Gaussian Blue Noise

ABDALLA G. M. AHMED, KAUST, KSA
JING REN, KAUST, KSA and ETH Zurich, Switzerland
PETER WONKA, KAUST, KSA

Fig. 1. Point distributions (1K sets) and power spectra (1K realizations) of different blue-noise optimization techniques including: (a) BNOT [de Goes et al. 2012], (b) KDM [Fattal 2011], (c) VnC [Ulichney 1993], (d) FPO [Schlömer et al. 2011], (e) BlueNets [Ahmed and Wonka 2021], (f) SOT [Paulin et al. 2020], and (g) GBN (Ours). Among the various approaches for producing point distributions with blue noise spectrum, we argue for an optimization framework using Gaussian kernels. We show that with a wise selection of optimization parameters, this approach attains unprecedented quality, provably surpassing the current state of the art attained by the optimal transport (BNOT) approach. Further, we show that our algorithm scales smoothly and feasibly to high dimensions [Kuipers and Niederreiter 1974], but even in the 2D plane that characterizes graphical images, it was established early in the works of Dippé and Wold [1985] and Cook [1986] that regular sampling leads to excessive aliasing artifacts in rendering, while a Poisson (random) distribution of samples leads to excessive amounts of noise. Hence, Poisson-disk sampling emerged as a reasonable compromise, where a minimal spacing, known as the conflict or Poisson-disk radius $r_1$, is required between otherwise random samples. Jittering of the regular grid, also known as stratification, was suggested as a cheaper alternative. In the context of halftoning, Ulichney [1987; 1998] concluded the superiority of the more-or-less same distribution, for which he coined the name blue noise (BN) to describe a distribution of samples whose frequency power spectrum is characterised by a low-energy low-frequency band, a sharp transition towards a small peak (corresponding to $r_1$), followed by a flat spectrum in higher frequencies.

The loose definition of blue noise and the lack of a deterministic mean for generating it lead to a substantial amount of literature devoted to describing algorithms for approximating the blue-noise distribution, as well as means for evaluating its quality, as we will detail in the following Section 2. The very nature of blue noise, however, remained unclear, lacking a mathematical theory to characterize this important distribution.

Among the wide variety of blue noise generation algorithms, we may broadly identify two distinct approaches: cellular and kernel-based. Cellular methods are based on a divide-and-conquer principle that assigns one-and-only sample point to represent a partition of the domain, while kernel-based techniques try to maintain a uniform density of kernels placed at the sample point. For a long time, cellular techniques, flag-shipped by BNOT [de Goes et al. 2012], were considered the reference methods, while kernel-based techniques were thought of as just an alternative mean of achieving
comparable results. Indeed, in addition to its noticeable higher quality, BNOT also offers a plausible theoretical justification rooted in least-square fitting, in contrast to the heuristic motivation in known kernel-based methods.

Quite recently, Ahmed and Wonka [2021] presented a formulation of a loss function for Gaussian-kernel-based blue noise, and used it for optimization in a specific discretized context, namely dyadic nets. In this paper, we extend this approach to continuous domains and high-dimensional spaces, and show that, with the appropriate design choices, kernel-based methods can outperform cellular methods in terms of scalability in dimensions, coding complexity, and even quality. The key insight is that cellular methods work by modeling interactions between points and their immediate neighbors, whereas kernel-based methods enable modeling a longer range of interactions.

1.1 Contributions
In summary, our main contributions are:

1. We derive an analytical formulation of the power spectrum of Gaussian-kernel-based blue noise, demonstrate that it is provably superior to BNOT, and show empirical results to support the claim.

2. Starting from the loss function in [Ahmed and Wonka 2021], and following a series of objective design choices, we present a robust algorithm for our Gaussian blue noise (GBN) optimization over a uniform toroidal domain of any number of dimensions, and an adaptation for bounded domains.

3. We present a new algorithm for adaptive sampling that improves the current state of the art, along with a reconstruction algorithm.

The paper is organized as follows. We start by reviewing related literature in Section 2. In Section 3 we discuss the theoretical foundation of GBN, and in Section 4 and Section 5 we discuss practical details of how to realize it, and present two efficient algorithms for uniform and adaptive blue-noise optimization. We then showcase actual results in Section 6, and compare to state-of-the-art methods, before making concluding remarks in Section 7.

2 RELATED WORK
Thanks to the special nature of sampling in CG discussed in the introduction, blue noise is recognized as an important local product of the graphics community, and received a lot of attention evident in the large bulk of related literature. In the following subsections we briefly outline the most related work to this paper.

2.1 Generating Blue Noise
Most of the literature on blue noise is devoted to presenting generation and optimization algorithms of blue-noise point sets. There are algorithms that use dart throwing [Cook 1986; Dippé and Wold 1985; Ebeida et al. 2011; Gamito and Maddock 2009; McCool and Fiume 1992; Wei 2008; Yan and Wonka 2013; Yuksel 2015] or advancing front [Bridson 2007; Dunbar and Humphreys 2006; Jones 2006; Mitchell et al. 2018] techniques for the direct non-iterative generation of sample points that maintain a Poisson-disk property, and hence bear a blue noise spectrum. While some of these algorithms are very fast, the quality of the point distributions is relatively poor.

There are also general spectral tailoring algorithms [Heck et al. 2013; Kaikkhura et al. 2016; Öztireli and Gross 2012; Zhou et al. 2012; Öztireli 2020] that may be used for the production of blue noise, but these algorithms are typically costly, and are mainly of theoretical importance.

Our concern in this paper is on optimization algorithms aimed at producing high-quality blue noise point sets. There is a wide range of such algorithms, but they can be grouped into two distinct categories. In one category we have cellular methods that associate each point with a partition of the domain: a Voronoi or power cell, and optimize the local neighborhood of the points [Ahmed et al. 2017a; Balzer et al. 2009; Chen et al. 2012; de Goes et al. 2012; Deussen et al. 2000; McCool and Fiume 1992; Ostromoukhov 1993; Paulin et al. 2020; Schläomer et al. 2011; Second 2002; Xin et al. 2016a; Xu et al. 2011]. Then we have kernel-based methods that use a decaying kernel to model the influence of sample points [Ahmed and Wonka 2021; Fattal 2011; Hanson 2003, 2005; Jiang et al. 2015; Öztireli et al. 2010; Schmalz et al. 2010; Ulchney 1993]. While most of these methods were developed heuristically, they are closely related to kernel density estimation [Terrell and Scott 1992a]. More recently, Leimkühler et al. [2019] presented a machine-learned algorithm for blue-noise optimization that seems closely related to our work, but the main difference is that the kernels are learned by the optimizer, while we explicitly work with a Gaussian kernel.

