Differentiable Owen Scrambling

BASTIEN DOIGNIES, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, France
DAVID COEURJOLLY, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, LIRIS, France
NICOLAS BONNEEL, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, LIRIS, France
JULIE DIGNE, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, LIRIS, France
JEAN-CLAUDE IEHL, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, France
VICTOR OSTROMOUKHOV, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, France

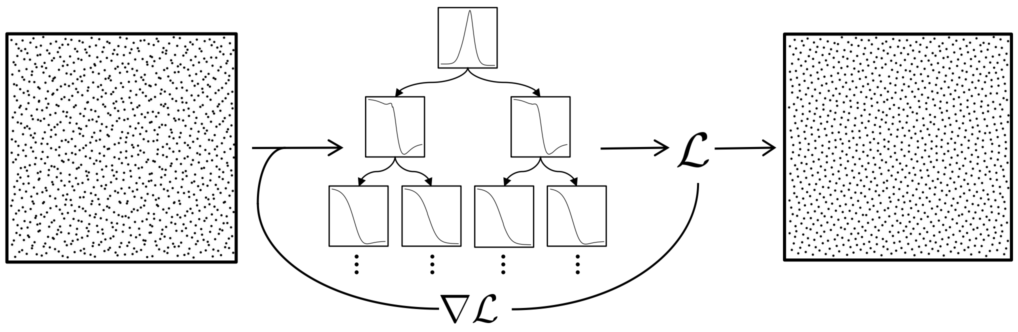

Fig. 1. Owen scrambling is a popular tool in Quasi-Monte Carlo to randomize samples by permuting elementary intervals of $[0,1)^s$. It relies on a tree of boolean flags swapping digits of the positional decomposition of the sample coordinates (see Fig. 2). Our Owen permutation tree replaces permutations by smooth transitions between intervals, rendering the Owen tree differentiable. This can be used in a smooth optimization framework to improve the quality of low-discrepancy point sets while preserving their low discrepancy. Here, starting with a random Owen tree (left), our optimization results in low discrepancy samples with minimized optimal transport cost (right). These samples give lower errors for equal sample counts in simple rendering settings. We also show for each node of the tree the effect of smoothly exchanging two intervals on an optimal transport loss when starting from the identity permutation (middle).

Quasi-Monte Carlo integration is at the core of rendering. This technique estimates the value of an integral by evaluating the integrand at well-chosen sample locations. These sample points are designed to cover the domain as uniformly as possible to achieve better convergence rates than purely random points. Deterministic low-discrepancy sequences have been shown to outperform many competitors by guaranteeing good uniformity as measured by the so-called discrepancy metric, and, indirectly, by an integer t value relating the number of points falling into each domain stratum with the stratum area (lower t is better). To achieve randomness, scrambling techniques produce multiple realizations preserving the t value, making the construction stochastic. Among them, Owen scrambling is a popular approach that recursively permutes intervals for each dimension. However, relying on permutation trees makes it incompatible with smooth optimization frameworks. We present a differentiable Owen scrambling that regularizes permutations. We show that it can effectively be used with automatic differentiation tools for optimizing low-discrepancy sequences to improve metrics such as optimal transport uniformity, integration error, designed power spectra or projective properties, while maintaining their initial t-value as guaranteed by Owen scrambling. In some rendering settings, we show that our optimized sequences improve the rendering error.

CCS Concepts: • **Mathematics of computing** → Automatic differentiation; • **Theory of computation** → **Pseudorandomness and derandomization**; • **Computing methodologies** → *Computer graphics.*

Additional Key Words and Phrases: Sampling, Owen scrambling, Quasi-Monte Carlo, Automatic differentiation

ACM Reference Format:
Bastien Doignies, David Coeurjolly, Nicolas Bonneel, Julie Digne, Jean-Claude Iehl, and Victor Ostromoukhov. 2024. Differentiable Owen Scrambling. *ACM Trans. Graph.* 43, 6, Article 255 (December 2024), 12 pages. https://doi.org/10.1145/3687764

Authors' addresses: Bastien Doignies, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, France, bastien.doignies@liris.cnrs.fr; David Coeurjolly, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, LIRIS, France, david.coeurjolly@cnrs.fr; Nicolas Bonneel, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, LIRIS, France, nicolas.bonneel@liris.cnrs.fr; Julie Digne, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, LIRIS, France, julie.digne@liris.cnrs.fr; Jean-Claude Iehl, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, France, jean-claude.iehl@liris.cnrs.fr; Victor Ostromoukhov, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, France, victor.ostromoukhov@liris.cnrs.fr.

1 INTRODUCTION

Monte Carlo integration is widely used in computer graphics, such as in rendering [Kajiya 1986; Cook 1986; Veach 1997; Keller 2013; Pharr et al. 2016], geometry processing [Sawhney and Crane 2020; Hermosilla et al. 2018] or image processing [Chan et al. 2014]. In the classical formulation, Monte Carlo techniques estimate the integral of a function by averaging evaluations of a function at random locations. Since, the expected numerical error is formally related to the estimator variance, many variance reduction techniques have been developed to accelerate convergence. Among these techniques, correlated sampling reduces variance by designing point sets that cover the integration domain much more uniformly, resulting in orders of magnitude lower integration errors [Sobol 1967; Niederreiter 1992] – a process called Quasi-Monte Carlo.

Following the Koksma-Hlawka inequality that bounds the integration error by a measure of samples uniformity [Niederreiter 1992], low-discrepancy point sets and low-discrepancy sequences are constructed with the objective of providing good sampling locations for most integration problems. A popular construction imposes multiple simultaneous stratification constraints [Owen 1997b,a; Grünschloß et al. 2008; Paulin et al. 2021, 2022; Ahmed et al. 2023b] and leads to so-called (t, m, s)-nets. Their quality is guaranteed by a low t value, an integer that determines the number of points falling inside each stratum, hence characterizing uniformity. However, these point sets are sometimes difficult to obtain, and producing many of them is intractable. For rendering, where each pixel may be estimated independently, using the same point set for all pixels introduces aliasing, and it is often better to use a different point set per pixel.

Introducing diversity often means running an entire machinery to produce a limited number of point sets (e.g., changing a random seed before solving a linear program [Paulin et al. 2022] or sieving among many candidates [L'Ecuyer and Munger 2016; Paulin et al. 2021]). But it can also be achieved by scrambling an existing point set via simple and cheap operations. For instance, Cranley-Patterson rotations translate points on a toroidal domain, which preserves lattices but negatively affects the t value of (t, m, s)-nets. For (t, m, s)-nets, Owen scrambling recursively permutes intervals for each dimension independently so that the number of points in each interval remains unchanged, preserving the t value. Owen scrambling may even improve the quasi-Monte Carlo convergence rate for smooth integrands [Owen 1997b,a]. Multiple attempts have thus been made at optimizing Owen scrambling to improve this integration convergence rate [Perrier et al. 2018; Perrier 2018]. However, these methods rely on random and blind exploration of a huge search space that grows exponentially fast with the number of points.

In this paper, we propose a novel differentiable formulation of Owen scrambling, and an optimization scheme that works in any dimension. Since our differentiable scrambling is based on Owen permutation trees, it exactly preserves the t value of the input point set. Our differentiable scrambling then allows optimizing other uniformity criteria for the input point set, such as optimal transport energies on the point set or its projections, integration error, or energies enforcing a prescribed power spectrum. Our implementation is available at https://github.com/liris-origami/DifferentiableOwenScrambling.

2 RELATED WORKS

We review the literature focusing on optimizing point sets and sequences for quasi-Monte Carlo applications.

