NILE: Nested Interleaving of Low-Dimensional Elements

ABDALLA G. M. AHMED, Shenzhen University, China
MATT PHARR, NVIDIA, USA
VICTOR OSTROMOUKHOV, Univ Lyon 1/CNRS, France
HUI HUANG*, Shenzhen University, China

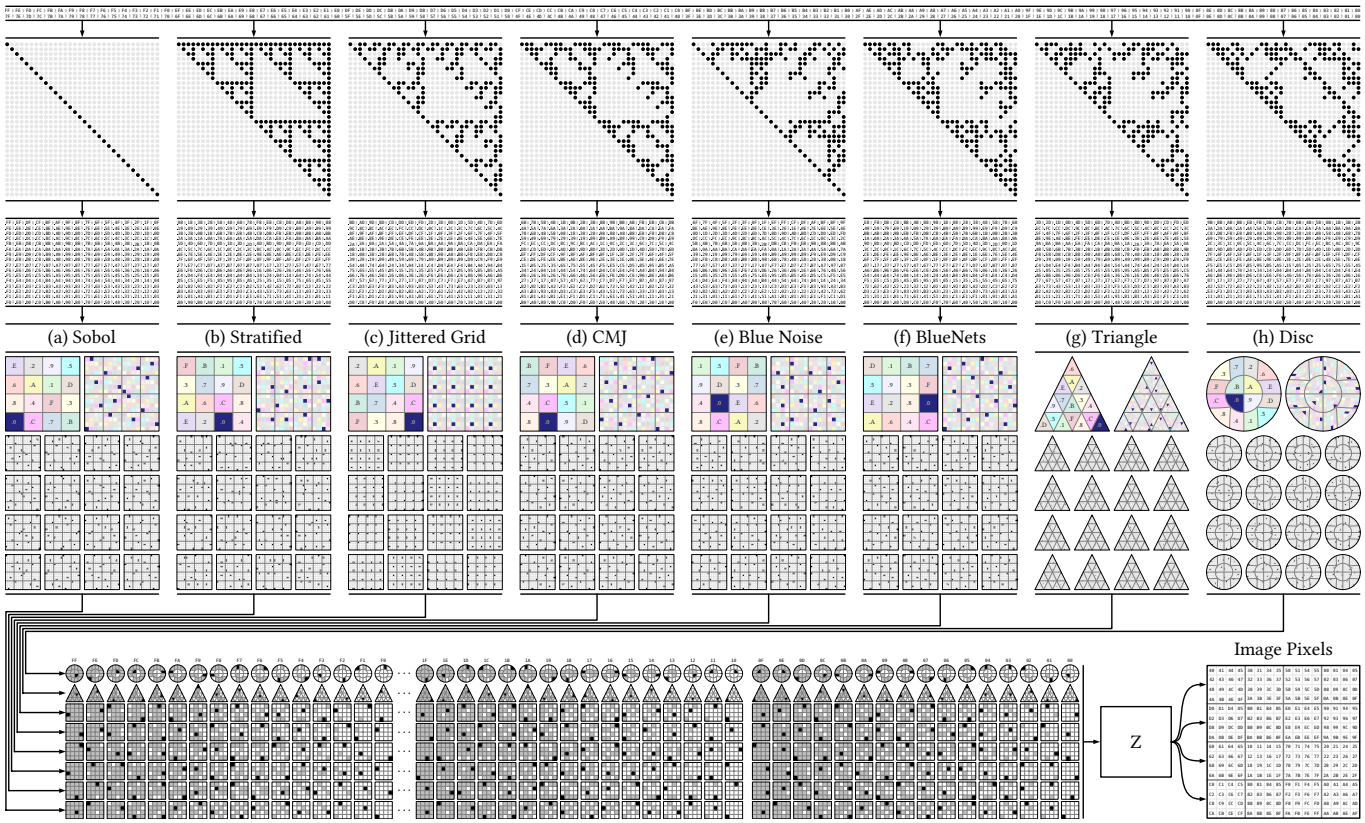


Fig. 1. Schematic visualization of our proposed NILE sampler architecture, showing the data flow for distributing samples from eight 2D subspaces into a miniature 4×4-pixel image at 16 samples per pixel. The model comprises three major parts: (i) an expandable bank of *elements*, each generating a sequence of low-dimensional samples, (ii) a high-dimensional low-discrepancy sequence, for which we use SZ [Ahmed et al. 2025], that orchestrates the indexing of samples, and (iii) a Z-indexing unit that maps the samples to (sub-)pixels in the image plane. The flow in this diagram is top-down, left-to-right: First, sample indices are fed to the SZ sequence, which bit-reverses and then shuffles them. Each dimension of the SZ sequence is used to index one *nested* element in a way that interleaves the samples so that there is uniform coverage of the joint high-dimensional space. A color-coded static indexing scheme is shown for major and sub-intervals of each element, along with the 16-sample sets it would deliver in non-shuffled order. The stream at the bottom shows the major interval of each sample component in its corresponding subspace, and the gray shading traces the sampled intervals across 16-sample blocks. The 16-sample block delivered to each pixel covers all major intervals of each element, while all 256 sub-intervals are covered over the 4×4-pixel block. Further, the samples delivered to the 4×4-pixel block comprise all 16×16 combinations of intervals for each pair of elements. Numbers in the pixel area indicate the index of the composite sample assigned to the subpixel sample there, e.g., composite scene sample #0 is received by the image-plane sample in the top-left subpixel of the top-right pixel.

*Corresponding author: Hui Huang (hhzhiyan@gmail.com)

Authors' Contact Information: Abdalla G. M. Ahmed, Visual Computing Research Center (VCC), College of Computer Science and Software Engineering (CSSE), Shenzhen University, China, abdalla_gafar@hotmail.com; Matt Pharr, NVIDIA, USA, matt@pharr.org; Victor Ostromoukhov, Univ Lyon 1/CNRS, France, victor.ostromoukhov@liris.cnrs.fr; Hui Huang, VCC, CSSE, Shenzhen University, China, hhzhiyan@gmail.com.

Sampling strategies in computer graphics have long been divided between local approaches, which optimize sample distributions independently at each pixel, and global approaches, which use high-dimensional low-discrepancy sequences to ensure uniformity across all dimensions, including adjacent pixels. While global samplers are most-commonly used thanks to generally

ACM 1557-7368/2026/7-ART105
https://doi.org/10.1145/3811283

having better convergence rates, it comes at a cost of very limited user control over the distribution of samples on subspaces, making it difficult to handle common aliasing artifacts.

We introduce a novel modular meta-sampler architecture that bridges local and global sampling, allowing integrator designers to employ specialized low-dimensional samplers while still achieving high-dimensional uniformity. Our approach leverages high-dimensional low-discrepancy sequences to orchestrate sample generation by a collection of local samplers operating over power-of-two hierarchical intervals. We demonstrate how existing local sampling techniques, including stratified, blue noise, and dyadic nets sampling, can be reformulated to be used with this framework, enabling hybrid sampling strategies that combine the benefits of both paradigms.

ACM Reference Format:
Abdalla G. M. Ahmed, Matt Pharr, Victor Ostromoukhov, and Hui Huang. 2026. NILE: Nested Interleaving of Low-Dimensional Elements. *ACM Trans. Graph.* 45, 1, Article 105 (July 2026), 15 pages. https://doi.org/10.1145/3811283

1 Introduction

Sampling is a fundamental operation in computer graphics (CG), underlying many applications, ranging from traditional halftoning to modern machine learning. Sampling is especially essential in rendering, where pixel values are computed via Monte Carlo integration of radiance over complex light transport paths. Unlike Monte Carlo-based simulation in many other fields, where all samples contribute to an overall aggregate, the pixel-based nature of CG, along with relatively high dimensions and broad frequency spectra, make sampling particularly challenging, and even very high sampling rates are insufficient to reach the Nyquist rate of classic sampling theory, calling for special handling and many heuristics when devising effective and efficient sampling strategies.

Early research on sampling generally focused on *local sampling* techniques that sampled individual per-pixel low-dimensional constituents [Cook et al. 1984; Dippé and Wold 1985; Glassner 1994]. This naturally reflected the nature of light paths, leading most researchers to consider high-dimensional uniformity "at best a secondary concern" [Pharr and Humphreys 2004, p. 339]. A new opportunity for effective high-dimensional sampling was introduced by the crucial work of Grünschloß et al. [2012], who proposed efficient algorithms for inverting common low-discrepancy (LD) sequences and paved the way for the adoption of *global sampling* strategies, where a single high-dimensional LD sequence is used to generate the samples at once for all pixels over all dimensions.

Unfortunately, local and global sampling approaches remain disconnected: one must choose one or the other. In practice, global samplers are now the most widely used, even though it can be scene-dependent whether a local or global approach will be more effective. This state of affairs has also reduced interest in new research on local samplers, even though some local distributions are demonstrably superior in controlled experiments, and handcrafted samplers for specific geometries (e.g., triangles and disks) have been shown to be highly effective in some situations [Pharr 2019]. More broadly, it forces the light transport integrator to conform to the sampler, rather than having a sampler that adapts to the integrator and the problem at hand.

In this paper, we seek to reconcile competing sampling models by introducing a hybrid, modular *meta-sampler* architecture—*Nested Interleaving of Low-Dimensional Elements* (NILE)—that bridges local and global sampling techniques by combining high-dimensional LD sequences with classic per-subspace distributions within a single framework, as illustrated in Fig. 1. In contrast to prior attempts that fuse local distribution properties, such as blue noise (BN), directly into LD sequences, our approach assembles these components as distinct modules within a higher-level architecture, allowing each to retain its identity and fulfill a complementary role. Specifically, we use explicitly controlled low-dimensional elements to generate the actual samples, rely on a global LD sequence to interleave them into high-dimensional samples, and finally apply Z-indexing to distribute the samples across pixels. This three-stage pipeline is designed to ensure a fair distribution of the samples across their source subspaces, the joint high-dimensional space, and the pixels.

Following the necessary technical background and a brief review of related work in Section 2, we provide an abstract overview of our proposed model in Section 3, analyzing success factors of established samplers, distilling design targets, and assembling the model to combine best practices. The proposed model is modular, allowing explicit control over local sampling, and in Section 4 we provide example modules covering a range of classic local distributions. We then present results and discuss some implications in Section 5, and conclude with remarks and an outlook for future work in Section 6.

2 Essential Background and Related Work

In this section, we review related work and discuss the essential technical background needed to understand the current paper.

2.1 Tiling Techniques

Most favorable local sampling distributions are produced by costly iterative optimization, making them unsuitable for direct use in rendering [Lagae and Dutré 2008]. To overcome this bottleneck, a range of lookup-based techniques were proposed, evolving through four technical generations, from (i) a single periodic tile [Glassner 1994], through (ii) Wang tile sets with matching rules to suppress repetition artifacts [Cohen et al. 2003; Lagae and Dutré 2006; Wang 1965], to (iii) recursive tiling enabling adaptive sampling [Kopf et al. 2006; Lagae and Dutré 2006; Ostromoukhov 2007; Ostromoukhov et al. 2004], and ultimately (iv) an abstract grammar for indexing over a multi-resolution regular grid [Ahmed et al. 2017]. We will rely mostly on tiling methods to provide local sample distributions, but combine them with other modules to interleave the samples across dimensions and distribute them over pixels.

2.2 Digit-Reversed Indexing

A significant advancement in tiling techniques that underpins our current model is digit-reversed indexing, originally developed by Ostromoukhov [2007] for Polyominoes, as illustrated in Fig. 2, yet extends to other self-similar tilings [Ahmed 2019; Wachtel et al. 2014]. The model is defined by four aspects; the terms are ours:

Geometry: The shape of tiles and their integer self-tiling ratio b.
Alphabet: Each distinct tile is given an id that captures its local neighborhood in a recursive tiling. Polyominoes use geometric neighborhoods, but the actual process differs across tiling methods.

