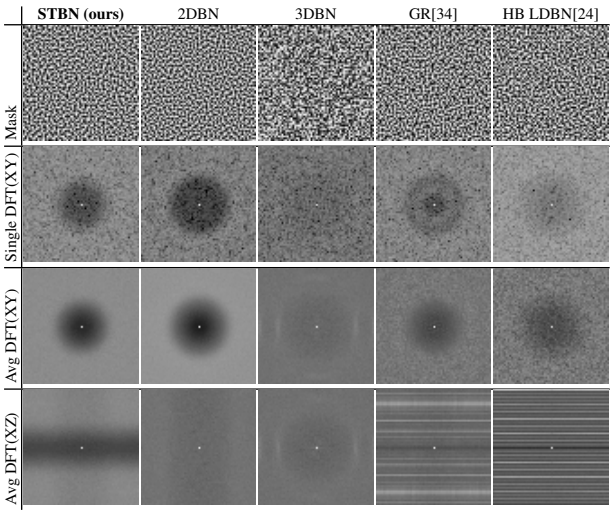


Figure 3: Spatiotemporal frequencies of various masks. Only ours (STBN) has blue noise over space and time (z-axis). Golden ratio animated blue noise (GR) and Heitz Belcour (HB LDBN) show specific frame numbers where the blue noise is much lower quality.

The reasoning behind the design of this energy function is that we want noise which is blue over space (the xy plane) and also we want each pixel individually to be blue over time (the z axis). The motivation for blue noise on the z axis is only to give better convergence than white noise. While our formulation gives equal weighting to both the spatial and temporal requirements, we've found that this gives good results.

4.2. Vector Valued Masks

Vector masks have a vector value per pixel and are useful in rendering algorithms that want random vector values per pixel, such as the ambient occlusion in Section 6.2 and Fig. 9. To generate them, we make the same algorithmic modifications. We distribute energy in three dimensions, but only return nonzero energy between two pixels if they are from the same two-dimensional spatial layer, or if they are the same pixel at different points in time. We set σ_i to 1.9 and σ_s to 1.0. The $d/2$ exponent in Equation 3 becomes $d/3$ because we are working in three dimensions now. Our energy function is

$$E(\mathbf{p},\mathbf{q}) = \begin{cases} \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{\sigma_i^2} - \frac{\|\mathbf{V}_p-\mathbf{V}_q\|^{d/3}}{\sigma_s^2}\right), & \text{if } \mathbf{p}_{xy}=\mathbf{q}_{xy} \text{ or } p_z=q_z \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Importance sampling is often desired when rendering, but putting samples through importance sampling functions distorts the points. This damages their otherwise desirable properties such as convergence speed or frequency content [CRW09], causing developers to have to choose between importance sampling or good sampling sequences. We realized that the texture can be initialized to nonuniform white noise vectors, before the algorithm is executed, allowing for blue noise masks which are also importance sampled. The PDF of each pixel can either be stored in, e.g., the alpha channel of the mask, or it can be derived, for instance, by a dot product in the case of cosine weighted hemispherical samples. This allows a developer to have both importance sampling and (spatiotemporal) blue noise sampling properties without compromising on either.

This algorithm is able to generate scalar masks as well, but execution time is longer than void and cluster, and stopping is controlled by either an error threshold, or a maximum swap count, both of which are tuneable parameters. Because of this, void and cluster is able to make higher-quality scalar masks more quickly, although it is limited to uniform PDFs.

5. Analysis

We now analyze the spatiotemporal blue noise produced by our method, and also discuss the use with temporal antialiasing (TAA).

5.1. Scalar Spatiotemporal Blue Noise Analysis

We generated scalar valued masks of resolution 64^3 of the following types: spatiotemporal blue noise (STBN, our method), independent 2D blue noise (2DBN), 3D blue noise (3DBN), and the low discrepancy sampler by Heitz and Belcour [HB19] (HB LDBN). We also repeatedly added the golden ratio to a single 2D blue noise mask (GR) to make it low discrepancy over time, which is equivalant to using a spatial blue noise texture to Cranley-Patterson rotate the golden ratio rank 1 lattice.

Figure 3 shows that only our masks have attenuated low frequencies on both the XY spatial plane as well as the Z time axis. 2D blue noise has spatial blue noise but is white noise temporally due to each slice being generated independently. 3D blue noise is blue over neither space nor time. Golden ratio animated blue noise and HB LDBN both damage the blue noise spatially and their time axes have pronounced frequency peaks which result in strobing.