Since its introduction by de Goes et al. [2012], blue noise through optimal transport (BNOT) — a cellular method — has widely been accepted as the reference algorithm for best-quality blue noise, while kernel-based methods are thought of as secondary alternatives. Subsequent research tried to find faster implementations, e.g., [Xin et al. 2016], port it to higher dimensions, e.g., [Paulin et al. 2020], or emulate it with faster methods, e.g., [Jiang et al. 2015], but we are not aware of published works that claimed improved quality. The fact that the algorithm could not be improved upon, until now, is a testament to the amazing quality of BNOT. In this paper we demonstrate that, with informed choices of optimization parameters and settings, kernel-based methods can actually outperform BNOT.

2.2 Distributing Blue Noise Samples
The high cost of generating blue noise samples makes it not suitable for direct generation on demand. Instead, blue noise samples are typically generated offline and stored in lookup tables [Glassner 1995]. Beyond the direct storage of a fixed list of samples, different techniques were proposed to distribute arbitrary numbers of blue-noise samples, including Wang tiling techniques [Cohen et al. 2003; Kopf et al. 2006; Lagae and Dutré 2006], self-similar tiling [Ahmed 2019; Ahmed et al. 2017b; Lagae and Dutré 2006b; Ostromoukhov 2007; Ostromoukhov et al. 2004; Wachtel et al. 2014], and AA Patterns [Ahmed et al. 2015].

While our focus in this paper is on generating blue noise, a good understanding of how the samples are eventually distributed is important in favoring a generation technique. Notably, the kernel-based methodology we are advocating is more versatile for optimizing the samples in tiling techniques thanks to the localized definition of the energy target.
2.3 Evaluation
The third line of research on blue noise is devoted to developing measures for evaluating and comparing blue noise distributions. The primary tool for assessing blue noise is the frequency power spectrum of the point process that generates it, typically estimated empirically by averaging the periodograms of a reasonable number of realizations. This tool was introduced by Ulrich [1987; 1988], who also defined two radially averaged plots for summaries: radial power, obtained by averaging the frequency power over rings of different radii, and anisotropy, which measures the variance over each frequency ring, and detects directional bias and regularity. Schlöemer et al. [2011] developed a tool, PSA, that standardizes the computation and presentation of frequency spectra. We extended that tool to work with high dimensions, so all the plots in this paper use exact evaluation of the periodograms.

Many alternative and additional tools were developed subsequently to assess the quality of blue noise sets, including auto-correlation plots [Wei and Wang 2011] for anisotropic and adaptive point processes, and their 1D profiles for stationary processes [Heck et al. 2015; Öztireli and Gross 2012]. In addition to these plots, there are also scalar means to assess blue noise point sets. Conflict radius [Lagae and Dutré 2008] prevailed before the advent of capacity-constrained cellularization by Balzer et al. [2009], Heck et al. [2013] then introduced three additional measures: effective Nyquist rate $\nu_{\text{eff}}$, oscillation $\Omega$, and bond orientation order $Q_b$, to assess the quality of blue noise for reconstruction.

More recently, Ahmed and Wonka [2021] derived an energy term for evaluating blue noise based on a Gaussian kernel that filters it. This is the most relevant work to our current work, and we take it further to characterize the frequency spectrum of kernel-based blue noise processes, and to study its realizability in 2D and its extensibility to higher dimensions. We actually show that blue noise is realizable even in one dimension, and we discuss the effect of the number of dimensions on the generated blue noise.

2.4 Integration with Blue Noise
While blue noise was originally intended for reconstructing a visual signal from samples to minimize coherent aliasing, there has been a long-standing curious question about its utility for Monte Carlo integration. Hanson [2003] empirically demonstrated the advantage of blue noise sets over Halton low-discrepancy distribution. The seminal report by Durand [2011] on the frequency analysis of numerical integration established a theoretical base, followed up by Ramamoorthi et al. [2012], Subr and Kautz [2013], and Pilleboue et al. [2015]. Öztireli [2016] took a different path by studying the same problem in the spatial domain, using auto-correlation. More recently, Singh et al. [2019] reported various aspects of the subject. In this paper we show analytical cues and present empirical data that confirms the suitability of blue noise for numerical integration, especially in higher dimensions, where it is less hit by the curse of dimensionality than the competing alternatives.

3 GAUSSIAN BLUE NOISE
The idea of using a Gaussian kernel for blue noise optimization dates back to Ulrichney’s void-and-cluster algorithm [1993], and was re-introduced at least three times thereafter by Hanson [2003; 2005], Öztireli et al. [2010], and Fattal [2011], with a different motivation each time. The idea is to place Gaussian kernels at the points, and optimize the placement of the points so that a uniform density is maintained everywhere. The concept was mostly developed heuristically, but is closely related to kernel density estimation, as can be seen by comparing [Fattal 2011, Eq. (1)] to [Terrell and Scott 1992, Eq. (1.7)]. Ahmed and Wonka [2021] have recently derived an analytical formulation,

$$\text{Var}(A(X)) = \frac{\pi \sigma^2}{2N} \sum_{k=1}^{N} \sum_{j=1}^{N} \exp \left( -\frac{||x_k - x_j||^2}{4\sigma^2} \right) - \left( \frac{2\pi \sigma^2}{4} \right)^2,$$

(1)

that underlies these methods, defined as the variance of a sum

$$\text{Var}(A(x)) = E\left(\hat{A}^2(x)\right) - \left( E(A(x)) \right)^2$$

(2)

of a sum

$$A(X) = g(x) * \delta(X) = \sum_{k=1}^{N} \exp \left( -\frac{||x - x_k||^2}{2\sigma^2} \right)$$

(3)

of Gaussian kernels

$$g(x) = \exp \left( -\frac{||x||^2}{2\sigma^2} \right)$$

(4)

placed at a set

$$X = \{ x_k \}_{k=1}^{N}$$

(5)

of sample points.

In this section, we derive an analytical formula for the power spectrum of a point distribution that minimizes this variance to serve as a theoretical reference for all Gaussian-kernel-based algorithms, and we discuss its inherent superiority over cellular methods.

3.1 Frequency Analysis
We start by analyzing the variance in Eq. (2) directly in the frequency domain. As already established by Durand [2011, Eq. (15)], “the variance is the integral of the power spectrum minus the DC,”

$$\text{Var}(A(X)) = \int |\hat{A}(\omega)|^2 \, d\omega - |\hat{A}(0)|^2,$$

(6)

which follows directly from Parseval’s theorem. The right-side terms of Eq. (6) correspond directly to their counterparts in Eq. (1). The DC term is invariant, and minimizing the variance in Eq. (1) is therefore equivalent to attenuating the whole power spectrum except for the DC, which is intuitive, since our goal is to come close to a constant. An idealized optimization process would not have spectral bias, and would therefore bring the power spectrum of the filtered point set below some level

$$|\hat{A}(\omega)| \leq \epsilon$$

(7)

that manifests as random bumping on the surface of the filtered signal $A(x)$.

