Author's manuscript. Published in The Visual Computer, first online: 10 May 2017
The final publication is available at Springer via http://dx.doi.org/10.1007/s00371-017-1392-7.

Forced Random Sampling

Fast Generation of Importance-Guided Blue-Noise Samples

Daniel Cornel[1] · **Robert F. Tobler**[1] · **Hiroyuki Sakai**[2] · **Christian Luksch**[1] · **Michael Wimmer**[2]

Abstract In computer graphics, stochastic sampling is frequently used to efficiently approximate complex functions and integrals. The error of approximation can be reduced by distributing samples according to an importance function, but cannot be eliminated completely. To avoid visible artifacts, sample distributions are sought to be random, but spatially uniform, which is called blue-noise sampling. The generation of unbiased, importance-guided blue-noise samples is expensive and not feasible for real-time applications. Sampling algorithms for these applications focus on runtime performance at the cost of having weak blue-noise properties.

Blue-noise distributions have also been proposed for digital halftoning in the form of precomputed dither matrices. Ordered dithering with such matrices allows to distribute dots with blue-noise properties according to a grayscale image. By the nature of ordered dithering, this process can be parallelized easily.

We introduce a novel sampling method called Forced Random Sampling that is based on Forced Random Dithering, a variant of ordered dithering with blue noise. By shifting the main computational effort into the generation of a precomputed dither matrix, our sampling method runs efficiently on GPUs and allows real-time importance sampling with blue noise for a finite number of samples. We demonstrate the quality of our method in two different rendering applications.

Keywords Blue noise · Importance sampling · Ordered dithering

Daniel Cornel
E-mail: cornel@vrvis.at

[1] VRVis Research Center, Vienna, Austria
[2] TU Wien, Vienna, Austria

1 Introduction

Stochastic sampling is a widely used technique in computer graphics and is typically used for anti-aliasing or the evaluation of complex integrals like the rendering equation [15]. Compared to regular sampling, stochastic sampling uses random sample locations, which mitigates aliasing artifacts. However, this does not guarantee the function to be sampled spatially uniformly. Random samples do not cover the sampling domain evenly, but tend to cluster and form holes. An ideal sample distribution would have a high spatial uniformity and a low regularity, such as the *Poisson disk* distribution. Its power spectrum has distinctive *blue-noise* properties, i.e., it is isotropic, has no concentrated spikes and no low-frequency energy. Sampling with such a distribution is called *blue-noise sampling*.

Creating a blue-noise sampling pattern with a given, constant density of samples can be easily achieved. However, in most applications, an *importance function* is available that provides an estimate of the function to be sampled. By varying the local sample density according to the importance function, the error of the approximation can be significantly reduced. In recent years, several efforts have been made to create fast sampling algorithms that generate blue-noise sampling patterns according to importance functions. However, the methods are still either limited in their performance or the blue-noise characteristics of the resulting sampling patterns are comparatively weak.

In digital halftoning, blue noise is used to reduce visible artifacts in the arrangement of ink dots. Ordered dithering, a common halftoning technique for grayscale images, overlays the image with an infinite tiling of a precomputed dither matrix. The decision whether to place an ink dot is the result of a threshold compari-

son of each image pixel with the corresponding dither matrix element. These threshold comparisons are independent from each other, which makes ordered dithering inherently parallelizable and therefore very efficient. However, due to the finite size of the dither matrix, only a finite number of unique dots can be drawn before the arrangement starts to repeat itself. Thus, ordered dithering trades high quality and performance at a small scale for repetition artifacts at a large scale.

In this paper, we introduce *Forced Random Sampling* (FRS), a simple and parallelizable blue-noise importance sampling algorithm based on ordered dithering. The main idea behind FRS is to threshold a large precomputed dither matrix with an importance function to distribute two-dimensional samples instead of ink dots. Although the algorithm works with any dither matrix, we use a *Forced Random Dithering* matrix [25], which results in a sampling pattern that resembles a blue-noise distribution. The matrix is created by randomly adding points into a repulsive force field near the location of lowest energy, which results in a spatially uniform, but irregular placement. As with dithering, using a non-constant importance function for thresholding allows the generation of samples with local densities matching the corresponding importance values. A dither matrix of size 64×64 with values scaled to $[0, 255]$ for display is illustrated in Figure 1 together with results of uniform and non-uniform sampling.

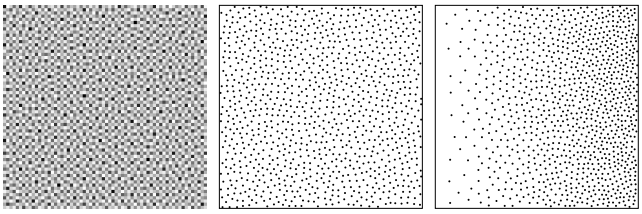

Fig. 1: Example of a 64×64 Forced Random Dithering matrix (left), and the results of Forced Random Sampling with a constant (middle) and a linear importance function (right).

2 Related Work

In 1983, Yellott [37] related the absence of aliasing in the vision systems of primates to the blue-noise distribution of photoreceptors and proposed such a distribution – called *Poisson disk distribution* – for artificial imaging. Cook [9] suggested the simple, but expensive *dart throwing* algorithm for generating such a distribution for a constant importance function. Sample distributions with even more pronounced blue-noise characteristics can be achieved with iterative approaches [3, 12,26]. Other algorithms for creating Poisson disk samples are based on tiling the plane [13,17,18,27]. Wei [32], Bowers et al. [5], and Xiang et al. [36] proposed parallel Poisson disk algorithms that are capable of blue-noise sampling on the GPU at a rate of several million samples per second. Recent approaches to generate Poisson disk samples include sample elimination [38] and rearrangement of low-discrepancy samples [2].

Relaxation dart throwing by McCool and Fiume [20] extends dart throwing to non-constant importance functions and will serve as reference for all spectral analyses in this work. For real-time importance sampling, several tile-based approaches have been proposed [23,24,31]. In particular, the *Recursive Wang Tiles* (RWT) by Kopf et al. [16] allow fast near-blue-noise importance sampling by recursively subdividing self-similar Wang tiles with progressive point sets. We compare the distribution properties of our method to those of RWT. Unlike RWT, our method is not recursive, which prohibits progressive sampling, but enables efficient GPU parallelization. In a survey on Poisson disk algorithms in 2008, Lagae and Dutré [19] concluded that algorithms based on tiling are the only option for real-time blue-noise sampling, despite their inferior spectral properties.