To test convergence speeds, we use these types of noise to integrate simple 1D functions. Figure 4 shows that while our masks are not always the fastest converging (compared to low-discrepancy temporal sequences), they are competitive, and are more temporally stable than the competition. This is because of the low aliasing property of blue noise, due to it being made from high-frequency randomization, instead of the structural patterns you see in a low discrepancy sequence. For more information about the temporal instability, please see Burley's work [Bur20]. Convergence levels off

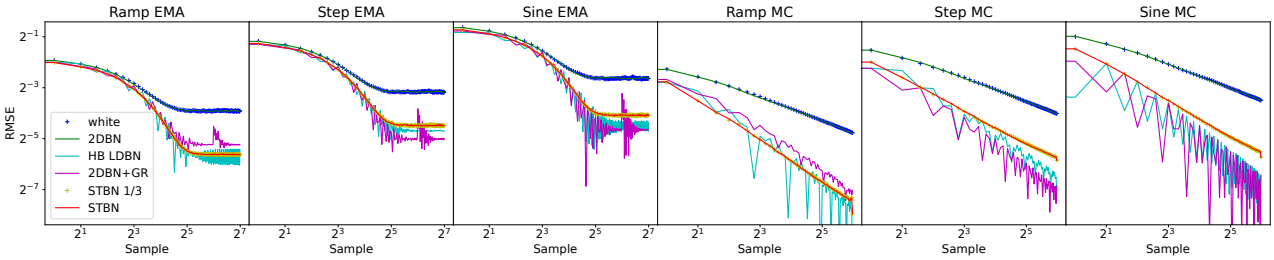


Figure 4: RMSE for sampling simple functions. Each sample is a frame of a 64×64 noise texture. Samples are combined using EMA ($\alpha = 0.1$) (left 3) and Monte Carlo integration (right 3). EMA simulates TAA without reprojection, motion vectors, or history rejection. Independent slices of 2D blue noise (green) are white noise over time, STBN shows toroidal progressiveness in the offset sequence STBN 1/3 (yellow), and the low discrepancy samples show good but erratic convergence (magenta, teal). Golden ratio animated blue noise (magenta) has a spike under EMA where the sequence restarts to avoid numerical issues.

to a plateau in EMA because it is a leaky integrator and converges to a biased value. Leaky integration is desired in dynamic, interactive real-time rendering situations, over Monte Carlo integration, which is appropriate only for static conditions.

Only our noise gives good spatial error distributions while also giving competitive convergence speeds, and desired temporal stability. Another important feature of our method is toroidal progressiveness, which is discussed in Section 5.3.

5.2. Vector Spatiotemporal Blue Noise Analysis

Fig. 5 shows spatial and temporal frequency content of multiple types of spatiotemporal blue noise masks, including importance sampled masks. The scalar mask made with the void and cluster algorithm stores the same type of data as the Vec 1 mask made with the swap algorithm, but the quality generated by the void and cluster algorithms is noticeably higher. Note that the void and cluster algorithm is unable to make vector valued or importance sampled blue noise masks. Frequency comparisons versus non spatiotemporal masks is omitted as the results are very similar as in Fig. 3. In all cases of spatiotemporal blue noise shown, the low frequencies are attenuated on both the XY spatial plane as well as the Z time axis. While vector valued blue noise masks have been generated by others [GF16], vector valued spatiotemporal blue noise masks and nonuniform distributions of vectors within them are novel.

To test convergence speeds, we use two-dimensional vector valued noise types to integrate simple 2D functions in Fig. 6. We compare independently generated 2D blue noise (2DBN), scalar STBN, which reads at two different pixel offsets to get two values, importance sampled STBN (IS STBN), which is importance sampled to the specific function being integrated, and uniform STBN (STBN), which has uniformly distributed vectors.

Our importance sampled spatiotemporal blue noise generate results with substantially lower RMSE in all cases in Fig. 6. 2DBN once again performs the same as white noise over time, due to each frame being generated independently. A noteworthy result is that while vector valued STBN beats scalar STBN in both circle and Gauss, scalar wins when integrating the step function. This is due to the step function being an inherently scalar function, and it is well known that single axes of blue noise sample points are not themselves blue when looked at independently [RRSG16].

5.3. Use with TAA

Temporal antialiasing uses an exponential moving average (EMA) for each pixel, correlating pixels from frame to frame to account for motion. When history is determined to be invalid due to events like disocclusion, a pixel will reject the history and restart the integration. This is a problem when using a global progressive sequence because different pixels restart on their own timelines, defeating the benefits of the progressive sampling by starting sampling in the middle of a sequence. To counter this, sequences are restarted fairly often, limiting the number of effective samples a pixel can integrate, and thus limiting image quality [YLS20]. Our masks avoid this by being toroidally progressive, providing a seamless, progressive sampling sequence for each pixel starting at any index. This allows for longer sampling sequences, and thus higher effective sample counts and image quality. This is demonstrated in Fig. 4, where "STBN 1/3" shows the masks starting at frame index 11 instead of 0, and resulting in the same convergence rate.

Golden ratio animated blue noise needs to restart the sequence periodically to keep from hitting numerical problems, and creates a spike of error and a sampling discontinuity when that happens. Our noise does not have that problem by being seamless, as can be seen by the clear frequency content on the DFT in Figure 3, which assumes infinite repetition of the sampling pattern, and is thus seamless for the frequencies shown.