We further analyze this power spectrum. Applying the kernels in the spatial domain is a convolution process that translates into a multiplication in the frequency domain; hence

$$|\hat{A}(\omega)| = |\hat{g}(\omega)| \cdot |\mathcal{F}(\omega)|,$$

(8)
where \( \hat{g} \) is the Fourier transform of the kernel \( g \), and

\[
\mathcal{F} (\omega) = \sum_{k=1}^{N} \exp(-i\omega \cdot x_k)
\]

is the frequency spectrum of the point set \( X \). This second factor in Eq. (8) is, by definition, the power spectrum

\[
P(\omega) = \mathcal{F}(\omega) \cdot \mathcal{F}^*(\omega)
\]

of the point set. Combining Eqs. (7, 8, 10) gives

\[
P(\omega) \leq \epsilon|\hat{g}|^{-2}(\omega).
\]

This equation characterizes the power spectrum of kernel-based methods, and applies to any square-integrable kernel used to filter the point set. The only assumption is that the optimization process is not frequency biased; otherwise the constant \( \epsilon \) would have to be replaced by a frequency profile of the optimization process.

For the case of a Gaussian kernel \( g(x) \) in Eq. (4), the Fourier transform is another Gaussian:

\[
\hat{g}(\omega) = \exp\left(-\frac{\sigma^2}{2} \|\omega\|^2\right).
\]

Substituting Eq. (12) in Eq. (11) gives

\[
P(\omega) \leq \epsilon e^{\sigma^2 \|\omega\|^2}.
\]

### 3.2 Feasibility

This Eq. (13) characterizes the frequency spectrum of an idealized Gaussian-kernel-based blue noise when the filtered set is close to a constant, which is the target of minimizing the variance in Eq. (1). As can be seen in Fig. 1(h), such a frequency spectral profile is actually realizable, and is faithfully attained by Gaussian-based methods like KDM [Fattal 2011] and VnC [Ulichney 1993]. It is interesting that FPO [Schlömer et al. 2011] bears a similar profile and the same exponent as VnC, which possibly comes from the fact that they both search for farthest points. Cellular methods, in contrast, follow a fundamentally different polynomial power profile, which is already noted in the literature [Pilleboue et al. 2015]. BlueNets [Ahmed and Wonka 2021] exhibit a mixed behavior, reflecting the combination between their stratified nature and the kernel-based optimization.

A very important note about the Gaussian-based spectrum is that it is not only good for reconstruction, but also for numerical integration. Indeed, Pilleboue et al. [2015] characterized the variance of numerical integration of (semi-)stochastic point sets by the growth-rate of the frequency power spectrum of the point process, and they favored a higher polynomial degree for the curve of the spectrum. The idealized blue noise model in Eq. (13), however, has an exponential growth, or even better, quadruplexponential, which means that, starting from the same noise floor \( \epsilon \) at the lowest frequency, it will always stay for a wide range below the polynomially-shaped power spectra, as can be visualized by comparing the linearly-sloped spectra in Fig. 1(h) with the curved ones.

This intrinsic advantage of Gaussian-based blue noise, however, remained undiscovered in the past possibly because the known algorithms all stop at a very shallow noise floor \( \epsilon \), as seen in Fig. 1(e), either due to an inherent limit of the algorithm, e.g. FPO, or a numerical limit, e.g. VnC, or an arbitrary stop to avoid developing regular patterns. BNOT, in contrast, is able to attain a very low noise floor without suffering quality issues. We recall, however, that BNOT was proceeded by Lloyd’s algorithm [Lloyd 1982; McCool and Fiume 1992], which suffered from similar problems to the mentioned ones in kernel-based methods. It was not until Balzer et al. [2009] introduced the capacity constraint, 17 years later, and de Goes et al. developed a theory of it, that the cellular-based approach was able to unlock its full potential and reach the BNOT quality. Analogously, the preceding analysis furnishes as theoretical basis to develop an algorithm that is able to unlock the full potential of kernel-based optimization and reach its extent, and in the following two sections we go through the many practical aspects that need to be taken into consideration. Through these objective design choices we were able to reach a 10 orders of magnitudes lower noise level than the state-of-the-art kernel-based algorithms, as seen in Fig. 1(h), and even surpass BNOT itself by two orders of magnitudes. The actual improvement over BNOT is all the highlighted volume in Fig. 1(h), which manifests visually as less noise in the the point plots in Fig. 1(a, g) and later Figs. (10, 13, 14), or as a reduction in the numerical integration variance, as will be demonstrated in Section 6.2.

It is worth noting that Eq. (13), defined in the frequency domain, is not used as an objective function in our optimization, but guides the choices, and also provides the possibility to verify if a particular algorithm or optimization behaves as desired. Instead, we use the corresponding spatial-domain Eq. (1), and can empirically verify that the result of our algorithm yields a power spectrum that has the desired shape. Fig. 2 schematically summarizes the optimization process. This (idealized) abstract model is meant to highlight important aspects. The essence of filtering is to mask the high-frequency range from the optimization process. The optimization strives to attenuate the energy in the visible low-frequency range, and if it achieves a tight fit to its target level \( \epsilon \) then the power spectrum of the point set would automatically attain the profile of Eq. (13) in this range. When the kernels are removed, the masked frequency zone should ideally be restored into their original flat spectrum, but in practice some frequency oscillation occurs harmonically as a side effect.

\[ \text{White} \rightarrow \text{Filter} \rightarrow \text{Gaussian} \rightarrow \text{Optimize} \rightarrow \text{Disc} \rightarrow \text{Unfilter} \rightarrow \text{Blue} \]
effect of the low-frequency optimization. Specifically, the energy excavated from the low frequency region of interest can not vanish, but manifests as a primary peak followed by decaying harmonic oscillation.

3.3 Optimization Method

Deriving a gradient-descent minimization algorithm from Eq. (1) is straightforward. We first write the Gaussian in a standardized form [Ahmed and Wonka 2021, Eq. (36)]:

$$\sigma^2 \leftarrow 2\sigma^2 \text{Filtering}$$

(14)

The second (DC) term is invariant, so our target energy becomes

$$E(X) = \frac{\pi\sigma^2}{2N} \sum_{k=1}^{N} \sum_{l \neq k} \exp \left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right).$$

(15)

We can then extract a loss function

$$E(x_k) = \frac{\pi\sigma^2}{N} \sum_{l \neq k} \exp \left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right)$$

(16)

for each point $x_k$ such that

$$\arg\min_x \text{Var} (A(X)) = \arg\min_x \sum_{k=1}^{N} E(x_k).$$

(17)

Note that each point counts twice in Eq. (15): once as $x_k$ and once as $x_l$, hence the factor of two in Eq. (16). Finally, we compute the gradient

$$\nabla E(x_k) = -\frac{\pi}{N} \sum_{l \neq k} \exp \left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right) (x_k - x_l)$$

(18)

of the loss function. This was already derived by Öztireli [2010] from a different approach. Note that the energy is halved between each point and the remaining set.