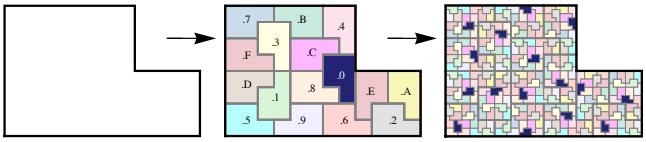


Fig. 2. A particular polyomino, the P-pentomino (left), is subdivided into 16 identical pentominoes (middle). Each resulting pentomino can in turn be further subdivided into 16 pentominoes (right). The first pentomino selected after each subdivision (shown in dark blue) forms a well-spaced, nearly blue-noise pattern. This process can be iteratively repeated until the desired level of subdivision is achieved.

Grammar: A tile is recursively subdivided sufficiently to capture all distinct neighborhoods; then a production rule is extracted for every distinct tile listing child tile ids.
Ranking: A rank in $\{0, \dots, b-1\}$ is assigned to each child of every tile. This is the critical user-controlled parameter for this model.

With this definition, the tileset can be optimized as follows. First, a tile is recursively subdivided deeply enough to capture all distinct neighborhoods. Each tile is subdivided once more, and the child-tile ranks of each distinct tile are optimized to match a prescribed target across all instances in the tiling. The target is usually a blue-noise distribution, and the resulting optimized pattern is a dithering mask [Ulichney 1993]. While the optimized mask has only b grades, it is turned into an infinite-resolution mask via digit-reversed indexing: each child tile at any depth has a unique global rank obtained by recursively concatenating its rank to that of its parent into a base-b digit-reversed index. For our current paper, this is best viewed as a positionally encoded fraction.

2.3 ART Grammar

Adaptive Regular Tiles (ART) [Ahmed et al. 2017] replaced the color-coded and geometrically induced grammar of earlier methods with an abstract axis-wise grammar derived from the Thue–Morse word [Lothaire 2002]

$$T = 0110100110010110100101100110100 1\cdots, \tag{1}$$

defined as the fixed point of the Thue–Morse production rule

$$\mu: \begin{array}{l} 0 \mapsto 01 \\ 1 \mapsto 10 \end{array}. \tag{2}$$

An alphabet Σ_l is identified by the distinct *factors* (subwords) of T of a prescribed length l; for example,

$$\Sigma_2 = \{A, B, C, D\} : (A, B, C, D) \leftrightarrow (01, 11, 10, 00)\,. \tag{3}$$

Production rules are obtained by applying the Thue–Morse production in Eq. (2) to the identification words, and then reading back the resulting symbols through a sliding window of l bits:

$$\begin{pmatrix} A, \\ B, \\ C, \\ D \end{pmatrix} \mapsto \begin{pmatrix} 01, \\ 11, \\ 10, \\ 00 \end{pmatrix} \overset{\mu}{\mapsto} \begin{pmatrix} 0110, \\ 1010, \\ 1001, \\ 0101 \end{pmatrix} \mapsto \begin{pmatrix} (0110,\ 0110), \\ (1010,\ 1010), \\ (1001,\ 1001), \\ (0101,\ 0101) \end{pmatrix} \mapsto \begin{pmatrix} (A, B), \\ (C, A), \\ (C, D), \\ (A, C) \end{pmatrix}. \tag{4}$$

A 2D alphabet and associated grammar are obtained as a Cartesian product of 1D alphabets.

An abstract tiling is obtained by recursively applying the production rules, and can be used to index any tiling with the same dimension and a power-of-2 subdivision rate. More broadly, the production tree can be used to index the nodes of any aligned tree. The essence of this scheme is that it inherits multiple desirable properties from the underlying Thue–Morse word, including provable repetition avoidance, low complexity [Lothaire 2002], and a convenient setup for optimization. The most significant advantages for our current work, however, are the square tile geometry and the binary production rate, which make ART highly compatible with dyadic LD sequences.

2.4 The van der Corput Sequence

The (binary) van der Corput [1935] (vdC) construction applies a transformation $\phi_2 : \mathbb{N}_0 \to [0, 1)$:

$$(\cdots v_3 v_2 v_1)_2 = \sum_{i=1} v_i\, 2^{i-1} \;\mapsto\; \sum_{i=1} v_i\, 2^{-i} = (.v_1 v_2 v_3 \cdots)_2\,, \tag{5}$$

which maps a binary-encoded sample index v to a binary-encoded fraction, $\phi_2(v)$, representing the indexed sample. Intuitively, a vdC sample is obtained by mirroring the bits across the fraction point. This digit-reversal mapping provably produces well-distributed sequences of samples, and serves as a building block for high-dimensional LD sequences.

2.5 Nets and (t, s)-Sequences

Niederreiter [1987; 1992] established a theoretical framework that encompasses almost all high-dimensional extensions of the vdC sequence, which are constructed to minimize a sample quality measure called *discrepancy*. In Niederreiter's notation, a "(t, m, s)-net in base b" is a set of b^m points in the s-dimensional hypercube such that for all possible subdivisions into b^{m-t} congruent hyperrectangular strata, each stratum contains exactly b^t points. A "(t, s)-sequence in base b" is an infinite sequence of points such that, for all values of m, every consecutive block of b^m points is a (t, m, s)-net in base b. To minimize discrepancy, we prefer the base b and "quality parameter" t to be small. The vdC sequence is a dyadic (i.e., base 2) $(0, 1)$-sequence.

2.6 Low-Dimensional Samplers

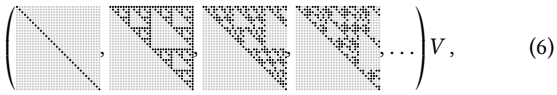
Early work on sampling in CG generally focused on low-dimensional sampling techniques. Jittered stratified points were applied by Cook et al. [1986; 1984] and by Dippé and Wold [1985], soon to be improved by Mitchell's [1987] use of error diffusion in pursuit of a blue-noise distribution. Subsequent work included Kollig and Keller's [2002] use of dyadic $(0, m, 2)$-nets and $(0, 2)$-sequences for 2D sampling, multi-jittered sampling and its descendants [Chiu et al. 1994; Christensen et al. 2018; Helmer et al. 2021; Kensler 2013], and sampling using rank-1 lattices [Dammertz and Keller 2006; Keller 2004]. There has also been work on developing direct sampling techniques for domains other than $[0, 1]^n$, including the unit sphere [Marques et al. 2013], triangles [Basu and Owen 2015], and disks [Christensen 2018; Shirley and Chiu 1997].

With such samplers, higher-dimensional patterns were generally generated by *padding*: randomly pairing lower-dimensional samples [Cook et al. 1984; Kollig and Keller 2002]. Mitchell [1991]

investigated explicitly optimizing 5D samples from 1D and 2D distributions, and more recently Jarosz et al. [2019] introduced orthogonal array sampling to CG and showed its value for combining lower-dimensional samplers.

2.7 Sobol' and SZ Sequences

Sobol' [1967] introduced a powerful dyadic sequence for sampling in arbitrary dimensions, obtained by linear shuffling:



$$V, \tag{6}$$

where V is a column vector encoding the vdC sequence. Each matrix in the ordered tuple generates a component (dimension) of the sample. The matrix-vector multiplication uses arithmetic over the Galois field GF(2), that is, bitwise AND and XOR for multiplication and addition, respectively [Bratley and Fox 1988; Sobol' et al. 2011]. This shuffling is rank-preserving, thanks to the upper-triangular form of these matrices [Faure and Tezuka 2002], meaning that each individual dimension is a $(0, 1)$-sequence. The matrices in Eq. (6) are constructed from GF(2) polynomials of increasing degree

$$e = (1, 1, 2, 3, \ldots), \tag{7}$$

and this order plays a pivotal role in the success of the Sobol' sequence in rendering, as we will see later. Any subspace is a dyadic (t, s)-sequence satisfying

$$t \leq \sum_i (e_i - 1), \tag{8}$$

summed over the underlying polynomial degrees [Lemieux 2009; Sobol' et al. 1992].

The described polynomial-based construction gives Sobol' sequences a preference for earlier over later dimensions in terms of subspace coverage. The model, however, embodies multiple degrees of freedom for constructing variants. For example, Joe and Kue [2008], using the same original set of primitive polynomials, optimized the arbitrary initialization bits toward better pairwise 2D projections. More recently, Ostromoukhov et al. [2024] suggested a variant using base-3, while Bonneel et al. [2025] proposed a binary variant with a different choice of polynomials, both aimed at improving lower-dimensional projections.

Apart from Sobol', Ahmed et al. [2025] recently introduced a new family of LD sequences, SZ, that similarly employs a bank of GF(2) upper-triangular matrices, but recursively shuffle the matrices themselves over 2^{2^k} bands of dimensions, yielding dyadic $(0, 1)$-sequences over all dimensions, dyadic $(0, 2)$-sequences over consecutive pairs, base-4 $(0, 4)$-sequences over consecutive quadruples, etc.

2.8 Global Samplers

A milestone in sampling practice was the introduction of LD sequences to rendering by Shirley [1991] and Mitchell [1992]. Subsequent work on inversion by Grünschloß et al. [2012] was crucial in enabling the global sampler model, as it allowed the use of LD sequences with parallel processing based on screen-space decomposition. In this formulation, samples are drawn from a single high-dimensional LD sequence shared across all pixels and dimensions: the first two dimensions parameterize the image plane, while the remaining sample components are obtained by inverting the samples associated with each pixel to retrieve indices. This approach provided the first effective alternative to random coupling of low-dimensional components, and achieved notable practical success, particularly with the Sobol' sequence, despite characteristic pre-convergence artifacts. Several other works by Keller and colleagues further contributed to advances in sampling practice [Grünschloß and Keller 2009; Keller 1996, 2006].

2.9 Z Sampler

Georgiev and Fajardo [2016] demonstrated that the desirable blue-noise characteristics relevant to rendering are associated with the screen-space distribution of error rather than the spatial distribution of the input samples themselves, and presented convincing results that influenced later research [Heitz and Belcour 2019; Heitz et al. 2019]. Along this direction, Ahmed and Wonka [2020] introduced a sampler, Z, based entirely on an LD sequence, namely the 2D Sobol' sequence, yet producing blue-noise-like error distributions without explicit optimization.

3 Core Ideas

In this section, we present our core contribution, starting by analyzing relevant characteristics of established samplers, which we distill into the NILE model.

3.1 Demystifying Global Sobol

To appreciate the true power of the global Sobol' sampler, we start by reviewing its intrinsic distortions in Fig. 3, which shows the actual samples it would produce for the same miniature 4×4 image as in Fig. 1, at the same 16 samples per pixel (spp). There we also show the *effective matrices* that generate these samples. These matrices account for the sample reordering due to global sampling. Here, they would first generate all 16 samples for the first Z-indexed pixel, then 16 for the second pixel, etc. They are never explicitly constructed, but serve to illustrate the effect of global sampling.

While it is commonly thought that global Sobol' sampling benefits from the Joe and Kuo [2008] tables optimized for pairwise 2D projections, it turns out that the effective matrices are significantly reshaped by the inversion process. Intuitively, the samples associated with each pixel do not come from a contiguous block of sample points from the original Sobol' sequence, and hence are not the ones one would expect from the optimized tables: the pairwise optimization is lost. The matrices are not even upper-triangular, and hence individual dimensions are not necessarily $(0, 1)$-sequences, as can be clearly seen in the sixth pair, where half the domain has no samples for each 16-sample block. This gives a clear explanation of the checkerboard patterns observed in images rendered using this sampler when higher dimensions are used for sampling surfaces that are indirectly visible at a pixel, for example after multiple specular reflections.