The whole purpose of using temporal antialiasing is being able to have a moving, dynamic scene which is able to amortize rendering costs by integrating a render over time. When pixels are under motion in TAA, pixel history migrates between the pixels but the sampling sequence does not. If the desire is to have sampling with specific properties over space, and specific properties over time, a moving pixel has to choose whether to preserve the spatial or temporal properties after it has moved. As the goal of our work is to maximize image quality at the lowest of sample counts, we opt to preserve spatial properties and not migrate the sampling sequence.

Figure 7 shows that our masks have spatial blue noise quality under TAA when pixels are in motion, keeping full benefit of spatial blue noise (including the ability to use importance sampled blue noise vectors spatially), and that they do better for any pixels which are not in motion. Any still pixels will converge faster and better, and that quality will be carried around if those pixels start to move.

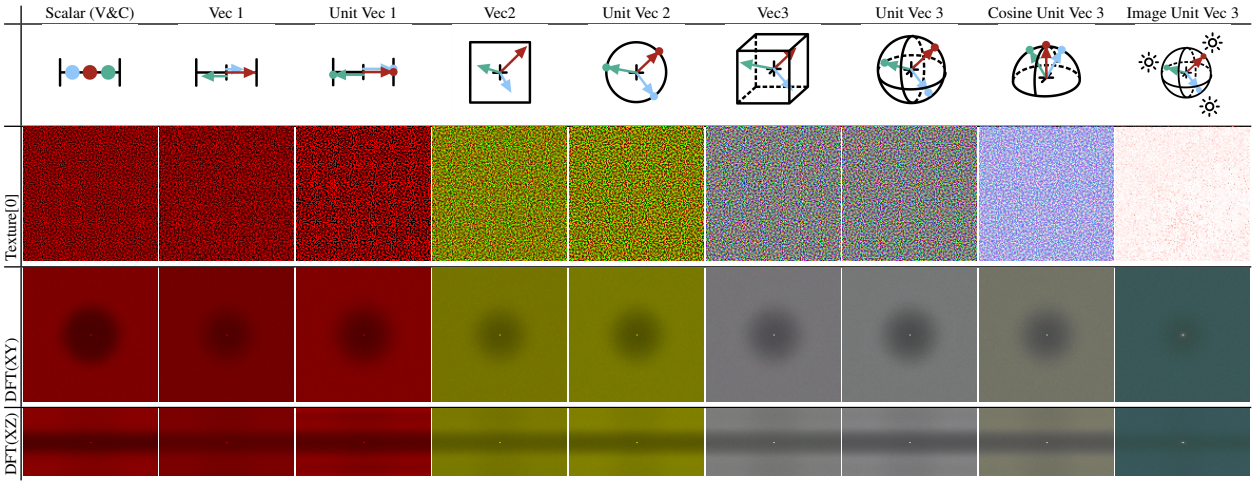


Figure 5: Masks and their spatial (xy) and temporal (xz) frequency content. All masks have uniform distributions except for cosine unit vec 3, which involved cosine weighted hemispherical unit vectors, and an HDR image which used an HDR skybox image as a source for importance sampling. Scalar was made with the V&C process while the others were made with the swap process.

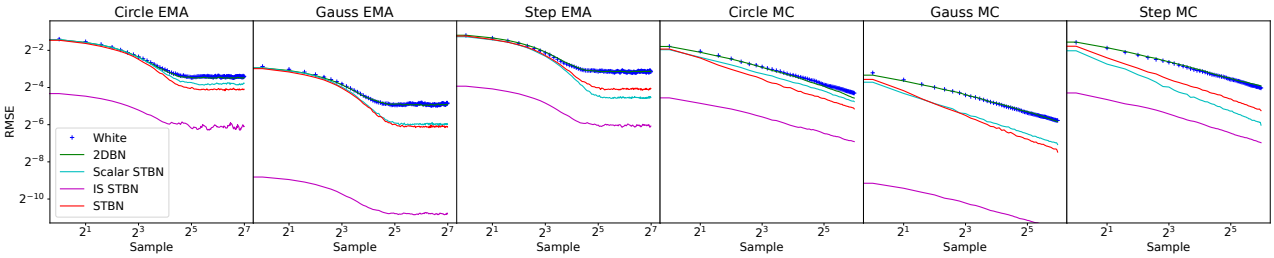


Figure 6: RMSE for sampling simple functions. Each sample is a frame of a 64×64 noise texture. Samples are combined using EMA ($\alpha = 0.1$) (left 3) and Monte Carlo integration (right 3). EMA simulates TAA without reprojection, motion vectors, or history rejection. Independent blue noise (green) shows the same convergence as white noise (blue), importance sampled STBN (magenta) shows the best convergence, and scalar STBN (teal) wins for step which is inherently 1D, while vector STBN (red) wins for the other functions that are genuinely 2D.