The current state of the art in real-time importance sampling and our reference for runtime performance is *Hierarchical Sample Warping* (HSW) by Clarberg et al. [7]. HSW warps an initial, uniformly distributed sample set recursively according to a hierarchical representation of the importance function in order to adapt the density of the samples to the local probabilities. However, HSW cannot preserve the distribution properties of the initial sample set and is therefore not suited for blue-noise sampling. Extensions to this technique have been proposed by Cline et al. [8], Huang et al. [14] as well as Clarberg and Akenine-Möller [6].

As with sampling, dithering with blue noise has been a research topic for decades [28, 29], of which *ordered dithering* is particularly important. Mitsa and Parker [21] proposed the use of blue-noise matrices instead of using the matrix by Bayer [4], which produces objectionable regularity artifacts. Similar matrices have been proposed by Ulichney [30], Purgathofer et al. [25], Newbern et al. [22], and Abe [1]. We base our method on Forced Random Dithering by Purgathofer et al. because of the simplicity and parallelizability of the matrix generation, which we revisit in Section 3. Recently, blue-noise dither matrices have been used for color banding reduction and volumetric light sampling in the video game INSIDE [11] and to reduce the perceivable noise in Monte Carlo rendered images by correlating samples between pixels [10].

3 Review of Forced Random Dithering

Forced Random Sampling (FRS) is based on Forced Random Dithering, proposed by Purgathofer et al. [25]. It is a variant of ordered dithering [4] that aims at a spatially uniform placement of dots without regularity artifacts. The creation of the dither matrix $\mathcal{M}$ uses the principle of repulsion to control the placement of dither values. To create a matrix of size $w_\mathcal{M} \times w_\mathcal{M}$, the values $0, \dots, w_\mathcal{M}^2 - 1$ are sequentially inserted into a discrete force field. In this force field, all values that have already been inserted repulse new values according to a force-field function f. For the most spatially uniform distribution of values in $\mathcal{M}$, the location at which the next value should be inserted is the global minimum of the force field. However, this deterministic choice could possibly lead to regular patterns, which is why a local minimum is chosen instead. This is done by randomly selecting half of all free locations of the force field and choosing from these the location with the minimal accumulated repulsion.

The design of the force-field function f follows from the requirements formulated by Purgathofer et al.: For isotropic images, f should be radially symmetric, and to avoid clumping of values, f should penalize closeness. Thus, as a function of the distance

$$r = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

of any location (x_2, y_2) to an already set location (x_1, y_1) in the force field, the repulsion is expressed as

$$f(r) = \exp\left(-\left(\frac{r}{s}\right)^p\right), \tag{2}$$

where p and s are parameters to control the steepness and deviation of f. For both values, Purgathofer et al. suggest to use $1/2$, which is used for all dither matrices throughout this paper.

The algorithm has a computational complexity of $\mathcal{O}(w_\mathcal{M}^4)$. With our GPU implementation – which is included in the supplementary material – a matrix of size 2048×2048 took roughly 2 hours to generate, a matrix of size 4096×4096 took about 20 hours.

4 Forced Random Sampling

Forced Random Sampling (FRS) is the application of Forced Random Dithering for importance sampling. Instead of halftoning an image, an importance function $\mathcal{I}$ defined in the sampling domain $\Omega \subseteq [0, 1)^2$ is considered. Distribution properties such as randomness and spatial uniformity are already included in the precomputed dither matrix and do not need to be computed at runtime. In this regard, FRS is similar to tile-based sampling methods. The main advantage of these methods is the separation of sample generation and sampling into an expensive offline and an inexpensive online computation step. The dither matrix $\mathcal{M}$ can be seen as a large set of possible sample distributions, including non-uniform ones. One set of samples is drawn from $\mathcal{M}$ by thresholding, where the importance function $\mathcal{I}$ controls the density of samples.

Uniform Sampling. To introduce the idea of FRS, we first consider the special case of uniform sampling, i.e., $\mathcal{I} = 1$ over the entire sampling domain. Let $\mathcal{M}$ be of size $w_\mathcal{M} \times w_\mathcal{M}$ and let $\mathbf{x}_\mathcal{M} \in [0, w_\mathcal{M} - 1]^2$ be the index of a matrix element with the precomputed dither value $\mathcal{M}(\mathbf{x}_\mathcal{M})$. The dither values stem from the order of insertion at matrix creation and range from 0 to $w_\mathcal{M}^2 - 1$. This means that thresholding $\mathcal{M}$ with a constant threshold function $\mathrm{T} = n, n \in \mathbb{N}$, will leave the first n elements that have been inserted. The index $\mathbf{x}_\mathcal{M}$ of each of these elements corresponds to a point $\mathbf{x}_\mathcal{I} = \mathbf{x}_\mathcal{M}/w_\mathcal{M}$ in the sampling domain Ω, which can be used for sampling. By the design of the matrix, these points are distributed spatially uniformly within Ω.

Tiling. From the dither matrix $\mathcal{M}$, a maximum of $w_\mathcal{M}^2$ samples can be generated with the described thresholding, which is one for every element of the matrix. To generate more than $w_\mathcal{M}^2$ samples without using a larger matrix, a tiling of $\mathcal{M}$ can be used. Since $\mathcal{M}$ is toroidal, it can be repeated seamlessly. From the theoretically infinite tiling of $\mathcal{M}$, a finite section is used for thresholding, which is illustrated in Figure 2 (left). This section, in the following called *window* of $\mathcal{M}$, has to be large enough to provide the desired number of samples, but should be as small as possible to minimize the computational effort of thresholding. The window can additionally be offset relative to the first element of $\mathcal{M}$ to vary the sample distribution for the same $\mathcal{I}$ and $\mathcal{M}$ over several sampling runs. Since this window offset $\mathbf{\Delta}_W$ does not influence thresholding, its value can be chosen randomly from $[0, w_\mathcal{M} - 1]^2$.