The gradient formulation in Eq. (18) suggests a quadratic complexity, and it is tempting to look for alternative implementations that would accelerate convergence, or otherwise reduce computational complexity. Unfortunately, a Hessian matrix of the energy function is inherently singular, ruling out second order techniques such as Newton’s method. Another choice is discretization, as in [Ahmed and Wonka 2021; Fattal 2011; Hanson 2003; Ulichney 1993], that differ in design parameters. In this section we discuss important design choices, and make informed decisions in the light of the preceding discussion about the target frequency spectrum.

4.1 Choice of Kernel

Our focus in this paper is on Gaussian kernels, but it is worthwhile having a brief discussion, in the light of our theoretical model, about the merits of this specific kernel among alternatives such as the inverse distance [Schmaltz et al. 2010] or SPH [Jiang et al. 2015]. A great advantage of the Gaussian kernel is that it has a well-defined parametric frequency transform, and this transform happens to be quite favorable in having a very fast decaying rate towards high-frequencies, making it possible to band-limit the optimization to the more important low-frequency range, as will be discussed in the following Section 4.2. In contrast, an inverse distance kernel, for example, is not even square-integrable to fit the theory, and any shaping to escape the singularity adds complexity to the frequency profile. We are not claiming optimality, though; only that the Gaussian kernel leads to an excellent blue noise profile.

Besides this analytical merit, there is another very important unique advantage of the Gaussian kernel, that it is separable in dimensions, making the optimization scales with dimension in a feasible manner.

4.2 Kernel Parameter

The choice of $\sigma$ is an essential design decision. Values used in previous methods include 1.5 [Fattal 2011; Ulichney 1993], ≈1 [Öztireli et al. 2010], and 0.5 [Ahmed and Wonka 2021]. These values were chosen experimentally for the different algorithms, coordinated with other parameters.

Only considering the spatial domain can lead to wrong choices of $\sigma$. Specifically, it is tempting to set a small $\sigma$ which will concentrate the energy in the nearby neighborhood, leading to a visually quick convergence towards a large Poisson-disk radius in the spatial domain, as demonstrated in Fig. 3(a). However, a more thorough investigation of the radial power, i.e., in the spectral domain, would reveal that the spectra of these point sets are quite shallow.

The key insight to guide the choice of a kernel width $\sigma$ is that "you get what you pay for". The point locations are the only degree
of freedom we have for optimization; that is, 2N real parameters for a set of N points in 2D, for example. Thus, we have a limited budget of information bits to use for shaping the spectrum. A rough measure for the attenuated energy is to count the number of zeros to the right of the fractional point in the frequency range (0, 1), which is proportional to the area below 1 in a log-linear scale of the spectrum. In Fig. 4(a) we show the spectra of our obtained GBN with different choices of $\sigma$. We note that for a small $\sigma$ the frequency spectrum is distorted significantly from the spectrum of an idealized blue noise model, having residual low-frequency content and too-much oscillation in the high-frequency range. A possible explanation is that a narrow kernel transforms into a wide one in the frequency domain, assigning substantial weights to higher frequencies where our algorithm tries to optimize worthlessly. On the other hand, optimizing for a large $\sigma$ gives too much improvement in the low-frequency range, at the cost of narrowing the low-energy band. We found that $\sigma = 1$, relative to a regular-grid $N^{-1/d}$ spacing between the points, gives an excellent trade-off, so we recommend it. With $\sigma = 1$, the energy weight of a regular grid spacing drops to $e^{-(2\pi)^2} \approx 7.2 \times 10^{-18}$ relative to the DC weight. With a proper implementation, this guards the optimization from seeking regular or triangular structures, and leads to an elegant formula

$$P(\omega) = e^{-|\omega|^2}$$

for our reference blue-noise power-spectrum profile demonstrated in Fig. 1(g, h). We restate that this formula is only for the low-frequency band that is targeted by the optimization.

We can also realize high-dimensional (e.g., 8D) Gaussian Blue Noise with comparable noise floor to 2D GBN, as illustrated in Fig. 4(b). An interesting, unexpected result we obtained is that higher dimensions actually converge faster towards the target noise floor, or, alternatively, attain even lower noise levels. Our first guess is that this is due to the availability of more degrees of freedom with dimensions (i.e., more budget for information bits), but this needs further investigation.

4.3 Kernel Support

Similar to the misleading spatial-domain intuition to use a small kernel width $\sigma$, as discussed in Section 4.2, it is also tempting to consider only a local neighborhood for optimization: (i) it has linear complexity for optimization compared to the quadratic complexity when considering all the sample points, and (ii) it seems to quickly converge towards a large Poisson-disk radius in the spatial domain [Fattal 2011; Öztireli et al. 2010]. However, in fact, considering only a local neighborhood is detrimental to the optimization, since it brings harmful distortion to the frequency structure of the energy kernel. Specifically, only considering a local neighborhood is equivalent to truncating the kernel $h$, which can be formulated as multiplying $h$ with a box function. In the frequency domain, multiplying $h$ with a box function transforms into a convolution between $\hat{h}$ and a sinc function. Note that a sinc function is known to be slowly decaying, which suggests that truncating the kernel would expand its frequency support. As illustrated in Fig. 5, the smaller the range of the truncation is, the more are the higher frequencies leaking in, which makes most of the common algorithms tend to settle at patches of regular structures. Fig. 6 shows actual results obtained by truncated kernels, revealing different kinds of distortions. The $\sigma$ support effectively addresses the first ring of Voronoi neighbors, leading to a similar result to Centroidal Voronoi Tessellation (CVT).

To avoid these distortions, we therefore use full kernel support by considering all the point pairs when evaluating the energy term in Eq. (16). This choice of global optimization is arguably the second most important design choice we had to make, following the choice of $\sigma$. It leads to quadratic time complexity, but gives more accurate and much higher-quality results. It is also the gateway to high-dimensional blue noise, as we will see in Section 5.2.

There is a limit, though, to the effective kernel support by the numerical precision of the machine, and the mutual energy is effectively zero after $6/9-\sigma$ steps with float/double data types. A practical implementation may take advantage of this, especially with a large number of points, but for the point counts used in this paper the saving would not offset the extra coding complexity. Note that these neighborhoods are still considerably wider than the ranges used in common algorithms, e.g., [Fattal 2011; Öztireli et al. 2010].