(t, m, s)-nets. A low-discrepancy sequence (LDS) guarantees low integration error by controlling the *discrepancy* of generated sample points, a measure of their uniformity [Niederreiter 1992]. To obtain a LDS, a typical construction requires that when stratifying the domain, a point set of $n = b^m$ samples has exactly b^t samples in each stratum of volume b^{t-m}. Here, b is a fixed base (typically $b = 2$), and t is an integer characterizing the uniformity of the point set, related to its discrepancy [Niederreiter 1992]. Ideally, $t = 0$ indicates the best possible quality, where each stratum of volume b^{-m} contains a single sample. For $t = 1$, this enforces b samples per b times bigger stratum, so uniformity is less enforced than for $t = 0$. This property is at the core of (t, m, s)-nets, a construction of a low-discrepancy point set of b^m s-dimensional points of specific t value. Similarly, (t, s)-sequences are sequences of points for which the (t, m, s)-net property holds for all m, i.e., they remain low discrepancy when adding more points. Following Ahmed et al. [2023b] and Ostromoukhov et al. [2024], we denote (t, m, s)-progressive, point sets that are (t, m', s)-nets for all $m' \leq m$ given a fixed maximum value of m. (t, m, s)-nets can be obtained via algebraic construction [Sobol 1967; Niederreiter 1992; Bratley and Fox 1988], or solving complex systems of constraints [Ahmed and Wonka 2021; Paulin et al. 2022]. The resulting point sets and sequences offer the best Monte Carlo convergence rate. Our work allows to further optimize them according to other metrics using efficient convex optimization routines, while maintaining (t, m, s)-progressivity.

Point set randomization. Making point sets stochastic is desirable in many applications, and in particular for rendering where a different point set may be needed for each pixel. This can be done via Cranley-Patterson rotations that randomly but rigidly translate the point set modulo 1 [Cranley and Patterson 1976], although this operation degrades t. Digital shift applies a XOR operation (when $b = 2$) to the point coordinates with a random mask. This preserves t but provides little degrees of freedom in our context (typically 2^{32s} possible permutations) and does not allow for improved Monte Carlo convergence rate [Owen 2003]. Similarly, Linear Matrix scrambling [Hickernell 1996; Matoušek 1998] multiplies the bit vector by a random invertible triangular matrix, which offers much higher degrees of freedom (typically $2^{31 \cdot 30 \cdot s/2}$).

Owen scrambling recursively permutes half-spaces (when $b = 2$) for each dimension based on a decision tree, which also preserves t. This produces a vast exploration space that amounts to a different digital shift *per point*, but the decision tree is inherently not differentiable. Owen permutations do not cover *all* permutations that would preserve t [Ahmed and Wonka 2021], but the alternative permutations of Ahmed and Wonka [2021] covering them all are restricted to two dimensions. The number of degrees of freedom is again exponentially higher, typically $2^{(2^{32})s}$ and in practice, Owen scrambling performs better than Linear Matrix Scrambling [Owen 2003]. While Owen's theoretical construction is based on trees of infinite depth that preserve the (t, s)-sequence property, practical implementations require to fix their depth. In the general case, strictly speaking, this produces (t, m, s)-progressive samples for arbitrarily large m (where m equals the tree's depth). Producing a sequence would involve sequentially increasing the tree depth, but this would impact the lower significant digits of all previously generated samples. Restricting the space of permutations allows to produce true (t, s)-sequences by imposing that previously generated samples are not affected by lower tree levels [Perrier et al. 2018].

Blue noise. Blue noise point sets have attenuated low-frequencies in their power spectrum, which also offers low integration error guarantees [Durand 2011; Subr and Kautz 2013; Pilleboue et al.

2015]. The prevailing method to attain such spectrum is energy minimization. One of the earlier algorithms that exploits this idea is Lloyd's relaxation [Lloyd 1982] and its direct extension using capacity-constrained Voronoi's diagrams [Balzer et al. 2009; Li et al. 2010; Xu et al. 2011; Du et al. 1999; Chen et al. 2012]. The connection to semi-discrete optimal transport was made explicit and extended in BNOT [de Goes et al. 2012] which led to many applications in computer graphics. While these methods theoretically extend to any dimension, they rely on Voronoi or Power Diagrams that are hard to construct in high dimensions. To overcome this limitation, the use of a sliced optimal transport energy (SOT) has been proposed [Paulin et al. 2020; Salaün et al. 2022]. A similar spectrum can be obtained by optimizing distance-based filters [Fattal 2011; Heck et al. 2013]. The special case of Gaussian filters was recently studied in depth and produces state-of-the-art blue noise point sets in arbitrary dimensions [Öztireli et al. 2010; Ahmed et al. 2022]. The filters themselves can also be learned to target any (projectively) isotropic spectra [Leimkühler et al. 2019]. These energies can be used in our framework to further optimize low-discrepancy sequences.

Blue noise with (t, m, s)-net properties. Blue noise and low-discrepancy properties are hard to combine. The low-discrepancy property requires very fine structures that leave little room to move samples and control their power spectrum. An attempt to directly build blue noise low-discrepancy point sets can be found in the PMJ construction [Christensen et al. 2018] but this approach yields subpar results in terms of power spectra. An effective way to combine the two properties is to use permutations that preserve the low discrepancy property. In LDBN [Ahmed et al. 2016], the authors propose permutations on local 2D tiles that slightly affect discrepancy, but not sufficiently to significantly affect integration convergence rate. Starting with an exhaustive search on the first few levels of the Owen scrambling permutation tree (see Sec. 3.1) on local tiles, BNLD [Perrier et al. 2018] computes new local permutations that extend the blue noise property to the whole point set but this method only enforces blue noise on 2D projections. More recently, Ahmed and Wonka [2021] introduced a novel scrambling technique that allows a brute force optimization to target a blue noise spectrum, although their approach remains limited to 2D and would be intractable for generic energy functions (*e.g.*, optimal transport). In higher dimensions, another approach by Ahmed et al. [2023a] produces Owen trees that could be used to minimize any loss by similar local permutations, and offers preliminary 2D optimization results.

While differentiable approaches to optimize general permutations have been explored using Sinkhorn Networks [Mena et al. 2017], this would only permute the *order* of the points, without affecting their spatial relationships within each dimension.

3 A DIFFERENTIABLE OWEN SCRAMBLING FORMULATION

We first review the original Owen scrambling approach for completeness, and then describe our differentiable version.

3.1 Random Owen scrambling

Owen scrambling or Nested Uniform Scrambling is a widely used scrambling technique for (t, m, s)-nets. It has numerous interesting

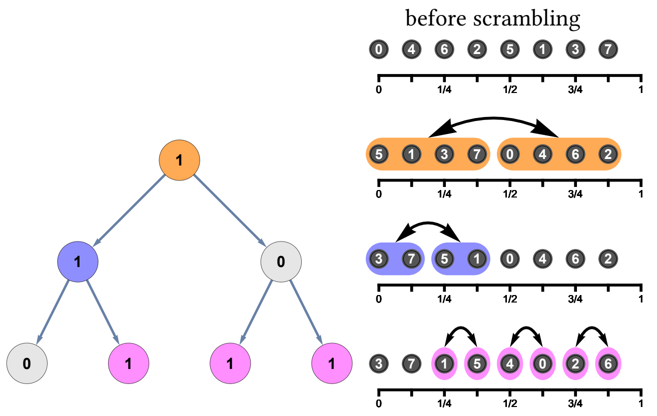

Fig. 2. *L*eft: Representation of an Owen scrambling permutations as a tree of depth 3. *R*ight: Discrete swaps applied on $2^3 = 8$ points of the second Sobol' dimension. Above the unit intervals, point indices (white numbers) are represented at their corresponding spatial locations. Each level of the tree decides if halves, quarters or eighths of intervals need to be permuted (tree node value = 1, colored) or not (tree node value = 0). Our framework makes these permutations differentiable.

properties: the space of different permutations is huge (in our case, with $b = 2$ and $n = 2^m$ points in s dimensions, there are $2^{(n-1)s}$ degrees of freedom), it is fast with moderate storage, and it can be done progressively one point at a time. Unlike other scrambling methods, not only does it preserve the point set quality as defined by its t value, but it can also improve it [Owen 1997a, 2003].

