Eurographics Symposium on Rendering 2019
T. Boubekeur and P. Sen
(Guest Editors)

Volume 38 (2019), Number 4

Orthogonal Array Sampling for Monte Carlo Rendering

Wojciech Jarosz[1] Afnan Enayet[1] Andrew Kensler[2] Charlie Kilpatrick[2] Per Christensen[2]

[1]Dartmouth College [2]Pixar Animation Studios

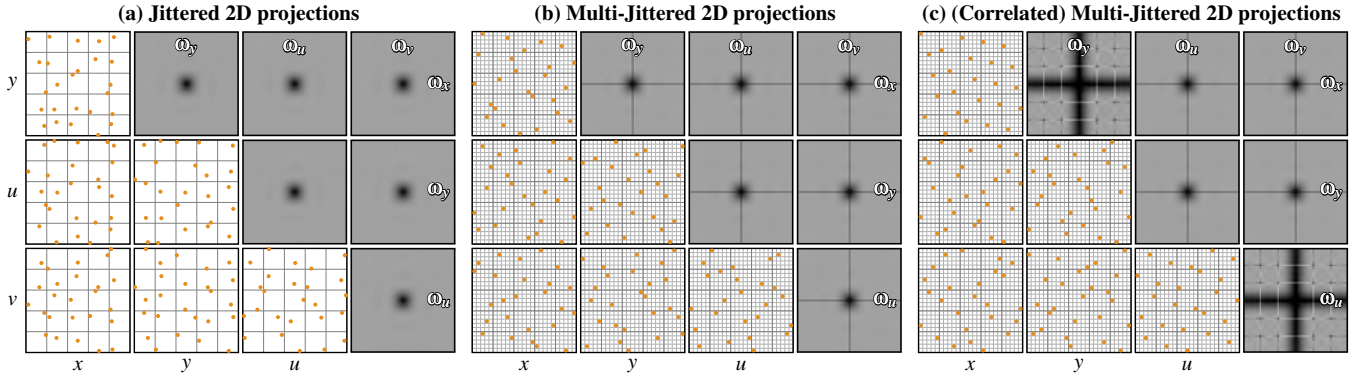


Figure 1: *We create high-dimensional samples which* simultaneously *stratify all bivariate projections, here shown for a set of 25 4D samples, along with their six stratified 2D projections and expected power spectra. We can achieve 2D jittered stratifications (a), optionally with stratified 1D (multi-jittered) projections (b). We can further improve stratification using correlated multi-jittered (c) offsets for primary dimension pairs (xy and uv) while maintaining multi-jittered properties for cross dimension pairs (xu, xv, yu, yv). In contrast to random padding, which degrades to white noise or Latin hypercube sampling in cross dimensional projections (cf. Fig. 2), we maintain high-quality stratification and spectral properties in all 2D projections.*

Abstract

We generalize N-rooks, jittered, and (correlated) multi-jittered sampling to higher dimensions by importing and improving upon a class of techniques called orthogonal arrays *from the statistics literature. Renderers typically combine or "pad" a collection of lower-dimensional (e.g. 2D and 1D) stratified patterns to form higher-dimensional samples for integration. This maintains stratification in the original dimension pairs, but looses it for all other dimension pairs. For truly multi-dimensional integrands like those in rendering, this increases variance and deteriorates its rate of convergence to that of pure random sampling. Care must therefore be taken to assign the primary dimension pairs to the dimensions with most integrand variation, but this complicates implementations. We tackle this problem by developing a collection of practical, in-place multi-dimensional sample generation routines that stratify points on all t-dimensional and 1-dimensional projections* simultaneously. *For instance, when t=2, any 2D projection of our samples is a (correlated) multi-jittered point set. This property not only reduces variance, but also simplifies implementations since sample dimensions can now be assigned to integrand dimensions arbitrarily while maintaining the same level of stratification. Our techniques reduce variance compared to traditional 2D padding approaches like PBRT's (0,2) and Stratified samplers, and provide quality nearly equal to state-of-the-art QMC samplers like Sobol and Halton while avoiding their structured artifacts as commonly seen when using a single sample set to cover an entire image. While in this work we focus on constructing finite sampling point sets, we also discuss potential avenues for extending our work to progressive sequences (more suitable for incremental rendering) in the future.*

CCS Concepts

• ***Computing methodologies*** → ***Computer graphics;*** *Ray tracing;* • ***Theory of computation*** → ***Generating random combinatorial structures;*** • ***Mathematics of computing*** → *Stochastic processes; Computations in finite fields;*

1. Introduction

Rendering requires determining the amount of light arriving at each pixel in an image, which can be posed as an integration problem [CPC84]. Unfortunately, this is a particularly challenging integration problem because the integrand is very high dimensional, and it often has complex discontinuities and singularities. Monte Carlo (MC) integration—which numerically approximates integrals by evaluating and averaging the integrand at several sample point locations—is one of the only practical numerical approaches for such problems since its error convergence rate (using random samples) does not depend on the dimensionality of the integrand. Taking samples often corresponds to tracing rays, which can be costly in

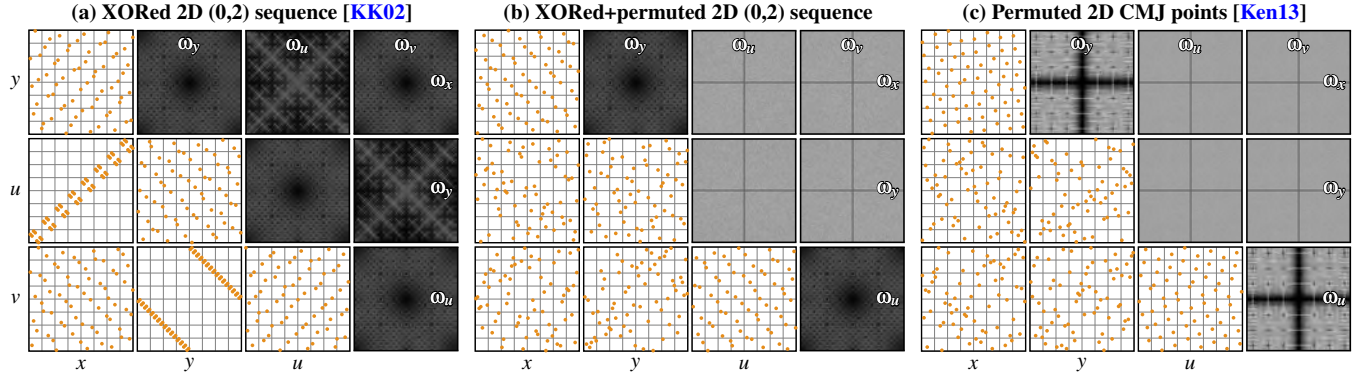


Figure 2: *A common way to create samples for higher-dimensional integration is to pad together high-quality 2D point sets. (a) Kollig and Keller [KK02] proposed scrambling a* $(0,2)$ *sequence by XORing each dimension with a different random bit vector. While odd-even dimension pairings produce high-quality point sets, even-even (xu) or odd-odd (yv) dimensions have severe deficiencies. These issues can be eliminated by randomly shuffling the points across dimension pairs (b and c), but this decorrelates all cross dimensions, providing no 2D stratification.*

complicated scenes, so a common goal is to optimize speed by rendering accurate images using as few samples as possible.

While N *uncorrelated* random samples result in variance proportional to N^{-1}, using carefully *correlated* samples (with no dense clumps or large gaps) can produce high-quality images at a much faster rate [Coo86; DW85; Mit91; Mit96; Özt16; PSC*15; RAMN12; SJ17; SK13; SMJ17; SNJ*14]. A common approach to generate such samples is stratification (or jittering) which divides the sampling domain into regular, even-sized strata and places one sample in each [Coo86]. Multi-jittering [CSW94] takes this one step further by ensuring stratification in both 2D and 1D. Rendering is, however, an inherently high-dimensional integration problem, but it is unclear how to extend such random sampling approaches to simultaneously stratify in successively higher dimensions.

It is tempting to switch to deterministic (optionally randomized) quasi-Monte Carlo (QMC) [Nie92] sampling techniques which extend to higher dimensions and are carefully crafted to provide additional stratification guarantees. Unfortunately, common QMC sequences like Halton [Hal64] and Sobol [Sob67] have increasingly poor stratification in (pairs of) higher dimensions [CFS*18; CKK18] (even with good randomization [JK08]), resulting in erratic convergence. They are also prone to structured artifacts which can be visually more objectionable and harder to filter away [ZJL*15] than the high-frequency noise arising from random MC integration.

To overcome these challenges, renderers in practice typically combine or "pad" a collection of 2D points to form higher-dimensional samples [CFS*18; KK02; PJH16]. In this way, each successive pair of dimensions is well stratified, providing some variance reduction. Unfortunately, this does not provide stratification over the higher-dimensional domain or in 2D projections which span different stratified dimension pairs, see Fig. 2. In practice, this also means that the primary dimensions of the sampler must be carefully assigned to the important dimensions of the integrand to best leverage the stratification. Even so, the lack of some stratification means that the convergence rate is destined to degrade back to that of uncorrelated random sampling [SJ17]. In short, none of the sampling patterns used in rendering are well-stratified simultaneously in all higher dimensions and all lower-dimensional projections [CFS*18].

Contributions. In this paper we take a step to overcome these limitations by extending jittered and multi-jittered samples so that they are stratified across all (and not just some) lower-dimensional (e.g. all 2D) projections, see Fig. 1. We accomplish this by building on the concept of orthogonal arrays (OAs) [HSS99; Rao47], which, to our knowledge, are unexplored in the rendering community. Literature on orthogonal arrays typically targets statistical experiment design, where the computational constraints and terminology is quite different from our application in Monte Carlo integration. One of the goals of this paper is to introduce these ideas to graphics and provide a sort of "Rosetta stone" (Sec. 3) into this literature for our community. Since these techniques are often applied in fields where construction speed is not a concern (e.g., where a single set of samples is used for an experiment that spans days or months), efficient implementations are often unavailable. To fill this gap, we propose novel, in-place construction routines (Sec. 4) which make orthogonal array sampling practical for rendering. In this paper, we focus on constructing *finite sampling point sets* and leave progressive sequences (more suitable for incremental rendering) as future work—see the discussion in Sec. 6. Our collection of sampling routines results in a useful hybrid between the strengths of stochastic approaches like (multi-) jittered sampling (but stratified in more dimensions) and scrambled deterministic QMC sampling (but without the structured artifacts and poor stratification in projections). To demonstrate the benefits and trade-offs of the proposed methods we perform an empirical evaluation (Sec. 5) on both simple analytic integrands and rendered scenes. Finally, we discuss limitations to our approach (Sec. 6) and identify additional promising work from this field which could provide fruitful avenues for further investigation.

2. Related Work

MC and QMC sampling in graphics. Pure MC integration using *uncorrelated* random samples has a fixed convergence rate independent of the (dimensionality of the) integrand. However, today we have a large arsenal of carefully *correlated* sample patterns which are stochastic, yet enforce certain *spatial* (stratification) or *spectral* (Fourier) properties to obtain improved variance and convergence rate. Recent surveys [ÖS18; SÖA*19; SSJ16] provide an excellent historical perspective and overview of the latest developments.