4.4 Convergence

As mentioned in Section 3.3, the Hessian of the energy function in Eq. (15) is inherently singular, which makes any gradient descent algorithm eventually linear, hence non-converging to a complete rest: the only converging algorithms we are aware of are those applied to discrete domains, e.g., [Ahmed and Wonka 2021; Hanson 2005; Ulichney 1993]. An intuitive explanation to this is that each point should end up at a trough of the Gaussian field induced by
We note that, apart from the time step, the optimization process is linearly at the beginning, and starts to slow down from around 500 iterations, when the spectrum takes its designated shape. We do not claim any optimality of our algorithm, and better convergence rates might be attainable.

In our experiments we used a parallel implementation on GPU. We note that, apart from the time step, the optimization process is deterministic, hence the resulting blue noise depends only on the initial distribution. Looking at the convergence behavior, we note that some of the previous methods may not have used a sufficient number of iterations. For example, Öztireli [2010] considers only 10 iterations, while Fattal [2011] uses 15 iterations per scale in a logarithmic subdivision. This clearly shows the importance of correct parameter settings: the favorable blue noise profile actually appears around 500 iterations, and we recommend 10k iterations.

We conclude this section by a brief comparison between the schematic model of Fig. 2 and the actual realized one in Fig. 7. A very low noise level $\epsilon$ is reached that only becomes visible in log-log plots. To facilitate the comparison, we shaded the area below $e^{-\sigma^2}$ and 1 in the discrete points spectrum, and the corresponding areas $e$ and $e^{-\omega^2}$, respectively, in the filtered set spectrum. The actual performance is close to the idealized model, but the spectrum oscillates around these two curves. Most notably, the first approximately describes the low-frequency band of interest, while the other is associated with the high-frequency range. Together, they are very close to the original specification of Ulichney [1987; 1988].

5 ALGORITHMIC DETAILS

Having nominated a generic Gaussian kernel, in this section we discuss domain-specific details, and show how the kernel may be manipulated accordingly.

5.1 Toroidal Domain

Many application scenarios favor point sets optimized with toroidal boundaries. This allows the points to be tiled seamlessly into larger sets, and also enables toroidal shifting to randomize the set [Cranley and Patterson 1976]. Additionally, having toroidal boundaries is also helpful during kernel-based optimizations to keep a balance for the points at the boundaries.

The essence of a toroidal domain is that each point “sees” every other point on both sides of each axis, which raises a question about which image of a point to use? The obvious answer is to take the nearest, which works well as long as the number of points is large enough relative to the effective kernel support of the numeric precision, but is incorrect otherwise: in theory, all replicas, in all dimensions, must be considered to attain the correct frequency profile of the energy kernel.

A toroidal domain can be seen as an infinite tiling of a primary tile that contains the samples, as illustrated in Fig. 8. The section plots reveal the influence of kernels at distant points, but also demonstrate...
that kernels near the edges continue outside the domain boundaries, while their replicas in other tiles provide contributions in the primary tile across the opposite edges. As a result, the net effect appears as if the Gaussian kernel is folded over the toroidal domain.

The replicas of an individual point form a regular grid at the domain period [Glassner 1995], as illustrated in Fig. 8(c). The accumulated influence over the toroidal boundary of the replicas of a single point is independent of the locations of the other points, but is affected by the number of these points via $\sigma$, which is scaled down by $\sqrt{N}$ for $N$ points in a $d$-dimensional unit domain. This makes the periodic images of a point more influential with a small point count, and the effective kernels, obtained by aggregating all point images, (i) are anisotropic, (ii) dominated by a DC level, and (iii) look more sinusoidal; see Fig. 8(c). The kernels gradually restore their original Gaussian shape as more points are added and $\sigma$ becomes smaller relative to the domain period; see Fig. 8(d).

We may now use these insights to compute the energy and gradient terms over a toroidal (periodic) domain. To account for all replicas of all points, the correct energy term $E_{ij}$ at point $p_i = (x_i, y_i)$ due to another point $p_j = (x_j, y_j)$ in a toroidal domain is

$$E_{ij} = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \exp\left(-\frac{(x_i - x_j - k)^2 + (y_i - y_j - l)^2}{2\sigma^2}\right)$$

where we dropped the scaling factors for sake of simplicity. This formulation of $E_{ij}$ is helpful in developing scalable algorithms and understanding the frequency spectrum of blue noise in a toroidal domain. The first sum of Gaussians in Eq. (22) can be shown to satisfy

$$\sum_{k=-\infty}^{\infty} \exp\left(-\frac{(x_{ij} - k)^2}{2\sigma^2}\right) = \sqrt{2\pi\sigma} \theta_3 \left(-\pi x_{ij}, \exp(-2\pi^2\sigma^2)\right).$$

where $x_{ij} = x_i - x_j$, and $\theta_3$ is a Jacobi theta function, which also satisfies

$$\theta_3 \left(-\pi x_{ij}, \exp(-2\pi^2\sigma^2)\right) = 1 + \sum_{f=-\infty}^{\infty} e^{-2\pi^2\sigma^2f^2} \cos(2\pi x_{ij}f).$$

(24)

At first glance this model of Eq. (22) may sound infeasible to compute, but the separable nature of the Gaussian kernel comes in quite handy here.

5.2 High Dimensions

All the previous derivations automatically scale to higher dimensions. A tricky note, though, is that the scaling factor $\sqrt{N}$ drops quickly below 2, making $\sigma$ comparable to the domain size, and the energy kernel profile gets shallow, as in the few-points 2D case in Fig. 8(row c column B). Then it is natural to ask whether the blue noise energy is well-defined or meaningful in 20 dimensions, for example? The answer is yes, since even a small slope of the energy kernel profile will get scaled up quickly after we multiply it over the dimensions, which will lead to a considerable variance in the energy for different point distributions. For a visual example, compare the deepest to the shallowest kernel profiles in Fig. 8(c)(B, C). We then note that the energy function is smooth and continuous, which means that, applying a descent algorithm, the points will continue to move down the valleys of the energy field. As shown in Fig. 4(b), we actually managed to obtain blue noise point sets in higher dimensions of the same quality as the 2D point sets, judging by their frequency spectra. Algorithm 1 lists our steps for optimization in a uniform unit torus of any number of dimensions, and we provide our actual implementations in the supplementary materials.

5.3 One Dimension

It is worth noting that Eq. (24) gives the energy function in terms of harmonics, offering an alternative implementation. We actually verified empirically that almost identical results are obtained by summing the Gaussians or the Gaussian-weighted cosines. An interesting application of this is to truncate the cosine series at some
harmonic below $\sqrt{N}$, which restricts the optimization to a small frequency range, producing a step blue noise as in Fig. 9.

A special application to this is producing one-dimensional blue noise, which is challenging otherwise. The problem with 1D optimization is that pushing any point away from another would inevitably bring it closer to another point on the other side: there are strictly two neighbors that cannot be escaped. This makes any optimization of the BN energy quickly descend towards the inevitable global minimum of a regular grid. Truncation of harmonics, however, offers better masking to the high-frequency range than mere filtering. Our experiments so far worked only with a very narrow band, but they provide an empirical demonstration of the existence of 1D blue noise.