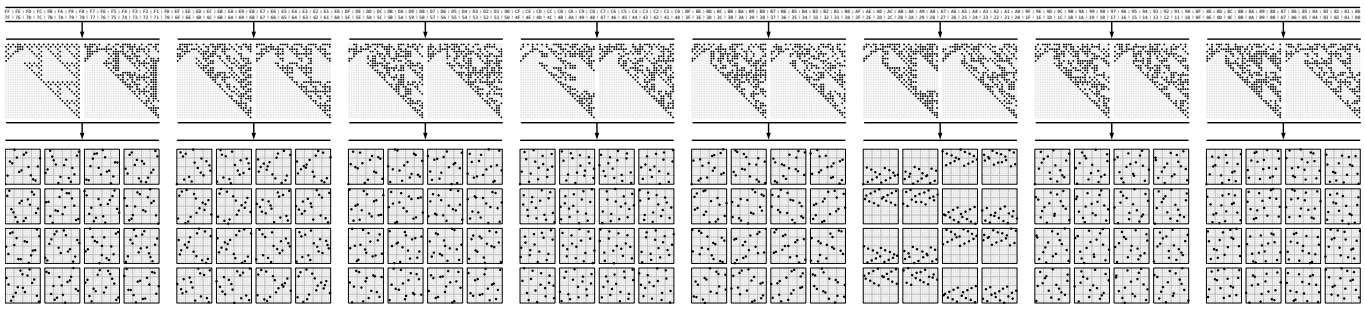


Fig. 3. The 2D samples generated by the classic global Sobol' sampler for dimension pairs $(2, 3)$ through $(16, 17)$, corresponding to those in Fig. 1. Here we visualize the 2D sample distribution that is provided to each pixel in a small 4×4 image. The matrices shown are the effective matrices accounting for the assignment of samples to pixels; i.e., the first 16 samples they generate are those that are provided to the first Z-indexed pixel, the next 16 to the second, etc. Note that, unlike the base Sobol' matrices, they are no longer upper triangular.

Yet the global sampler is highly effective in practice; we see that its success cannot be explained by low discrepancy alone. Thus, it takes a deeper understanding of the structure of the sequence to explain this performance, and the key is in Eqs. (7, 8), where the first two matrices, responsible for the image plane, contribute nothing to the t value of joint subspaces. Consequently, the quality level of the sequence is the same for each pixel's samples as for the global sequence. The implication is twofold: (1) each pixel receives good-quality samples on the joint high-dimensional space, and (2) all pixels receive the same sample quality, minimizing variance between different pixels. In short, global sampling takes advantage of the specific structure of Sobol, not only its discrepancy.

An additional advantage is how error is distributed among pixels: global sampling ensures that 2×2 blocks of pixels act like a super-pixel that receives 4× as many well-distributed samples, and so on. A careful look at the problematic pair of dimensions in Fig. 3 reveals that the samples are complementary not only over 2×2 pixels, but also 4×1 and 1×4 blocks, reflecting the dyadic net structure of 2D Sobol.

To summarize, for the global Sobol' sampler we have excellent coverage over pixels, excellent coverage of the high-dimensional space, and while there are cases of poor coverage over subspaces, they are ameliorated by screen-space redistribution of error.

3.2 Analyzing the Z Sampler

The Z sampler model is simple: pixels are Z-indexed [Morton 1966], the induced quad-tree is shuffled independently for each subspace, and a dyadic $(0, 1)$- or $(0, 2)$-sequence of samples is distributed over the space-filling Z-curve of each 1D or 2D subspace. Thanks to the *modular* structure of these sequences, all consecutive 2^m blocks are respectively dyadic $(0, m, 1)$- or $(0, m, 2)$-nets. Hence, analogously to global Sobol' sampling, (1) each pixel receives good-quality samples on each subspace, and (2) all pixels receive the same sample quality. Further, power-of-2 blocks of pixels share complementary samples and therefore, like global Sobol, hierarchically distribute the sampling error across pixels, though only for square blocks in this case. Thus, Z sampling trades the uniform high-dimensional coverage of global Sobol' for uniform coverage of subspaces, and the comparable rendering results suggest that this is a fair trade, emphasizing the importance of both high-dimensional and per-subspace uniform coverage.

3.3 Nested Interleaving

We now proceed to present our proposed model, which aims to provide balanced coverage across all three relevant domains: pixels, low-dimensional subspaces, and the joint high-dimensional space. In addition, we seek to retain explicit user control over sample distributions on elementary low-dimensional subspaces. To this end, we build upon recursive tiling (Sections 2.1–2.3) and the Z sampler.

Our approach to high-dimensional coverage is simple in principle: we delegate this responsibility to an existing high-dimensional LD sequence. Specifically, we associate each element sampler with a single dimension of the LD sequence and use that dimension to nest the generated sample stream. The LD sequence then interleaves these streams in the same manner that it interleaves its native components. In this way, the LD sequence provides global high-dimensional uniformity while the nested streams retain control over low-dimensional sample structure.

More specifically, the vdC sequence in Eq. (5) maps a stream of indices to an abstract stream

$$.0, .1, .01, .11, .001, \ldots \tag{9}$$

of binary fractions at progressively increasing resolution. While these are most commonly interpreted as sample *points* on the unit interval, this interpretation is arbitrary. Alternatively, the shuffling mechanism of nets and (t, s)-sequences leads to a different interpretation in which they identify multi-resolution intervals. For example, a dyadic $(0, 4, 2)$-net is constructed by shuffling a list $(.x_1x_2x_3x_4)_{.0000}^{.1111}$ of 16 binary-encoded fractions, and pairing it with a list $(.y_1y_2y_3y_4)_{.0000}^{.1111}$, in such a way that for all 4-bit combinations

$$\{.y_1x_1x_2x_3,\ .y_1y_2x_1x_2,\ .y_1y_2y_3x_1\} \tag{10}$$

of leading bits, we have the full set .0000 through .1111. The essence of these concatenated-bit fractions is that they represent intervals on the 2D space, i.e., strata in one of the possible stratifications, and a full power-of-two-sized list gives fully populated strata. This is the most fundamental conceptual shift behind our vision of nested interleaving: thinking in terms of intervals instead of points.

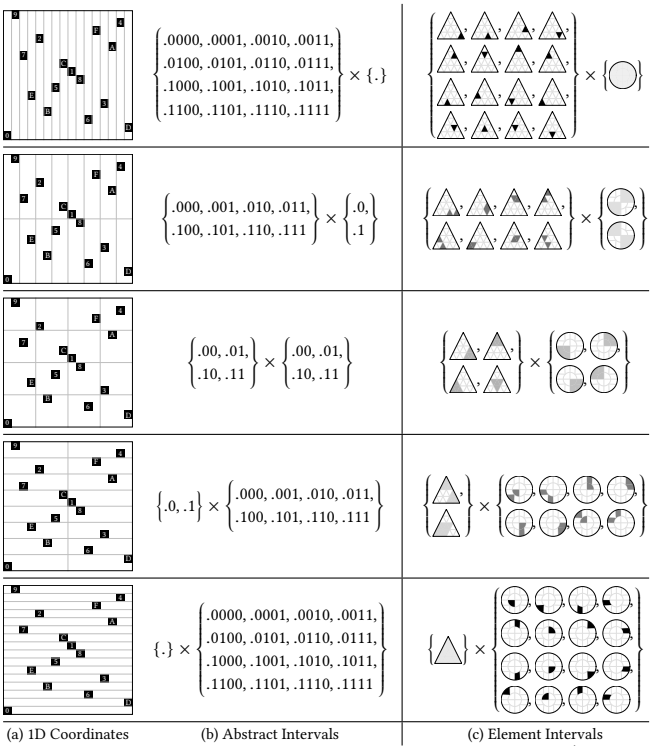


Fig. 4. Interleaving of the first 16 samples of the triangle and disc in Fig. 1(g, h), indexed by SZ dimensions 6 and 7 that form a dyadic (0, 2)-sequence. (a) When indices are treated as 1D coordinates, they make a classic multi-stratified dyadic (0, 4, 2)-net, namely a shuffled 2D Sobol' sequence. (b) Each stratification can be abstracted into a Cartesian product of abstract intervals. (c) Corresponding stratifications in the triangle×disc space can be viewed through a Cartesian product of indexed intervals. These samples are assigned to the top-right pixel in Fig. 1, and each other pixel receives a similar transformed dyadic net in the joint 4D triangle×disc space.

To nest our element sample distributions, we thus need to recursively partition each sampled domain into a dyadic tree of our choice, and to map its intervals to those of the nesting vdC sequence. As long as we respect this dyadic indexing, the LD sequence will transparently interleave our samples, with the basic difference that the unit hypercube spanned by the LD sequence is interpreted now as a combination space spanning the Cartesian product of our customized intervals, as demonstrated in Fig. 4. Section 4 presents a number of such samplers.

The sample indexing bits used for interleaving may be partitioned into three parts:

$$.o_1 o_2 \dots o_s d_1 \cdots d_r j_1 j_2 j_3 j_4 j_5 \cdots \tag{11}$$

For our example in Fig. 1, the order bits o_1 through o_4 select the major interval containing the sample point, according to the designated 4-bit samples-per-pixel rate. Since all major intervals are visited once to pick the 16 samples for each pixel, these bits only affect the order of the samples, but not their locations. These bits can therefore serve to shuffle the sample indices, in order to decorrelate local dimensions of different elements. The following design bits d_1 through d_r control the actual layout up to the designated design resolution. The remaining trailing jitter bits j_i, up to numerical precision, decide the sample placement within the sub-intervals, and can therefore serve as scrambling bits to jitter the sample sets.

3.4 Element Construction

The construction of hierarchically ordered, sequentially indexed sample distributions differs fundamentally from the construction of their point-centric counterparts. The distinction is not merely algorithmic, but reflects a conceptual shift in how sampling is understood. From this perspective, sampling is no longer viewed as placing points in a domain, but as assigning to each pixel sample a structured representation—an image—of the sampled domain itself.

Each pixel sample initially receives a single-pixel image of the domain, representing it at very low resolution. As samples accumulate over spatial neighborhoods, the reflected domain image is progressively refined: blocks of pixel samples collectively share higher-resolution images of the domain. Notably, the assigned domain samples themselves remain unchanged: only their associated domain-image pixels are refined as they are combined with those of neighboring pixel samples. In this way, the Z-indexed mapping ensures that each block of pixels receives a faithful, progressively refined view of the sampled domain.

The same interval-centric perspective extends naturally to higher-dimensional interleaving. Rather than combining points, the LD sequence effectively synthesizes voxels of joint spaces from pixels of individual subspaces, in order to convey an image of the combined spaces to pixels of the rendered image.

Following this approach, element sampler construction in NILE is fundamentally concerned with how pixel representations are defined and progressively refined across sampled subspaces. The following concepts and requirements will be used to guide element design, ensuring proper interleaving and uniform coverage of the combined sample space:

Sequencing: The sampled element must be subdivided to pixel-count×samples-per-pixel resolution, i.e., the overall sample count, and each domain pixel at this resolution must be assigned exactly once to a pixel sample.

Modularity: We require a *modular sequence*: a sequence of sample sets, just like (t, s)-sequences; this is what is missing from previous recursive tiling techniques that were optimized to deliver a single *progressive sequence*.