We first formally define Owen scrambling in 1d. Without loss of generality, we assume $b = 2$. Starting with a point expressed by its fixed-point binary representation on q bits (typically $q = 32$)

$$x = \sum_{i=1}^{q} a_i 2^{-i}, \tag{1}$$

the scrambled point is given by:

$$x' = \sum_{i=1}^{q} e_i 2^{-i}, \tag{2}$$

where the $\{e_i\}$ are obtained as: $e_i = \pi_{a_1 a_2 \ldots a_{i-1}}(a_i)$ and the $\{\pi_j\}$ are random (and possibly independent) permutations of $\{0, 1\}$. In base 2, there are only two permutations ($0 \leftrightarrow 1$), hence it is common to talk about *bit flipping* and it can be efficiently implemented via XOR operations. This scrambling is classically represented as a tree with branches encoding the original digits and nodes encoding the permutation to apply (see Figure 2). Alternatively, flipping bits can be interpreted as swapping half (or quarter, eighth, ...) spaces.

Scrambling trivially extends to any dimension by considering one permutation tree per dimension. Evaluating a scrambled point set of n points has $O(snq)$ time complexity and requires $O(s2^q)$ storage. In practice, for (t, m, s) inputs, we may consider using $q' = m$ bits instead of q, in which case complexity is $O(sn \log n)$ and storage is $O(sn)$. We may alternatively consider $m \leq q' \leq q$.

Owen permutation trees may be computed on the fly to avoid storing $s \cdot n$ values, resulting in no storage. A straightforward implementation relies on seeding a fast counter-based random number generator with random access [Salmon et al. 2011]. This also allows to run the scrambling at maximum depth (typically $q = 32$) obtaining a much larger diversity of point sets. Even faster scrambling can be achieved in base $b = 2$ with carefully designed hash structures [Burley 2020], and Owen scrambling and Sobol' construction of points sets can be performed simultaneously [Helmer et al. 2021].

3.2 Differentiable Owen scrambling

In this section, we present our continuous formulation of Owen scrambling. The scrambling process relies on bit flipping, which is a discrete operation transforming a 0 to a 1, or a 1 to a 0. We design a smooth function that achieves this effect. More formally, we give the following definition for our *differentiable binary flip* (DBF).

Definition 3.1 (Differentiable binary flip). Let f be a differentiable, bijective, strictly increasing function defined on $[0, 1]$ with values in $[0, 1]$ such that $f(0) = 0$ and $f(1) = 1$. We define a corresponding binary flip function $DBF_f : \{0, 1\} \times [0, 1] \mapsto [0, 1]$ as

$$DBF_f(\beta, \theta) := (1 - \beta)f(\theta) + \beta(1 - f(\theta)), \tag{3}$$

where β is the bit value, and $f(\theta)$ indicates by how much this value should change. While this formulation allows for the bit to take non integer values –a condition for its differentiability– it can still represent a *true* binary flip because $DBF_f(0, 0) = DBF_f(1, 1) = 0$ and $DBF_f(0, 1) = DBF_f(1, 0) = 1$. Since f defines a bijection from $[0, 1]$ to $[0, 1]$, DBF_f has no local minimum with respect to its second argument θ. In practice we use a tanh function for f: $f(\theta) = \frac{1}{2}\left(\tanh(\alpha(\theta - 0.5)) + 1\right)$, where α is the smoothing parameter. We use $\alpha = 5$ in our experiments.

ALGORITHM 1: Differentiable Owen Scrambling

```
Data: $x$: the coordinate to scramble, $q'$: the scrambling depth, $\theta$: the
      fuzzy tree parameters.
Result: $x'$: the continuous owen scrambling of $x$ with parameters $\theta$.
1 $select \leftarrow 1$;
2 $bits \leftarrow$ BinaryExpansion$(x, depth)$;
3 for $i \leftarrow 1$ to $q'$ do
4     $bits[i] \leftarrow \mathrm{DBF}_f(bits[i], \theta[select])$;
      // classical Owen scrambling:
      // $bits[i] \leftarrow bits[i]$ xor $tree[select]$
5     $select \leftarrow 2 \cdot select + bits[i]$;   // Binary heap traversal
6 end
7 $x' \leftarrow \sum_{i=1}^{q} 2^{-i} bits[i]$;
```

A DBF function may return non-integer values that represent fuzzy bits. However, points are represented using their binary decomposition $x = \sum_{i=1}^{q} a_i 2^{-i}$ with integers $a_i \in \{0, 1\}$. Our differentiable Owen scrambling keeps this representation, but uses fuzzy scrambled bits instead $x' = \sum_{i=1}^{q} a'_i 2^{-i}$ with $a'_i \in [0, 1]$. This still results in points within the unit square, but they are not guaranteed to lie on the classical dyadic grid anymore (i.e., where coordinates are multiple of negative powers of 2). In our implementation (see Algorithm 1), we store a flattened binary tree, and cannot benefit from fast hash-based or on the fly evaluation of the scrambling.

Our Algorithm 1 differs from the classical Owen scrambling in line 4, where we replace Owen scrambling's XOR operation to compute $bits[i]$ by our differentiable bit flip.

3.3 Optimization scheme

Equipped with our differentiable formulation, we are now ready to optimize low-discrepancy point sets. We minimize various differentiable losses (see Sec. 4) using either gradient descent or stochastic gradient descent (SGD). We require explicitly storing the entire Owen tree, and optimize all nodes values. In our context, Adam would require nearly 3 times higher memory usage, and we excluded it given the limited memory. To further reduce memory use, we use an explicit derivative $f'(\theta) = \frac{\alpha}{2}(1 - \tanh^2(\alpha(\theta - 0.5)))$ (DBF_f is only differentiated w.r.t. θ). We apply the chain rule for a given loss $\mathcal{L}$: $\frac{\partial \mathcal{L}(x(\theta))}{\partial \theta} = \left[\frac{\partial x_i}{\partial \theta_j}\right]_{ij} \nabla \mathcal{L}(x)$, where the loss gradient is explicited in Sec. 4. The (sparse) Jacobian matrix $\left[\frac{\partial x_i}{\partial \theta_j}\right]_{ij}$ is obtained using f', and its ratio of nonzero values for row j is $n/2^{\ell}$ with $\ell = \lfloor \log_2(j + 1) \rfloor$ (i.e., the number of points affected by level ℓ of the tree). We also optimize values over a fixed tree depth $q' = 16$.

Due to fixed-point arithmetic, components of loss gradients with respect to low significant bits rapidly diminish. However, moderately significant bits ($i \approx \log_2(N)$) are the ones most likely to significantly affect the energy (see Sec. 4.5 and Fig. 11). For this reason, we increase the learning rate linearly with the number of points.

After the optimization has terminated, each θ_i is rounded to either 0 or 1. The effect of this step is evaluated in Sec. 4.

3.4 Optimized losses

We use four losses: Optimal transport to uniform distribution, Gaussian Kernel Energy, Integration Error, Pair Correlation Function.