Having a window larger than the dither matrix influences thresholding because the same dither values occur multiple times in the window. In the case of a window of size $\mathbf{s}_W = (w_W, h_W) = (w_\mathcal{M}, w_\mathcal{M})$, the first n elements of the dither matrix pass thresholding. If, however, $w_W = h_W = 2w_\mathcal{M}$, each value $\in [0, w_\mathcal{M}^2 - 1]$ of the dither matrix appears four times in the window. Thus, the threshold needs to be scaled to a quarter of its original value to account for that. Similarly, if $w_W = h_W = w_\mathcal{M}/2$, the window would contain only a quarter of all possible dither values, which is why the

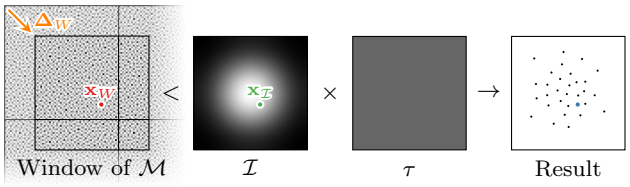

Fig. 2: Illustration of the thresholding inequality of Forced Random Sampling.

threshold would have to be multiplied by four. If the size of the window does not match the size of $\mathcal{M}$, the threshold function T therefore has to be extended to

$$\mathrm{T} = n \, \frac{w_\mathcal{M}^2}{w_W h_W} \, . \tag{3}$$

Sparsity. The size of the window $\mathbf{s}_W$ depends on multiple parameters and determines the quality and performance of sampling. The smallest possible window size for n samples is $\mathbf{s}_W = (\sqrt{n}, \sqrt{n})$. In this case, every element inside the window would pass thresholding, which would result in a completely regular sample placement. This degeneration to regular sampling has to be prevented by enforcing an average distance between samples, which is achieved with the *sparsity* parameter σ. It introduces a constraint that on average, only one out of σ^2 window elements passes thresholding. The choice of σ influences the quality of the resulting samples. If it is too low, the samples exhibit strong regularity artifacts. If it is too high, the window becomes large and too many elements are compared for thresholding, which reduces the runtime performance. σ should therefore be as small as possible, but as large as necessary to avoid artifacts. We propose a value of $\sigma = 8$, which is – independently of the matrix size $w_\mathcal{M}$ – a fair compromise between quality and performance.

Non-Uniform Sampling. Until now, only uniform sampling has been discussed, in which case it suffices to threshold the window with T as shown in Equation 3 to get n samples. But if $\mathcal{I}$ is non-constant, more elements of the window should remain in regions of higher importance than in regions of low importance such that the density of the resulting samples approximates $\mathcal{I}$. Thus, in the general case, the threshold of each window element $\mathbf{x}_W$ depends on the importance $\mathcal{I}(\mathbf{x}_\mathcal{I})$ at the corresponding point $\mathbf{x}_\mathcal{I} = {}^{\mathbf{x}_W}\!/_{\mathbf{s}_W}$ in the sampling domain, i.e.,

$$\mathrm{T}(\mathbf{x}_\mathcal{I}) = \tau \, \mathcal{I}(\mathbf{x}_\mathcal{I}), \tag{4}$$

where τ is a constant scaling factor. If $\tau = n \, {}^{w_\mathcal{M}^2}\!/_{(w_W h_W)}$, as in the uniform case, less than n samples would be created if $\mathcal{I}(\mathbf{x}_\mathcal{I}) < 1$ for any $\mathbf{x}_\mathcal{I}$. To create approximately n samples, τ needs to account for the average importance

$$\mu_\mathcal{I} = \int_\Omega \mathcal{I}(\mathbf{x}_\mathcal{I}) \, \mathrm{d}\mathbf{x}_\mathcal{I}, \tag{5}$$

leading to a scaling factor

$$\tau = \frac{n}{\mu_\mathcal{I}} \, \frac{w_\mathcal{M}^2}{w_W h_W} \, . \tag{6}$$

In Figure 2, the complete threshold inequality of FRS, $\mathcal{M}(\mathbf{x}_\mathcal{M}) < \mathcal{I}(\mathbf{x}_\mathcal{I}) \, \tau$, is illustrated using $n = 32$ and a Gaussian blob as importance function. For each element $\mathbf{x}_W$ of the window (red), the coordinates relative to the first element of the dither matrix are given by $\mathbf{x}_\mathcal{M} = (\boldsymbol{\Delta}_W + \mathbf{x}_W) \bmod w_\mathcal{M}$. The dither value $\mathcal{M}(\mathbf{x}_\mathcal{M})$ value is compared to the product of the corresponding importance $\mathcal{I}$ at point $\mathbf{x}_\mathcal{I}$ (green) and the threshold scaling factor τ. Approximately n elements pass this comparison, and their locations (blue) are used for sampling. The resulting samples are adapted to the importance function while still being well distributed.

Window Size Selection. The window is the section of an infinite tiling of $\mathcal{M}$ that is large enough to provide the desired number of samples n through thresholding. Its size $\mathbf{s}_W = (w_W, h_W)$ is a result of the sparsity constraint, which ensures that on average, only one out of σ^2 window elements passes thresholding, and thus prevents the sample placement from being too regular. For a constant importance $\mathcal{I} = 1$, the sparsity constraint can be expressed as

$$n\sigma^2 = w_W h_W. \tag{7}$$

This means that the dither matrix window has to have a size of at least $w_W \times h_W$ to provide n samples for a given σ. For a non-constant $\mathcal{I}$, the sample density should vary according to it. Therefore, for regions where $\mathcal{I}(\mathbf{x}_\mathcal{I}) < 1$, less dither values than one out of σ^2 should pass thresholding. Instead of having a probability of ${}^1\!/_{\sigma^2}$ to pass thresholding, as in the uniform case, the probability that window element $\mathbf{x}_W$ passes thresholding should be ${}^{\mathcal{I}(\mathbf{x}_\mathcal{I})}\!/_{\sigma^2}$. From this follows that with the $w_W h_W$ window elements from the entire sampling domain, only approximately

$$\frac{w_W h_W}{\sigma^2} \int_\Omega \mathcal{I}(\mathbf{x}_\mathcal{I}) \, \mathrm{d}\mathbf{x}_\mathcal{I} = n\mu_\mathcal{I} \tag{8}$$

samples could be created, which is less than desired. To still provide n samples in total, more samples would need to be placed in the more important regions, which would violate the sparsity constraint. Thus, the minimum window size that follows from Equation 7 is not sufficient in this case. Instead, the window size has to

be scaled by $1/\mu_\mathcal{I}$ beforehand to compensate for the low importance in some regions of $\mathcal{I}$.