![Algorithm 1](image)

**Algorithm 1**: Uniform blue-noise optimization. The variable names $\{i,j,k\}$ are used for the nested loop counters for a reference point, another point, and a replica index of the other point. The variable $g$ stands for Gaussian, while $g'$ stands for its gradient. The value of “periods” is $6/9$-$\sigma$ for single/double precision float.

**Input**: A list $p$ of $N$ point locations in the $d$-dimensional unit torus $[0,1)^d$.

**Output**: An optimized list of locations that minimizes the BN energy for a filtering kernel size of $\sigma^2$.

**repeat**

for $i \leftarrow 0$ to $N - 1$ do

for $j \leftarrow 0$ to $N - 1$, $j \neq i$ do

$g[j] \leftarrow 0$;

$g_{ij} \leftarrow 0$;

$g'_{ij} \leftarrow 0$;

for dim $\leftarrow 0$ to Dimensions $- 1$ do

$x \leftarrow p_i[dim] - p_j[dim]$;

if $x < 0$ then

$x \leftarrow x + 1$;

for $k \leftarrow 1$ to Periods do

$x_k \leftarrow x - k$;

$g_{ij}[dim] \leftarrow g_{ij}[dim] + e^{-\frac{x^2}{2\sigma^2}}$;

$g'_{ij}[dim] \leftarrow g'_{ij}[dim] + x_k \cdot e^{-\frac{x^2}{2\sigma^2}}$;

end for

end for

for dim$_{ref} \leftarrow 0$ to Dimensions $- 1$ do

for dim $\leftarrow 0$ to Dimensions $- 1$ do

if dim $\neq$ dim$_{ref}$ then

$g'_{ij}[dim] \leftarrow g'_{ij}[dim] \cdot g_{ij}[dim]$;

end if

end for

end for

$p[i] \leftarrow p[i] + g'[i]$

end repeat

until Optimization criteria met;

5.4 Bounded Domain

To optimize points in a bounded domain we need a strategy to keep them within the domain boundary, since otherwise the minimum energy is obtained just by letting the points go infinitely far apart. Just restricting the point locations does not work: the points will condensate at the boundaries, since that gives the minimal energy. The more appropriate way is to give the domain itself an appropriately scaled energy to attract the points. This is equivalent to the semantically more meaningful model of simulating a fictitious presence of points outside the domain. Modeling the domain as a continuum of infinitesimal points, the analytic nature of the gradient force in Eq. (18) offers a neat analytical solution. For a point $x_i$ in 2D, the energy gradient due to the domain can be modeled as

$$\nabla E(x_i) = - \int_0^1 \int_0^1 \exp \left(-\frac{(x-x_i)^2 + (y-y_i)^2}{4\sigma^2}\right) \left( x-x_i \right) dx \: dy + \left( y-y_i \right) \cdot \frac{1}{2\sigma}.$$ (25)
The x-axis component can be computed by splitting the integration:
\[ \nabla \mathcal{E}(x_i) = -\int_0^1 \exp \left(-\frac{(y-y_i)^2}{4\sigma^2}\right) dy \int_0^1 \exp \left(-\frac{(x-x_i)^2}{4\sigma^2}\right) (x-x_i) \, dx, \]
which eventually evaluates to
\[ \nabla \mathcal{E}(x_i) \propto \left[ \text{erf} \left( \frac{1-y_i}{2\sigma} \right) + \text{erf} \left( \frac{y_i}{2\sigma} \right) \right] \left[ e^{-\frac{(x-x_i)^2}{2\sigma^2}} - e^{-\frac{(x_i)^2}{2\sigma^2}} \right]. \]

The second factor is the 1D gradient, which is just the difference between the Gaussian-weighted distance to the edges of the domain, while the first term gives a Gaussian-weighted sum of gradients at different vertical stripes. The y-axis component is computed similarly, and the model scales smoothly to any dimensions just by incorporating the respective erf weights.

Noting that we no longer need to consider replicas of the points, we can see that optimization over a bounded domain is simpler and more efficient than over a toroidal domain. One noteworthy aspect of the resulting distribution, though, is that it will be offset from the domain boundaries by an \( O(\sigma) \) distance.

5.5 Adaptive Sampling

Typical contemporary blue-noise samplers are expected to support adaptive and importance sampling, where the sample density is spatially varied in accordance with a given density map. We combine four ideas to extend our model to adaptive sampling. From Schmaltz et al. [2010] we borrow the idea of treating pixels as negatively weighted kernels that attract the sample points. In our model, the idea is to minimize the variance of a zero-mean sum of Gaussians comprising the (positive) points and the (negative) pixels. From Fattal [2011, Figure 2] we borrow the idea of shaping the kernels in accordance with the local density, so we introduce a shaping factor \( a_k \) to shape the normalized energy kernels
\[ g_k(x) = a_k \exp \left(-a_k \frac{||x-x_k||^2}{2\sigma^2} \right) \]
so that they shrink in high-density areas to capture higher frequency details. We then developed our own Algorithm 2 for adapting the kernel shapes. Finally, we extend the variance term in Eq. (1) to the adaptive kernels case:
\[ \text{Var}(A(X)) = \frac{1}{N} \sum_{k=1}^N \sum_{l=1}^N \frac{2\pi \sigma^2}{a_k + a_l} \exp \left(-a_k a_l \frac{||x_k - x_l||^2}{2\sigma^2} \right) \]
from which we extract a loss function
\[ \mathcal{E}(x_k) = \frac{\pi \sigma^2}{N} \sum_{l=1}^N a_{kl} \exp \left(-a_{kl} \frac{||x_k - x_l||^2}{2\sigma^2} \right) \]
and its gradient
\[ \nabla \mathcal{E}(x_k) = \frac{\pi \sigma^2}{N} \sum_{l=1}^N a_{kl} \exp \left(-a_{kl} \frac{||x_k - x_l||^2}{2\sigma^2} \right) (x_k - x_l) \]
for individual points, where
\[ a_{kl} = \frac{2a_k a_l}{a_k + a_l} \]
is the mutual shaping factor of two kernels. Details of these derivations are provided in the supplementary materials. Optimization for adaptive domains then proceeds similar to the uniform case, and observes the same guidelines for kernel width and support, iteration counts, etc., but alternates between variance minimization and kernel shaping. Thus, we propose a novel and efficient auxiliary algorithm for adaptive samples (see Algorithm 2) that may also be used for reconstruction from adaptive samples. In practice, we use a single iteration of Algorithm 2 per optimization step, and 15 total iterations in reconstruction.

6 RESULTS AND COMPARISON

In this section we discuss various practical details of applying the discussed principles and models, and demonstrate the superiority of our method over baselines, achieving higher-quality blue noise, and better results on adaptive sampling for stippling and reconstruction.