Techniques typically target so-called "blue-noise" frequency spectra [BSD09; Coo86; HSD13; LD08; RRSG16; SZG*13]/Poisson-disk sampling [MF92; Mit91] or enforce stratification constraints to reduce clumping using e.g., uniform [PKK00; RAMN12] jitter [Coo86], Latin hypercube [MBC79]/N-rooks [Shi91], correlated [Ken13], in-place [Ken13], and progressive [CKK18] variants of multi-jittered [CSW94] sampling, and various flavors of low-discrepancy [Shi91] QMC sampling [Hal64; HM56; Kel13; KK02; KPR12; LP01; Nie92; Sob67]. Recent work has even combined both spatial and spectral properties in the form of low-discrepancy blue-noise [PCX*18]. We introduce a new class of multi-dimensional stratified sampling techniques to graphics from statistics.

Higher-dimensional integration. While rendering is an inherently high- (or infinite-) dimensional integration problem, most state-of-the-art techniques focus entirely on the 2D problem. Independent random sampling and most QMC approaches generalize (in theory), but higher-dimensional variants of the aforementioned techniques are unfortunately either unknown (multi-jitter [CKK18; CSW94; Ken13]), are prohibitively expensive to generate (blue noise), have poor stratification properties in higher dimensions or their projections [JK08] (QMC sequences), produce distracting structured artifacts (a single QMC sequence applied to an image), or provide diminishing returns due to the curse of dimensionality (jitter [CPC84]). The orthogonal array (OA) sampling techniques we introduce provide a middle ground between stratified- and QMC-sampling which directly address each of these limitations.

In light of these high-dimensional sampling challenges, practical implementations [KK02; PJH16] typically generate multiple low-dimensional samples which are randomly combined to form higher-dimensional points. For instance, a 4D sample (for e.g., pixel anti-aliasing plus depth of field) could be created by combining the 2D coordinates of two separate 2D stratified points (one for xy and one for uv, see Fig. 2) while permuting ("shuffling") the order across pairs of dimensions. This was originally proposed in graphics under the name "uncorrelated jitter" by Cook et al. [CPC84], but is now more commonly referred to as "padding." Such samples can be combined using independent random permutations, Owen scrambling [Owe97], or random digit XOR scrambling [KK02]. Padding is a generalization of Latin hypercube sampling (which pads n 1D stratified points to form an nD sample). Unfortunately, just as with Latin hypercube sampling, while the primary dimension pairs may be well stratified, the resulting samples are not well stratified in the higher-dimensional domain ($xyuv$) or in cross-dimension projections (xu, xv, yu, yv), see Fig. 2. Recent work has made progress on preserving stratification for some subset of the non-primary dimension pairs via precomputed permuted tables [CFS*18] or "magic" shuffling [KCSG18], but a general solution has remained elusive. Unfortunately, lacking stratification in *any* projection or the full high-dimensional domain will degrade convergence rate to that of independent random sampling, $\mathcal{O}(N^{-1})$ [SJ17].

The OA sampling techniques we introduce stratify simultaneously in all 1-dimensional and t-dimensional projections, for $t \geq 2$.

Orthogonal Arrays. Orthogonal arrays (OAs) [BB52; BP46; Bus52; HSS99; Rao47] are a mathematical concept from combinatorial design which have become essential in the statistical design of scientific experiments, automated software testing [Pre10], and cryptography. Owen [Owe92] later applied them to numerical integration and provides a recent overview [Owe13] of these techniques in relation to other stratification methods and QMC. In essence, orthogonal arrays generalize N-rooks, jittered, and multi-jittered sampling and can be interpreted as producing a dD point set which together is stratified in all possible tD projections, for some $t \leq d$ (e.g., $d = 5$ and $t = 2$). Orthogonal array-based Latin hypercube designs [Tan93; Tan98] impose this stratification for some chosen t but additionally ensure that the samples form a valid Latin hypercube where all samples project to distinct strata in each 1D projection. Jittered samples are OAs where $t = d = 2$, and multi-jittered is the corresponding "Latinized" version, but importantly OAs allow for both t and d to be greater than 2. While this is an old but still active area of research [AKL16; CQ14; DS14; HCT18; HHQ16; HQ11; HT12; HT14; LL15; PLL09; SL06; SLL09; Wen14; YLL14], most work in this field is difficult to adapt to rendering since it is often concerned merely with existence proofs, or theoretical variance analyses, with less emphasis on efficient construction algorithms. (A brute-force search is acceptable when a single OA is needed before a much more time-consuming physical experiment. Pre-tabulated arrays are also common.)

To the best of our knowledge, OAs have not previously been used in computer graphics, though they are mentioned in passing by Veach (who was co-advised by Owen) in his dissertation [Vea97]. Our primary contributions are to introduce these techniques to rendering, to demonstrate their benefit in rendering using novel construction routines, and to provide a bridge between the OA literature and rendering to guide future work. In the supplemental document we additionally provide a "further reading list" with our suggested sequence of references for graphics researchers to most quickly get ramped up on the concepts.

3. Background

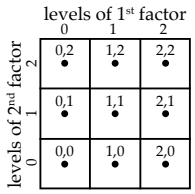
3.1. A primer on orthogonal arrays

Experiment design. An experiment typically seeks to determine how one or more *factors* (or *variables*), the values of which range over a finite set of *levels* (or symbols), influence a particular outcome. For instance, we might want to know how well a plant grows based on two factors: amount of sunlight and presence of fertilizer; or whether a chemical reaction occurs based on five factors: the presence of three possible compounds, a catalyst, and the temperature. To determine this, we need to perform a number of *runs* or *trials* of our experiment, where each run specifies particular *levels* (or values) for the combination of factors, e.g., no sun but lots of fertilizer.

If the number of factors and levels is small, then it may be feasible to perform a *full factorial experiment* which tests all possible combinations of levels across all factors. The total number of such combinations for d factors with s levels each is s^d, so this is unfortunately only practical for small problems. When a full exploration of the parameter space is infeasible, an alternative is to use a *fractional factorial design* that significantly reduces the sampling space by omitting certain combinations. Orthogonal arrays are a mathematical tool to determine a suitable subset of combinations to evaluate.

Table 1: *An orthogonal array* $OA(N,d,s,t)$ *with* $N = 9$ runs, $d = 4$ factors (x,y,u,v), $s = 3$ levels $(0,1,2)$ *and* strength $t = 2$ *expressed (left) as a table of levels with runs as columns and factors as rows. In each pair of factors, every possible* 2*-tuple occurs exactly the same* ($\lambda = N/s^t = 1$) *number of times* (λ *is called the* index *of the array). By interpreting each run as a sample point, and each level as a coordinate, any choice of two distinct factors (e.g., x and y) plots (right) a regular grid of* $s^t = 9$ *points.*

	runs:	1	2	3	4	5	6	7	8	9
factors	x:	0	0	0	1	1	1	2	2	2
	y:	0	1	2	0	1	2	0	1	2
	u:	0	1	2	1	2	0	2	0	1
	v:	0	1	2	2	0	1	1	2	0



Orthogonal arrays. An orthogonal array specifies the levels to test for each factor across a series of runs. We can encode this in 2D tabular form with *runs* or *trials* of the experiment enumerated along one axis and the *factors* or *variables* whose effects are being analyzed along the other. The entries in the array are elements from an alphabet of *symbols* (e.g., the numbers $0,1,2$) that indicate the levels at which the factors will be applied.

Table 1 (left) shows an example orthogonal array with 4 rows enumerating the factors and 9 columns enumerating the runs (note that the literature sometimes swaps the mapping of rows/columns to runs/factors, so we will typically refer to runs and factors to avoid confusion). Each array entry in this example takes on one of 3 levels: $0,1,2$, which could indicate three possible levels of sunlight for one factor or three possible types of fertilizer for another factor. Here we have simply labeled the four factors abstractly as x,y,u,v.

This is an orthogonal array of *strength* 2, which means that we can take any two factors (say the first two),

x:	0	0	0	1	1	1	2	2	2
y:	0	1	2	0	1	2	0	1	2

,

and we see that each of the 9 possible 2-tuples from $\{0,1,2\}^2$: $\{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$, occur as runs *the same number of times* t (in this case once each, so $t = 1$). Basing an experiment on an orthogonal array of strength t ensures that we analyze not just the effect of individual factors, but the *interaction* of all possible combinations of up to t factors.

Definition: An $N \times d$ array A, with entries from a set $S = \{0,\ldots,s-1\}$ of $s \geq 2$ symbols, is said to be an **orthogonal array** of s *levels*, *index* λ, *size* N, d *factors/variables/dimensions*, and *strength* t (for some t in the range $0 \leq t \leq d$) if each t-tuple drawn from S occurs as a run exactly λ times in every $N \times t$ subarray of A.

We denote the array $OA(N,d,s,t)$, where the integers N,d,s,t and λ are its parameters. Note that there is no need to explicitly mention λ since it is determined by the other parameters: $\lambda = N/s^t$. We will mostly be interested in OAs of *unit index*, $\lambda = 1$.

Existence and construction of orthogonal arrays. Much of the theoretical work on orthogonal arrays is interested in proving for which values of these parameters an orthogonal array can exist. The more practical question, however, is: if one does exist, how can we algorithmically construct such an array?

Every $N \times d$ array has strength 0. Unit index orthogonal arrays of strength 1, $OA(N,d,s = N,1)$, also always exist regardless of the values of the other parameters. We can trivially construct such OAs for an arbitrary number of factors by simply assigning $A_{ij} = i$ where i and j indicate a run and factor of the array A, for $0 \leq i < N$ and $0 \leq j < d$. We will see later that unit index OAs of strength 1 correspond to Latin hypercube stratification [MBC79; Shi91].

Several construction algorithms also exist for unit index orthogonal arrays with strength $t \geq 2$, but they are more involved. In Sec. 4 we will introduce novel, and extend existing, construction algorithms to make them practical for Monte Carlo rendering. First, however, we need to understand how such OAs relate to MC integration.

3.2. Monte Carlo sampling using orthogonal arrays

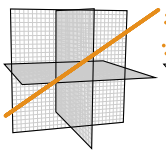
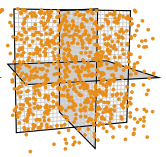
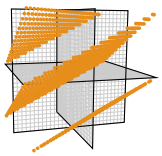
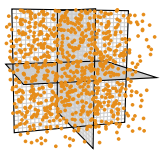
The tabular form of OAs is quite abstract, but we can also interpret OAs visually (Table 1, right) as a collection of N d-dimensional points: one for each of the N runs, with each factor specifying the coordinate along one of the d dimensions. The strength property of OAs tells us that by plotting any choice of t distinct dimensions/factors, the N points will be arranged on a regular s^t grid, with each point repeated λ times.