However, this assumes that the maximum of $\mathcal{I}$,

$$m_\mathcal{I} = \max_{\mathbf{x}_\mathcal{I} \in \Omega} \mathcal{I}(\mathbf{x}_\mathcal{I}), \tag{9}$$

is 1, which is not necessarily true. If $m_\mathcal{I} < 1$, the window would be scaled up unnecessarily to ensure the sparsity constraint in regions of $\mathcal{I}(\mathbf{x}_\mathcal{I}) = 1$, which do not exist. Thresholding would thus compare more dither values than necessary. This can be avoided by scaling the importance function by $1/m_\mathcal{I}$ such that it has a maximum of 1. Including both average and maximum importance into Equation 7, the sparsity constraint for sampling with a non-constant $\mathcal{I}$ can be expressed as

$$n\sigma^2 \frac{m_\mathcal{I}}{\mu_\mathcal{I}} = w_W h_W. \tag{10}$$

As the window is thresholded with the importance function, their aspect ratios should match. In the following, the extents of $\mathcal{I}$ are denoted with $\mathbf{s}_\mathcal{I} = (w_\mathcal{I}, h_\mathcal{I})$, which is the pixel size of the discretized importance function, i.e., importance map. Without loss of generality, it is assumed that $w_\mathcal{I} \leq h_\mathcal{I}$, and therefore $w_W \leq h_W$. With the aspect ratio $a = h_\mathcal{I}/w_\mathcal{I} = h_W/w_W \geq 1$, Equation 10 can be rewritten as

$$n\sigma^2 \frac{m_\mathcal{I}}{\mu_\mathcal{I}} = a w_W^2, \tag{11}$$

from which follows

$$w_W = \sigma \sqrt{\frac{n m_\mathcal{I}}{\mu_\mathcal{I} a}}, \quad h_W = a w_W. \tag{12}$$

These equations describe the minimum window size to provide enough dither matrix elements for thresholding with $\mathcal{I}$ and sparsity σ.

Thresholding. Once the size of the window has been determined, every element $\mathbf{x}_W$ of the window can be thresholded with $\mathrm{T}(\mathbf{x}_\mathcal{I})$. For each $\mathbf{x}_W$, the dither value at the corresponding position $\mathbf{x}_\mathcal{M}$ within the dither matrix is compared to the threshold function and a sample $\mathbf{x} \in \Omega$ is created if $\mathcal{M}(\mathbf{x}_\mathcal{M}) < \mathrm{T}(\mathbf{x}_\mathcal{I})$. The location of the sample is $\mathbf{x} = \mathbf{x}_\mathcal{I} + \boldsymbol{\Delta}_\mathbf{x}/\mathbf{s}_W$, where $\boldsymbol{\Delta}_\mathbf{x} \in [0,1)^2$ is an offset inside the window element. We use $\boldsymbol{\Delta}_\mathbf{x} = (0.5, 0.5)$ for FRS and a random offset for jittered FRS.

For an implementation of FRS, thresholding can be parallelized because the thresholding inequality does not depend on any intermediate results other than the scaling factor τ and the window size. Similar to other sampling techniques, it is not guaranteed that exactly n samples are retrieved, but the deviation is negligible.

Since the dither matrix is repeated in the case of $w_W > w_\mathcal{M}$ or $h_W > w_\mathcal{M}$, repetitions in the arrangement of the samples can occur, depending on $\mathcal{I}$. In the case of a constant $\mathcal{I}$, this is certain. Repetitions cannot be avoided completely in general, but they can be reduced by either applying random jittering to the samples or increasing the size of the dither matrix. In the following, all qualitative tests will be performed on both jittered and unjittered samples.

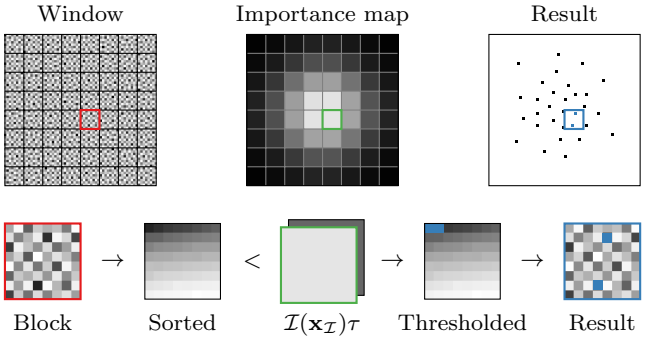

Fig. 3: Implementation of FRS. Sorting of dither values within each block allows fast thresholding with early termination.

Also, a dither matrix of size $w_\mathcal{M} = 2048$ is used. This size is a fair compromise between the expensive matrix creation and the number of unique samples that can be created. The size of the dither matrix does not influence the runtime complexity of FRS, but the quality of the samples. As the dither matrix is a regular grid, samples obtained from it always exhibit regularity to some extent because of their discrete element positions. With an increasing dither matrix size, this regularity becomes less apparent in the sample distribution. Furthermore, only a finite number of unique samples can be obtained from one dither matrix. Using Equation 7, this number is $n_\mathrm{max} = w_\mathcal{M}^2/\sigma^2$. For the parameters used in this paper ($w_\mathcal{M} = 2048$, $\sigma = 8$), $n_\mathrm{max} = 65536$.

5 Implementation

A naive implementation of Forced Random Sampling (FRS) compares every dither value of the window with the threshold function (Equation 4). Due to the introduced sparsity value (in our case 8), this would require a prohibitive number of comparisons (in our case 64) for each generated sample. In order to speed up the sampling process, a number of implementations have been investigated, including a hierarchical comparison of dither matrix values. The fastest version on both the CPU and the GPU is based on a reorganization of the values, which is illustrated in Figure 3. The dither matrix window is subdivided into $w_\mathcal{I} h_\mathcal{I}$ blocks of size $w_B \times w_B$ (red), each corresponding to one pixel of the importance map (green). Following Equation 7, the number

of samples that can be obtained from such a block is w_B^2/σ^2 on average. Accounting for the random distribution of values in the dither matrix, a conservative estimation of the maximum number of samples per block is twice that value, i.e., $n_B = 2w_B^2/\sigma^2$. If the dither values of a $w_B \times w_B$ block are sorted, thresholding for this block can stop after n_B comparisons. Furthermore, an early termination after the first element greater than the threshold can be performed (if the importance is constant within the block), or after the matrix value is greater than the maximum importance within the block (for non-constant importance).

To achieve this, the dither values are sorted in ascending order within each block. Additionally, an element index $\in [0, w_B^2 - 1]$ corresponding to the former position of the pixel in the block is kept for each value. All of these blockwise-sorted lists can be stored in one buffer of length $w_\mathcal{M}^2$, which is the size of the original dither matrix. Since the dither values are sorted and only a maximum of n_B values per block are required, only the first n_B elements of each block need to be stored, adding up to $n_B w_\mathcal{I} h_\mathcal{I}$ elements for the whole dither matrix. This is especially useful for the GPU implementation, as only a small portion of the dither matrix needs to be uploaded to and stored in the graphics card memory.

All $w_\mathcal{I} h_\mathcal{I}$ blocks are independent from each other and can be processed in parallel. The thresholding inside each block uses only basic operations that can be executed efficiently on the GPU. As window and block size are dictated by the maximum value of $\mathcal{I}$, the need for recursion of tile-based sampling algorithms such as Recursive Wang Tiles (RWT) is avoided. However, this comes at the cost of potentially many threshold rejections, especially if $\mathcal{I}$ has a high dynamic range.