Optimal transport. We optimize the (squared) semi-discrete optimal transport cost between our point set $X = \{x_i\}_{i=1..n}$ and a uniform distribution, using a quadratic ground distance:

$$\mathcal{W}_2(X) = \inf_T \sum_i \int_{T^{-1}(x_i)} \|x - x_i\|^2 \mathrm{d}x\,. \tag{4}$$

This can be efficiently computed in 2D and 3D by an optimization process using a Newton solver [Lévy and Schwindt 2018], since $T^{-1}(x_i)$ results in a cell of a power diagram. It characterizes the uniformity of the point set and relates to the integration error [Paulin et al. 2020]. The gradient $\nabla W_2(X)$ is given by:

$$\nabla \mathcal{W}_2(X) = \left(\frac{\partial W_2(X)}{\partial x_i}\right)_i = \frac{2}{n}\left(\mathrm{centroid}(T^{-1}(x_i)) - x_i\right)_i, \tag{5}$$

where centroid$(T^{-1}(x_i))$ is the centroid of the power cell of x_i. We use a gradient descent to optimize our Owen permutation tree.

Gaussian kernel energy. The blue-noise enforcing energy introduced by Ahmed et al. [2021; 2022] describes the difference between the point set smoothed by a Gaussian kernel and a constant function:

$$\mathcal{G}(X) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} e^{-\|x_i - x_j\|^2/(2\sigma'^2)}\,. \tag{6}$$

Its gradients follows

$$\nabla \mathcal{G}(X) = \left(\frac{1}{n\sigma^2} \sum_{\substack{j=1 \\ j \neq i}}^{n} (x_j - x_i) e^{-\frac{\|x_j - x_i\|^2}{2\sigma^2}}\right)_i. \tag{7}$$

We use $\sigma = 0.5n^{1/s}$ and a gradient descent optimization.

Integration error. We optimize a sum of integration errors to a set of K random Gaussian distributions in the cases of 2D, 4D, 6D, or 8D point sets. We use $K = 65,536$ Gaussians with uniformly random mean and covariance matrices using UTK [UTK 2018]. The integral of each Gaussian $g_i(x)$ is pre-computed at high precision using 2^{28} samples. The integration error is defined as:

$$\mathcal{I}(X) = \frac{1}{K}\sum_{k=1}^{K}\left|\int g_k(x)dx - \frac{1}{n}\sum_{i=1}^{n} g_k(x_i)\right|^2 . \quad (8)$$

Its gradient is simply obtained by

$$\nabla\mathcal{I}(X) = -\frac{2}{nK}\sum_{k=1}^{K}\left(\int g_k(x)dx - \frac{1}{n}\sum_{i=1}^{n} g(x_i)\right)\nabla g_k(x) . \quad (9)$$

We use an SGD to optimize our Owen permutation tree, using batches of 512 Gaussians. When testing, we use $K' = 16,384$ different random Gaussians.

Pair Correlation Function (PCF).. A pair correlation function characterizes the spectrum of a point set as a distribution of distances between points [Öztireli and Gross 2012]. It can be explicited for a point set X as a 1d function of a radial parameter r:

$$\text{PCF}(X, r) = \frac{1}{\alpha(r)}\sum_{i=1}^{n}\sum_{j=1}^{n} e^{-\left(\|x_i - x_j\| - r\right)^2/(2\sigma'^2)} , \quad (10)$$

with the normalization $\alpha(r) = 2n(n-1)(\pi - 4r - r^2)r$. We use a square ℓ_2 distance between a reference PCF (obtained by averaging ≈ 128 PCFs of n points with the desired distribution) $\widetilde{\text{PCF}}(r)$ and that of the current low-discrepancy point set X:

$$\mathcal{P}(X) = \int_{r=r_{\min}}^{r_{\max}} |\text{PCF}(X, r) - \widetilde{\text{PCF}}(r)|^2 \text{dr} . \quad (11)$$

Its gradient is obtained using:

$$\frac{\partial\mathcal{P}(X)}{\partial x_i} = \frac{-2}{\alpha(r)\sigma'^2}\int_{r=r_{\min}}^{r_{max}}\left(\text{PCF}(X, r) - \widetilde{\text{PCF}}(r)\right)\cdot$$
$$\sum_{\substack{j=1\\ j\neq i}}^{n}(x_i - x_j)\left(1 - \frac{r}{\|x_i - x_j\|}\right)e^{-\left(\|x_i - x_j\| - r\right)^2/(2\sigma'^2)} dr . \quad (12)$$

We discretize the PCF on 100 bins, we use $\sigma' = 10^{-3}$, $r_{\min} = 0.01$, $r_{\max} = 0.1$, and a gradient descent Owen tree optimization.

We may also optimize a mixture of the above losses, notably to enforce them for different projections (see sec. 4.7) or to enforce them at different sample counts (sec. 4.6).

4 NUMERICAL RESULTS

4.1 Comparisons to other methods

We compare our approach that mathematically preserves the low discrepancy property and the actual value of t while optimizing other metrics, to methods that either only optimize these metrics or also preserve low discrepancy properties. Representative point sets for our results and for the identified best performing competing approaches are shown in Fig. 3, and direct comparisons in terms of metrics for a larger set of related works are shown in Fig. 5.

When optimizing $\mathcal{W}_2$, we compare to the state-of-the-art optimal transport-based BNOT approach of de Goes et al. [2012] in 2D or to the sliced optimal transport (SOT) of Paulin et al. [2020] which conveniently works in arbitrary dimensions. We also compare to the 2D low discrepancy blue noise (LDBN) approach of Ahmed et al. [2016] mimicking a BNOT spectrum while remaining low discrepancy. We also compare to Gaussian Blue noise (GBN, [Ahmed et al. 2022]) for completeness. In 2D, we obtain a similar $\mathcal{W}_2$ as BNOT while preserving low-discrepancy and t properties, and outperform LDBN. In 3D, the limited degrees of freedom do not allow us to reach the $\mathcal{W}_2$ level of SOT, but we improve over GBN and reduce the $\mathcal{W}_2$ energy from the initialization.

When optimizing the Gaussian Kernel Energy $\mathcal{G}$, we compare to Gaussian Blue Noise [Ahmed et al. 2022], specifically designed for minimizing this loss. We also compare to the 2D approach of Ahmed and Wonka [2021] (Blue-Nets) which preserves t and optimizes the Gaussian Kernel energy $\mathcal{G}$. For completeness, we also similarly compare to SOT and LDBN. While most methods perform similarly for this metric, we show modest improvements metric-wise over non-optimized Owen that remain visually noticeable (Fig. 3).

Regarding the integration loss $\mathcal{I}$, while no other method aims at directly minimizing this exact loss, low-discrepancy sequences such as Owen-scrambled Sobol' are state-of-the-art, at least in low-dimensional quasi-Monte Carlo integration contexts. For all metrics, we compare to an (unoptimized) ART-Owen scrambling [Ahmed et al. 2023a] on 16 symbols (ART16). Specifically for the integration loss, we have implemented a preliminary optimization strategy over ART (denoted ART16/Int.) following the paper's pseudocode [Ahmed et al. 2023a]. For completeness, we also compare to other techniques. Our Gaussian integrands cover a much wider range of covariance matrices than those used in the GBN paper [Ahmed et al. 2022], which explains the different results. In all tested dimensions (2D, 4D, 6D, 8D), our method performs best. In 2D, its closest competitor is Sobol', but in higher dimensions, optimal-transport based methods come second, outperforming Sobol'. Unoptimized ART-Owen over 16 symbols performs similarly to Owen on 32 bits.

For the PCF loss $\mathcal{P}$, we target a step function PCF, and we mainly compare to Heck et al. [2013] and LDBN [2016]. The approach of Heck et al. does not directly seek to minimize the same square ℓ_2 distance between PCFs, but can produce point sets following a step PCF. Similarly, LDBN matches a point set of step PCF but it is not driven by an energy minimization. For this reason, we only show qualitative comparisons in Figs. 6 and 3. For references, Figure 4 presents the Power spectra of our optimized point sets.