Orthogonal-array-based jittered sampling: In canonical form, such points are ill-suited for Monte Carlo integration since (even though their low-dimensional projections are evenly spaced) the points are not evenly distributed in the high-dimensional space (see Table 2). To enable unbiased MC integration, Owen [Owe92] proposed mapping the integer lattice representation of the OA to points uniformly distributed in the unit hypercube $[0,1)^d$ via randomized shuffling and jittering:

$$X_{ij} = \frac{\pi_j(A_{ij}) + \xi_{ij}}{s} \in [0,1)^d, \text{ for } 0 \leq i < N \text{ and } 0 \leq j < d, \quad (1)$$

where $\xi_{ij} \in [0,1)$ independently jitters each point's dimensions by a uniformly distributed random value, and $\pi_j(A_{ij})$ simply accesses the A_{ij}^{th} element from the j^{th} independent random permutation of the levels (a.k.a. strata) in S. We discuss ways to efficiently generate A and X in Sec. 4. Table 2 shows an example.

Table 2: *In the canonical arrangement, samples lie along a single diagonal for Latin hypercube sampling (left) and along planes for Orthogonal-array-based sampling (right). Random shuffling and jitter distributes the samples uniformly over the hypercube.*

Latin hypercube sampling $OA(s,3,s,1)$		**Orthogonal array sampling** $OA(s^2,3,s,2)$	
Canonical	Shuffled	Canonical	Shuffled
			

When A has unit strength and index, Eq. (1) produces Latin hypercube (LH) samples, which stratify all 1-dimensional projections. More generally, Eq. (1) stratifies all t-dimensional projections for an OA of strength t. In simple terms, if A is a strength $t = 2$ OA, Eq. (1) produces uniformly distributed d-dimensional points which are simultaneously stratified in all 2D projections as if they had been produced by jittered sampling. Fig. 1(a) shows the six different stratified 2D projections of a 4D point set generated this way.

Orthogonal array-based multi-jittered sampling: The disadvantage of using strengths $t \geq 2$ with such jittered offsets is that while t-dimensional projections are stratified into s^t strata, the projections for all lower dimensions $r < t$ are only stratified into s^r levels with λs^{t-r} samples in each stratum, e.g., the 1-dimensional projections are now only stratified into s^1 intervals, not the $s^t = N$ intervals ensured by LH sampling. Tang [Tan93] showed how to achieve both LH stratification and OA stratification by arranging, for each dimension, all $N/s = \lambda s^{t-1}$ points that would fall into the same stratum to instead each fall into finer, distinct, sub-strata. This is analogous to how multi-jittered sampling [CSW94] enforces both LH and jittered stratifications; but while multi-jittered sampling is restricted to 2D points, OAs allow generating d-dimensional points with t-dimensional (plus 1-dimensional) stratification. Fig. 1 shows the multi-jittered (b) and correlated multi-jittered (c) 2D projections of a 4D point set generated using an efficient in-place construction algorithm we propose in Sec. 4.

4. Practical construction techniques for rendering

In this section we translate the previously introduced mathematical definitions of orthogonal arrays into concrete construction algorithms suitable for MC integration in rendering. While some papers in the OA literature include "constructions", this typically refers to an existance proof by construction rather than a practical and efficient numerical algorithm.

Since practical rendering systems typically consume sample dimensions one by one, we seek construction algorithms that support such on-the-fly generation. Our goal is to efficiently construct each sample and dimension without much precomputation or storage, ideally in a way where each dimension of each sample can be generated independently and in any order.

We propose two practical implementations of classical OA construction approaches (Secs. 4.1 and 4.2)—which target OAs of strength $t = 2$ and $t > 2$, respectively—while enriching them with jittered, multi-jittered and correlated multi-jittered offsets. We also propose a (to our knowledge) novel construction algorithm (Sec. 4.3) that generalizes Kensler's CMJ sampling to arbitrary dimensions, producing full-factorial OAs of strength $t = d$.

4.1. High-dimensional points with (C)MJ 2D projections

Bose [BN41; Bos38] proposed a construction technique for orthogonal arrays of type $\mathrm{OA}(s^2, s+1, s, 2)$ where s is a prime number. From an MC rendering point of view, this will allow us to generalize Latin hypercube [MBC79; Shi91], padded 2D jittered [CPC84], and padded 2D (correlated [Ken13]) multi-jittered [CSW94] sampling to be stratified simultaneously in all 1D and 2D projections.

Listing 1: *Computing an arbitrary sample from a Bose OA pattern.*

```
1  float boseOA(unsigned i,      // sample index
2               unsigned j,      // dimension (< s+1)
3               unsigned s,      // number of levels/strata
4               unsigned p,      // pseudo-random permutation seed
5               OffsetType ot) { // J, MJ, or CMJ
6      unsigned Aij, Aik;
7      i             = permute(i % (s*s), s*s, p);
8      unsigned Ai0  = i / s;
9      unsigned Ai1  = i % s;
10     if (j == 0) {
11         Aij       = Ai0;
12         Aik       = Ai1;
13     } else if (j == 1) {
14         Aij       = Ai1;
15         Aik       = Ai0;
16     } else {
17         unsigned k = (j % 2) ? j-1 : j+1;
18         Aij       = (Ai0 + (j-1) * Ai1) % s;
19         Aik       = (Ai0 + (k-1) * Ai1) % s;
20     }
21     unsigned stratum    = permute(Aij, s,     p * (j+1) * 0x51633e2d);
22     unsigned subStratum = offset(Aij, Aik, s, p * (j+1) * 0x68bc21eb, ot);
23     float    jitter     = randfloat(i,        p * (j+1) * 0x02e5be93);
24     return (stratum + (subStratum + jitter) / s) / s;
25 }
26
27 // Compute substrata offsets
28 unsigned offset(unsigned sx, unsigned sy, unsigned s,
29                 unsigned p, OffsetType ot) {
30     if (ot == J)   return permute(sy, s, (sy * s + sx + 1) * p);
31     if (ot == MJ)  return permute(sy, s, (sx + 1) * p);
32                    return permute(sy, s, p);   // CMJ
33 }
```

Bose's construction sets the j-th dimension of the i-th sample to:

$$A_{i0} = \lfloor i/s \rfloor, \quad A_{i1} = i \bmod s \qquad \text{for } j = 0 \text{ and } j = 1; \text{ and} \tag{2}$$

$$A_{ij} = A_{i0} + (j-1)A_{i1} \bmod s \qquad \text{for } 2 \leq j \leq s+1. \tag{3}$$

The division and modulo operations for the first two dimensions (2) cycle through the s^2 possible 2-tuples by converting the value i into a two-digit number in base s. This is equivalent to the standard way of mapping a linear index i into a regular 2D $s \times s$ grid. The remaining dimensions (3) are simply a linear combination of the first two, modulo s. The example orthogonal array in Table 1 was created using this construction technique.

The orthogonal array produced by Eqs. (2) and (3) must be properly scaled to fit into the unit hypercube for MC integration. Trivially mapping an OA by taking $X_{ij} = A_{ij}/s$ creates points that lie within just a few lines or planes (see Table 2) so it is essential to randomize and jitter the OA beforehand to ensure the points are uniformly distributed (e.g. using Eq. (1)). In Listing 1 we provide pseudo-code for computing an arbitrary sample i from the $N = s \times s$ set of stratified d-dimensional Bose samples *in-place* (without requiring significant precomputation or storage). We model this after Kensler's in-place CMJ construction, but generalized to d dimensions, and enhanced to allow for three different flavors of substrata offsets: jittered, multi-jittered, and correlated multi-jittered.

Line 7 permutes the index `i` so that the samples are obtained in random order. We rely on Kensler's hashing-based `permute(i,l,p)` function, which returns the element at position `i` in a random permutation vector of length `l` where the value `p` is used to select one of the $l!$ possible permutations (we include the definition of `permute()` as well as the pseudorandom `randfloat()` in Listing 4 for completeness). Lines 8 and 9 implement Eq. (2) and Line 18 implements Eq. (3) for the remaining dimensions of the sample. These are then

permuted (Line 21) to assign the sample to a random one of the major $s \times s$ strata. Lines 22–23 together implement the three different strategies of substrata offsets. We first offset (Line 22) the sample within the cell to the appropriate substratum, and then add a small bit of additional random jitter (Line 23) within the substratum.

The `offset()` function takes on one of three forms depending on whether jittered (J), multi-jittered (MJ), or correlated multi-jittered (CMJ) offsets are desired. Jittered-style chooses one of the s substrata offsets at random, independently for each stratum. For multi-jittered, samples that fall within the same x stratum use a single x-substratum permutation of their y strata coordinates, and likewise for y. Correlated multi-jittered uses the *same* permutation for the x substrata in each y stratum, and vice-versa.

The *substratum* offset computation for a dimension depends on the *stratum* coordinate for the "other" dimension (Lines 10–20), e.g. for the first two dimensions: the x substrata depend on the y stratum (Lines 11 and 12), and the y substrata depend on the x stratum (Lines 14 and 15). Once we move beyond two dimensions we have many choices for how to pair up the remaining dimensions. For jittered and multi-jittered offsets, pairing dimension j with any other dimension will produce correct and statistically identical results. However, this choice will influence the appearance of the point set for correlated multi-jittered offsets. We chose to generalize the xy case by pairing each odd dimension with the dimension immediately preceding it, and vice-versa (Line 17). This ensures that the point sets look like CMJ2D in each of the primary 2D projections. Projection onto non-primary pairs of dimensions produces somewhat inferior correlations, so we instead force those projections to use multi-jittered offsets by introducing an additional shuffle.

Bose's construction shares many similarities to multi-jittered sampling. In fact, the first two dimensions are identical. Bose's construction, however, extends this to higher dimensions. On the other hand, the Bose construction also places more restrictions on the stratification grid and number of allowed samples. Multi-jittered sampling can produce an $n \times m$ stratification of points for arbitrary positive integers n and m, while Bose is restricted to an $s \times s$ stratification where s must be prime. Bose's full construction is in fact more general, and can allow s to be a prime raised to an arbitrary positive power [Bos38], but this requires arithmetic in a finite (Galois) field, instead of modular arithmetic.

4.2. High-dimensional points with (M)J tD projections

The Bose construction stratifies all 1D and 2D projections, but 3D and higher projections of the points are randomly distributed. If significant integrand variation exists in such projections, the convergence rate of MC integration will deteriorate.

Bush [Bus52] generalized Bose's construction to arbitrary strength t, producing arrays of type $\mathrm{OA}(s^t, s, s, t)$ for prime s (using modular arithmetic), or primes powers (using finite field arithmetic).

Bush's construction sets the j-th dimension of the i-th sample to:

$$A_{ij} = \phi_i(j) \bmod s, \quad \text{with} \quad \phi_i(x) = \sum_{l=0}^{t-1} c_{il} x^l, \tag{4}$$

where the ϕ_is are polynomials with *distinct* coefficient vectors (one for each of the $i = 0, \ldots, s^t - 1$ sample points) with elements drawn from S. A simple approach is to convert the sample index i to base-s, and set c_{il} to the l-th digit. Evaluating the polynomial $\phi_i(j)$ then amounts to reinterpreting those digits as a base-j number.