Sample implementations of FRS for CPU and GPU can be found in the supplementary material for this paper together with dither matrices of different sizes.

6 Evaluation

Spectral Analysis. In order to evaluate the quality of a sample distribution for importance sampling, a *differential domain analysis* (DDA) [34] is performed. As with Fourier analysis, the results of this method can be shown in a spectral plot. The quality of a sample distribution is assessed by comparing its power spectrum to the power spectrum of relaxation dart throwing. Distinctive patterns are identified and interpreted according to the reference spectra given by Wei and Wang [34]. The optimal spectrum is isotropic, has a low-energy annulus around the center followed by a narrow high-energy annulus corresponding to the minimum differential of any two samples and then transitions into noise. As the average distance between samples varies according to the importance map and is the lowest at the most important region, the radii of the annuli also vary between different importance maps. Their general behavior, however, stays the same.

Estimated power spectra of Forced Random Sampling (FRS) and jittered FRS for different importance maps are shown in Figure 4. They are compared to the estimated power spectra of Hierarchical Sample Warping (HSW) [7] and Recursive Wang Tiles (RWT) [16]. For RWT, the implementation and the tiles included in the supplementary material for the paper by Kopf et al. have been used (8 tiles with 2048 base points and 4×4 sub-tiles with 1 subdivision rule each). For HSW, different input sample distributions have been used, which can be identified in Figure 4 by the labels *random*, *regular*, *Halton* and *Poisson disk*. All power spectra were estimated based on 10 different sets with 1024 samples each. For FRS and jittered FRS, the sparsity $\sigma = 8$ and matrix size $w_\mathcal{M} = 2048$ have been chosen. The DDA was performed with the tool provided by Wei [33], using the parameters that have been recommended in the paper by Wei and Wang [34] for an easy comparison.

The power spectra of relaxation dart throwing and FRS show similar distribution properties for all importance maps. In particular, they exhibit almost no low-frequency energy and are highly isotropic. This suggests the use of FRS as a fast alternative to relaxation dart throwing. The power spectrum of FRS also clearly shows the low-energy annulus around the center followed by a high-energy annulus, which is slightly fringed outwardly as compared to the sharp transition in the reference spectrum. This indicates a higher variation of the differentials and thus an increased randomness of FRS compared to Poisson disk sampling. Figure 5 illustrates a second difference to the reference spectrum: a second peak in the radial mean corresponding to the double of the minimum differential. In the power spectrum, this manifests itself as a second annulus of high energy around the first one. Furthermore, Figure 4 demonstrates that FRS is well suited for importance sampling, as its good spectral properties are preserved locally when using a non-constant importance function. These properties are even preserved for highly anisotropic importance functions such as the gradient, which pose a problem for HSW and RWT.

Another observation from the power spectra is that jittering the obtained FRS samples does not significantly influence the quality. Although the regular arrangement of values in the dither matrix leads to regular sample locations to some extent, the sparsity con-

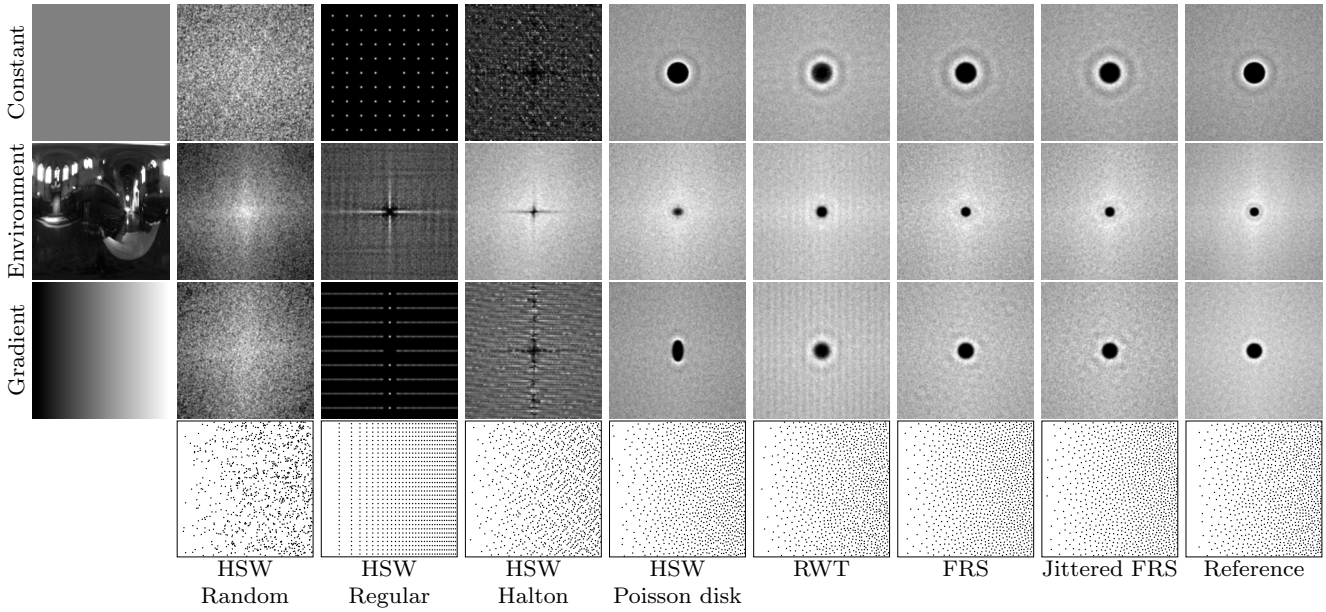

Fig. 4: Estimated DDA power spectra of different importance-guided sample distributions. The leftmost image column depicts the importance map. The bottom row shows a set of 1024 samples of each sample distribution for the linear gradient importance map. The reference sample distribution was generated with relaxation dart throwing.

straint causes a sufficiently irregular sample placement, which makes additional jittering unnecessary.

Depending on the size of the window, the dither matrix might need to be repeated to provide enough samples. To investigate the impact of these repetitions on the spectral properties of the sample distribution, large sample sets have been generated for the importance functions used for Figure 4 with predefined numbers of repetitions. The number of repetitions can be controlled by choosing a window size $\mathbf{s}_W$ that is a multiple of the dither matrix size $w_{\mathcal{M}}$.

In Figure 6, the DDA power spectra of FRS are shown for all unique samples of 1×1, 2×2, 4×4 and 8×8 repetitions of the dither matrix. For up to 4×4 repetitions, the power spectra for all importance functions are isotropic and do not exhibit any spikes of concentrated energy. These indicators for repetitive patterns only become apparent after 64 repetitions, in particular for the environment importance function. Depending on the application, the resulting artifacts might still be tolerable, but as a guideline, $w_{\mathcal{M}}$ should be chosen such that this amount of repetitions is not reached.