Using a weighted sum of two losses (Integration and Gaussian kernels, Fig.7) results in a tradeoff between both losses as expected, while improving both energies with respect to the initial point set (`Sobol'+Owen`).

4.2 Timings and bruteforce comparisons

The running time of our optimization is largely dominated by the evaluation of the loss and its gradient at each iteration. All our experiments were run on an AMD Ryzen 7 1700X 8-Core computer. Typical optimization times for 1k points in 2D at depth 16 range from 2s for Gaussian Kernel energy to 30s for Integration error, due

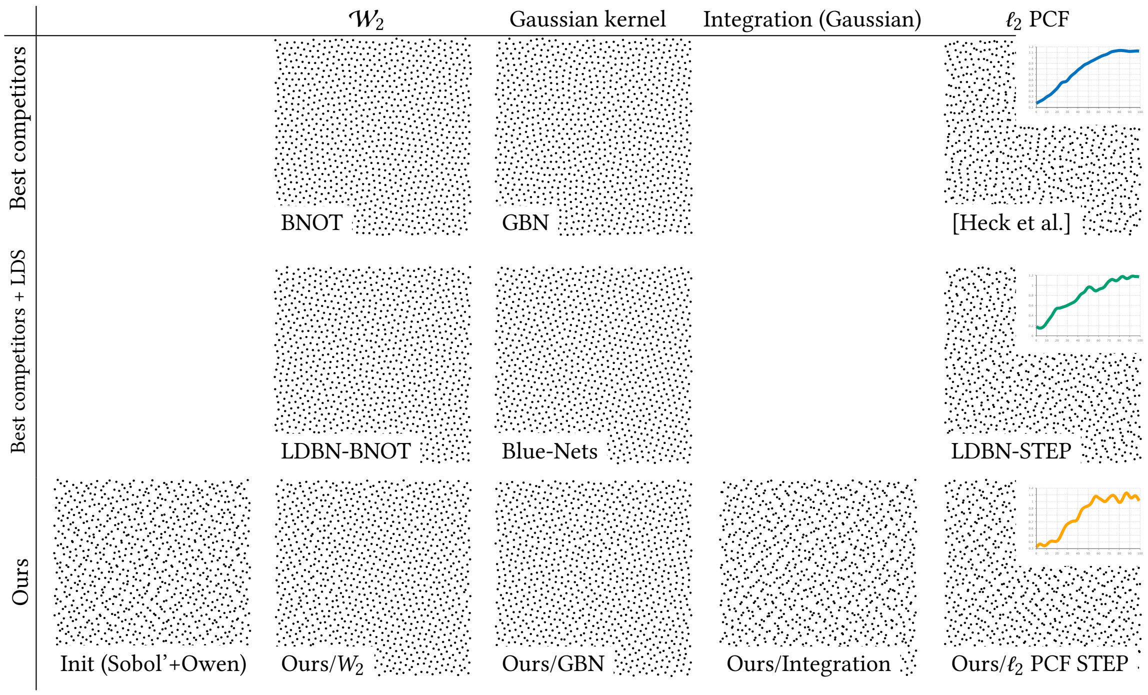

Fig. 3. For each loss, we identify a method that optimizes best ("Best competitors"), a method that optimizes best under the constraint that points remain of low discrepancy ("Best competitors + LDS") and compare the generated point sets to our method. Our best competitors are BNOT [de Goes et al. 2012] (in 2D), and the blue noise approach of Heck et al. [2013]. For best competitors under LDS constraints, we identified LDBN (in 2D) with a BNOT and STEP targets [Ahmed et al. 2016], and Blue-Nets (in 2D) [Ahmed and Wonka 2021].

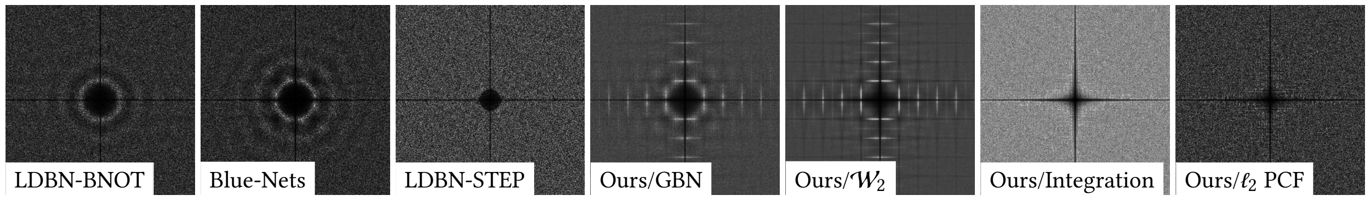

Fig. 4. For point sets given in Fig. 3, our Fourier Power Spectra exhibit the low discrepancy property (low energy on the axes, i.e., a black cross over the frequency domain), and some blue noise property (low energy in the low frequencies, i.e., a black disk at the origin), especially for GBN and $\mathcal{W}_2$ losses.

to the stochasticity of the optimization procedure. It also requires 1 GB of RAM to store the trees and Jacobian matrices. When applying a static Owen tree (eq. 2 on 16 bits), typical timings are 63.34M samples per second (averaging 64k realizations of 8D point sets of 256 samples). For the comparison, ART16 [Ahmed et al. 2023a] outputs 18.59Ms/sec. For completeness, fast hash-based techniques such as Burley's `FastOwenScrambler` in PBRT 4 [Burley 2020] can generate 103.48Ms/sec but without any control of the scrambled set.

As our method is tailored for smooth optimization, we provide equal time comparisons with respect to a more naive discrete optimization technique. Specifically, we compare to a simple optimization scheme that builds N random Owen trees and keeps the best performing one. We set N such that the total running time matches that of our method for the same loss. In Fig. 8, we show the relative reduction in loss for $\mathcal{W}_2$, $\mathcal{G}$, and $\mathcal{I}$ when using our approach and bruteforce optimization compared to the initial loss. Our optimization largely outperforms a bruteforce search in most cases.

4.3 Low discrepancy preservation

Our optimization scheme preserves t by construction. We illustrate this property in Fig. 9 by showing that all our optimized Owen trees result in the same discrepancies as Sobol' with random Owen permutations, which is the best discrepancy attained among all samplers we tested.

4.4 Effect of clamping

During optimization, all our fuzzy bit variables take continuous values in $[0, 1]$. Upon completion, bits are rounded to their nearest integer value (and all results presented in the paper use integer