Listing 2: *Computing an arbitrary sample from a Bush OA pattern.*

```
1  float bushOA(unsigned i,      // sample index
2               unsigned j,      // dimension (< s)
3               unsigned s,      // number of levels/strata
4               unsigned t,      // strength of OA (0 < t <= d)
5               unsigned p,      // pseudo-random permutation seed
6               OffsetType ot) { // J or MJ
7      unsigned N          = pow(s, t);
8      i                   = permute(i, N, p);
9      auto iDigits        = toBaseS(i, s, t);
10     unsigned stm        = N / s;     // pow(s, t-1)
11     unsigned k          = (j % 2) ? j - 1 : j + 1;
12     unsigned phi        = evalPoly(iDigits, j);
13     unsigned stratum    = permute(phi % s, s, p * (j+1) * 0x51633e2d);
14     unsigned subStratum = offset(i, s, stm,   p * (j+1) * 0x68bc21eb, ot);
15     float    jitter     = randfloat(i,        p * (j+1) * 0x02e5be93);
16     return (stratum + (subStratum + jitter) / stm) / s;
17 }
18
19 // Compute the digits of decimal value 'i' expressed in base 'b'
20 vector<unsigned> toBaseS(unsigned i, unsigned b, unsigned t) {
21     vector<unsigned> digits(t);
22     for (unsigned ii = 0; ii < t; i /= b, ++ii)
23         digits[ii] = i % b;
24     return digits;
25 }
26
27 // Evaluate polynomial with coefficients a at location arg
28 unsigned evalPoly(const vector<unsigned> & a, unsigned arg) {
29     unsigned ans = 0;
30     for (unsigned l = a.size(); l--; )
31         ans = (ans * arg) + a[l];        // Horner's rule
32     return ans;
33 }
34
35 // Compute substrata offsets
36 unsigned offset(unsigned i, unsigned s, unsigned numSS,
37                 unsigned p, OffsetType ot) {
38     if (ot == J) return permute((i / s) % numSS, numSS, (i + 1) * p);
39                  return permute((i / s) % numSS, numSS, p);  // MJ
40 }
```

Listing 2 implements Eq. (4) and maps A_{ij} to the unit hypercube for MC integration. We do this while enabling jitter- and multi-jittered-style substrata offsets (we were not able to attain correlated multi-jittered-style offset for Bush OAs) using an efficient *in-place* approach that does not require significant precomputation or storage.

As before, for jittered offsets, each of the i points chooses a random substratum independently. Multi-jittered offsets seek to assign each of the s^{t-1} samples that fall into the same stratum into different substrata. We achieve this by assigning a unique index (e.g. $i/s \bmod s^{t-1}$) to each of these samples, and setting the substratum offsets as a random permutation of these indices.

4.3. High-dimensional CMJ

While both Bose and Bush can be generalized to support non-prime bases using Galois field arithmetic, this significantly complicates their implementation, and still only allows s to be a power of a prime. We propose a novel OA construction algorithm inspired by both Kensler's 2D CMJ algorithm and Bush's construction technique. The new construction makes a design trade-off: requiring that the pattern be a full factorial design ($t = d$), which relaxes the prime power requirement, allowing any positive value for s. Our algorithm results in orthogonal arrays of type $\mathrm{OA}(s^d, d, s, t)$ simultaneously for all

Listing 3: *CMJ sampling generalized to arbitrary dimensions d.*

```
1  float cmjdD(unsigned i,  // sample index
2              unsigned j,  // dimension (<= s+1)
3              unsigned s,  // number of levels/strata
4              unsigned t,  // strength of OA (t=d)
5              unsigned p { // pseudo-random permutation seed
6      unsigned N        = pow(s, t);
7               i        = permute(i, N, p);
8      auto     iDigits  = toBaseS(i, s, t);
9      unsigned stm1     = N / s;     // pow(s, t-1)
10     unsigned stratum  = permute(iDigits[j], s,  p * (j+1) * 0x51633e2d);
11     auto     pDigits  = allButJ(iDigits, j);
12     unsigned sStratum = evalPoly(pDigits, s);
13              sStratum = permute(sStratum, stm1, p * (j+1) * 0x68bc21eb);
14     float    jitter   = randfloat(i,            p * (j+1) * 0x02e5be93);
15     return (stratum + (sStratum + jitter) / stm1) / s;
16 }
17
18 // Copy all but the j-th element of vector in
19 vector<unsigned> allButJ(const vector<unsigned> & in, unsigned omit) {
20     vector<unsigned> out(in.size()-1);
21     copy(in.begin(), in.begin() + omit, out.begin());
22     copy(in.begin() + omit + 1, in.end(), out.begin() + omit);
23     return out;
24 }
```

$1 < t \leq d$, but we enrich them with denser 1D LH stratification, also satisfying $\mathrm{OA}(s^d, 1, s^d, d)$. For $d = 4$, for instance, this construction provides $(s \times s \times s \times s)$ 4D jittered stratification with $\lambda = 1$ sample in each 4D stratum, $(s \times s \times s)$ 3D jittered stratification with $\lambda = s$ in each 3D stratum, $(s \times s)$ 2D stratification with $\lambda = s^2$ samples in each 2D stratum, and finally proper LH stratification into s^4 intervals with $\lambda = 1$ sample in each 1D stratum.

Listing 3 shows our complete algorithm, which mirrors the structure of Listing 2: We first map the linear sample index i to t numbers indexing the coordinates of the regular s^t grid by using the digits of the base-s representation of i (Line 8). We then randomly permute the order of the strata in each dimension (Line 10). Initially, all $N = s^t$ points project to just s locations along any dimension j. Our goal is to offset these samples in each dimension j so that they each project to a distinct substratum. To ensure this 1D LH stratification for dimension j, we consider the s^{t-1} strata in all dimensions excluding j. We can uniquely index into this set by using all but the j^{th} base-s digit of i (Line 11). Finally, we push the samples in each stratum together by one of s^{t-1} distinct offsets (Line 12) so that they each fall into a different substratum.

In addition to the stratification properties mentioned above, this construction also has a nesting property related to nested/sliced orthogonal arrays [HHQ16; HQ11; YLL14]. Specifically, for 3D, the points falling into the s equal sized slices along any of the three dimensions ($3 \times s$ slices total) are themselves a proper 2D $s \times s$ CMJ pattern. Any 2D projection in this case is the superposition of s separate 2D CMJ patterns which together form a Latin hypercube design. Fig. 3 illustrates this property for a 3D pattern with 27 points. We show the xy projection of all 27 points as well as the 2D projections of each of the 3 z-slices, each of which are CMJ2D sets. Together the points satisfy both strength 3 and strength 1 with index unity. For general d, taking any of the s-slices along any dimension will give an $\mathrm{OA}(s^{d-1}, d-1, s, t)$ with $\lambda = s$ points in each stratum.

5. Variance Analysis and Results

Prior work on orthogonal arrays has examined their variance properties extensively. We briefly summarize the most relevant results here and aggregate them in Table 3. We will also quantitatively evaluate the error of OA-based sampling to existing methods both on synthetic integrands and in real renderings.

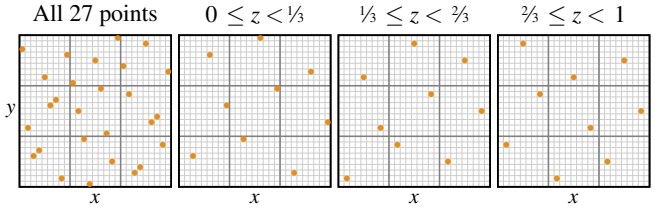


Figure 3: *For a CMJ3D pattern with $3^3 = 27$ points, the nesting property ensures that all 3 z-slices produce CMJ2D points when projected onto xy. The same nesting property simultaneously holds for the x and y slices when projected onto yz and xz.*

5.1. Theoretical variance convergence rates

Variance for purely random sampling is $\mathcal{O}(N^{-1})$ regardless of the dimensionality of the integrand. Full grid-based stratification improves this to $\mathcal{O}(N^{-1-2/d})$ for d-dimensional integrands with continuous first derivatives and $\mathcal{O}(N^{-1-1/d})$ for discontinuous ones [Owe13; PSC*15]. For low-dimensional integrands this is a sizable improvement (e.g., going from $\mathcal{O}(N^{-1})$ to $\mathcal{O}(N^{-2})$ or even $\mathcal{O}(N^{-3})$ for 1D), but the benefits diminish at higher dimensions.

Intuitively we would expect strength-t OA sampling (and Latin hypercube sampling where $t = 1$) to perform well whenever stratification on some t of the d dimensions would be effective. One benefit of OA sampling over padding t-dimensional stratified samples (when $t > 1$) is that we do not need to know in advance which of the $\binom{d}{t}$ subsets of the dimensions are the important ones to stratify, since in fact all such projections are well stratified!

Asymptotically, orthogonal array sampling will perform no better than pure random sampling for general d-dimensional integrands if $t < d$. This is because while all t-dimensional projections are stratified, the points are not stratified at the full dimensionality unless $t = d$. Stein [Ste87] showed, however, that Latin hypercube sampling performs much better if the d-dimensional integrand is

Table 3: *The convergence rate improvement (b in $\mathcal{O}(N^{-1-b})$) as a function of the dimensionality and smoothness of the integrand for various samplers. The 1- and t-additive integrands are d-dimensional, where $t < d$. Best case for each integrand is bold.*

Sampler	**Convergence rate improvement b**							
Integrand:	d-dim.		t-dim.		t-additive		1-additive	
Discontinuity:	C^1	C^0	C^1	C^0	C^1	C^0	C^1	C^0
Random	0	0	0	0	0	0	0	0
d-stratified	**$2/d$**	**$1/d$**	$2/d$	$1/d$	$2/d$	$1/d$	$2/d$	$1/d$
Padded t-stratified	0	0	**$2/t$**	**$1/t$**	0	0	0	0
Padded t-stratified+LH	0	0	**$2/t$**	**$1/t$**	0	0	**2**	**1**
OA strength-t	0	0	**$2/t$**	**$1/t$**	**$2/t$**	**$1/t$**	$2/t$	$1/d$
OA strength-t+LH	0	0	**$2/t$**	**$1/t$**	**$2/t$**	**$1/t$**	**2**	**1**

nearly an additive function of the d components of X, e.g., for $d = 3$,

$$f(x,y,z) \approx f_x(x) + f_y(y) + f_z(z). \tag{5}$$

When the integrand is exactly additive, the asymptotic convergence improves to that of stratified sampling in 1D: $\mathcal{O}(N^{-3})$ for smooth or $\mathcal{O}(N^{-2})$ for discontinuous integrands. Similar results were later derived from a Fourier perspective [SJ17]. Orthogonal array sampling generalizes this so that integrands that are additive functions of t-tuples of the d components of X, e.g., for $t = 2, d = 3$

$$f(x,y,z) = f_{xy}(x,y) + f_{xz}(x,z) + f_{yz}(y,z), \tag{6}$$

obtain convergence rates of t-dimensional stratification [Owe92]: $\mathcal{O}(N^{-1-2/t})$ for smooth and $\mathcal{O}(N^{-1-1/t})$ for discontinuous ones. Orthogonal array-based Latin hypercubes further ensure that if the integrand happens to be additive in just the individual dimensions, this improves to 1D stratified convergence: $\mathcal{O}(N^{-3})$ or $\mathcal{O}(N^{-2})$ [Tan93]. Table 3 summarizes all these cases.