Resolution: We use this term to refer to the number of slices the element designer wants to see and control before recursive subdivision. Apart from the power-of-2 constraint, resolution is a free design parameter, though a high resolution may degrade the quality of the interleaving process. The geometry of the domain may dictate a minimal resolution, e.g., four for rectangular and triangular elements, to subdivide into congruent slices as needed for a good coverage.

Partitioning: Apart from the slicing geometry, usually implied by the geometry of the domain, the only degree of freedom for the designer is the indexing order of the slices. We are bound to partition the set into subsets of the target size that all comply with the design criteria, as dictated by the modularity target. For example, the 256

Algorithm 1: Retrieving a sample from a 2D NILE element

Input: A 2D element domain Ω and a binary fractional index $V \in [0, 1)$

Output: A sample $\mathbf{x} \in \Omega$

1 $S \leftarrow \Omega$; `// Initialize slice to the whole domain`
2 **repeat**
3 $\quad q \leftarrow \lfloor 4V \rfloor$; `//` $q \in \{0, 1, 2, 3\}$
4 $\quad V \leftarrow 4V - q$; `// Discard consumed digit`
5 $\quad S \leftarrow \text{subslice}(S, q)$; `// Replace` S `by its` q`-th child`
6 **until** *numeric precision is reached*;
7 Choose an arbitrary point $\mathbf{x} \in S$;
8 **return** $\mathbf{x}$;

slices in Fig. 1(f) are partitioned into 16 sets, each is a (i) non-digital (ii) dyadic $(0, 4, 2)$-net (iii) exhibiting blue noise properties.

Tiling: It is not only impractical to design for all possible sample counts, but a bad practice, too. Suppose, for example, that we aim at distributing 16 samples per pixel: we may choose a resolution of 256, as in Fig. 1(e, sub-intervals) and partition into 16 subsets, but then we would need not only to decide on the layout but also the indexing order of each set. A better plan is to slice into only 16, and use tiling techniques like ART to coordinate the placement of a single point per tile. That is, we design over two subdivision levels of the chosen resolution. The blue noise distribution in Fig. 1(e) is actually obtained at the native 2×2 resolution of the square, giving a good example of a conservative use of resolution.

Scrambling and Shuffling: We use these terms in the same sense as in the quasi-Monte Carlo literature, but through a different mechanism reflected in Eq. (11). Notably, shuffling has a tangible application here to decorrelate different elements.

Deep Nesting: Given the different roles of the sample indices described in Eq. (11), we can claim more freedom by nesting different elements inside others. For example, the StART element in Section 4.1 is optimized for random indexing, and we can therefore use it to provide the shuffling and scrambling components of the sample indices.

We conclude this section by explicitly highlighting an implied design constraint that the indexing order of intervals has to be static. That is, interval .011, for example, has to be strictly inside interval .01 and consistently visited after interval .010. This is crucial because permuting the order of intervals can ruin the orchestrated shuffling process of the LD sequence.

4 Example Elements

In this section, we present a few practical examples of elements designed to deliver a range of per-subspace sample distributions proposed previously while staying compliant with the binary indexing condition of the NILE architecture. While all elements share the same run-time logic outlined in Algorithm 1, they vary primarily in step 5, namely in how child slices are defined and indexed. Notably, this indexing is typically determined recursively and never materialized explicitly.

4.1 StART

Stratified sets commonly serve the first example of better-than-random distributions [Cook et al. 1984], so it is instructive to start by designing an element that provides this distribution, and use it as an example to demonstrate the design procedure.

A stratified set of n^2 points in the unit square is obtained by partitioning the domain into $n \times n$ strata (cells), placing a sample point randomly within each stratum. Obtaining a standalone set like this straightforwardly follows from the definition. Our goal is to build a modular sequence of samples such that, for all values of m, every consecutive block of $N = 2^{2m}$ points is stratified.

To imagine how to generate such a sequence, let us first consider the 1D case. The analogous definition of a modular stratified sequence aligns perfectly with the definition of dyadic $(0, 1)$-sequences. With the additional constraint of a static interval visiting order, it coincides precisely with the set of Owen-scrambled vDC sequences [1995; 1998]. Analogously, a quad-based modular sequence of stratified samples precisely coincides with Owen-scrambled base-4 vdC sequence, where digits are interpreted as quadrants. By definition, every consecutive power-of-4 block of such a sequence is indistinguishable from a stratified set of that size, and the sequence constraints only become apparent when considering successive blocks, which is exactly what we intend.

Thus, building a NILE element to serve stratified samples reduces to finding an efficient good-quality algorithm for base-4 Owen scrambling. This is a free design choice. In our implementation, we used the recently introduced Q-ART algorithm [Ahmed 2025], and obtained satisfactory results. It is based on the ART Grammar in Section 2.3, hence we use *Stratification with ART* (StART) to name the resulting element. We conclude this Section by making a historical note that the idea of incremental stratified sampling was proposed quite early in the work of Dippé and Wold [1985], but we are unaware of follow-ups, and this may count as successful realization of an old vision.

4.2 Jittered Grid

Instead of randomly jittering the location of individual sample points within each cell of a stratified grid, Ramamoorthi et al. [2012] argue for jittering the grid (JG) as a whole while preserving its regular structure. Constructing a basic element that serves a modular sequence of this distribution is surprisingly straightforward: the Z-order indexing [Morton 1966]

$$z = .y_1 x_1 y_2 x_2 y_3 x_3 \cdots \tag{12}$$

of quadrants readily provides such an element when applied over a vdC-indexed stream of samples. Replacing Morton's Z ordering with Bayer's used in ordered dithering further enables diagonal grids for handling odd powers of 2. Random jittering is implicitly achieved through the shuffled Z-ordering of pixels, which ensures that any displacement of the grid within the sampled domain has equal probability of being assigned to any pixel.

While the interleaving process can still attain a good coverage over combinations of intervals, the Morton ordering imposes strong correlation between individual axes across different subspaces. To see this, note that Morton indexing is a linear operation that implies

a pair of matrices for the two axes, with the net effect of replacing the indexing matrix in the LD sequence with a pair of matrices for independently sampling the two axes of the element. To avoid this, we nest the plain model above in a StART element that shuffles the indexing order of strata. Doing so preserves the jittered grid of each sample set, but union sets over pixel blocks are no longer regular grids.

4.3 Correlated Multi-Jitter

Chiu et al. [1994] introduced a variant of stratification, *multi-jitter*, where the samples are stratified in their 1D projections as well as the joint 2D space, thus satisfying the so-called N-Rook or Latin-hypercube property. More recently, Kensler [2013] proposed an improved variant, correlated multi-jitter (CMJ), where each dimension's sample points have the same offset within their cell, as illustrated in Fig. 1(d). This approach was further refined and extended by Christensen et al. [2018]. Another significant improvement is due to Jarosz et al. [2019], who extended CMJ sampling to higher dimensions.

Constructing a standalone CMJ distribution of $N = n \times n = 2^{2m}$ comes down to choosing two permutations of $\{0, \dots, n-1\}$, and reading through these sequences along columns and rows to assign sample offsets. To create an element that serves a modular sequence of this distribution, we need to augment each permutation by an orthogonal set of n non-overlapping permutations. The binary size of the permuted set readily suggests an easy solution using XOR scrambling against codes in $\{0, \dots, n-1\}$. This, however, will only cover the 1D projections, and to fill the 2D space we need a Cartesian product $\{0, \dots, n-1\} \times \{0, \dots, n-1\}$ of scrambling codes.

A geometric intuition of this can be obtained by carefully examining Fig. 1(d): each set must contain a point in the origin of the stratum; the 2D-pair of XOR-scramble codes translates that point uniquely for each complementing set, and all other points are bitwise-XOR translated by the same offset. Indexing the samples in the set, the sequencing of translations, and the jittering within the $2^{-m} \times 2^{-m}$ intervals may all be conveniently delegated to StART.

4.4 Blue Noise

With its digit-reversed indexing and binary production rate, the original ART model [Ahmed et al. 2017] fits readily in our scheme as an element for providing blue-noise sample distribution. Optimizing an ART tileset originally follows the same procedure discussed is Section 2.2. but we introduce a few modifications repurposing the model to deliver modular sequences for rendering. First, we replace the ranks by class indices, and optimize to obtain a blue-noise distribution over each class. At most we can optimize jointly for pairs of child indices to account for odd-power-of-2 sample counts, but child indices are not meant to coexist beyond that; instead, each index from different tiles would end up together in a sample set. This follows readily from the vdC indexing. Secondly, especially for non-adaptive sampling, we do not have to, and actually should not use the self-similar property of the optimized tileset to index over all generations. Instead, we use StART to randomly scramble/shuffle the trailing/indexing bits to improve the spectral quality within/across subspaces, respectively. Here we are taking advantage of the shared underlying ART scheme, with an opportunity of reusing the same tables.

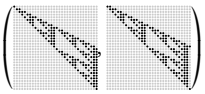
4.5 Triangles and Discs

The idea of implementing stratified sampling as Owen scrambling extends naturally to different domain geometries. In Fig. 1, for example, we use StART sampling with (g) triangles, using Basu–Owen indexing [2015], and (h) discs [Christensen 2018; Shirley and Chiu 1997]. The concept also admits different slicing patterns. For example, the Basu–Owen subdivision of a triangular domain can be replaced by the Talbot–Heitz [Pharr 2019] parametrization. Finally, the subdivision ratios can also be varied, subject to the provision of an efficient Owen-scrambling algorithm in the corresponding base.

4.6 3D Elements

A $(0, 2)$-sequence is arguably the most natural candidate for nesting. In its original form, using the identity–Pascal pair of matrices (I, P), the 2D Sobol' $(0, 2)$-sequence permutes the visiting order of intervals, violating the required static indexing-tree condition. Indeed, nesting (I, P) into the first dimension of the (I, P) pair itself gives (I, P, P), the worst possible outcome. A compliant version, however, is obtained by multiplying an appropriate matrix to undo this permutation, bringing the sequence to its *finite-row* matrix form [Hofer and Pirsic 2011]. Alternatively, the self-similar $(0, 2)$-sequence [Ahmed et al. 2023], generated by the pair of self-similar matrices (ξ, ξ^+), readily satisfies the static indexing of intervals thanks to its tile-based construction. We may consider the upper-triangular factors

$$(\xi_u, \xi_u^+) = \left(\quad \right) \tag{13}$$

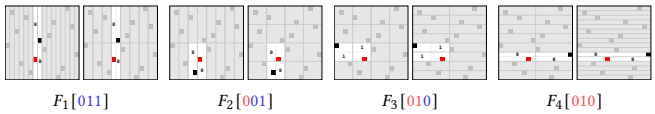


for faster evaluation via diagonal factoring [Ahmed 2024]. It so happens that ξ_u is exactly the anti-permuting matrix needed for Sobol, so both roads lead to the same pair of matrices in Eq. (13) that generate the same 2D Sobol' sequence but in an invariant visiting order of 2D intervals.