6.1 Spatial Properties

We start by benchmarking our GBN against common techniques using the classic measures of blue noise, as shown in Fig. 10. Possibly the plots do not reveal a clear difference, except the zoneplate, which manifests the lower noise floor in our point sets in the low-frequency...
band. The actual difference is orders of magnitude, however, as revealed in Fig. 1(h).

Careful inspection of the radial power also reveals that our blue noise has the most flat low-frequency region combined with the smallest peak, which means that it is the least noised and also the least aliased, as reflected in the zoneplate plot. Our algorithm offers two handles for controlling the noise-aliasing trade-off: the value of $\sigma$ and the number of iterations, as discussed in Section 4.2 and Section 4.4. For the results in Fig. 10 we set $\sigma = 1$ and use 10K iterations.

6.2 Numerical Integration
In Fig. 11 we show numerical integration comparisons using various sample distributions in two, three, and eight dimensions, using different integrands. As a representative of smooth isotropic functions we use a toroidal sum of 64 Gaussians placed at random points [Ahmed and Wonka 2021], as illustrated in the top-left corner of the 2D plot. The spectrum of this integrand is a Gaussian multiplied by white noise; that is, a noisy Gaussian, which represents a wide range of signals dominated by a DC level. We adjust the variance of the Gaussians to a regular grid frequency of 512 points:

$$\sigma' = \frac{512^{-1}}{\text{dimensions}},$$

(33)

so that we can see the behavior of the sampling sets in both under-sampling and over-sampling conditions.

We first note that the variance of random sampling faithfully follows the $N^{-1}$ convergence rate that characterizes Monte Carlo integration. This offers a good reference for comparison. In all dimensions, GBN evidently outperforms the other distributions by orders of magnitudes. It exhibits an interesting sigmoid curve that faithfully follows analytical prediction. Indeed, if the integrand is $O\left(e^{-ax^2}\right)$, and the the spectrum of the point process is $O\left(e^{\beta x^2}\right)$, as we demonstrated, then Pilleboue et al. [2015] estimate the variance to be the inner product of the two spectra, which is $O\left(e^{(\beta-a)x^2}\right)$. When $\beta$ is smaller than $\alpha$, as is the case for $N < 512$, the variance spectra have an exponential decay, which explains the quick convergence rate at smaller point counts. On the other hand, the variance spectrum grows exponentially for a large number of points, but that is already bounded by the small peak in the points spectrum, no more than 4 times the white noise level, while the decay of the integrand should have already reached a very low level. Thus, GBN offers a very decent behavior in integration, being very competent in under-sampling conditions, and losing its advantage only at already-redundant sampling rates. This conclusion is very interesting, because it implies that blue noise faithfully brings its reconstruction advantage to numerical integration.

On the other extreme of directionally biased integrands, we use a half-space step by picking a random point and a random direction. While all distributions exhibit a similar $O(N^{-1})$ convergence to random sampling, we observe that GBN and some other blue noise distributions (BNOT, KDM, FPO) are outperformed by Owen-scrambled Sobol and, notably, SOT [Paulin et al. 2020]. Our guess is that this improvement arises from the cases where the direction of the step comes aligned with one or another axis in Owen-scrambled Sobol or slicing direction in SOT.

Finally, in the third column of Fig. 11 we show results for a bilinear $xy$ integrand [Christensen et al. 2018], and its high-dimensional equivalent, which combine between a smooth gradient and a sharp discontinuity at the domain edges. SOT clearly leads the competition in this kind of signals, even though low-discrepancy constructions like Owen-scrambled Sobol and BlueNets seem to eventually overtake. That mentioned, while our focus in this paper is on a practical number of samples, it is still worthwhile taking a look at the large-scale behavior of our method. The variance plots in Fig. 11(a) suggest that Owen-scrambled Sobol, BNOT, and possibly SOT, are going to outperform GBN, even for isotropic integrands, for a large number of points. In fact, the same piece of information can be read directly from the power spectral profiles in Fig. 1(h), by extending the lines to the left. We note, however, that GBN may benefit from more iterations and better machine precision, as illustrated in Fig. 12, and it is therefore too early to draw a conclusion without analyzing the intrinsic processes, which we leave for future research.

From these comparisons we conclude that it is difficult to nominate a specific method as the best all-purpose blue noise. Indeed, edges (e.g., object boundaries) and gradients (e.g., textures) are both met quite often in graphical scenes [Ahmed et al. 2016]. In all cases, however, there seems to exist a feasible blue noise optimization (GBN or SOT) that outperforms classic low-discrepancy distributions like Owen-scrambled Sobol, as envisioned long ago by Handson [2003]. As a side note, we find the superior performance of SOT in the shown cases quite interesting, since it could not be predicted by any of the common assessment tools. This is a clear gap in the assessment tools discussed in Section 2, and warrants further future research.

6.3 Adaptive Sampling and Reconstruction
Similar to the uniform sampling case, we could obtain superior results for adaptive sampling, as illustrated in Fig. 13. As predicted by the spectral profile in Fig. 1(g), our blue noise is much less noisy than BNOT. We used our auxiliary Algorithm 2 for reconstructing the density map back from the samples, and it proves faithful in capturing the noise visible to the eye. We also note that our method seems to be structure-aware, making the samples follow the feature lines. More results are provided in the supplementary materials. In Fig. 14 we show the convergence behavior of the optimization process. It bears close resemblance to the uniform case in Fig. 7, suggesting the correctness of our model. Finally, we note that our method is parametric, enabling uniformity-noise trade-offs by controlling the $\sigma$ parameter and the number of iterations.

6.4 Complexity
The time complexity of Algorithm 1 is clearly quadratic, which may be taken as a disadvantage. Thanks to the separability of the Gaussian kernel, however, as well as the associated theta function, the algorithm scales linearly with dimension, rather than exponentially. Thus, even though the quadratic complexity is inferior to the log-linear performance of BNOT [de Goes et al. 2012] and the linear convergence of KDM [Fattal 2011], our algorithm remains quadratic in all dimensions, while the time complexity of computing the underlying power diagram in BNOT grows exponentially, as
well as the space complexity of the computational grid of KDM. Fig. 15 shows actual execution speeds for different dimensions and point counts. It confirms the forecast linear time complexity with dimension, as evident in the uniform spacing of the curves. Notably, the quadratic complexity appears only beyond 8K points, which seems to be related to hardware configuration.

Our algorithm is inherently parallel, and lends itself to a GPU implementation, which gives a significant boost in performance compared to CPU-based algorithms like BNOT and KDM. In fact, the actual speed performance of our implementation is significantly faster than BNOT with reasonably-sized point sets. Orders of 100K points can still be computed in a reasonable time, but the quadratic time complexity makes it prohibitively slow to work with orders of millions. In that case the algorithm may be adapted to work within a local neighborhood of $\sigma$, which is quite sufficient.