5.2. Empirical variance analysis on analytic integrands

We carried out a quantitative variance analysis to evaluate how these theoretical results translate into practice. Critically, we are interested in analyzing multi-dimensional integration, and how this differs from prior analyses that have focused primarily on 2D [CKK18; PSC*15; SSJ16]. To this end, we designed a few simple analytic functions which we can combine to form integrands of arbitrarily high dimensions d, while controlling their level of continuity and whether they are fully d-dimensional or t-additive for some $t < d$.

Full d-dimensional integrand. We define our full d-dimensional integrand as a radial function using some 1D kernel function $g(r)$:

$$f_d^D(p_1,\ldots,p_d) = g^D(\|\vec{p}\|), \text{ for } \vec{p} = (p_1,\ldots,p_d) \in [0,1)^d, \tag{7}$$

where $\|\cdot\|$ denotes vector norm.

To control the level of continuity, we choose $g^D(r)$ from one of the following three step-like functions with different orders of discontinuity D:

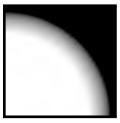
$$g^0(r) = 1 - \text{binaryStep}(r, r_{\text{end}}), \tag{8}$$

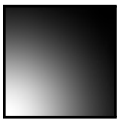
$$g^1(r) = 1 - \text{linearStep}(r, r_{\text{start}}, r_{\text{end}}), \tag{9}$$

$$g^\infty(r) = \exp(-r^2/(2\sigma^2)), \tag{10}$$

where we set the parameters $r_{\text{end}} = 3/\pi$, $r_{\text{start}} = r_{\text{end}} - 0.2$, and $\sigma = 1/3$. Intuitively, when coupled with Eqs. (8)–(10), f_d^D is a d-sphere with a binary (g^0) or fuzzy (g^1) boundary, or a radial Gaussian (g^∞), as visualized for $d = 2$ in the insets to the right.


f_2^0: Binary Step


f_2^1: Linear Step


f_2^∞: Gaussian

Additively and multiplicatively separable integrands. To test whether t-additive functions really do gain improved convergence rates with OAs, we additionally build up separable d-dimensional integrands which are t-additive ($f_{d,t+}^D$) and t-multiplicative ($f_{d,t\times}^D$) by summing or multiplying lower-dimensional functions f_t for each possible t-tuple of the d dimensions:

$$f_{d,t+}^D(p_1,\ldots,p_d) = \sum_{i_1=1}^{d} \cdots \sum_{i_t=i_{t-1}+1}^{d} f_t^D(p_{i_1},\ldots,p_{i_t}), \tag{11}$$

$$f_{d,t\times}^D(p_1,\ldots,p_d) = \prod_{i_1=1}^{d} \cdots \prod_{i_t=i_{t-1}+1}^{d} f_t^D(p_{i_1},\ldots,p_{i_t}). \tag{12}$$

Intuitively, an e.g. 3D 2-additive or 2-multiplicative integrand would add or multiply together one of the 2D functions above evaluated using all choices of the two dimensions:

$$f_{3,2+}^D(x,y,z) = f_2^D(x,y) + f_2^D(x,z) + f_2^D(y,z), \tag{13}$$

$$f_{3,2\times}^D(x,y,z) = f_2^D(x,y) \times f_2^D(x,z) \times f_2^D(y,z). \tag{14}$$

We extended the open-source Empirical Error Analysis (EEA) [SSJ16] framework to support multidimensional integrands. Fig. 4 shows variance graphs for a representative set of test integrands (Figs. 3 and 4 of the supplemental include more integrands and variance graphs). Each data point is the unbiased sample-variance from 100 runs. We plot these on a log-log scale of samples vs. variance, where the graphs' slopes indicate the rate of convergence. We compare various dimensionalities and strengths of our OA-based samplers to several baseline methods including random sampling, Latin hypercubes, PBRT's padded stratified 2D sampler using jittered samples as well as our extension using padded 2D CMJ patterns, PBRT's padded $(0,2)$ QMC sequence, and truly high-dimensional Sobol and Halton samplers.

We observe good agreement to the theoretically predicted convergences rates in Table 3 for the various samplers and integrands, confirming that strength-t OA+LH-based sampling provides a sizable benefit compared to 2D padded samplers when the integrands are up to t-additive (left two columns). When the strength t of the OA is higher than the additivity of the integrand (e.g. BushMJ(t=4) for the 2-additive integrand) we still obtain improved convergence rates, but the improvements are most dramatic when t matches the additivity. When the integrand is 1-additive (leftmost column), all our samplers provide dramatically improved convergence rates (down to -3, faster even than QMC techniques which max out at -2), since they all enforce Latin hypercube stratification on all 1D projections regardless of strength. On the other hand, when the integrand is not t-additive (right two columns), the asymptotic convergence rates for all the padded and OA-based samplers degrade to that of random sampling (unless $t = d$), while truly high-dimensional QMC sequences like Sobol or Halton are able to maintain a better asympotic rate. Interestingly, despite all OA and padding approaches having the same slope when $t < d$, OAs of increasing strength seem to provide successively stronger constant-factor variance reduction.

5.3. Evaluation on rendered images

From our analysis on simple analytic integrands, we would expect OA-based sampling to always be at least as good as standard padding approaches asymptotically, and also provide a noticeable constant-factor variance reduction.

Since real rendering integrands are high-dimensional and likely not t-additive, we also compared our OA+LH-based sampling approach to a similar suite of other samplers within PBRT [PJH16] to

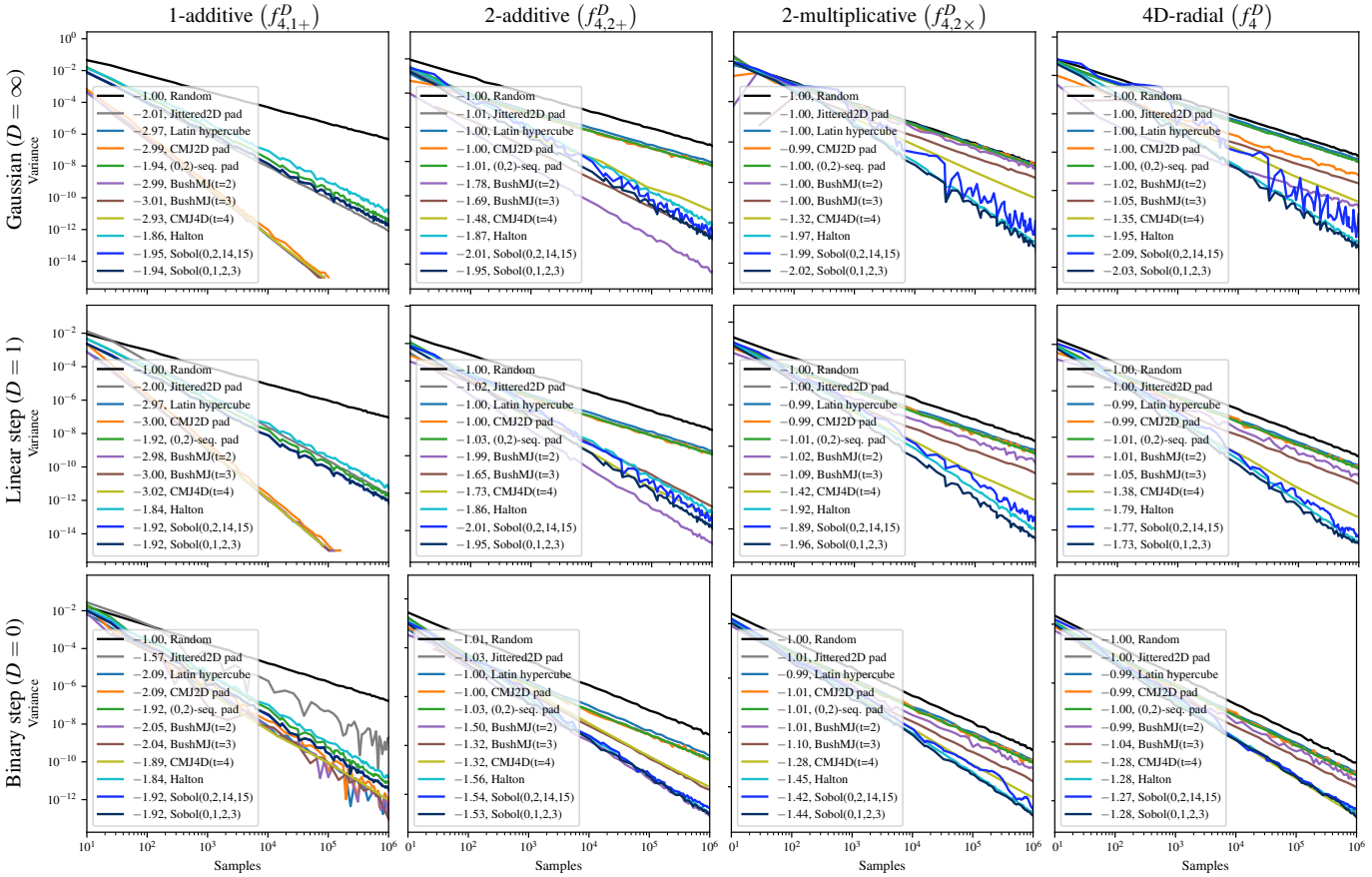


Figure 4: *Variance behavior of 11 samplers on 4D analytic integrands of different complexity (columns) and continuity (rows). We only show one OA sampler for each strength since these tend to perform similarly (see supplemental for additional variants). We list the best-fit slope of each technique, which generally matches the theoretically predicted convergences rates (Table 3). Our samplers always perform better than traditional padding approaches, but are asymptotically inferior to high-dimensional QMC sequences for general high-dimensional integrands. When strength $t < d$ (right two columns), convergence degrades to $\mathcal{O}(N^{-1})$, but higher strengths attain lower constant factors.*

see whether these benefits extend to more practical scenarios. Fig. 6 shows qualitative and quantitative variance comparisons among these samplers on three scenes. BLUESPHERES features a combined 9D integrand consisting of simultaneous antialiasing, depth of field, motion blur, and interreflections, and CORNELLBOX has antialiasing, shallow depth of field, and direct illumination for a combined 7D integrand. In these two scenes we compare the approaches at $11^2 = 121$ samples per pixel (spp) for the samplers that support it, and round up to the next power-of-two at $2^7 = 128$ spp for QMC samplers. BARCELONA is a much more complicated scene featuring several bounces of global illumination (a 43D integrand), so we use $63^2 = 3969$ spp for all samplers that support it, but round QMC samplers up to $2^{12} = 4096$ spp. Our OA-based sampler is able to consistently beat all the 2D padded approaches and provides results close to that of multi-dimensionally stratified global samplers like Halton and Sobol, but without the risk of structured artifacts of Sobol (see blue inset for CORNELLBOX). While these results confirm that stratification in general is important, BARCELONA shows that the improvement diminishes as the scene/integrand complexity/dimensionality increases.