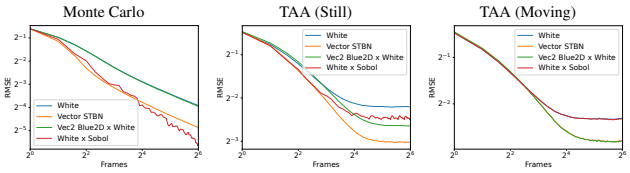


Figure 7: Convergence rate of ray traced AO. STBN (our method) and White×Sobol have best convergence under Monte Carlo at low sample counts. Under TAA, STBN has the best convergence for regions not in motion, but under motion it is equivalent to spatial blue noise.

5.4. Properties After Spatiotemporal Filtering

Fig. 1 shows that after multiple frames of EMA in a realistic rendering situation, spatiotemporal blue noise retains spatial blue noise but has a lower error magnitude than animated 2D blue noise. To answer the question of whether this is a property of the noise or the rendering algorithm, Fig. 8 shows a more direct evaluation by integrating a constant function. Note that the DFT of a constant function consists of zeroes everywhere except at frequency 0 (DC component). Spatiotemporal blue noise shows a strong blue noise spectrum for few samples in the figure, but converges substantially faster to the expected result as can be seen in the bottom right image. This can be seen as evidence that a spatiotemporal blue noise mask provides a better integral value compared to spatial blue noise masks. More results supporting this can be seen in the supplemental material.

6. Results

Our masks have broad applications in a range of rendering and related problems. We show three representative examples, involving respectively scalar, vector, and importance sampled masks. As our masks are meant to be used in animated images, and not still images, we encourage readers to view the interactive html demos in the supplemental material.

6.1. Scalar Masks: Volumetric Rendering

We present a simple algorithm for rendering single scattering heterogeneous participating media with very low sample counts, sim-

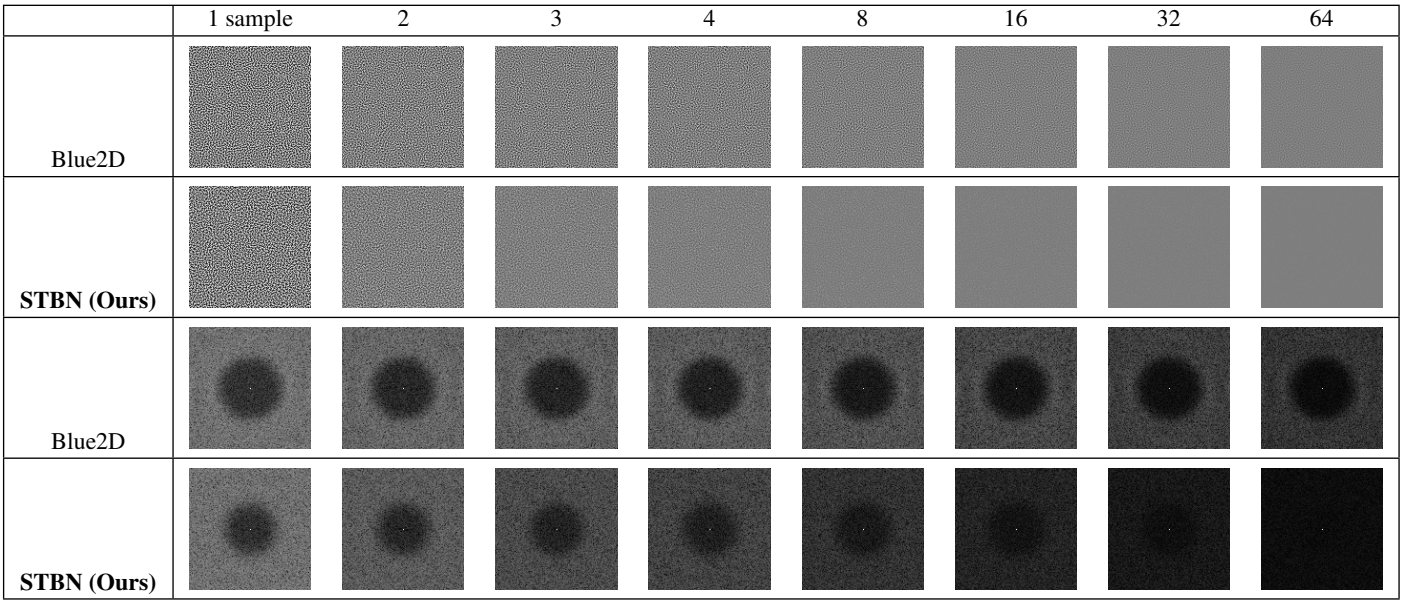


Figure 8: Top: Integrating a constant function with 64 noise textures using 2D blue noise and spatiotemporal blue noise. Bottom: The discrete Fourier transform magnitude of those integrated textures. Both types of noise show blue noise spectrum spatially, but spatiotemporal blue noise converges to the expected result substantially faster.