Density Analysis. Because of its simplicity, the density ρ_X of a sample set X is a popular measure for the quality of a sample distribution. Lagae and Dutré [19] proposed the calculation of the density based on the minimum Euclidean distance between any two samples in X, which only returns meaningful results if X is distributed according to a constant importance function. By using the minimum transformed differential $d_{\min}$ after the differential domain transform as proposed by Wei and Wang [34], the density can also be calculated for non-uniform sample distributions:

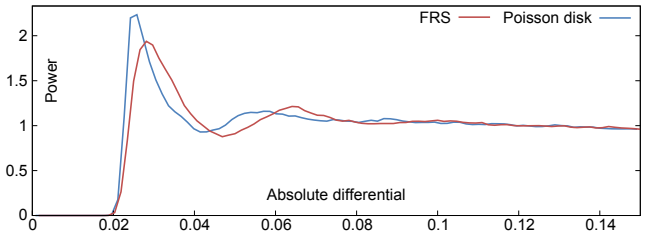

Fig. 5: Radial means of FRS and relaxation dart throwing.

$$\rho_X := \frac{d_{\min}}{d_{\max}}, \quad d_{\max} = \sqrt{\frac{2}{\sqrt{3}n}}. \tag{13}$$

A density $\rho_X \in [0.65, 0.85]$ is considered as a good density for blue-noise sampling, while a lower value indicates randomness and a higher value indicates regularity. Table 1 shows the densities for the importance maps and sampling methods used for Figure 4. The densities were averaged over 10 different sample sets with $n \approx 1024$ per distribution and per importance map. The reference sample distribution with a target density $\rho = 0.7$ is dart throwing in the uniform case and relaxation dart throwing in the non-uniform case.

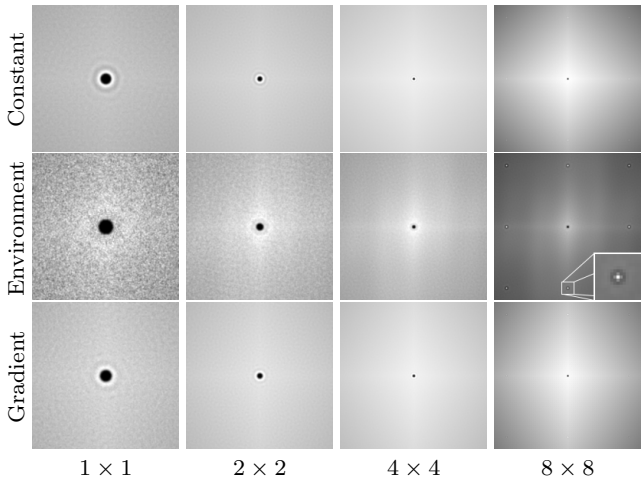

Fig. 6: Estimated DDA power spectra of Forced Random Sampling for windows larger than the dither matrix. One of the spike artifacts in the spectra generated with 8×8 repetitions of the dither matrix is enlarged for better visibility.

	Constant	Environment	Gradient
HSW/Random	0.003	0.008	0.001
HSW/Regular	0.930	0.357	0.439
HSW/Halton	0.066	0.041	0.138
HSW/Poisson disk	0.700	0.142	0.370
RWT	0.339	0.315	0.294
FRS	0.681	0.644	0.662
Jittered FRS	0.668	0.661	0.617
Reference	0.700	0.700	0.700

Table 1: Densities of different sampling methods (rows) for different importance functions (columns).

It can be seen that, independent of the input distribution, HSW is not able to preserve the density of the samples when warping non-uniformly. The density of RWT is similar for all importance functions, but very low, as already pointed out by Lagae and Dutré. The density of FRS is similar for all importance maps and is in the lower end of the interval suggested by Lagae and Dutré. The density of jittered FRS tends to be slightly lower than the density of FRS, which can be explained by the small amount of randomness added by jittering.

Performance Analysis. FRS, HSW with Halton samples, and RWT have been implemented based on the .NET Framework 4.0. For the GPU implementation, OpenCL 1.2 was used. RWT has not been implemented on the GPU, since the recursive nature of the algorithm prevents a high-performance GPU implementation. The tests were performed in a 64-bit environment on a system with an Intel i7-950 3.07 GHz quad-core processor and an NVIDIA GeForce GTX 680 graphics card.

In Figure 7, the benchmarks of FRS are compared to those of HSW with Halton points and RWT by averaging the results for the three different importance maps that were also used in the spectral analysis in Figure 4 (empirical experiments that were omitted for the sake of brevity have verified that the performance of FRS is independent of the used importance map). On the CPU, HSW is faster for up to 2^9 samples, but is outperformed by FRS for more than 2^{17} samples by a factor of 49. RWT is faster for up to 2^{13} samples, but is also outperformed by FRS for more than 2^{17} samples by a factor of 7.4. For both HSW and RWT, the computational effort grows linearly with the number of samples. The FRS implementations, in contrast, show a transition from constant to linear growth.

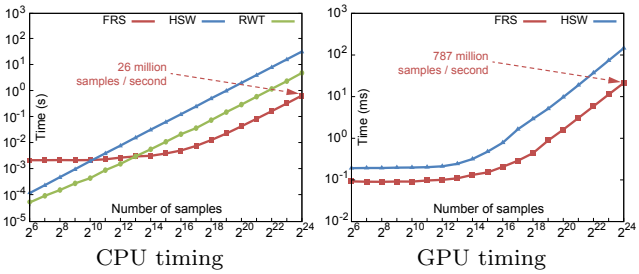

Fig. 7: Execution time of FRS on the CPU (left) in seconds and on the GPU (right) in milliseconds compared to HSW with Halton samples and RWT.

The runtime of FRS mainly depends on the window size $\mathbf{s}_W$ rather than the number of samples, which in turn is directly related to the sparsity σ. In order to prevent sampling from being too regular, σ was set to a reasonable value of 8 for the performance tests. For the importance maps of size $\mathbf{s}_\mathcal{I} = (128, 128)$, this results in a minimum window size $\mathbf{s}_W = (1024, 1024)$. Following Equation 7, a maximum of 2^{14} samples can be generated for this window size, resulting in an almost constant runtime of the FRS implementations up to this number of samples.

In contrast to the CPU implementation, the computational effort of HSW on the GPU also seems to be constant for smaller values of n and then transitions to the linear growth as observed above. Thus, FRS is faster for all sample counts up to a factor of 6.8.