In Fig. 5 we also plot variance as a function of samples on a log-log scale for a pixel in the yellow inset of BLUESPHERES and the blue inset of CORNELLBOX. We see that all 2D padded approaches perform about the same with a convergence rate of $\mathcal{O}(N^{-1})$. Our strength-2 and 3 OA approaches have consistently lower variance, and surprisingly also steeper convergence rates of roughly $\mathcal{O}(N^{-1.25})$ (for non-2-additive integrands we should ex-

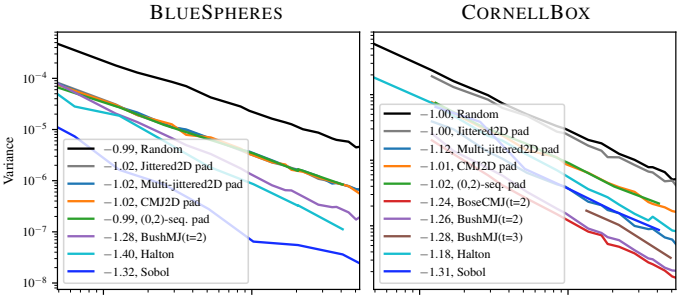


Figure 5: *Variance behavior and best-fit slope of various samplers for a pixel in the yellow inset in* BLUESPHERES *and the blue inset of* CORNELLBOX *in Fig. 6. Our samplers always perform better than traditional padding approaches and even outperform the global Halton and Sobol samplers in* CORNELLBOX.

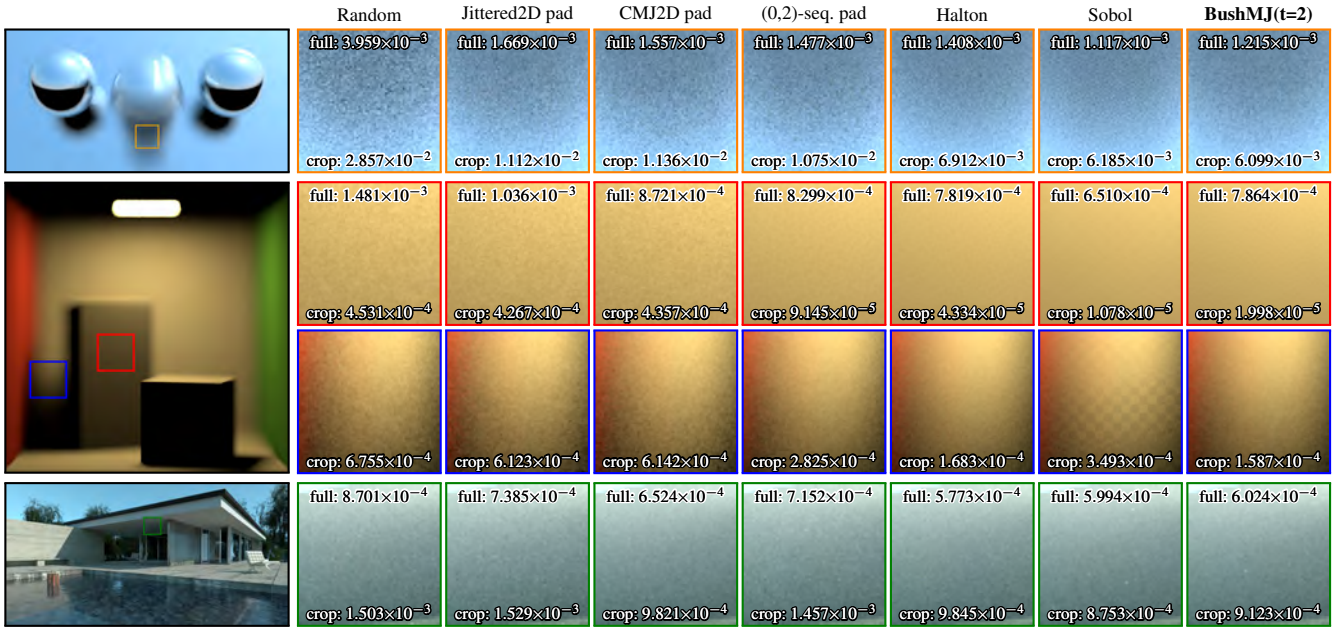


Figure 6: *The* BLUESPHERES, CORNELLBOX, *and* BARCELONA *scenes feature different combinations of pixel antialiasing, DoF, motion blur, and several bounces of indirect illumination for combined integrands of 9D, 7D, and 43D respectively. The relative MSE numbers for the entire image (top) and each inset (bottom) show that our OA-based sampling technique is able to out-perform 2D padded samplers (first 4 columns), and is close to the quality of multi-dimensionally stratified global samplers like Halton and Sobol.*

pect an asymptotic rate of $\mathcal{O}(N^{-1})$ for strength-2 OAs). We suspect these slopes will degrade to -1 in the limit, but at practical sample counts our OA-based samplers are still able to benefit from a faster rate, suggesting that the integrands may be *approximately* 2-additive. In BLUESPHERES the global QMC samplers do still provide superior variance and slope, though the difference to OAs is modest for Halton. In CORNELLBOX, our samplers outperform both QMC approaches, though this ranking depends on which image region is used, as the two insets in Fig. 6 for CORNELLBOX show. Overall, the multi-projection stratification of OAs substantially outperform 2D padding, while multi-dimensional QMC samplers maintain a slight edge depending on the scene/image region.

6. Conclusions, Discussion, & Limitations

We have introduced orthogonal array-based Latin hypercube sampling and proposed novel, in-place construction routines suitable for efficient Monte Carlo rendering. While our approach already provides a practical drop-in improvement to traditional 2D padding, neither the theory nor our implementation are without limitations.

Sample count granularity. None of our OA implementations can set the number of samples arbitrarily. Our full factorial ($t = d$) cmjdD construction requires $N = s^d$ samples. While this may be feasible for low-dimensional integrands, the 43D integrand in BARCELONA would require a prohibitive number of samples (2^{43}) even with just two strata per dimension, so we excluded cmjdD from the rendering comparisons. Partial factorial designs, in contrast, require a minimum of $N = s^t$ samples determined by the strength t and not the dimension d. The Bose and Bush constructions, however, produce at most $s+1$ and s dimensions, respectively, so patterns with higher strengths require finer stratifications s, and therefore more points. Our modular arithmetic implementations of these constructions further require that the number of levels s be a prime.

Implementation improvements. It should be relatively straightforward to extend our implementations of Bose and Bush to support positive prime powers, relaxing these restrictions at the cost of using arithmetic on a Galois Field. Even restricted to binary Galois Fields, which are particularly efficient, this would allow creating OAs with any power-of-two number of samples much like many QMC patterns. It may also be useful to explore other construction techniques, which impose different constraints on the relationship between the OA parameters, e.g., the strength-2 construction by Addelman and Kempthorne [AK61]. The OAPackage [EV19; SEN10] also contains C++ and Python bindings for generating, manipulating and analyzing various OAs for experimental design.

While we chose to use OAs in a pixel-based sampler, the 2D stratification guarantees would allow us to use a single strength-2 OA pattern across an entire image while ensuring a prescribed number of samples per pixel. This would effectively remove the aforementioned dimensionality limitation ($d \approx s$) since s will be very large for all but the smallest rendered images. This would bring OAs more in line with PBRT's global Halton and Sobol samplers, while likely avoiding the associated correlated/checkerboard artifacts since multi-jittered points do not exhibit structured patterns.

QMC and strong orthogonal arrays. Our results show that high-dimensional QMC samples like Halton and Sobol still provide better convergence compared to our techniques. An exciting direction to help narrow this gap are "strong" orthogonal arrays (SOAs), recently introduced by He and Tang [HT12]. While OA-based Latin hypercubes ensure stratification along all 1D and tD projections, all other

dimensional projections lack stratification. SOAs improve on this by constructing points that are stratified not just in 1- and t-dimensional projections, but in all r-dimensional projections for $r \leq t$. Moreover, the stratifications are more dense, e.g., both an OA and SOA of strength 3 would achieve $s \times s \times s$ (3D jittered) stratification in any three-dimensional projection, but the SOA version would achieve $s^2 \times s$ and $s \times s^2$ stratifications in any two-dimensional projection, while the OA guarantees only $s \times s$ stratification. These additional stratification guarantees are like the elementary intervals of (t,m,s)-nets [Nie92; Sob67], and, in fact, SOAs are a generalization of such nets. He et al. [HCT18] also proposed a middle-ground called OAs of strength "2-plus" which do not extend stratification to more dimensions, but do impose the denser elementary interval stratification on strength-2 OAs. He and Tang [HT12] showed that it is possible to generate an SOA from a regular OA by sacrificing some of its factors. Current construction algorithms are unfortunately fairly complex [HCT18; HT12; HT14; LL15; Wen14], often involving an intermediate conversion to a so-called generalized orthogonal array [Law96]. Nonetheless, we believe exploring this connection more carefully is a fruitful line of future work to extend the benefits of OAs further, and bridging the connection to QMC more strongly.

Progressive sample sequences. Another limitation of our work is that we currently only support finite point sets and not progressive sample sequences [CKK18; Hal64; Sob67]. These are particularly attractive for progressive rendering, since the total number of samples does not need to be known a priori and any prefix of the sample sequence is well-distributed. Luckily, there is already promising work in the OA literature on which future work could build.

"Nested" or "sliced" Latin hypercubes/orthogonal arrays [AZ16; HHQ16; HQ10; HQ11; QA10; QAW09; Qia09; RHVD10; WL13; YLL14; ZWD19] create two layered point sets: a low-sample-count sparse set, and a denser point set which is a proper superset of the sparse one. In statistics, these are used for multi-fidelity computer experiments, cross-validation, and uncertainty quantification. In rendering these could be used to provide two-level adaptive sampling control as a stepping stone to fully incremental sample sequences: rendering could start with a low spp stratified point set, and adaptively increase to a higher spp stratified (superset) pattern. While most of these techniques create two nested "layers" of points, this has recently been generalized to more than two nested layers by [SLQ14]. This could allow incrementally adding more stratified points in multiple steps of an adaptive sampling routine.

Additionally, given the recently established equivalence between SOAs and (t,m,s)-nets [HT12], we believe that fully progressive OAs derived from (t,s)-sequences may be within reach. In fact, some recent work has used (t,s)-sequence construction routines to build nested OAs [HQ10]. Ultimately, OAs and QMC sets/sequences are tightly related, and our hope is that future work in graphics can now leverage developments in both of these areas.

7. Acknowledgements

We are grateful to the anonymous reviews for helpful suggestions for improving the paper. Wojciech and Afnan thank past and present members of the Dartmouth Visual Computing Lab, particularly Gurprit Singh, for early discussions about sample generation. Andrew, Charlie, and Per would like to thank everyone on Pixar's RenderMan team. This work was partially supported by NSF grants IIS-181279 and CNS-1205521, as well as funding from the Neukom Scholars Program and the Dartmouth UGAR office.

References

[AK61] ADDELMAN, S. and KEMPTHORNE, O. "Some main-effect plans and orthogonal arrays of strength two". *The Annals of Mathematical Statistics* 32.4 (Dec. 1961). DOI: 10/cw6xh3 10.