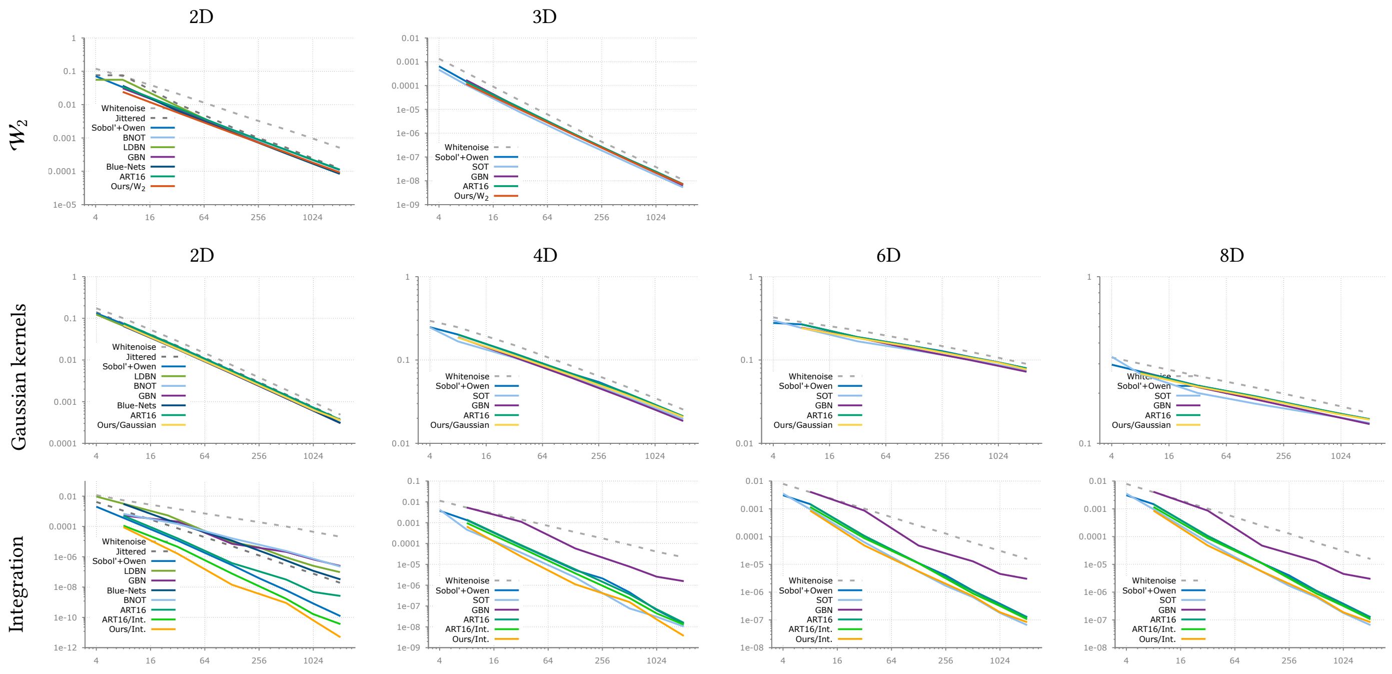

Fig. 5. We show the optimized point set losses for $\mathcal{W}_2$, Gaussian kernels $\mathcal{G}$, Integration $\mathcal{I}$, and compare them to state-of-the-art techniques.

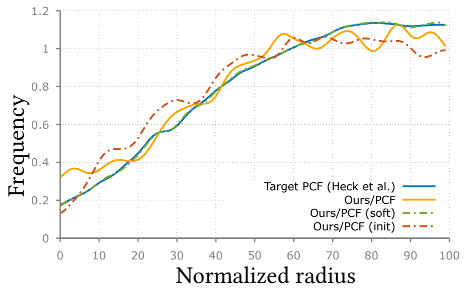

Fig. 6. For 1024 2D samples, we show the results of the PCF optimization using a squared ℓ_2 distance loss to a PCF from Heck et al. [2013]. Using dashed lines, we also illustrate the PCF of the initialization (Sobol'+Owen, labelled as *init*) and the PCF before the final clamping of the weights (*soft*). While we improve the PCF compared to the initialization, the clamping enforcing the LDS of the point set does affect the final PCF.

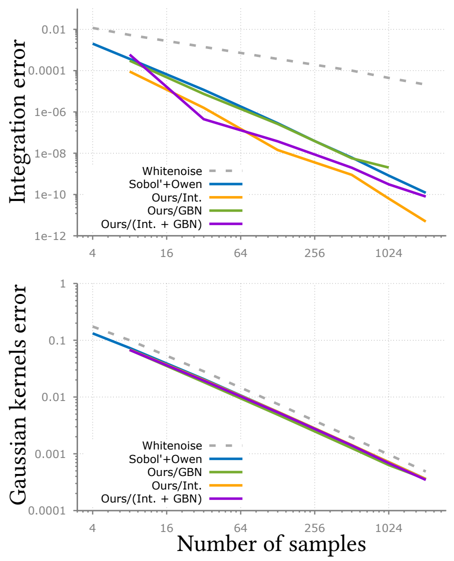

Fig. 7. Integration and Gaussian kernels errors while optimizing for each energy independently as well as a (weighted) sum of these two energies (denoted `Ours/(Int.+GBN)`).

bits in $\{0, 1\}$). While this is common in relaxed integer optimization [Burer and Letchford 2012], this may affect the energy, since the resulting tree is not optimal anymore for the integer-valued bits. We evaluate the effect of this rounding on the integration error energy $\mathcal{I}$ in Fig. 10.

4.5 Effect of tree level

It may appear surprising that smoothly exchanging two intervals would produce intermediate configurations meaningful enough for a smooth optimization solver to lower a loss often characterizing some uniformity measure. For instance, smoothly varying the most significant bit from 0 to 1 progressively exchanges the two halves of the unit domain, with a midpoint value representing a point set of much lower uniformity. However, smoothly exchanging bits of lower significance has a much less intuitive effect on the point set (recall that a decision to flip a given bit is sample-dependent, and depends on the value of the point's more significant bits). We run

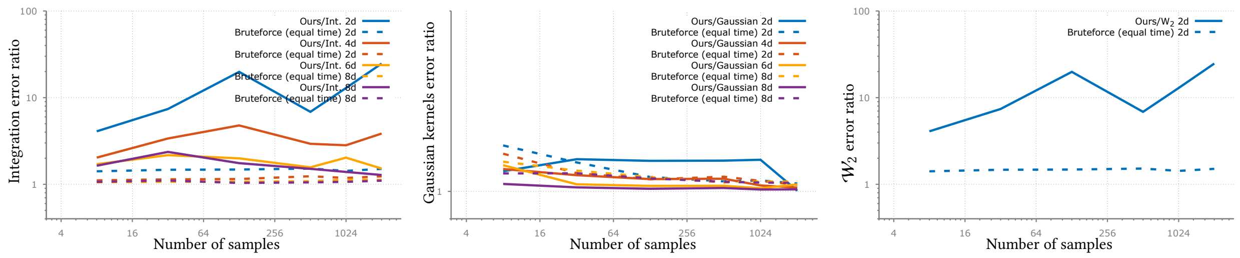

Fig. 8. We show the relative reduction in loss ($\mathcal{Loss}$(Sobol'+Owen32) / $\mathcal{Loss}$(optimized point set)) for our method compared to a bruteforce optimization scheme at equal running time (higher is better). Our smooth optimization largely outperforms a bruteforce search in most cases.

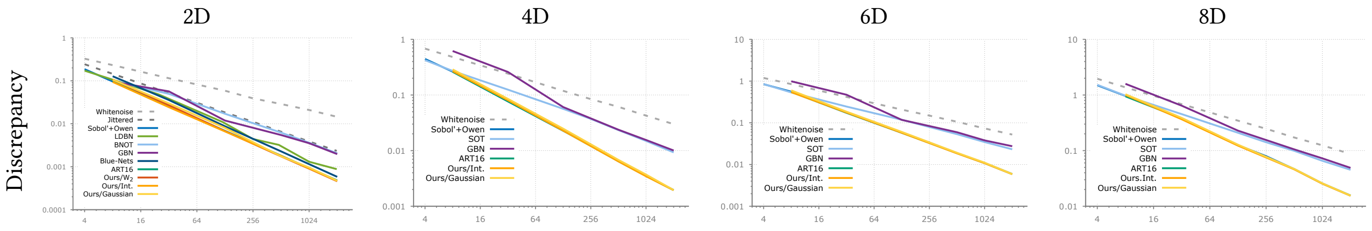

Fig. 9. Discrepancy evaluation. Our method is based on Owen scrambling and thus preserves the t value of the input point set. All our optimized permutations thus perform as well as Sobol' with random Owen permutations (our $\mathcal{W}_2$ and $\mathcal{I}$ discrepancy curves are superimposed with Sobol'+Owen).