7 Applications

In general, Forced Random Sampling (FRS) can be used for any sampling application based on a discretized two-dimensional importance function with a known average and maximum. This includes stippling for non-photorealistic rendering and automatic object placement. In the context of photorealistic rendering, a major objective is the evaluation of the rendering equa-

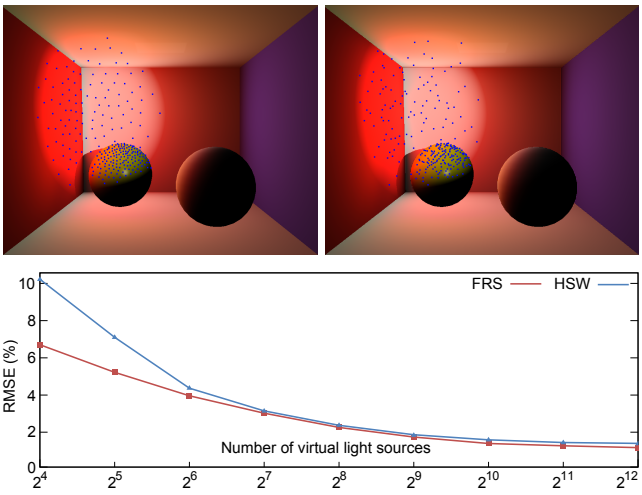

Fig. 8: Top: Placement of 256 virtual light sources (blue) for Reflective Shadow Mapping with FRS (left) and HSW (right). Bottom: Root-mean-square error (RMSE) of results with both sample distributions compared to a ground truth.

tion [15], which models the transport of light. Due to the complex nature of the equation, the evaluation generally requires the approximation of a significant number of multi-dimensional integrals. Real-time algorithms are based on simplified models that reduce the multi-dimensional problem to one with a more manageable number of dimensions. We demonstrate the merits of FRS based on such algorithms since the respective results are particularly descriptive.

Reflective Shadow Mapping. We have integrated FRS in an implementation of Reflective Shadow Mapping (RSM) [35]. RSM is a real-time rendering method, where indirect illumination is approximated by positioning virtual light sources according to an importance-guided point distribution. Hierarchical Sample Warping (HSW) with Halton points as input samples has been previously used to generate the positions. Compared to HSW, FRS provides a more spatially uniform distribution, as seen in Figure 8 (top), while maintaining real-time capability. As a result, the indirect illumination converges faster to a ground truth obtained offline by relaxation dart throwing with 16 384 virtual light sources. This is illustrated in Figure 8 (bottom) with the root-mean-square error (RMSE) of the rendered images compared to the ground truth, each averaged over ten different sample sets. For all typical numbers of virtual light sources, FRS exhibits a lower deviation.

Photon Tracing. Another application of FRS is the generation of well-distributed photon directions, especially for photometric light sources with non-uniform

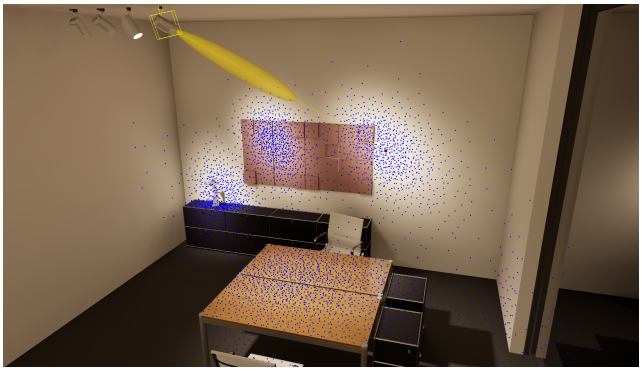

Fig. 9: FRS integrated in photon tracing for interactive light planning. The blue points show the first hits of 5000 photons emitted from the photometric light sources, the yellow lobe visualizes the emission profile of the selected light source.

emission profiles. We have integrated our method into a photon tracer for interactive light planning. The resulting photon distribution shown in Figure 9 exhibits no regularity or clumping.

8 Conclusion and Future Work

We have presented a novel algorithm for generating a finite number of importance-guided samples in real time that exhibit distinct blue-noise characteristics. Our algorithm has been shown to outperform the fastest existing method on the GPU and – if generating more than about 2^{13} samples – also on the CPU. Our method generates high-quality samples that are comparable to those generated by much slower reference methods. This is especially notable, since the second fastest method on the CPU, Recursive Wang Tiles (RWT), has been shown (in this paper and by Lagae and Dutré [19]) to generate samples of significantly lower quality.

Currently, our method cannot be easily extended to more than two dimensions since the creation of higher-dimensional dither matrices is prohibitively expensive (even though it is only a one-time precomputation step). Another challenge is to adapt our method to a curved sampling domain since the force-field matrix generation is based on the Euclidean distance (see Equation 1). We will direct our research effort into improved generation methods in order to overcome these limitations.

Acknowledgements The competence center VRVis is funded by BMVIT, BMWFW, and City of Vienna (ZIT) within the scope of COMET Competence Centers for Excellent Technologies. The program COMET is managed by FFG. Hiroyuki Sakai is partly supported by the Austrian Science Fund (FWF), project no. P 27974. We thank Christoph Weinzierl-Heigl for providing us access to his implementation of Reflective Shadow Mapping.