[AKL16] AI, M., KONG, X., and LI, K. "A general theory for orthogonal array based Latin hypercube sampling". *Statistica Sinica* 26.2 (2016). ISSN: 1017-0405. DOI: 10/gfznbq 3.

[AZ16] ATKINS, J. and ZAX, D. B. "Nested orthogonal arrays". (Jan. 24, 2016). arXiv: 1601.06459 [stat] 11.

[BB52] BOSE, R. C. and BUSH, K. A. "Orthogonal arrays of strength two and three". *The Annals of Mathematical Statistics* 23.4 (Dec. 1952). ISSN: 0003-4851. DOI: 10/dts6th 3.

[BN41] BOSE, R. C. and NAIR, K. R. "On complete sets of Latin squares". *Sankhyā: The Indian Journal of Statistics* 5.4 (1941). ISSN: 00364452 5.

[Bos38] BOSE, R. C. "On the application of the properties of Galois fields to the problem of construction of hyper-Graeco-Latin squares". *Sankhyā: The Indian Journal of Statistics* 3.4 (1938). ISSN: 00364452 5, 6.

[BP46] BURMAN, J. P. and PLACKETT, R. L. "The design of optimum multifactorial experiments". *Biometrika* 33.4 (June 1946). ISSN: 0006-3444. DOI: 10/fmhkrj 3.

[BSD09] BALZER, M., SCHLÖMER, T., and DEUSSEN, O. "Capacity-constrained point distributions: a variant of Lloyd's method". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 28.3 (July 27, 2009). ISSN: 0730-0301. DOI: 10/dcnbwb 3.

[Bus52] BUSH, K. A. "Orthogonal arrays of index unity". *The Annals of Mathematical Statistics* 23.3 (1952). ISSN: 00034851. DOI: 10/fgm6b8 3, 6.

[CFS*18] CHRISTENSEN, P., FONG, J., SHADE, J., WOOTEN, W., SCHUBERT, B., KENSLER, A., FRIEDMAN, S., KILPATRICK, C., RAMSHAW, C., BANNISTER, M., RAYNER, B., BROUILLAT, J., and LIANI, M. "RenderMan: an advanced path-tracing architecture for movie rendering". *ACM Transactions on Graphics* 37.3 (Aug. 2018). ISSN: 0730-0301. DOI: 10/gfznbs 2, 3.

[CKK18] CHRISTENSEN, P., KENSLER, A., and KILPATRICK, C. "Progressive multi-jittered sample sequences". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 37.4 (July 2018). ISSN: 0167-7055. DOI: 10/gdvj4n 2, 3, 8, 11.

[Coo86] COOK, R. L. "Stochastic sampling in computer graphics". *ACM Transactions on Graphics* 5.1 (Jan. 1986). ISSN: 0730-0301. DOI: 10/cqwhcc 2, 3.

[CPC84] COOK, R. L., PORTER, T., and CARPENTER, L. "Distributed ray tracing". *Computer Graphics (Proceedings of SIGGRAPH)* 18.3 (July 1, 1984). ISSN: 0097-8930. DOI: 10/c9thc3 1, 3, 5.

[CQ14] CHEN, J. and QIAN, P. Z. "Latin hypercube designs with controlled correlations and multi-dimensional stratification". *Biometrika* 101.2 (Feb. 2014). ISSN: 1464-3510. DOI: 10/f5549t 3.

[CSW94] CHIU, K., SHIRLEY, P., and WANG, C. "Multi-jittered sampling". *Graphics Gems IV*. Ed. by HECKBERT, P. S. San Diego, CA, USA: Academic Press, 1994. ISBN: 0-12-336155-9 2, 3, 5.

[DS14] DEY, A. and SARKAR, D. "A note on the construction of orthogonal Latin hypercube designs". *Journal of Combinatorial Designs* 24.3 (Aug. 2014). ISSN: 1063-8539. DOI: 10/gfznbv 3.

[DW85] DIPPÉ, M. A. Z. and WOLD, E. H. "Antialiasing through stochastic sampling". *Computer Graphics (Proceedings of SIGGRAPH)* 19.3 (July 1, 1985). ISSN: 0097-8930. DOI: 10/cmtt4s 2.

[EV19] EENDEBAK, P. T. and VAZQUEZ, A. R. "OApackage: A Python package for generation and analysis of orthogonal arrays, optimal designs and conference designs". *Journal of Open Source Software* 34.4 (2019). DOI: 10/gfznbw 10.

[Hal64] HALTON, J. H. "Algorithm 247: radical-inverse quasi-random point sequence". *Communications of the ACM* 7.12 (Dec. 1964). ISSN: 0001-0782. DOI: 10/dd3674 2, 3, 11.

[HCT18] HE, Y., CHENG, C.-S., and TANG, B. "Strong orthogonal arrays of strength two plus". *The Annals of Statistics* 46.2 (Apr. 2018). ISSN: 0090-5364. DOI: 10/gfznb6 3, 11.

[HHQ16] HWANG, Y., HE, X., and QIAN, P. Z. "Sliced orthogonal array-based Latin hypercube designs". *Technometrics* 58.1 (Jan. 2016). ISSN: 1537-2723. DOI: 10/f8cjrk 3, 7, 11.

[HM56] HAMMERSLEY, J. M. and MORTON, K. W. "A new Monte Carlo technique: antithetic variates". *Mathematical Proceedings of the Cambridge Philosophical Society* 52.03 (July 1956). ISSN: 1469-8064. DOI: 10/dshxdn 3.

[HQ10] HAALAND, B. and QIAN, P. Z. G. "An approach to constructing nested space-filling designs for multi-fidelity computer experiments". *Statistica Sinica* 20.3 (2010). ISSN: 1017-0405 11.

[HQ11] HE, X. and QIAN, P. Z. G. "Nested orthogonal array-based Latin hypercube designs". *Biometrika* 98.3 (Sept. 1, 2011). ISSN: 0006-3444. DOI: 10/cz9pj6 3, 7, 11.

[HSD13] HECK, D., SCHLÖMER, T., and DEUSSEN, O. "Blue noise sampling with controlled aliasing". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 32.3 (July 2013). ISSN: 0730-0301. DOI: 10/gbdcxt 3.

[HSS99] HEDAYAT, A. S., SLOANE, N. J. A., and STUFKEN, J. *Orthogonal Arrays: Theory and Applications*. Springer-Verlag, 1999. ISBN: 978-1-4612-7158-1. DOI: bqh3jj 2, 3.

[HT12] HE, Y. and TANG, B. "Strong orthogonal arrays and associated Latin hypercubes for computer experiments". *Biometrika* 100.1 (Dec. 2012). ISSN: 1464-3510. DOI: 10/gfznb5 3, 10, 11.

[HT14] HE, Y. and TANG, B. "A characterization of strong orthogonal arrays of strength three". *The Annals of Statistics* 42.4 (2014). ISSN: 00905364. DOI: 10/gfznb4 3, 11.

[JK08] JOE, S. and KUO, F. Y. "Constructing Sobol sequences with better two-dimensional projections". *SIAM Journal on Scientific Computing* 30.5 (2008). ISSN: 1064-8275. DOI: 10/c8tff9 2, 3.

[KCSG18] KULLA, C., CONTY, A., STEIN, C., and GRITZ, L. "Sony Pictures Imageworks Arnold". *ACM Transactions on Graphics* 37.3 (Aug. 2018). ISSN: 0730-0301. DOI: 10/gfjkn7 3.

[Kel13] KELLER, A. "Quasi-Monte Carlo image synthesis in a nutshell". *Monte Carlo and Quasi-Monte Carlo Methods*. Ed. by DICK, J., KUO, F. Y., PETERS, G. W., and SLOAN, I. H. Springer-Verlag, 2013. ISBN: 978-3-642-41095-6. DOI: 10/gfz9gw 3.

[Ken13] KENSLER, A. *Correlated Multi-Jittered Sampling*. 13-01. Pixar Animation Studios, Mar. 2013 2, 3, 5, 13.

[KK02] KOLLIG, T. and KELLER, A. "Efficient multidimensional sampling". *Computer Graphics Forum (Proceedings of Eurographics)* 21.3 (Sept. 1, 2002). ISSN: 0167-7055. DOI: 10/d2stpx 2, 3.

[KPR12] KELLER, A., PREMOZE, S., and RAAB, M. "Advanced (quasi) Monte Carlo methods for image synthesis". *ACM SIGGRAPH Course Notes* (Los Angeles, California). ACM Press, 2012. ISBN: 978-1-4503-1678-1. DOI: 10/gfzncb 3.

[Law96] LAWRENCE, K. M. "A combinatorial characterization of (t,m,s)-nets in base b". *Journal of Combinatorial Designs* 4.4 (1996). ISSN: 1520-6610. DOI: 10/c9hhmz 11.

[LD08] LAGAE, A. and DUTRÉ, P. "A comparison of methods for generating Poisson disk distributions". *Computer Graphics Forum* 27.1 (Mar. 1, 2008). ISSN: 0167-7055. DOI: 10/cvqh4r 3.

[LL15] LIU, H. and LIU, M.-Q. "Column-orthogonal strong orthogonal arrays and sliced strong orthogonal arrays". *Statistica Sinica* (2015). ISSN: 1017-0405. DOI: 10/f7z56h 3, 11.

[LP01] LARCHER, G. and PILLICHSHAMMER, F. "Walsh series analysis of the L_2-discrepancy of symmetrisized point sets". *Monatshefte für Mathematik* 132.1 (Apr. 2001). ISSN: 1436-5081. DOI: 10/bqkb9p 3.

[MBC79] MCKAY, M. D., BECKMAN, R. J., and CONOVER, W. J. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code". *Technometrics* 21.2 (May 1979). DOI: 10/gfzncd 3–5.

[MF92] MCCOOL, M. and FIUME, E. "Hierarchical Poisson disk sampling distributions". *Proceedings of Graphics Interface*. Morgan Kaufmann, 1992. ISBN: 0-9695338-1-0 3.

[Mit91] MITCHELL, D. P. "Spectrally optimal sampling for distribution ray tracing". *Computer Graphics (Proceedings of SIGGRAPH)* 25.4 (July 2, 1991). ISSN: 0097-8930. DOI: 10/btwnm3 2, 3.

[Mit96] MITCHELL, D. P. "Consequences of stratified sampling in graphics". *Annual Conference Series (Proceedings of SIGGRAPH)*. Ed. by RUSHMEIER, H. Addison-Wesley, Aug. 1996. ISBN: 978-0-89791-746-9. DOI: 10/dkw86x 2.

[Nie92] NIEDERREITER, H. *Quasi-Monte Carlo Methods*. Wiley Online Library, 1992 2, 3, 11.

[ÖS18] ÖZTIRELI, C. and SINGH, G. "Sampling analysis using correlations for Monte Carlo rendering". *ACM SIGGRAPH Asia Course Notes*. ACM Press, Nov. 2018. ISBN: 978-1-4503-6026-5. DOI: 10/gfzncg 2.

[Owe13] OWEN, A. B. *Monte Carlo Theory, Methods and Examples*. To be published, 2013. URL: https://statweb.stanford.edu/~owen/mc/ (visited on 06/07/2019) 3, 7.