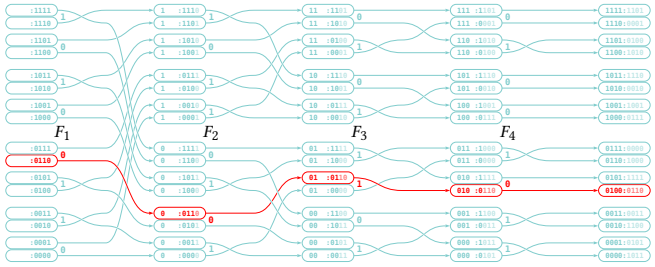
We experimented with nesting this sequence in the first dimension of the original 2D Sobol, and during assessment noted that the discrepancy plot of the resulting 3D sequence coincides with 3D Sobol. Further inspection revealed a very interesting result that the two sequences are actually identical, up to shuffling:

$$(\xi_u, \xi_u^+, P) = (I, P, S_2)\xi_u = (S_0, S_1, S_2)\xi_u\,, \tag{14}$$

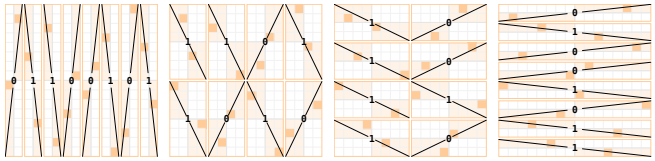
where S_0, S_1, and S_2 denote the first three matrices of the Sobol' sequence in Eq. (6). On the practical side, we see this finding as the most precious gem of nesting: it enables nesting a $(1, 3)$-sequence in any $(0, 2)$-sequence pair of the SZ sequence, providing a neat treatment to the 3D-sample alignment issue reported in the SZ-Sequences paper [Ahmed et al. 2025]. It is worth mentioning that the $(1, 3)$-sequence of 3D Sobol' is arguably the best available 3D sampling sequence, to our knowledge, and nesting avails it to any number of 3D subspaces in the scene. Depending on the context, we refer to this element and associated sampler using "O2", short for nested $(0, 2)$-sequence, or "One-Three", short for $(1, 3)$-sequence.



(a) Flags extracted from point $(0.0110, 0.0100)_2$. Red/blue bits belong to y/x, respectively.



(b) Netshuffle flow diagram, highlighting the point in (a) above



(c) Flags over the whole net

Fig. 5. (a) Geometric interpretation of flags evaluated from a single point. Strata are successively paired in each stratification. The construction of the net forces the two points sharing a pair to occupy diagonally-opposite quadrants of their shared area, and changing the diagonal is possible without violating the net structure, hence represents an independent bit parameter. (b) An abstraction of the Netshuffle parameterization for the whole net in (a), showing the influence of flags (large digits) in routing the association of xs to ys. Each column represents a stratification, and the number in each cell is a concatenation $y : x$ of m significant bits from the two coordinates, highlighting the relevant bits for indexing the strata. Adapted from [Ahmed and Wonka 2021, Fig. 4] to highlight the point in (a). (c) The geometric interpretation of the flags in (b) over the whole set, highlighting the occupied diagonal per pair per stratification. Flag vectors are read bottom-up in the flow diagram (b), and left-to-right bottom-up in the plots in (c).

4.7 BlueNets and WhiteNets

Ahmed and Wonka [2021] showed that any dyadic $(0, m, 2)$-net, up to its significant m-bit resolution, is uniquely defined by an ordered set

$$F = (F_1, F_2, \ldots, F_m) \tag{15}$$

of m vectors, each comprising 2^{m-1} binary flag bits. Each point defines m flags,

$$(0.x_1x_2\cdots x_m, 0.y_1y_2\cdots y_m) \;\mapsto\; (y_k \oplus x_{m-k+1})_{k=1}^{m}, \tag{16}$$

one for each vector in Eq. (15), and each flag in the set is defined by exactly two points: any violation disqualifies the set as a net. Fig. 5 illustrates the geometric interpretation of these flags. Briefly, for every consecutive pair of strata in each stratification, the two occupant points must lie in diagonally opposite quadrants, and the corresponding flag bit encodes which diagonal of quadrants is occupied. This compact parametrization implies an efficient construction algorithm, *Netshuffle*, and a handle for optimization via iterative manipulation of flags, giving rise to blue-noise-optimized nets, which Ahmed and Wonka termed *BlueNets*. They also showed that each net identified by a flag set like in Eq. (15) is associated with a unique complementary net,

$$\overline{F} = (\overline{F_1}, \overline{F_2}, \ldots, \overline{F_m}), \tag{17}$$

obtained by bitwise-complementing all flag vectors, which is the sole net that can pair with the given one to form a $(0, m+1, 2)$-net. Conversely, up to m-bit resolution, there is a unique pair of $(0, m, 2)$-nets, if any, to split a $(0, m+1, 2)$-net into, and their flags are all complements.

Starting with this definition of a complement net, our investigation reveals that a given net actually implies a whole *netset*

$$\left\{ \begin{array}{l} (F_1, F_2, \ldots, F_m), \\ (\overline{F_1}, F_2, \ldots, F_m), \\ (F_1, \overline{F_2}, \ldots, F_m), \\ (\overline{F_1}, \overline{F_2}, \ldots, F_m), \\ \quad\vdots \\ (\overline{F_1}, \overline{F_2}, \ldots, \overline{F_m}), \end{array} \right\} \tag{18}$$

of complements with the following interesting property:

Lemma 4.1. *A netset as defined in Eq.* (18) *partitions the domain at m-bit resolution.*

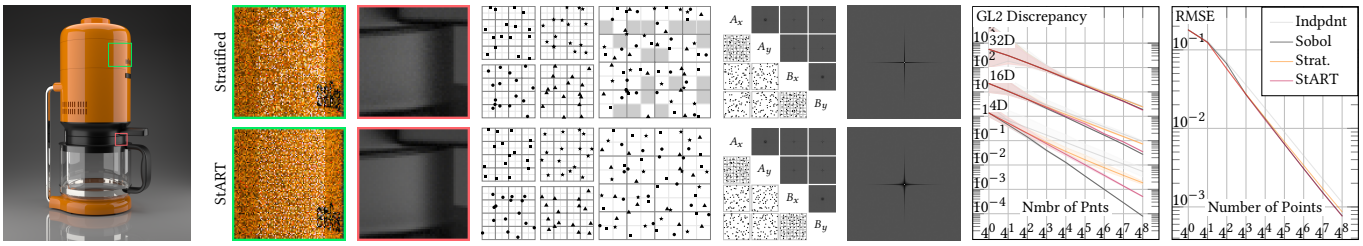
Proof. Every point traces a distinct path from x to y, as illustrated in Fig. 5, and marks a single flag bit in each flag vector at a position determined only by the point. If two nets in the set share a point then they would have at least one shared flag value per vector, leading to a contradiction that, by definition, they should have complementing flags in at least one vector. □

This netset formulation provides an easy way to produce a NILE element by applying an XOR scrambling code to the defining flag vectors, rather than directly to the points as in the CMJ case, reflecting the complex "multi-multi-jittered" nature of nets. Optimizing the netset is similar to optimizing an individual net, except that the optimization criteria need to be applied across all nets in the set. Alternatively, we can use unoptimized nets, which we refer to as "WhiteNets", in the sense of "white noise". We currently have WhiteNets in our supplementary code, leaving the actual optimization of BlueNets for future work. The demonstration in Fig. 1(e) is obtained by exhaustive search, which is feasible for 16-point nets over 2^{28} distinct netsets. Finally, as with CMJ, indexing, sequencing, and jittering may all be conveniently delegated to StART.

5 Results and Applications

Evaluating the NILE model is inherently challenging, as it is not a single sampler or a specific algorithm, but rather a blueprint for constructing customized samplers. To provide a meaningful evaluation, we therefore return to the model's original objective: enabling explicit user control of per-subspace sample distributions while preserving good coverage in the joint high-dimensional space. This motivates an evaluation plan that examines different aspects of the sampler and relates them to the expected rendering outcomes.

In what follows, unless otherwise noted, all scenes are rendered with *pbrt*-4 on GPU using the default scene settings, and all references are rendered with 2^{20}-spp StART. Global Sobol' uses default



(a) Rendered Scene (b) 4 spp (c) 4K spp (d) Same 2D across 2×2 Pixels (e) 2D×2D (f) $A_x \times B_x$ (g) GL2 Discrepancy (h) RMSE

Fig. 6. Rendering and sample-space comparison of traditional stratified sampling and the NILE-orchestrated StART sampler, showing: (a) a reference rendering of the Coffee Maker scene, (b) close-up views at low sample rates, (c) close-up views at high sample rates, (d) distribution of domain samples across 2×2-pixel blocks, (e) spatial and spectral plots of two subspaces and their axis-wise pairings at 64 spp, (f) close-ups of spectral plots of axis-wise pairings at 4K spp, showing the effect of LD interleaving, (g) generalized L_2 discrepancy over different dimensions, and (h) RMSE convergence plots. Coffee Maker scene from Benedikt Bitterli's *Rendering resources* [2016], rendered using a maximum path depth of 17.

Owen scrambling, while NILE samplers use ART-Owen scrambling—that is, 1D StART—for all 1D subspaces. Distribution names such as JG, BN, and CMJ refer to NILE-based samplers implementing these elements. We benchmark against standard *pbrt*-4 samplers, which serve as a common reference point for comparison with other samplers. For example, orthogonal arrays [Jarosz et al. 2019] pursue a similar goal of interleaving low-dimensional stratified sets into high-dimensional ones, but the authors report that "Our results show that high-dimensional QMC samples like Halton and Sobol' still provide better convergence compared to our techniques." In contrast, our results show that most NILE-based samplers consistently outperform Halton.

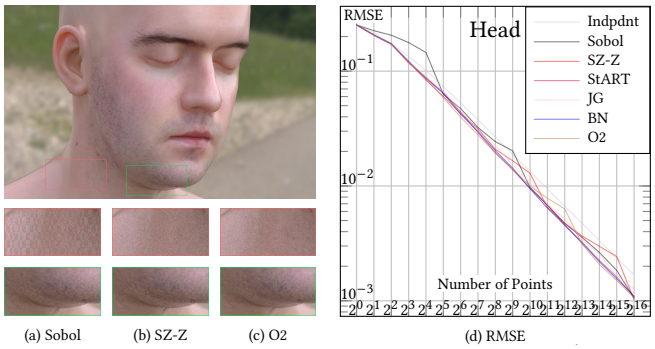
5.1 Z-Indexing and SZ-Interleaving

We begin by comparing a NILE-orchestrated distribution with its independently instantiated counterpart to evaluate the effectiveness of Z-indexing and interleaving. We choose our first stratified distribution, StART, and compare its performance to the native stratified sampler of *pbrt*-4, and benchmark against Independent and global Sobol' samplers for the worst and best expected performance. Fig. 6 shows the results we obtained for a selected scene, along with sample-space diagnostics trying to explain these results.

Exceeding our expectations, the StART sampler not only improves upon independent stratified sets, but in this scene even outperforms the long-established global Sobol' sampler. Specifically, thanks to the modular sequence structure, it is evidently benefiting from Z-indexing at low sampling rates, as visible in Fig. 6(b) and self-explained in (d), and leveraging SZ interleaving at high sampling rates, as seen in (c) and subtly reflected in (e, f).

5.2 Convergence

Next, we test from the other side, comparing NILE-augmented LD samplers to pure LD counterparts, for which we choose global Sobol' and Z-indexed SZ. Fig. 7 shows the convergence results we obtained for the "Head" test scene, which was reported as a failure case in the SZ paper [Ahmed et al. 2025]. All NILE-based samplers successfully avoid the artifacts of the underlying SZ sequence, but their convergence behavior differs depending on the local distribution. We first note that both Sobol' and native SZ experience worse-than-random



(a) Sobol (b) SZ-Z (c) O2 (d) RMSE

Fig. 7. RMSE convergence plots comparing miscellaneous NILE-based samplers to state-of-the-art LD samplers global Sobol' and Z-indexed SZ. The insets use (top) 64 and (bottom) 512 spp.

behavior for some sample rates, even high ones, most likely due to strongly-correlated pairs of dimensions. Notably, the O2-NILE sampler also experiences such a transient setback, and this is explainable by the linear operation: the net effect of nesting a $(0, 2)$-sequence in a $(0, 1)$ stream is replacing a single matrix of the underlying LD sequence with a pair of matrices, leading to a similar vulnerability to inter-axis correlation. The probability drops, though, thanks to (1) better alignment of subspaces with the SZ substructure, and (2) more effective utilization of earlier dimensions, which offer better joint-space coverage.