References

1. Abe, Y.: Digital halftoning with optimized dither array. The 2001 IEEE International Symposium on Circuits and Systems **2**, 517–520 (2001)
2. Ahmed, A.G.M., Perrier, H., Coeurjolly, D., Ostromoukhov, V., Guo, J., Yan, D.M., Huang, H., Deussen, O.: Low-discrepancy Blue Noise Sampling. ACM Transactions on Graphics (Proc. of SIGGRAPH Asia 2016) **35**(6), 247:1–247:13 (2016)
3. Balzer, M., Schlömer, T., Deussen, O.: Capacity-Constrained Point Distributions: A Variant of Lloyd's Method. ACM Transactions on Graphics (Proc. of SIGGRAPH 2009) **28**(3), 86:1–86:8 (2009)
4. Bayer, B.E.: An Optimum Method for Two-Level Rendition of Continuous-Tone Pictures. IEEE International Conference on Communication, Conference Record pp. (26–11)–(26–15) (1973)
5. Bowers, J., Wang, R., Wei, L.Y., Maletz, D.: Parallel Poisson Disk Sampling with Spectrum Analysis on Surfaces. ACM Transactions on Graphics (SIGGRAPH Asia 2010 Papers) **29**(6), 166:1–166:10 (2010)
6. Clarberg, P., Akenine-Möller, T.: Practical Product Importance Sampling for Direct Illumination. Computer Graphics Forum (Proc. of Eurographics 2008) **27**(2), 681–690 (2008)
7. Clarberg, P., Jarosz, W., Akenine-Möller, T., Jensen, H.W.: Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. ACM Transactions on Graphics (Proc. of SIGGRAPH 2005) **24**(3), 1166–1175 (2005)
8. Cline, D., Egbert, P.K., Talbot, J.F., Cardon, D.L.: Two Stage Importance Sampling for Direct Lighting. In: Proc. of the 17th Eurographics Conference on Rendering Techniques, pp. 103–113. Eurographics Association (2006)
9. Cook, R.L.: Stochastic Sampling in Computer Graphics. ACM Transactions on Graphics **5**(1), 51–72 (1986)
10. Georgiev, I., Fajardo, M.: Blue-noise Dithered Sampling. In: ACM SIGGRAPH 2016 Talks, pp. 35:1–35:1. ACM (2016)
11. Gjøl, M., Svendsen, M.: High Fidelity, Low Complexity – The Rendering of INSIDE. Game Developers Conference Europe 2016 (2016)
12. de Goes, F., Breeden, K., Ostromoukhov, V., Desbrun, M.: Blue Noise through Optimal Transport. ACM Transactions on Graphics (Proc. of SIGGRAPH Asia 2012) **31**(6), 171:1–171:11 (2012)
13. Hiller, S., Deussen, O., Keller, A.: Tiled Blue Noise Samples. In: Proc. of the Vision Modeling and Visualization Conference 2001, pp. 265–272. Aka GmbH (2001)
14. Huang, H.d., Chen, Y., Tong, X., Wang, W.c.: Incremental Wavelet Importance Sampling for Direct Illumination. In: Proc. of the 2007 ACM Symposium on Virtual Reality Software and Technology, pp. 149–152. ACM (2007)
15. Kajiya, J.T.: The Rendering Equation. In: Proc. of the 13th Annual Conference on Computer Graphics and Interactive Techniques, pp. 143–150. ACM (1986)
16. Kopf, J., Cohen-Or, D., Deussen, O., Lischinski, D.: Recursive Wang Tiles for Real-Time Blue Noise. ACM Transactions on Graphics (Proc. of SIGGRAPH 2006) **25**(3), 509–518 (2006)
17. Lagae, A., Dutré, P.: A Procedural Object Distribution Function. ACM Transactions on Graphics **24**(4), 1442–1461 (2005)
18. Lagae, A., Dutré, P.: An Alternative for Wang Tiles: Colored Edges Versus Colored Corners. ACM Transactions on Graphics **25**(4), 1442–1459 (2006)
19. Lagae, A., Dutré, P.: A Comparison of Methods for Generating Poisson Disk Distributions. Computer Graphics Forum **27**(1), 114–129 (2008)
20. McCool, M., Fiume, E.: Hierarchical Poisson Disk Sampling Distributions. In: Proc. of the Conference on Graphics Interface '92, pp. 94–105. Morgan Kaufmann Publ. Inc. (1992)
21. Mitsa, T., Parker, K.J.: Digital halftoning technique using a blue-noise mask. Journal of the Optical Society of America A **9**(11), 1920–1929 (1992)
22. Newbern, J.L., Bove, Jr., V.M.: Generation of Blue Noise Arrays by Genetic Algorithm. Proc. of SPIE Human Vision and Electronic Imaging II **3016**, 441–450 (1997)
23. Ostromoukhov, V.: Sampling with Polyominoes. ACM Transactions on Graphics (Proc. of SIGGRAPH 2007) **26**(3), 78–1–78–6 (2007)
24. Ostromoukhov, V., Donohue, C., Jodoin, P.M.: Fast Hierarchical Importance Sampling with Blue Noise Properties. ACM Transactions on Graphics (Proc. of SIGGRAPH 2004) **23**(3), 488–495 (2004)
25. Purgathofer, W., Tobler, R.F., Geiler, M.: Forced Random Dithering: Improved Threshold Matrices for Ordered Dithering. In: Proc. of the 1st IEEE International Conference on Image Processing, vol. 2, pp. 1032–1035. IEEE (1994)
26. Schmaltz, C., Gwosdek, P., Bruhn, A., Weickert, J.: Electrostatic Halftoning. Computer Graphics Forum **29**(8), 2313–2327 (2010)
27. Shade, J., Cohen, M.F., Mitchell, D.P.: Tiling Layered Depth Images. Tech. Rep. 02-12-07, Univ. of Washington, Dep. of Computer Science and Engineering (2002)
28. Ulichney, R.: Digital Halftoning. MIT Press (1987)
29. Ulichney, R.: Dithering with blue noise. Proc. of the IEEE **76**(1), 56–79 (1988)
30. Ulichney, R.: The void-and-cluster method for dither array generation. Proc. of SPIE, Human Vision, Visual Processing, and Digital Display IV **1913**, 332–343 (1993)
31. Wachtel, F., Pilleboue, A., Coeurjolly, D., Breeden, K., Singh, G., Cathelin, G., de Goes, F., Desbrun, M., Ostromoukhov, V.: Fast Tile-based Adaptive Sampling with User-specified Fourier Spectra. ACM Transactions on Graphics (Proc. of SIGGRAPH 2014 **33**(4), 56:1–56:11 (2014)
32. Wei, L.Y.: Parallel Poisson Disk Sampling. ACM Transactions on Graphics (Proc. of SIGGRAPH 2008) **27**(3), 20:1–20:9 (2008)
33. Wei, L.Y.: Public SVN repository of Li-Yi Wei, Revision 13 (2011). https://github.com/1iyiwei/noise/ (accessed February 19, 2017)
34. Wei, L.Y., Wang, R.: Differential Domain Analysis for Non-Uniform Sampling. ACM Transactions on Graphics (Proc. of SIGGRAPH 2011) **30**(4), 50:1–50:10 (2011)
35. Weinzierl-Heigl, C.: Efficient VAL-based Real-Time Global Illumination. In: Proc. of the 17th Central European Seminar on Computer Graphics (2013)
36. Xiang, Y., Xin, S.Q., Sun, Q., He, Y.: Parallel and Accurate Poisson Disk Sampling on Arbitrary Surfaces. In: SIGGRAPH Asia 2011 Sketches, pp. 18:1–18:2. ACM (2011)
37. Yellott Jr., J.I.: Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina. Science **221**, 382–385 (1983)
38. Yuksel, C.: Sample Elimination for Generating Poisson Disk Sample Sets. Computer Graphics Forum (Proc. of Eurographics 2015) **34**(2), 25–32 (2015)