[Owe92] OWEN, A. B. "Orthogonal arrays for computer experiments, integration and visualization". *Statistica Sinica* 2.2 (1992). ISSN: 10170405, 19968507 3, 4, 8.

[Owe97] OWEN, A. B. "Monte Carlo variance of scrambled net quadrature". *SIAM Journal on Numerical Analysis* 34.5 (Oct. 1, 1997). ISSN: 0036-1429. DOI: 10/bs8mtq 3.

[Özt16] ÖZTIRELI, A. C. "Integration with stochastic point processes". *ACM Transactions on Graphics* 35.5 (Aug. 2016). ISSN: 0730-0301. DOI: 10/f85k3g 2.

[PCX*18] PERRIER, H., COEURJOLLY, D., XIE, F., PHARR, M., HANRAHAN, P., and OSTROMOUKHOV, V. "Sequences with low-discrepancy blue-noise 2-D projections". *Computer Graphics Forum (Proceedings of Eurographics)* 37.2 (2018). DOI: 10/gd2j2d 3.

[PJH16] PHARR, M., JAKOB, W., and HUMPHREYS, G. *Physically Based Rendering: From Theory to Implementation*. 3rd. Cambridge, MA: Morgan Kaufmann, 2016. ISBN: 978-0-12-800645-0 2, 3, 8.

[PKK00] PAULY, M., KOLLIG, T., and KELLER, A. "Metropolis light transport for participating media". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. Vienna: Springer-Verlag, 2000. ISBN: 3-211-83535-0. DOI: 10/gfzm93 3.

[PLL09] PANG, F., LIU, M.-Q., and LIN, D. K. J. "A construction method for orthogonal Latin hypercube designs with prime power levels". *Statistica Sinica* 19.4 (2009). ISSN: 10170405, 19968507 3.

[Pre10] PRESSMAN, R. *Software Engineering: A Practitioner's Approach*. 7th ed. New York, NY, USA: McGraw-Hill, 2010. ISBN: 0-07-337597-7 3.

[PSC*15] PILLEBOUE, A., SINGH, G., COEURJOLLY, D., KAZHDAN, M., and OSTROMOUKHOV, V. "Variance analysis for Monte Carlo integration". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 34.4 (July 2015). ISSN: 0730-0301. DOI: 10/f7m28c 2, 7, 8.

[QA10] QIAN, P. Z. G. and AI, M. "Nested lattice sampling: a new sampling scheme derived by randomizing nested orthogonal arrays". *Journal of the American Statistical Association* 105.491 (2010). ISSN: 01621459. DOI: 10/fwjp8h 11.

[QAW09] QIAN, P. Z. G., AI, M., and WU, C. F. J. "Construction of nested space-filling designs". *The Annals of Statistics* 37 (6A Dec. 2009). ISSN: 0090-5364, 2168-8966. DOI: 10/fr79bb 11.

[Qia09] QIAN, P. Z. G. "Nested Latin hypercube designs". *Biometrika* 96.4 (2009). ISSN: 0006-3444. DOI: 10/cpmd77 11.

[RAMN12] RAMAMOORTHI, R., ANDERSON, J., MEYER, M., and NOWROUZEZAHRAI, D. "A theory of Monte Carlo visibility sampling". *ACM Transactions on Graphics* 31.5 (Sept. 2012). ISSN: 0730-0301. DOI: 10/gbbrnz 2, 3.

[Rao47] RAO, C. R. "Factorial experiments derivable from combinatorial arrangements of arrays". *Supplement to the Journal of the Royal Statistical Society* 9.1 (1947). ISSN: 14666162. DOI: 10/cck4gf 2, 3.

[RHVD10] RENNEN, G., HUSSLAGE, B., VAN DAM, E. R., and DEN HERTOG, D. "Nested maximin Latin hypercube designs". *Structural and Multidisciplinary Optimization* 41.3 (Apr. 1, 2010). ISSN: 1615-1488. DOI: 10/cpqtq3 11.

[RRSG16] REINERT, B., RITSCHEL, T., SEIDEL, H.-P., and GEORGIEV, I. "Projective blue-noise sampling". *Computer Graphics Forum* 35.1 (Feb. 1, 2016). ISSN: 0167-7055. DOI: 10/f8dm4x 3.

[SEN10] SCHOEN, E. D., EENDEBAK, P. T., and NGUYEN, M. V. M. "Complete enumeration of pure-level and mixed-level orthogonal arrays". *Journal of Combinatorial Designs* 18.2 (2010). DOI: 10/bc4c23 10.

[Shi91] SHIRLEY, P. "Discrepancy as a quality measure for sample distributions". *Proceedings of Eurographics*. Eurographics Association, 1991. DOI: 10/gfznch 3–5.

[SJ17] SINGH, G. and JAROSZ, W. "Convergence analysis for anisotropic Monte Carlo sampling spectra". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10/gbxfhj 2, 3, 8.

[SK13] SUBR, K. and KAUTZ, J. "Fourier analysis of stochastic sampling strategies for assessing bias and variance in integration". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 32.4 (July 2013). ISSN: 0730-0301. DOI: 10/gbdg7c 2.

[SL06] STEINBERG, D. M. and LIN, D. K. J. "A construction method for orthogonal Latin hypercube designs". *Biometrika* 93.2 (2006). ISSN: 00063444. DOI: 10/bns86t 3.

[SLL09] SUN, F., LIU, M.-Q., and LIN, D. K. J. "Construction of orthogonal Latin hypercube designs". *Biometrika* 96.4 (Oct. 2009). ISSN: 0006-3444. DOI: 10/dkmc5r 3.

[SLQ14] SUN, F., LIU, M.-Q., and QIAN, P. Z. G. "On the construction of nested space-filling designs". *The Annals of Statistics* 42.4 (2014). ISSN: 0090-5364. DOI: 10/gf3nmd 11.

[SMJ17] SINGH, G., MILLER, B., and JAROSZ, W. "Variance and convergence analysis of Monte Carlo line and segment sampling". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 36.4 (July 2017). ISSN: 01677055. DOI: 10/gfzncj 2.

[SNJ*14] SUBR, K., NOWROUZEZAHRAI, D., JAROSZ, W., KAUTZ, J., and MITCHELL, K. "Error analysis of estimators that use combinations of stochastic sampling strategies for direct illumination". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 33.4 (June 2014). ISSN: 1467-8659. DOI: 10/f6fgw4 2.

[SÖA*19] SINGH, G., ÖZTIRELI, C., AHMED, A. G., COEURJOLLY, D., SUBR, K., DEUSSEN, O., OSTROMOUKHOV, V., RAMAMOORTHI, R., and JAROSZ, W. "Analysis of sample correlations for Monte Carlo rendering". *Computer Graphics Forum (Proceedings of Eurographics State of the Art Reports)* 38.2 (Apr. 2019) 2.

[Sob67] SOBOL, I. M. "The distribution of points in a cube and the approximate evaluation of integrals". *USSR Computational Mathematics and Mathematical Physics* 7 (1967). DOI: 10/crdj6j 2, 3, 11.

[SSJ16] SUBR, K., SINGH, G., and JAROSZ, W. "Fourier analysis of numerical integration in Monte Carlo rendering: theory and practice". *ACM SIGGRAPH Course Notes*. ACM Press, July 2016. ISBN: 978-1-4503-4289-6. DOI: 10/gfzncn 2, 8.

[Ste87] STEIN, M. "Large sample properties of simulations using Latin hypercube sampling". *Technometrics* 29.2 (1987). DOI: 10/gfzncm 7.

[SZG*13] SUN, X., ZHOU, K., GUO, J., XIE, G., PAN, J., WANG, W., and GUO, B. "Line segment sampling with blue-noise properties". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 32.4 (July 2013). ISSN: 0730-0301. DOI: 10/gbdg4r 3.

[Tan93] TANG, B. "Orthogonal array-based Latin hypercubes". *Journal of the American Statistical Association* 88.424 (Dec. 1993). ISSN: 1537-274X. DOI: 10/gfzncp 3, 5, 8.

[Tan98] TANG, B. "Selecting Latin hypercubes using correlation criteria". *Statistica Sinica* 8.3 (1998). ISSN: 10170405, 19968507 3.

[Vea97] VEACH, E. "Robust Monte Carlo Methods for Light Transport Simulation". Ph.D. Thesis. Stanford University, Dec. 1997 3.

[Wen14] WENG, J. "Maximin Strong Orthogonal Arrays". M.Sc. Thesis. Simon Fraser University, 2014 3, 11.

[WL13] WANG, K. and LI, Y. "Constructions of nested orthogonal arrays". *Journal of Combinatorial Designs* 21.10 (2013). ISSN: 1520-6610. DOI: 10/gf2zf9 11.

[YLL14] YIN, Y., LIN, D. K., and LIU, M.-Q. "Sliced Latin hypercube designs via orthogonal arrays". *Journal of Statistical Planning and Inference* 149 (June 2014). ISSN: 0378-3758. DOI: 10/gfzncs 3, 7, 11.

[ZJL*15] ZWICKER, M., JAROSZ, W., LEHTINEN, J., MOON, B., RAMAMOORTHI, R., ROUSSELLE, F., SEN, P., SOLER, C., and YOON, S.-E. "Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering". *Computer Graphics Forum (Proceedings of Eurographics State of the Art Reports)* 34.2 (May 2015). ISSN: 1467-8659. DOI: 10/f7k6kj 2.

[ZWD19] ZHANG, T.-F., WU, G., and DEY, A. "Construction of some new families of nested orthogonal arrays". *Communications in Statistics - Theory and Methods* 48.3 (Feb. 1, 2019). ISSN: 0361-0926. DOI: 10/gf2zf4 11.

Listing 4: *Pseudorandom permutation and floating point number generator functions (reproduced from Kensler [Ken13]).*

```
unsigned permute(unsigned i, unsigned l, unsigned p) {
    if (p == 0) return i;       // identity permutation when p == 0
    unsigned w = l - 1;
    w |= w >> 1;
    w |= w >> 2;
    w |= w >> 4;
    w |= w >> 8;
    w |= w >> 16;
    do {
        i ^= p;             i *= 0xe170893d;
        i ^= p       >> 16;
        i ^= (i & w) >> 4;
        i ^= p       >> 8;  i *= 0x0929eb3f;
        i ^= p       >> 23;
        i ^= (i & w) >> 1;  i *= 1 | p >> 27;
                            i *= 0x6935fa69;
        i ^= (i & w) >> 11; i *= 0x74dcb303;
        i ^= (i & w) >> 2;  i *= 0x9e501cc3;
        i ^= (i & w) >> 2;  i *= 0xc860a3df;
        i &= w;
        i ^= i       >> 5;
    } while (i >= l);
    return (i + p) % l;
}

float randfloat(unsigned i, unsigned p) {
    if (p == 0) return 0.5f;    // always 0.5 when p == 0
    i ^= p;
    i ^= i >> 17;
    i ^= i >> 10;       i *= 0xb36534e5;
    i ^= i >> 12;
    i ^= i >> 21;       i *= 0x93fc4795;
    i ^= 0xdf6e307f;
    i ^= i >> 17;       i *= 1 | p >> 18;
    return i * (1.0f / 4294967808.0f);
}
```