We also tested multiple other common scenes, as shown in Fig. 8. For all tested scenes, we observe that, except for JG, all NILE-based samplers attain the same order of convergence as the underlying SZ sequence, with O2 being the best, often outperforming the native SZ. As the "Head" scene in Fig. 7 demonstrates, however, the other stochastic elements are better behaved and therefore preferable for problematic scenes of this type. JG consistently performs worst, which aligns with its poor discrepancy properties. We note, however, that this does not necessarily warrant discarding this distribution altogether: it may, for example, still be suitable for sampling smooth integrands or low-variation subspaces.

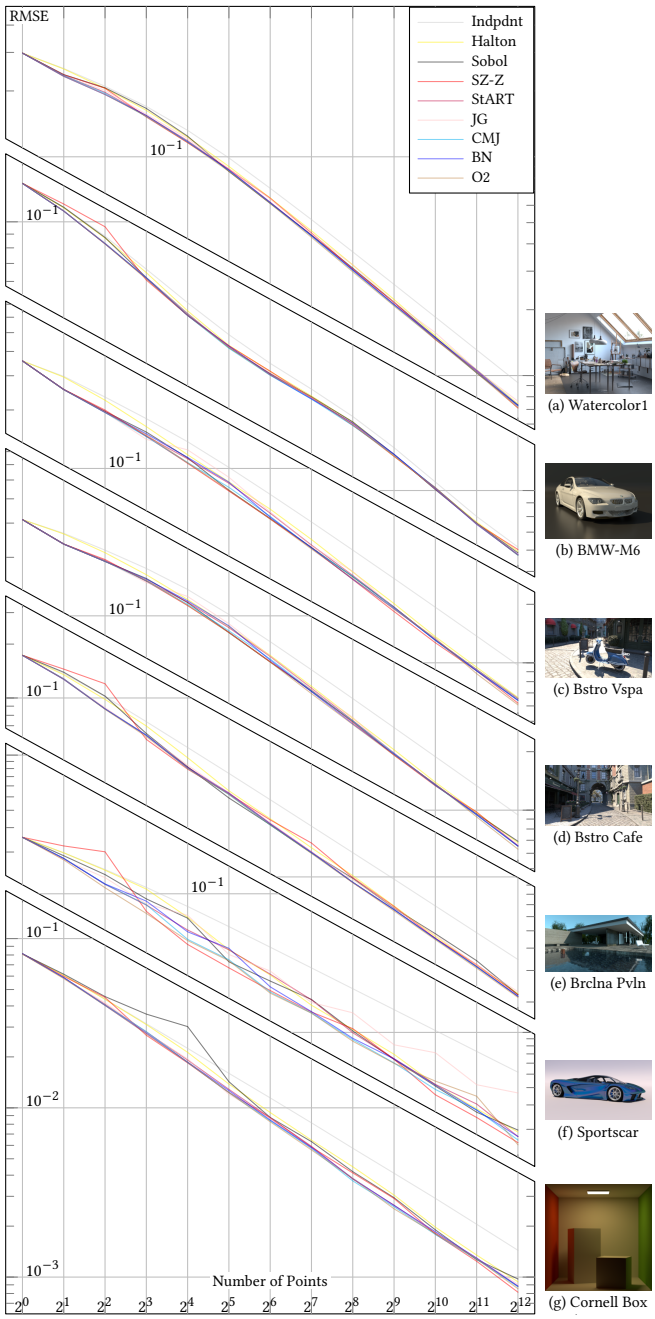


Fig. 8. Convergence plots for a few common *pbrt*-4 rendering scenes comparing NILE-based samplers to state-of-the-art LD samplers, along with the Independent sampler as a Monte Carlo baseline.

5.3 Screen-Space Performance

Here we evaluate the screen-space distribution of error, which is crucial for real-time rendering engines operating at low sampling rates. For this test, we use the BMW-M6 scene in Fig. 9, sampled at 4-spp, and compare NILE-based samplers to global Sobol' and Z-indexed SZ, and also to the native ZSobol sampler of *pbrt*-4, which is especially designed for this purpose. Insets are selected from two representative regions: a background area with a presumably smooth integrand, and a reflection zone exhibiting higher-dimensional light transport behavior. Overall, NILE-based samplers perform competitively in both: Sobol' exhibits the classic checkerboard patterns, SZ is evidently aliased in the reflection zone, while 2D Z is noisy in this region. Among the NILE variants, O2 is evidently the best performer, arguably followed by blue noise. The jittered grid performs well in the background region but suffers significantly in the reflection zone, supporting our argument in the preceding Section 5.2.

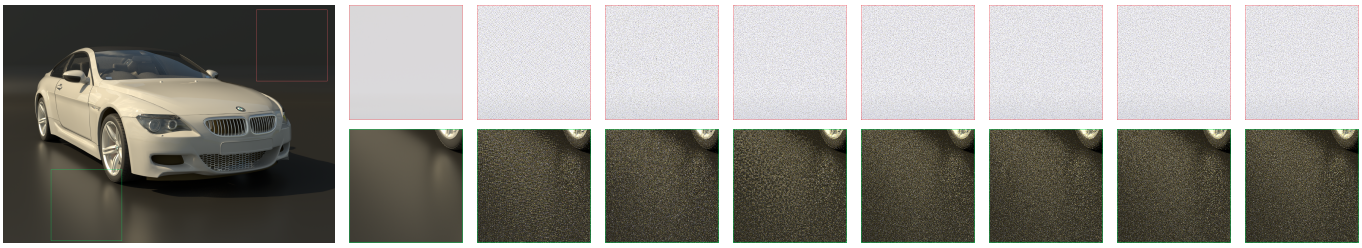
5.4 Sample-Space Distribution Quality

Next, we assess the impact of the modular sequence constraints on the distribution quality over the low-dimensional sample space. For this, we compare our modular ART-based blue noise to the native ART progressive blue-noise distribution, as demonstrated in Fig. 10, using the original code with recommended parameters from Ahmed et al. [2017]. The original progressive sequence model attains the best quality over the first 256-sample block, but the quality degrades and fluctuates over subsequent blocks, even reaching the stratified set baseline. This is because the progressive model is not designed to deliver separate blocks; instead, subsequent samples are optimized to complement preceding ones in a larger set. While our modularity constraint does degrade the distribution quality of the first set, it maintains the same quality over all blocks, and most notably attains a better overall average quality, as can be noted in the overall average spectrum. It is also worth noting that our modified sequence reduces the grid artifacts in the spectrum thanks to the random StART-based jittering suggested in Section 4.4.

5.5 Axis-Wise Decorrelation

Correlations between dimensions belonging to different subspaces are arguably a major source of artifacts when sampling with LD sequences, accounting for the characteristic checkerboard patterns of the global Sobol' sampler and the occasional low-frequency aliasing observed with Z-indexed SZ sampling. The sequential indexing of subspaces in NILE-based samplers provides an important byproduct in mitigating this issue through shuffling of intervals. Unlike axis-wise Owen scrambling, which preserves 2D interval structure, applying base-4 Owen scrambling to quadrants decorrelates the sampled domain dimensions from the indexing sequence, effectively occluding local domain axes from external visibility.

To demonstrate this property we choose the most regular element, the jittered grid, and benchmark against Sobol' and SZ sequences, as illustrated in Fig. 11. A useful property of the regular grid is that it literally instantiates the sampled-domain-as-image perspective on sampling discussed in Section 3.4. The plots in Fig. 11(a) show that a block of 1024 pixel samples receives a perfect 32×32-pixel image of each individual subspace, along with a noisy yet still recognizable image of paired dimensions from different subspaces, with skipped and duplicated pixels. In contrast, without shuffling, some subspaces form perfect images while others yield highly distorted



(a) Rendered Scene (b) Reference (c) Sobol (d) ZSobol (e) SZ-Z (f) StART (g) JG (h) BN (i) O2

Fig. 9. Screen-space error distribution comparing (f–i) NILE-based samplers to (b) global Sobol' and two Z-indexed samplers using (d) 2D Sobol' and (e) high-dimensional SZ. We invert the insets on the top for better visibility of the noise patterns. Full-resolution images are available in the supplementary materials.

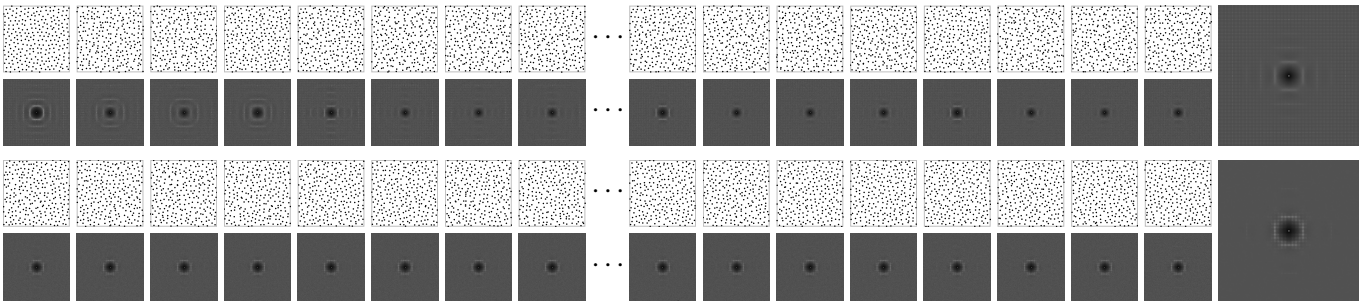
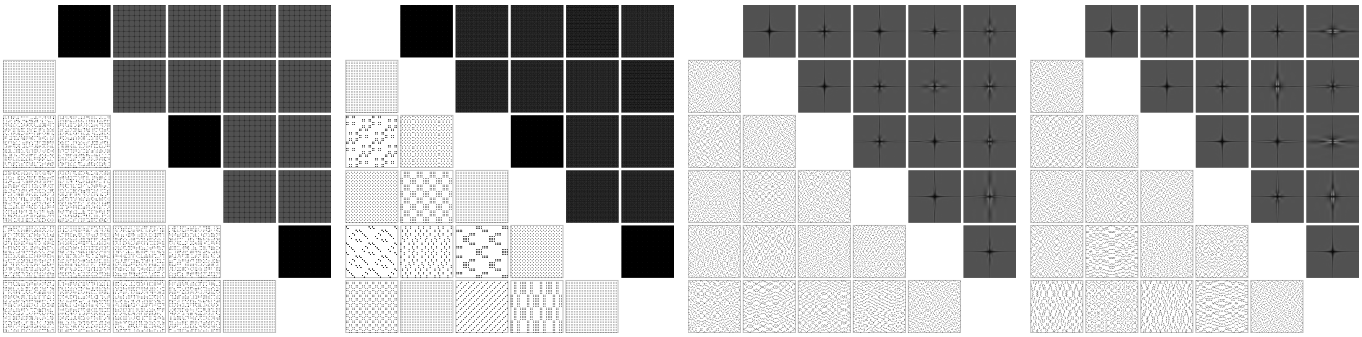


Fig. 10. Sample distribution and frequency power spectra (averaged over 100 realizations) of 256 consecutive blocks (showing only first and last eight) of 256 samples, along with the average spectrum over all sets, comparing (top) the original progressive ART blue noise to (bottom) our modular one.