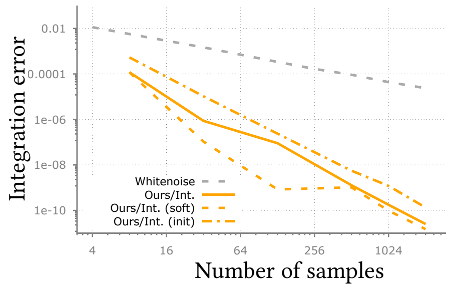

Fig. 10. Rounding the differentiable Owen tree to the nearest integer value affects the value of the loss being minimized. We show the energy value $\mathcal{I}$ in 2D before optimization (init), after optimization (soft), and with the full scheme (optimization+rounding, Ours/int).

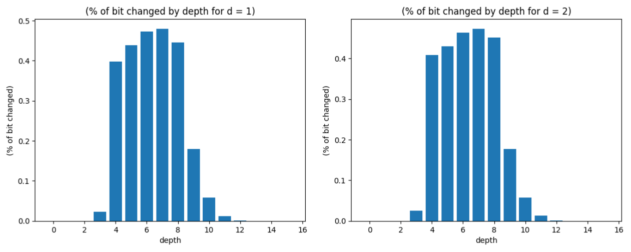

Fig. 11. We show the distribution of bits flipped as a function of the bit index (0 is the most significant) after optimizing a $\mathcal{W}_2$ loss over 2^8 2D samples (one histogram per dimension). Most bits are flipped in intermediate tree levels as applying our smooth permutation to highly significant bits makes little sense, while very low significant bits only slightly alters the loss.

an experiment on two trees of $q' = 16$ levels, optimizing a $\mathcal{W}_2$ loss on 2^8 2D points. We assess which levels of the permutations trees are more prone to change, i.e., which bit of the fixed-point representation of point coordinates are more likely affected by our optimization. We show the resulting histogram as a function of the bit index (bit 0 is the most significant) in Fig. 11. As expected, the most significant bit is never altered during the optimization. Most bit flips occur between the 4th and 8th bit, which can be explained as there are 2^8 samples. Bits of very low significancy also only have a minor effect on the energy as they correspond to subtle changes in point location. This behavior can be illustrated by the energy profile at each tree level: we start with an Owen tree entirely set to 0 and vary each node's value continuously from 0 to 1 independently. We show the resulting effect on the $\mathcal{W}_2$ loss in Fig. 1.

4.6 Multiscale optimization

Our method takes as input an Owen tree of fixed height $q' = 16$ and performs a single optimization. While this allows for the discrepancy and t value to be preserved, the losses we have introduced so far were only evaluated for a single sample count. The resulting low-discrepancy sequence thus only exhibits a minimal loss for a specific sample count. We evaluate the effect of minimizing an energy for a single sample count ($n = 2048$) on the Gaussian integration energy value of other sample counts in Fig. 12: the effect of the minimization process is only visible at the optimized sample count.

To alleviate this issue, we propose a multiscale loss that sums individual losses for multiple selected sample counts. We evaluate this strategy by summing Gaussian integration losses evaluated for 128, 512 and 2048 samples, and show the corresponding losses at other sample counts in Fig. 12 (see also Fig. 13 for point patterns illustration). We also show results for a more exhaustive loss accounting for sample counts of all powers of 2. This results in loss-specific

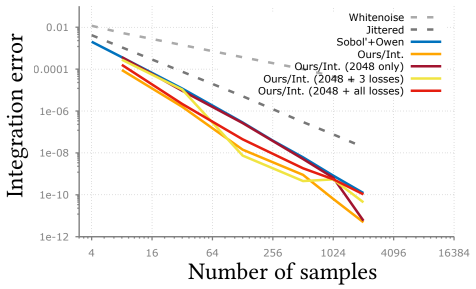

Fig. 12. Progressivity test. We compare various strategies for enforcing progressivity for the integration loss. We can either specifically optimize for n samples only ($n = 2048$), or sum of losses for a subset of the sample counts ($n = 128, 512, 2048$) or for all powers-of-two sample counts. We evaluate these strategies on the unoptimized sample counts, compared to reoptimizing a new tree per sample count (Ours/Int (pointset)).

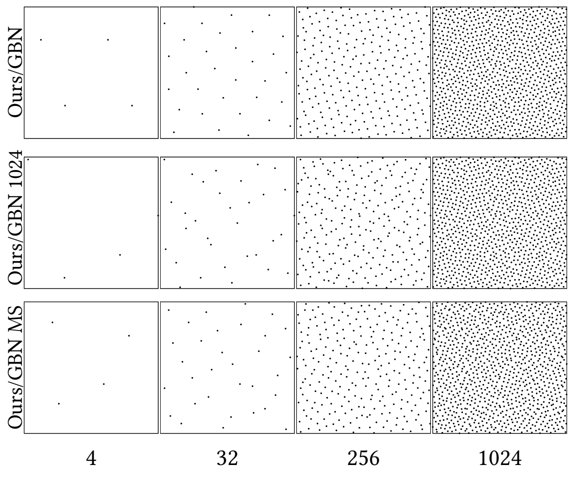

Fig. 13. Multiscale optimization in 2D. We show realizations when optimizing the Gaussian kernel loss independently for all sample counts (first row), only for 1024 samples (middle row), and jointly for all sample counts with the multiscale approach of Sec. 4.6 (last row).

progressive samplers, such as Gaussian integration-progressive or $\mathcal{W}_2$-progressive, in addition to remaining (t, m, s)-progressive by construction for (t, m, s)-progressive inputs. We can see a trade-off between progressivity and performance in terms of loss. While optimizing for all sample counts allows to reduce loss for all sample counts, quality remains lower than when minimizing for a few subsets of sample counts for these selected sample counts.

4.7 Rendering

We use our differentiable Owen scrambling to render images with a 6D integration domain. We integrate over the pixel area (2D), direct lighting (2D) and either depth of field or indirect lighting (2D). As a pre-process, a set of 64 different scramblings are optimized (starting from different initializations) to minimize

$$\mathcal{W}_2(\mathrm{proj}_{x^0x^1}(X)) + \mathcal{W}_2(\mathrm{proj}_{x^2x^3}(X)) + \mathcal{W}_2(\mathrm{proj}_{x^4x^5}(X)), \quad (13)$$

where $\mathrm{proj}_{x^kx^\ell}(X)$ projects the point set X onto 2 dimensions (k, ℓ). One of these realizations is picked at random for each pixel. Figure 14 illustrates the diversity of these realizations. Figure 15 (top) shows a smooth setting, with direct lighting, diffuse objects, a constant sky, a soft area light source and depth of field. Figure 15 (middle and bottom rows) shows scenes with one bounce of indirect lighting and diffuse and glossy materials. Starting from Sobol' [1967] or Cascaded Sobol' [Paulin et al. 2021], we perform the optimization for each sample count, and compare our results with the unoptimized point set in term of ℓ_1 error. Since Sobol' is (t, m, s)-progressive, a multiscale optimization denoted `"Ours (Sobol'- MS)"` has also been considered in Fig. 15 and 16. In the smooth setting, our optimized low-discrepancy point sets lower rendering error; in the discontinuous setting, our method performs equally (Fig. 15 bottom).

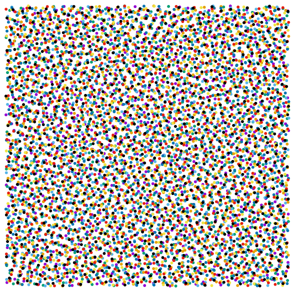

Fig. 14. Superimposed results of multiple optimization starting from different initial random Owen trees, exhbiting diversity (8 solutions here for 1024 samples for Gaussian kernel loss; each point set is color coded).