Finally, our algorithm leads the competition when it comes to coding complexity, as we only use elementary functions of the standard C or CUDA library, and simple array data structures.

### 7 CONCLUSION

In this paper, we analyzed kernel-based blue noise using a Gaussian kernel. We derived a simple formula for a particular high quality blue noise spectrum, and we demonstrated its realizability and feasibility with empirical results. The power spectrum that we can achieve follows an exponential function and has significantly less power in low frequencies than previous work. We show basic results in 2D, but we also show extensions to adaptive sampling and higher dimensions. Our results show unprecedented visual quality for adaptive sampling.

The results in the paper suggest multiple directions for future research. For example, we demonstrated that properly sampled adaptive BN distributions actually retain a lot more information than can be seen by the bare eye looking at a stippling, and we demonstrated a practically working algorithm for retrieving this information. This poses a natural question about the utility of blue noise for data representation and compression. Our results demonstrated the superiority of high-dimensional blue noise over known alternatives for numerical integration for some integrands, but not for others. Is there a way, then, to make further progress on Monte-Carlo...
integration using blue noise? Furthermore, since the (adaptive) reconstruction algorithm scales with dimension, then an intriguing question we ask is about the possibility of using a blue-noise set of samples as an equivalent of pixels in high dimensions.

Finally, we reply to a long-standing question: are the different families of algorithms to produce blue noise equivalent? The answer now is a clear No! Cellular methods are intrinsically different from kernel-based methods, as evident in the polynomial vs. quadr-exponential spectral profile. We have explained the superiority of the latter analytically, and demonstrated it empirically for isotropic integrands, and we conclude by giving an intuitive insight of the difference. The partitioning in cellular methods immediately introduces quantization noise all over the domain. Intuitively, we are introducing boundaries of our own that do not actually exist in the domain we are sampling. Kernel-based methods, in contrast, enable the “coverage” of sample points to overlap, as can be visualized in the sections of Fig. 8, which seems more natural in many application scenarios.

ACKNOWLEDGMENTS
Thanks to the anonymous reviewers for the valuable comments. Thanks to Mohanad Ahmed for his insightful discussions.
Abdalla G. M. Ahmed, Jing Ren, and Peter Wonka

Fig. 13. Adaptive sampling and reconstruction of a density map, using 20K samples, comparing GBN to BNOT.

Fig. 14. Convergence of our method in adaptive sampling at different iteration counts, using 10K sample points. BNOT [de Goes et al. 2012] is shown for benchmarking. We note that the convergence apparently follows the same behavior as the uniform case of Fig. 7, beginning in middle-range frequencies, and starting to attain the favorable spectrum from around 512. The last plot traces the actual paths of the points along the process. Input image courtesy of Zainab Eisa and Lamees Hassan.

Fig. 15. Typical computation times of 10K iterations for different point counts and different dimensions, using an NVIDIA Quadro RTX 4000 GPU. Adaptive timing is based on a 400×400 pixels image.

REFERENCES

Abdalla G. M. Ahmed. 2019. Sampling with Pinwheel Tiles. In Computer Graphics and Visual Computing (CGVC). The Eurographics Association. https://doi.org/10.2312/cgvc.20191271

A. G. M. Ahmed, J. Guo, D. M. Yan, J. Y. Franceschia, X. Zhang, and O. Deussen. 2017a. A Simple Push-Pull Algorithm for Blue-Noise Sampling. IEEE Transactions on Visualization and Computer Graphics 23, 12 (Dec. 2017), 2496–2508. https://doi.org/10.1109/TVCG.2016.2641963

Abdalla G. M. Ahmed, Hui Huang, and Oliver Deussen. 2015. AA Patterns for Point Sets with Controlled Spectral Properties. ACM Trans. Graph. 34, 6, Article 212 (Oct. 2015), 212:1–212:8 pages. http://doi.acm.org/10.1145/2816795.2818139

Abdalla G. M. Ahmed, Till Niese, Hui Huang, and Oliver Deussen. 2017b. An Adaptive Point Sampler on a Regular Lattice. ACM Trans. Graph. 36, 4 (July 2017), 158:1–158:13. http://doi.acm.org/10.1145/3072959.3073598

Abdalla G. M. Ahmed, Hélène Perrier, David Coeurjolly, Victor Ostromoukhov, Jianwei Guo, Dong-Ming Yan, Hui Huang, and Oliver Deussen. 2016. Low-Discrepancy Blue-Noise Sampling. ACM Trans. Graph. 35, 6, Article 247 (Nov. 2016), 247:1–247:13 pages. https://doi.org/10.1145/2901798.2902218

Abdalla G. M. Ahmed and Peter Wonka. 2021. Optimizing Dyadic Nets. ACM Trans. Graph. 40, 4, Article 141 (Jul. 2021). https://doi.org/10.1145/3450626.3459880

Michael Balzer, Thomas Schölmer, and Oliver Deussen. 2009. Capacity-Constrained Point Distributions: A Variant of Lloyd’s Method. ACM Trans. Graph. 28, 3, Article 86 (July 2009), 86:1–86:8 pages. https://doi.org/10.1145/1531326.1531392

R. Bridson. 2007. Fast Poisson-Disk Sampling in Arbitrary Dimensions. In ACM SIGGRAPH 2007 Sketches.

Zhonggui Chen, Zhan Yuan, Yi-King Liu, and Wenping Wang. 2012. Variational Blue Noise Sampling. IEEE Transactions on Visualization and Computer Graphics 18, 10 (Oct. 2012), 1784–1796. https://doi.org/10.1109/TVCG.2012.94

Per Christensen, Andrew Kensler, and Charlie Kilpatrick. 2018. Progressive Multi-Jittered Sample Sequences. In Computer Graphics Forum, Vol. 37. 21–33.

Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. 2003. Wang Tiles for Image and Texture Generation. ACM Trans. Graph. 22, 3 (July 2003), 287–294. https://doi.org/10.1145/882262.882265

Robert L. Cook. 1986. Stochastic Sampling in Computer Graphics. ACM Trans. Graph. 5, 1 (1986), 51–72. https://doi.org/10.1145/7529.8927

Roy Cranley and Thomas NL Patterson. 1976. Randomization of number theoretic methods for multiple integration. SIAM J. Numer. Anal. 13, 6 (1976), 904–914.

Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue Noise through Optimal Transport. ACM Trans. Graph. 31, 6, Article 171 (Nov. 2012), 171:1–171:11 pages. http://doi.acm.org/10.1145/2366145.2366190

Oliver Deussen, Stefan Hiller, Cornelius Van Overveld, and Thomas Strothotte. 2000. Floating Points: A Method for Computing Stipple Drawings. Computer Graphics...