(a) JG (shuffled) (b) JG (Morton) (c) Sobol (d) SZ

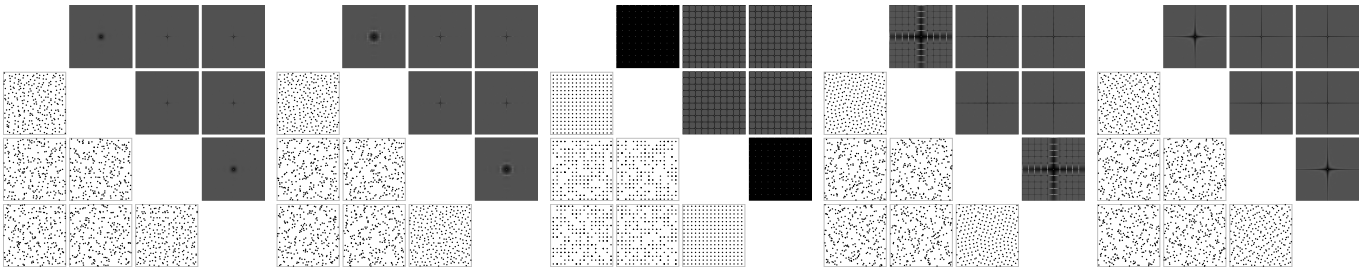
Fig. 11. Spatial and spectral plots for 3 consecutive 2D subspaces sampled at 1024 spp using jittered-grid NILE (a) with and (b) without shuffling, along with Owen-scrambled (c) Sobol' and (d) SZ shown for benchmarking.

ones, reproducing the behavior observed with LD sequences in a much more severe form.

More generally, we inspect the axis-wise sample-space interaction between subspaces sampled using similar and different elements, as demonstrated in Fig. 12 and Fig. 13, respectively. We note that pairwise plots of different elements capture features of both paired elements. For example, a StART×JG pair resembles StART×StART in one axis and JG×JG in the other.

5.6 Speed and Memory

Being binary-indexed sequences, the space and time complexity of nested elements is generally the same order as Sobol' and SZ sequences, differing only in the constant factors. While the two-layered processing and additional lookup tables may suggest an extra cost, there are in fact multiple cases where nested sampling actually reduces the computational and/or memory cost. For example, we use a 1kB lookup table for StART, equivalent to 8 dimensions of 32-bit Sobol' tables at 128 bytes each; the 2-in-1 packing makes the table sizes break even at 8×2D, and StART wins thereafter. The



(a) StART (b) BN (c) JG (d) CMJ (e) WhiteNet

Fig. 12. Spatial and spectral pairwise plots of 256 points spanning two 2D subspaces sampled using similar elements, indexed by consecutive SZ dimensions, demonstrating the interaction between similar elements.

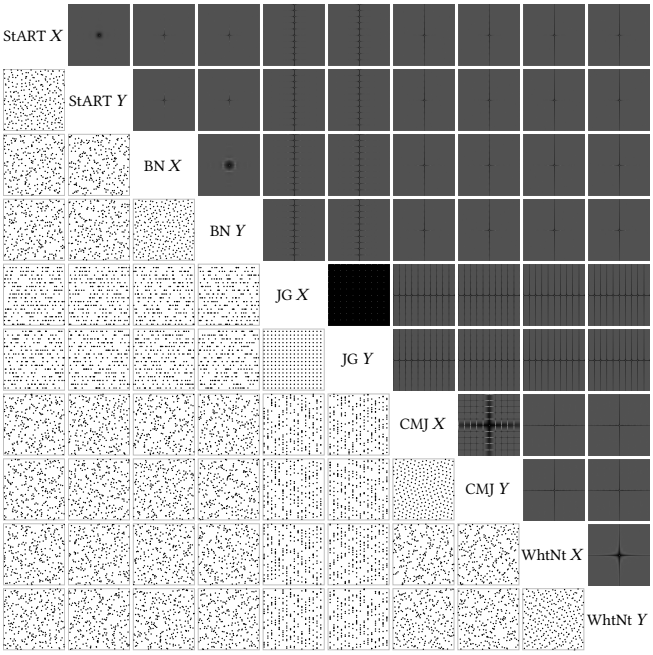


Fig. 13. Spatial and spectral pairwise plots of 256 points spanning five 2D subspaces sampled using different elements, indexed by consecutive SZ dimensions, demonstrating the interaction between different elements.

O2 NILE variant is tableless and therefore clearly advantageous in terms of memory. With diagonal factoring [Ahmed 2024], the evaluation code is extremely fast, hence O2 also wins in terms of speed, which is supported by empirical evidence throughout our experiments. Although the recursive traversal of ART trees is slower and less GPU-friendly than serial matrix processing in Sobol, the 2-in-1 feature makes the difference negligible. Besides such advantages, there is a subtle one that the computational cost of the underlying SZ sequence increases substantially after the first 16D, and nesting helps staying longer within these 16D.

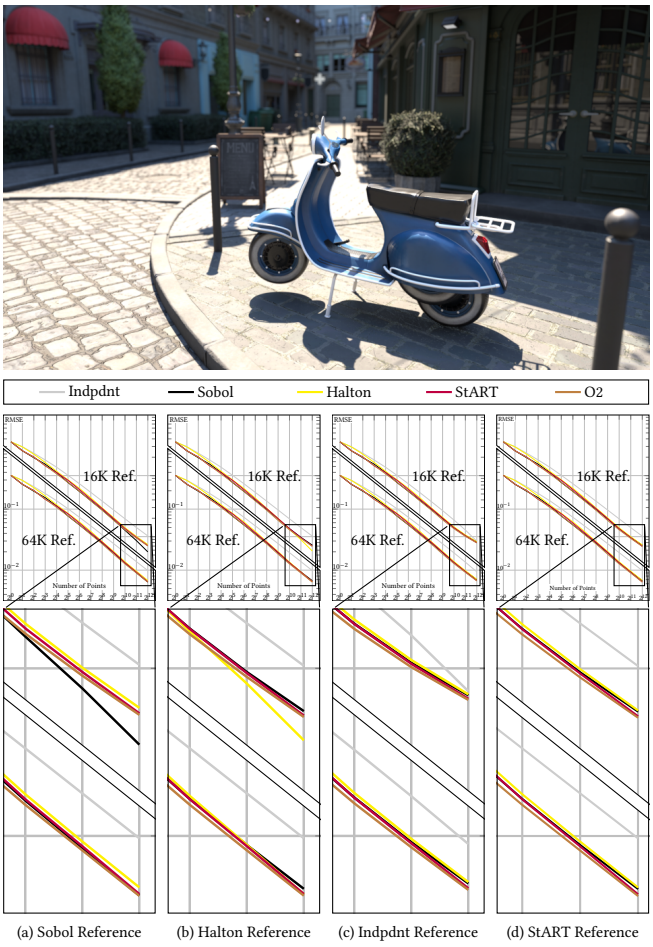
5.7 Bias

A tricky aspect of global samplers like Sobol' and Halton is that, when used to render reference images, they tend to favor candidates rendered with the same sampler, as demonstrated in Fig. 14. This behavior is now easier to understand in the light of our earlier analysis in Section 3.1. There is a good chance of sharing the same error on corresponding pixels between the reference and candidate images, and it may take multiple power-of-2 octaves to break the correlation. This has caused a long-standing challenge in benchmarking sampler performance, since generating a reliable reference may need impractical sample counts. In the other extreme, while the independent sampler is well-behaved, it unfortunately takes impractically large sample counts to converge.

A nice property of NILE samplers is that they apparently do not exhibit such a biased behavior, as illustrated in Fig. 14(d), where the plots obtained using 16K and 64K spp StART references are very similar, and neither claims the best place for StART-sampled images. This behavior is not exclusive to NILE, but is shared by all Z-indexed samplers. NILE, however, combines fast convergence with additional robustness by incorporating stochastic elements that reduce the likelihood of bias originating from the source subspaces. For this reason, we prefer StART over the faster-converging O2 when generating reference images, as the latter is more susceptible to aliasing.

5.8 Implementation Complexity

For most distributions, building a NILE-compliant element is admittedly more complex than independently generating the corresponding per-pixel sample set. That said, the NILE framework provides a platform for making fair comparisons. Indeed, established samplers like global Sobol, as well as newer ones like Z-indexed SZ, do not provide the researcher with an obvious handle to integrate their customized next-generation blue-noise distribution, for example, leaving them with no option but to build a complete sampler just to showcase their improvements. The schematic meta-sampler architecture in Fig. 1 gives an abstract idea about this saving in coding complexity, where the developer only needs to focus on a specific element. Furthermore, we will make our code freely available to encourage the adoption of this model.



(a) Sobol Reference (b) Halton Reference (c) Indpdnt Reference (d) StART Reference

Fig. 14. Convergence of a rendered scene measured against 16K- and 64K-spp references rendered using (a) global Sobol, (b) global Halton, (c) Independent, and (d) StART samplers. The Sobol' and Halton samplers exhibit evidently biased behavior at 4× the assessed sampling rates, favoring images rendered with the same sampler, and stay biased even at 16×. For ground truth, Fig. 8(c) shows the convergence plot for the same scene against a 2^{20}-spp StART reference, and we verified that a Sobol' reference at that rate produces almost identical curves.

5.9 Limitations

While NILE-based samplers have passed multiple empirical tests, the overall model is still heuristic and lacks analytically provable performance guarantees. Specifically, nested NILE sequences retain low discrepancy in the hypercube of subspaces when each subspace is treated as a single dimension, but not when subspaces are expanded into individual axes. Thus, we no longer have a known error bound such as the Koksma–Hlawka inequality. We believe this is an important area for future research.

On the practical side, we would like to emphasize that the convergence performance of NILE samplers comes primarily from the high-dimensional LD sequence. Hence, one should not expect NILE samplers to substantially outperform the underlying LD sampler in settings where full convergence is required. That said, the NILE framework still affords more user control at negligible cost.

6 Conclusion and Future Work

Our goal in this work was to bridge the long-standing gap in computer graphics between the local per-subspace and global high-dimensional perspectives on sampling. Our novel NILE meta-sampler has done so by using high-dimensional LD points to drive the operation of local samplers, leading to well-distributed high-dimensional points from low-dimensional components. In addition to reviving four decades of research on sample distribution over low-dimensional subspaces, we have also demonstrated that low-dimensional properties, e.g., blue noise, and high-dimensional coverage are not mutually exclusive and can be combined in a single sampler. Our results suggest that no specific distribution is always optimal; our modular sampler architecture enables the rendering engine architect to select the distribution that best matches their application. Even better, the model allows mixing-and-matching different elements, making it possible to fine-tune the sample distribution for a specific subspace without having to build a whole new sampler.

We hope that our work sparks a revival of interest in custom low-dimensional samplers and that it leads to greater adoption of existing ones in practice. We have shown that NILE works well with SZ as the underlying LD sequence and the Z-sampler used to distribute samples to pixels—these choices merit further investigation. It could be worthwhile to develop new LD sequences or sample distributors with this specific application in mind, for example. Having shown a practical use for the interval-interpretation of points generated by global samplers, we suspect that other applications of this idea will be found and that high-dimensional LD samplers will find further applications in computer graphics. Finally, the provision of traceable indexed samples in our model, as illustrated in Fig. 1, aligns naturally with modern machine-learning pipelines, opening new opportunities for learning-aware sampling.

Acknowledgments

This work was supported in part by Guangdong S&T Program (2024B0101050004), ICFCRT (W2441020), Shenzhen S&T Program (KQTD20210811090044003, KJZD20240903100022028), and Scientific Development Funds from Shenzhen University.