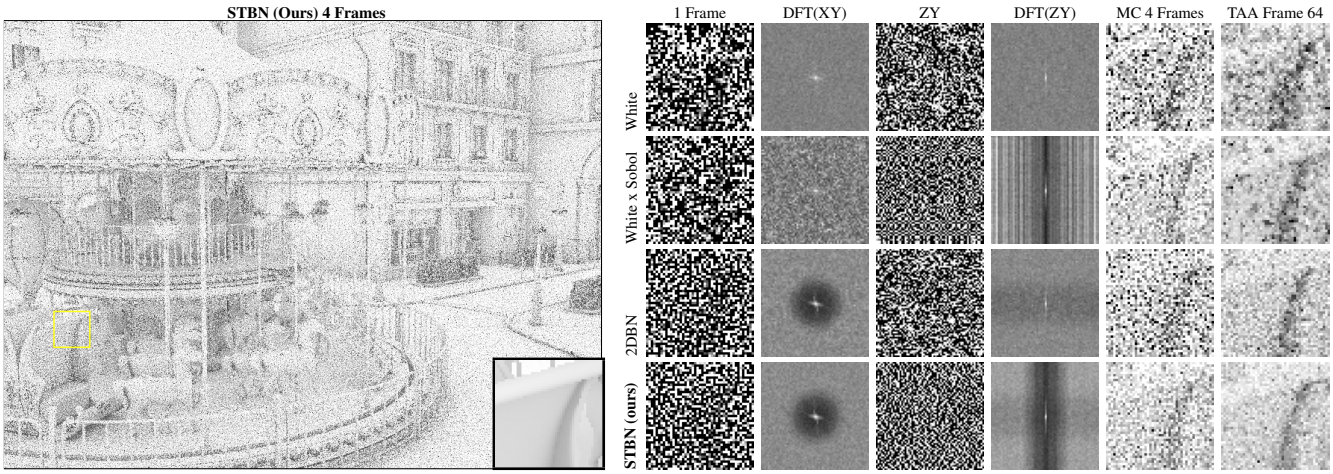


Figure 9: Ray traced ambient occlusion under Monte Carlo and TAA. Only our noise is blue over both space and time, and has the best quality shown for 4 samples under Monte Carlo, and 64 frames under TAA. Columns 2 and 4 are averaged to show expected spectra across all time slices. The ground truth is shown in the inset in the left image (lower right corner).

ilar to the airlight model by Sun et al. [SRNN05], but computed numerically for heterogeneous volumes.

First, a camera ray $\mathbf{o}+t\vec{\omega}_o$ is cast through the bounds of a heterogeneous volume, where an enter distance $t_{\min}$ and exit distance $t_{\max}$ through those bounds are recorded. From here, we use stochastic ray marching to march through the volume at n evenly spaced locations, where the space between each sample is $\frac{d}{n-1}$ units and $d = t_{\max} - t_{\min}$. The location $\mathbf{p}_s$ of a sample $s \in \mathbf{Z}[0, n-1]$ is then calculated as

$$\mathbf{p}_s = \mathbf{o} + \left(s\frac{d}{n-1} + t_{\min} \right) \vec{\omega}_o. \tag{6}$$

At each sample point $\mathbf{p}_s$, a volumetric density field F is sampled to get a density f_s. This density is assumed to represent the density for an entire step length of distance. We accumulate this density and proceed to the next sample point until the following criterion is met:

$$\sum_s f_s \geq -\ln(1-\xi), \tag{7}$$

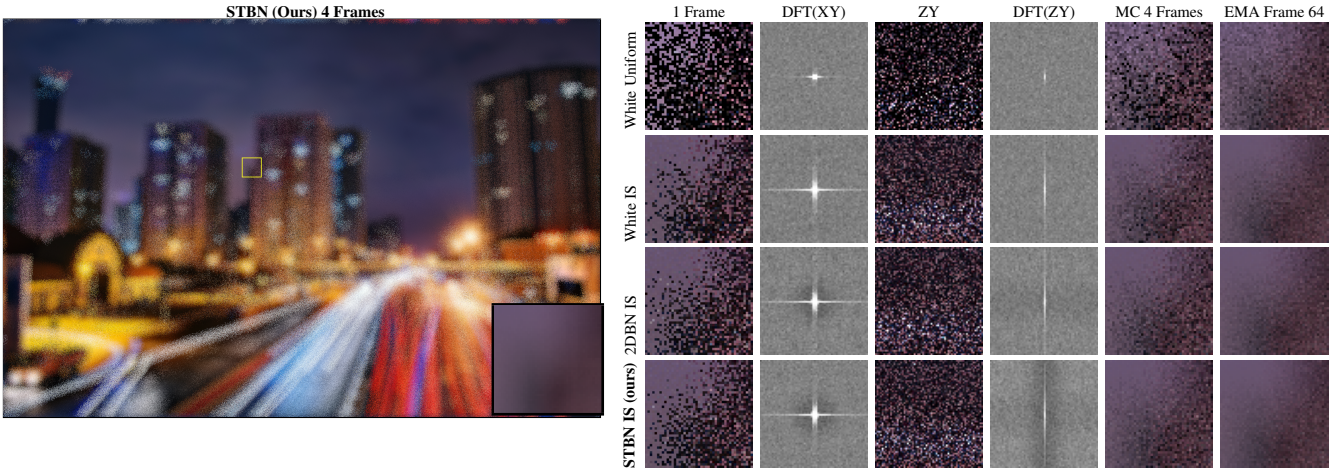


Figure 10: 1 SPP stochastic convolution under Monte Carlo and EMA. STBN converges faster under both, while also showing a blue noise spectrum over space and over time.

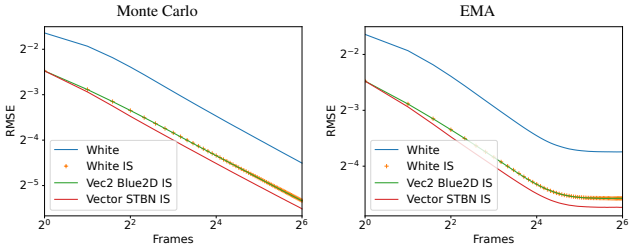


Figure 11: Convergence rate of stochastic convolution. The results are very similar for both Monte Carlo and EMA. Uniform white noise provides the highest RMSE. Importance sampled blue noise that stores a Vec2 per pixel but is white noise over time generates a similar results as importance sampled white noise. Our method (STBN) is the algorithm with the lowest RMSE.

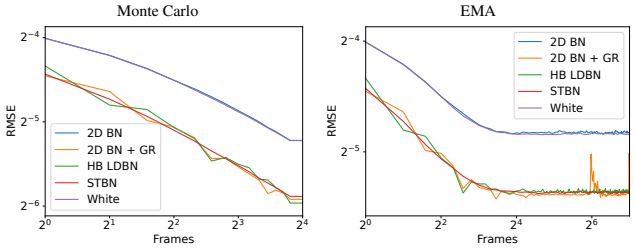


Figure 12: Convergence rate of single scattered volume rendering. Our spatiotemporal blue noise is competitive with the low discrepancy sequences but is much more stable and does not have the seam at the sequence restart point seen in EMA.

where ξ is a drawn random number, which is read from scalar noise masks. Once this condition is met, the point on the ray $\mathbf{p}_s$ will be approximately located at the sampled free-flight distance, or the distance at which the ray collides with a particle in the media. For more information, see the course by Novák et al. [NGHJ18].

At this point, a second ray originating at this collision point is traced toward a directional light source. Again, we march the ray through the volume. This second ray composites volumetric samples from front to back until the ray exits the volume. This composited value is then used to represent transmittance along the ray. To shade our collision point, we multiply the light intensity by the transmittance of light, and multiply by the albedo of the volume at the sampled collision location.

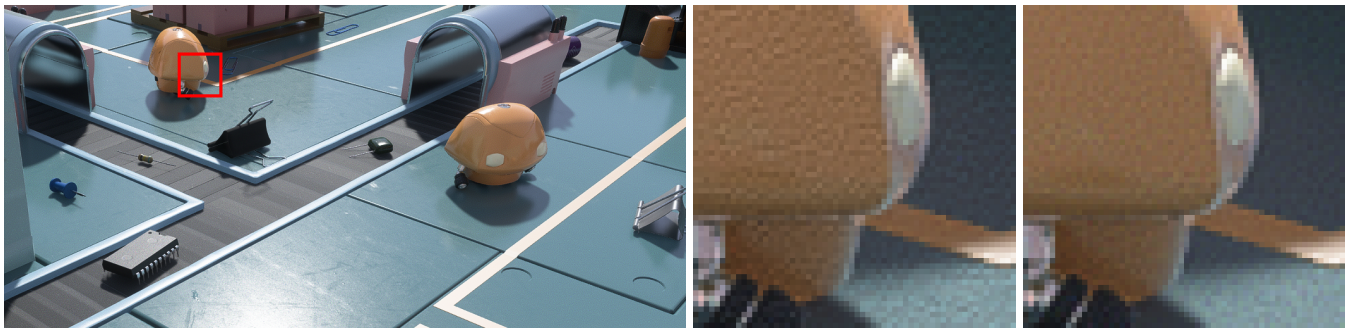
As shown in Fig. 1, using 2D blue noise for the free-flight collision distance turns the error into a pleasing 2D blue noise pattern. When using spatiotemporal blue noise, that pleasing 2D blue noise pattern still exists but is of a lower magnitude.

Fig. 12 shows convergence rates. Our spatiotemporal blue noise achieves competitive convergence with both the golden ratio LDS and the LDS by Heitz and Belcour [HB19]. However, note that our results do not damage the spatial blue noise properties, unlike these prior techniques. Moreover, our method is more temporally stable with somewhat faster convergence under EMA, which can also be seen more clearly in the supplemental material.

In Fig. 1, we also report image error metrics, namely RMSE, SSIM, and ꟻLIP [ANA*20], against a single scattering ground truth image. Although 2D blue noise already shows a favorable image error, our spatiotemporal blue noise further reduces this error by 36% for RMSE, 43% for SSIM, and by 30% for ꟻLIP.

6.2. Vector Masks: Ambient Occlusion

Next, we present a ray traced ambient occlusion algorithm. Each pixel reads a three-dimensional unit vector from a mask to use as the ray direction for a visibility query from the primary ray hit location. The ray hit length is used to interpolate the AO value from completely in shadow at $t = 0$ to completely in light at $t = T$, a scene dependent maximum AO ray length. This is multiplied by the cosine of the incident angle to get the final AO sample for that pixel. If multiple samples are taken, the samples would be averaged together for the final AO value.



2D blue noise STBN (ours)

Figure 13: Here we show images denoised using a spatiotemporal filter. The left inset used 2D blue noise during rendering, while the right inset used STBN. As can be seen, the level of noise in the final images are reduced to a greater extent with STBN.

Rendered results are shown in Fig. 9, where it can be seen that our spatiotemporal blue noise provides the best image quality under TAA, as well as low sample count Monte Carlo integration. The convergence graphs shown in Fig. 7 show the same. Only our noise gives both high quality blue noise spatially and faster convergence. Please also see our interactive html demo in the supplemental material where these differences are even more distinct.

6.3. Importance Sampled Vector Masks: Stochastic Convolution

Finally, we show a stochastic convolution algorithm which decreases the cost of convolving against a large kernel by convolving against a sparse subset of the kernel instead to save computation time. Each pixel reads a two-dimensional vector from a mask, samples the kernel at that location, and divides by the PDF. The kernel weights drive the PDF.

The rendered results in Fig. 10 show that STBN generates images with quality that is substantially closer to the ground truth image compared to the other methods, both under low sample count Monte Carlo and EMA. The convergence graphs in Fig. 11 show that our importance sampled vector valued spatiotemporal blue noise masks provide lower RMSE in those scenarios as well.

Imperfections in the importance sampling exist due to both the vectors and PDFs being stored in textures quantized to 8 bits per color channel.

6.4. Denoising

Since virtually all ray tracing based real-time rendering frameworks use some type of denoising technique, we also investigated how spatiotemporal blue noise works together with a denoising algorithm. In Fig. 13, an image was rendered using the PICA PICA ray tracing framework [BBHW*19] using a spatiotemporal filter. As can be seen, the insets show that the level of noise is substantially reduced using STBN compared to using 2D blue noise. This is a strong argument showing that STBN is valuable in low sample count real-time rendering scenarios.

7. Conclusions and Future Work

We have presented the first techniques that can generate scalar and vector valued masks with spatiotemporal blue properties. This was achieved by modifying existing algorithms for blue noise mask generation. We have shown how these masks can be useful for a variety of rendering algorithms and how resulting images have perceptually pleasing error and are easier to filter, while decreasing error magnitude through faster convergence.

A limitation of this work is the behavior under TAA when pixels are in motion. Our method does no worse than purely spatial blue noise when under motion, and does much better when not in motion. It would be good to address this situation better, perhaps by allowing all pixels within a similarly moving region (like on the surface of a moving object) to retain their sampling sequence after motion.

In Fig. 5, it can be seen that the scalar void and cluster noise provides higher quality than the scalar (Vec 1) valued noise made with the swapping algorithm. We also believe that STBN is a single member of a family of arbitrary dimensionality precomputed sampling textures optimized towards specific denoising techniques, rendering techniques, and artistic content. Both points suggest there is plenty of image quality left to be discovered at the lowest of sample counts.

In summary, spatiotemporal blue noise masks (STBN) have many applications, and are a drop-in replacement for independent 2D blue noise masks currently widely in use for real-time rendering.

Acknowledgements

This work was done while the authors were at NVIDIA. We thank Aaron Lefohn and NVIDIA Graphics Research for their support of the project. We express our gratitude to the anonymous reviewers for their thorough reading of the paper and many insightful suggestions. We would also like to thank EA SEED for the continued support of publishing this work, and also for the use of PICA PICA in our rendered results.