References

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

Abdalla G. M. Ahmed. 2024. An Implementation Algorithm of 2D Sobol Sequence Fast, Elegant, and Compact. In *Eurographics Symposium on Rendering (EGSR)*. The Eurographics Association. doi:10.2312/sr.20241147

Abdalla G. M. Ahmed. 2025. Q-ART Owen Scrambling. In *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association. doi:10.2312/cgvc.20251215

Abdalla G. M. Ahmed, Till Niese, Hui Huang, and Oliver Deussen. 2017. An Adaptive Point Sampler on a Regular Lattice. *ACM Trans. Graph.* 36, 4, Article 138 (July 2017), 13 pages. doi:10.1145/3072959.3073588

Abdalla G. M. Ahmed, Matt Pharr, Victor Ostromoukhov, and Hui Huang. 2025. SZ Sequences: Binary-Constructed (0, 2q)-Sequences. *ACM Trans. Graph.* 44, 6, Article 206 (Dec. 2025), 14 pages. doi:10.1145/3763272

Abdalla G. M. Ahmed, Mikhail Skopenkov, Markus Hadwiger, and Peter Wonka. 2023. Analysis and Synthesis of Digital Dyadic Sequences. *ACM Trans. Graph.* 42, 6, Article 218 (Dec. 2023), 17 pages. doi:10.1145/3618308

Abdalla G. M. Ahmed and Peter Wonka. 2020. Screen-Space Blue-Noise Diffusion of Monte Carlo Sampling Error via Hierarchical Ordering of Pixels. *ACM Trans. Graph.*

39, 6, Article 244 (Nov. 2020), 15 pages. doi:10.1145/3414685.3417881
Abdalla G. M. Ahmed and Peter Wonka. 2021. Optimizing Dyadic Nets. *ACM Trans. Graph.* 40, 4, Article 141 (July 2021), 17 pages. doi:10.1145/3450626.3459880
Kinjal Basu and Art B. Owen. 2015. Low Discrepancy Constructions in the Triangle. *SIAM J. Numer. Anal.* 53, 2 (Jan. 2015), 743–761. doi:10.1137/140960463
Benedikt Bitterli. 2016. Rendering Resources. https://benedikt-bitterli.me/resources/ Accessed: 2025-11-08.
Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, and Victor Ostromoukhov. 2025. Sobol' Sequences with Guaranteed-Quality 2D Projections. *ACM Trans. Graph.* 44, 4, Article 97 (July 2025), 16 pages. doi:10.1145/3730821
Paul Bratley and Bennett L Fox. 1988. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)* 14, 1 (1988), 88–100.
Kenneth Chiu, Changyaw Wang, and Peter Shirley. 1994. Multi-Jittered Sampling. In *Graphics Gems*, Paul S. Heckbert (Ed.). Academic Press, 370–374. doi:10.1016/B978-0-12-336156-1.50045-8
Per Christensen. 2018. Progressive Sampling Strategies for Disk Light Sources. *Pixar Technical Memo #18-02* (June 2018).
Per Christensen, Andrew Kensler, and Charlie Kilpatrick. 2018. Progressive Multi-Jittered Sample Sequences. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 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. doi:10.1145/882262.882265
Robert L. Cook. 1986. Stochastic Sampling in Computer Graphics. *ACM Trans. Graph.* 5, 1 (Jan. 1986), 51–72. doi:10.1145/7529.8927
Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed Ray Tracing. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '84)*. ACM, 137–145. doi:10.1145/800031.808590
Sabrina Dammertz and Alexander Keller. 2006. Image synthesis by rank-1 lattices. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, 217–236.
Mark A. Z. Dippé and Erling Henry Wold. 1985. Antialiasing through Stochastic Sampling. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*. ACM, 69–78. doi:10.1145/325334.325182
Henri Faure and Shu Tezuka. 2002. Another Random Scrambling of Digital (t,s)-Sequences. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*. Springer Berlin Heidelberg, 242–256.
Iliyan Georgiev and Marcos Fajardo. 2016. Blue-Noise Dithered Sampling. In *ACM SIGGRAPH 2016 Talks*. 1–1.
Andrew S. Glassner. 1994. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc.
Leonhard Grünschloß and Alexander Keller. 2009. (t, m, s)-Nets and Maximized Minimum Distance, Part II. In *Monte Carlo and Quasi-Monte Carlo Methods 2008*. Springer, 395–409.
Leonhard Grünschloß, Matthias Raab, and Alexander Keller. 2012. Enumerating Quasi-Monte Carlo Point Sequences in Elementary Intervals. In *Monte Carlo and Quasi-Monte Carlo Methods 2010*. Springer Berlin Heidelberg, 399–408.
Eric Heitz and Laurent Belcour. 2019. Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds Between Frames. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 149–158.
Eric Heitz, Laurent Belcour, V. Ostromoukhov, David Coeurjolly, and Jean-Claude Iehl. 2019. A Low-Discrepancy Sampler That Distributes Monte Carlo Errors as a Blue Noise in Screen Space. In *ACM SIGGRAPH 2019 Talks* (Los Angeles, California) *(SIGGRAPH '19)*. ACM, Article 68, 2 pages. doi:10.1145/3306307.3328191
Andrew Helmer, Per Christensen, and Andrew Kensler. 2021. Stochastic Generation of (t, s) Sample Sequences. In *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association. doi:10.2312/sr.20211287
Roswitha Hofer and Isabel Pirsic. 2011. An Explicit Construction of Finite-Row Digital (0,s)-Sequences. *Uniform Distribution Theory* 6, 2 (2011), 13–30.
Wojciech Jarosz, Afnan Enayet, Andrew Kensler, Charlie Kilpatrick, and Per Christensen. 2019. Orthogonal Array Sampling for Monte Carlo Rendering. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 135–147.
Stephen Joe and Frances Y. Kuo. 2008. Constructing Sobol Sequences with Better Two-Dimensional Projections. *SIAM J. Sci. Comput.* 30, 5 (Aug. 2008), 2635–2654. doi:10.1137/070709359
Alexander Keller. 1996. Quasi-Monte Carlo Radiosity. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (Porto, Portugal). Springer-Verlag, 101–110.
Alexander Keller. 2004. Stratification by rank-1 lattices. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*. Springer, 299–313.
Alexander Keller. 2006. Myths of Computer Graphics. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer, 217–243.
Andrew Kensler. 2013. Correlated Multi-Jittered Sampling. *Pixar Technical Memo 13-01* 7 (2013), 86–112.
Thomas Kollig and Alexander Keller. 2002. Efficient Multidimensional Sampling. In *Computer Graphics Forum*, Vol. 21. 557–563.
Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. 2006. Recursive Wang Tiles for Real-Time Blue Noise. *ACM Trans. Graph.* 25, 3 (July 2006), 509–518. doi:10.1145/1141911.1141916
Ares Lagae and Philip Dutré. 2006. An Alternative for Wang Tiles: Colored Edges versus Colored Corners. *ACM Transaction on Graphics*, 25, 4 (2006), 1442–1459.
Ares Lagae and Philip Dutré. 2006. Generating Well-Distributed Point Sets with a Self-Similar Hierarchical Tile. *Report CW* 462 (2006).
Ares Lagae and Philip Dutré. 2008. A Comparison of Methods for Generating Poisson-Disk Distributions. In *Computer Graphics Forum*, Vol. 27. 114–129.
Christiane Lemieux. 2009. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer. doi:10.1007/978-0-387-78165-5
M. Lothaire. 2002. *Algebraic Combinatorics on Words*. Cambridge University Press.
Ricardo Marques, Christian Bouville, Mickaël Ribardiere, Luis Paulo Santos, and Kadi Bouatouch. 2013. Spherical Fibonacci Point Sets for Illumination Integrals. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 134–143.
Don Mitchell. 1992. Ray Tracing and Irregularities of Distribution. In *Third Eurographics Workshop on Rendering Proceedings*. 61–69.
Don P. Mitchell. 1987. Generating Antialiased Images at Low Sampling Densities. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. ACM, 65–72. doi:10.1145/37401.37410
Don P. Mitchell. 1991. Spectrally Optimal Sampling for Distribution Ray Tracing. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. ACM, 157–164. doi:10.1145/122718.122736
GM Morton. 1966. A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. (1966).
Harald Niederreiter. 1987. Point Sets and Sequences with Small Discrepancy. *Monatshefte für Mathematik* 104, 4 (1987), 273–337.
Harald Niederreiter. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM.
Victor Ostromoukhov. 2007. Sampling with Polyominoes. *ACM Trans. Graph.* 26, 3, Article 78 (July 2007). doi:10.1145/1276377.1276475
Victor Ostromoukhov, Nicolas Bonneel, David Coeurjolly, and Jean-Claude Iehl. 2024. Quad-Optimized Low-Discrepancy Sequences. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 72, 9 pages. doi:10.1145/3641519.3657431
Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. 2004. Fast Hierarchical Importance Sampling with Blue-Noise Properties. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) *(SIGGRAPH '04)*. ACM, 488–495. doi:10.1145/1186562.1015750
Art B. Owen. 1995. Randomly Permuted (t, m, s)-Nets and (t, s)-Sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*. Springer, 299–317.
Art B Owen. 1998. Scrambling Sobol'and Niederreiter–Xing Points. *Journal of complexity* 14, 4 (1998), 466–489.
Matt Pharr. 2019. Adventures in Sampling Points on Triangles. (2019). https://pharr.org/matt/blog/2019/03/13/triangle-sampling-1.5 Accessed: 2025-12-24.
Matt Pharr and Greg Humphreys. 2004. *Physically-Based Rendering: from Theory to Implementation* (1st ed.). Morgan Kaufmann Publishers Inc.
Ravi Ramamoorthi, John Anderson, Mark Meyer, and Derek Nowrouzezahrai. 2012. A Theory of Monte Carlo Visibility Sampling. *ACM Trans. Graph.* 31, 5, Article 121 (2012), 16 pages. doi:10.1145/2231816.2231819
Peter Shirley. 1991. Discrepancy as a Quality Measure for Sample Distributions. In *Proc. Eurographics '91*, Vol. 91. 183–194.
Peter Shirley and Kenneth Chiu. 1997. A Low-Distortion Map between Disk and Square. *Journal of Graphics Tools* 2, 3 (1997), 45–52.
Il'ya Meerovich Sobol'. 1967. On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7, 4 (1967), 784–802.
Il'ya Meerovich Sobol', Danil Asotsky, Alexander Kreinin, and Sergei Kucherenko. 2011. Construction and Comparison of High-Dimensional Sobol' Generators. *Wilmott* 2011, 56 (2011), 64–79. doi:10.1002/wilm.10056
Il'ya Meerovich Sobol', VI Turchaninov, Yu L Levitan, and BV Shukhman. 1992. Quasi-Random Sequence Generators. *Keldysh Institute of Applied Maths, Moscow* (1992).
Robert A Ulichney. 1993. Void-and-Cluster Method for Dither Array Generation. In *IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology*. International Society for Optics and Photonics, 332–343.
J.G. van der Corput. 1935. Verteilungsfunktionen. *Proceedings of the Nederlandse Akademie van Wetenschappen* 38 (1935), 813–821.
Florent Wachtel, Adrien Pilleboue, David Coeurjolly, Katherine Breeden, Gurprit Singh, Gaël Cathelin, Fernando de Goes, Mathieu Desbrun, and Victor Ostromoukhov. 2014. Fast Tile-Based Adaptive Sampling with User-Specified Fourier Spectra. *ACM Trans. Graph.* 33, 4, Article 56 (July 2014), 11 pages. doi:10.1145/2601097.2601107
Hao Wang. 1965. Games, logic and computers. In *Computation, Logic, Philosophy: A Collection of Essays*. Springer, 195–217.