References

[Ahm20] AHMED, ABDALLA G. M. AND WONKA, PETER: Screen-Space Blue-Noise Diffusion of Monte Carlo Sampling Error via Hierarchical Ordering of Pixels. *ACM Transactions on Graphics 39*, 6 (Nov. 2020). 2

[ANA*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: ꟻLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 3*, 2 (2020), 15:1–15:23. 8

[ANS*19] ANDERSSON P., NILSSON J., SALVI M., SPJUT J., AKENINE-MÖLLER T.: Temporally Dense Ray Tracing. In *High-Performance Graphics - Short Papers* (2019). 2

[BBHW*19] BARRÉ-BRISEBOIS C., HALÉN H., WIHLIDAL G., LAURITZEN A., BEKKERS J., STACHOWIAK T., ANDERSSON J.: Hybrid Rendering for Real-Time Ray Tracing. In *Ray Tracing Gems*, Haines E., Akenine-Möller T., (Eds.). Apress, 2019, ch. 25, pp. 437–473. 9

[BH08] BALZER M., HECK D.: Capacity-Constrained Voronoi Diagrams in Finite Spaces. In *Symposium on Voronoi Diagrams in Science and Engineering* (2008), pp. 44–56. 2

[Bur20] BURLEY B.: Practical hash-based owen scrambling. *Journal of Computer Graphics Techniques (JCGT) 10*, 4 (December 2020), 1–20. URL: http://jcgt.org/published/0009/04/01/. 4

[BWP*20] BITTERLI B., WYMAN C., PHARR M., SHIRLEY P., LEFOHN A., JAROSZ W.: Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting. *ACM Transactions on Graphics 39*, 4 (July 2020). doi:10/gg8xc7. 2

[CRW09] CLINE D., RAZDAN A., WONKA P.: A comparison of tabular pdf inversion methods. *Computer Graphics Forum 28* (2009). 4

[dGBOD12] DE GOES F., BREEDEN K., OSTROMOUKHOV V., DESBRUN M.: Blue Noise through Optimal Transport. *ACM Transactions Graphics 31*, 6 (2012). 2

[Dis20] DISNEY: A large and highly detailed volumetric cloud data set. https://www.disneyanimation.com/resources/clouds/, 2020. 1

[GF16] GEORGIEV I., FAJARDO M.: Blue-Noise Dithered Sampling. In *ACM SIGGRAPH Talks* (2016). 2, 3, 5

[GS16] GJOEL M., SVENDSEN M.: Low complexity, high fidelity: The rendering of inside. In *Game Developer's Conference* (2016). 2

[HB19] HEITZ E., BELCOUR L.: 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. 2, 4, 8

[HBO*19] HEITZ E., BELCOUR L., OSTROMOUKHOV V., COEURJOLLY D., IEHL J.-C.: A Low-Discrepancy Sampler that Distributes Monte Carlo Errors as a Blue Noise in Screen Space. In *SIGGRAPH Talks* (July 2019). 2

[LNW*10] LI H., NEHAB D., WEI L.-Y., SANDER P. V., FU C.-W.: Fast Capacity Constrained Voronoi Tessellation. In *Symposium on Interactive 3D Graphics and Games* (2010). 2

[Mit91] MITCHELL D. P.: Spectrally Optimal Sampling for Distribution Ray Tracing. *Computer Graphics (SIGGRAPH) 25*, 4 (1991), 157–164. 2

[NGHJ18] NOVÁK J., GEORGIEV I., HANIKA J., JAROSZ W.: Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports) 37*, 2 (May 2018). 8

[RRSG16] REINERT B., RITSCHEL T., SEIDEL H.-P., GEORGIEV I.: Projective Blue-Noise Sampling. *Computer Graphics Forum 35*, 1 (2016), 285–295. 5

[SKW*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal Variance-guided Filtering: Real-time Reconstruction for Path-traced Global Illumination. In *High Performance Graphics* (2017), pp. 2:1–2:12. URL: https://research.nvidia.com/publication/2017-07_Spatiotemporal-Variance-Guided-Filtering%3A. 2

[SPD18] SCHIED C., PETERS C., DACHSBACHER C.: Gradient Estimation for Real-Time Adaptive Temporal Filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, 2 (2018). 2

[SRNN05] SUN B., RAMAMOORTHI R., NARASIMHAN S. G., NAYAR S. K.: A practical analytic single scattering model for real time rendering. *ACM Transactions on Graphics (TOG) 24*, 3 (2005), 1040–1049. 7

[Uli88] ULICHNEY R.: Dithering with blue noise. *Proceedings of the IEEE 76*, 1 (1988), 56–79. doi:10.1109/5.3288. 2

[Uli93] ULICHNEY R.: The Void-and-Cluster Method for Generating Dither Arrays. In *SPIE Symposium on Electronic Imaging Science & Technology* (1993), pp. 332–343. 2, 3

[Yel83] YELLOTT J.: Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina. *Science 221*, 4608 (1983), 382–385. 2

[YLS20] YANG L., LIU S., SALVI M.: A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum 39*, 2 (2020), 607–621. 2, 5