5 DISCUSSIONS & PERSPECTIVES

Our differentiable scrambling offers significant advantages to improve low-discrepancy sequences, but it also carries a number of limitations. First, while Owen trees are often evaluated on the fly thus requiring no storage, our optimized tree values need to be stored, requiring $O(n)$ storage. Our resulting trees could be further compressed as an interesting future work. Our process also requires $O(n^2)$ storage during the optimization to store the Jacobian, but due to its peculiar sparsity pattern and redundant values, this could supposedly be brought down to $O(n)$ with engineering efforts. Block coordinate descent could also help reduce memory use by only requiring blocks of Jacobian rows per iteration. Our implementation is limited to base $b = 2$ binary Owen trees, but extending it in higher bases [Faure and Lemieux 2016; Ostromoukhov et al. 2024] would be possible by considering an interpolation on a $(b - 1)$-dimensional simplex. Extending our approach to more complex Owen-like permutations such as the 2D approach of Ahmed and Wonka [2021] would be more difficult, as it requires non-trivial choices for permuting dimensions. Adapting our approach to the problem of smooth optimization of permutation sets [Mena et al. 2017] would be an interesting future work. Regarding progressivity with respect to given energies, we only minimize the energy for a specific subset of sample counts using a sum of losses approach. An interesting venue for future work would consist of a progressive per-level optimization of the tree. Nevertheless, we show that

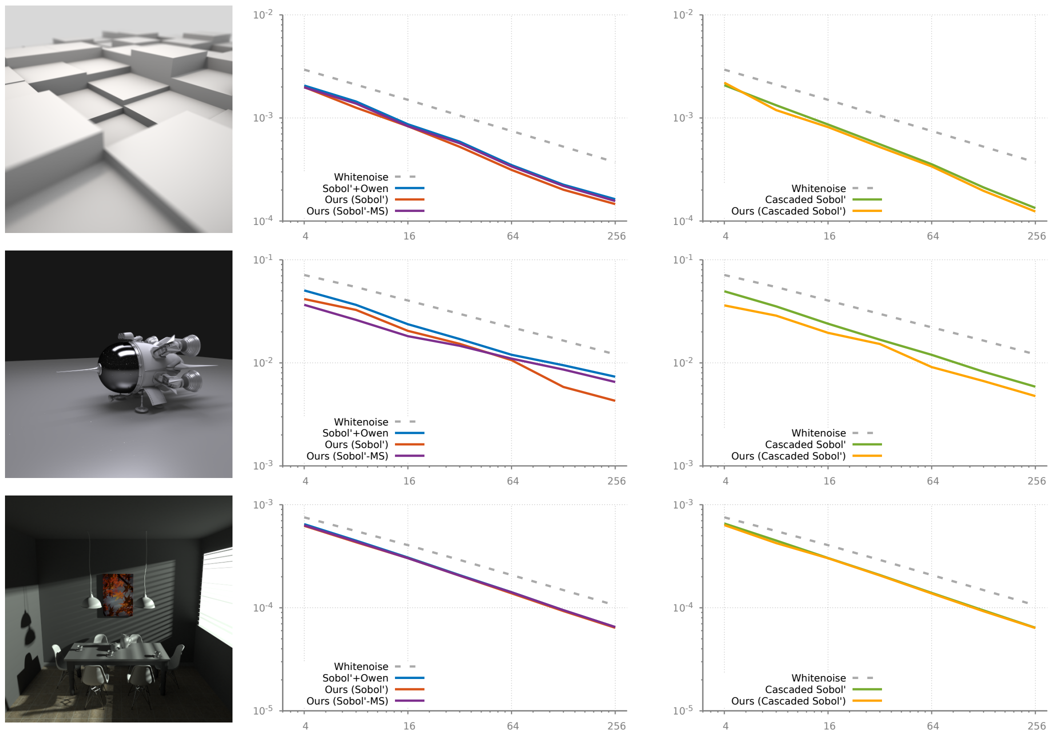

Fig. 15. 6D rendering ℓ_1 errors, before and after optimizing a sum of $\mathcal{W}_2$ losses on 2D projections, with the number of samples. (top) Smooth setting with pixel integration, direct lighting and depth of field. (middle) Pixel integration, direct and indirect lighting. (bottom) Same integration but less smooth setting.

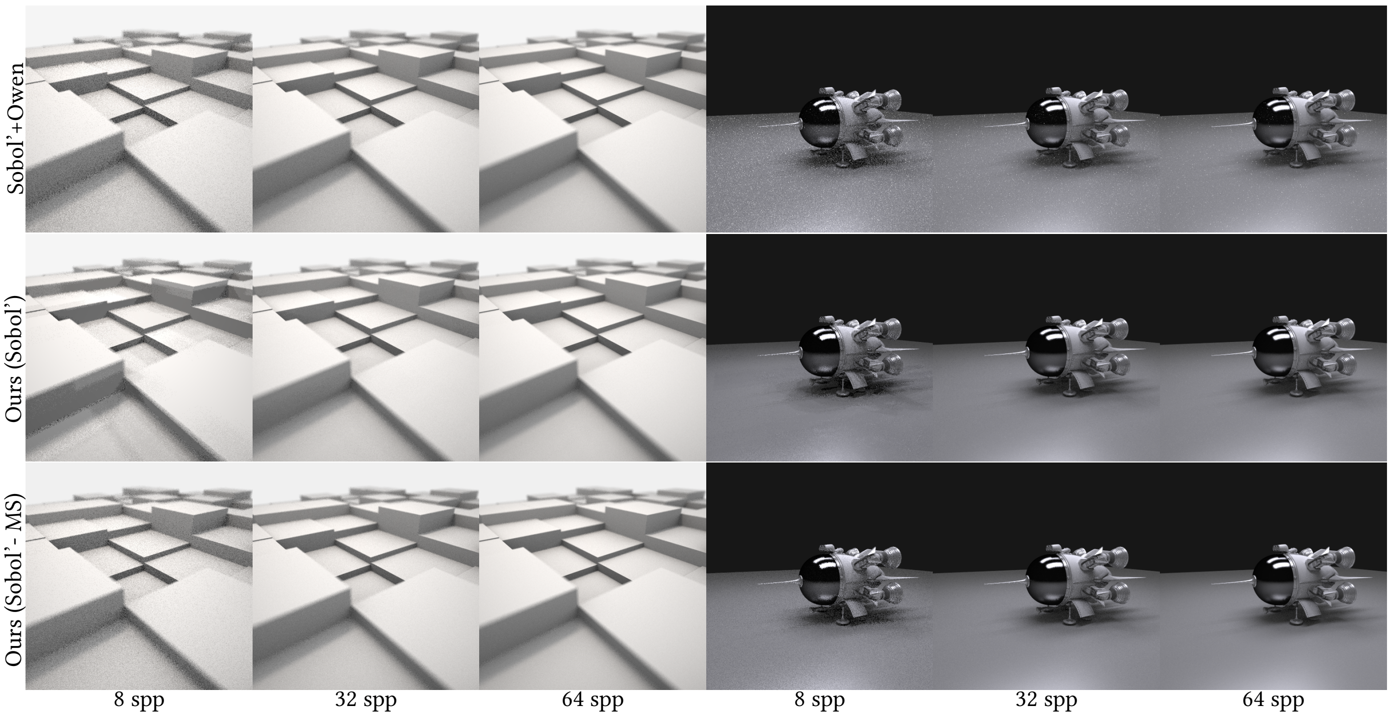

Fig. 16. We compare independently (middle) and jointly (bottom) optimized Sobol' with unoptimized Sobol'+Owen (top).

optimizing Owen trees can be performed in a smooth setting, and allows enforcing additional properties such as optimal transport uniformity, Gaussian integration efficiency, blue noise spectrum or other PCF while retaining the low discrepancy of the point sets.

ACKNOWLEDGMENTS

This work was partially funded by ANR-20-CE45-0025 (MoCaMed), by ANR-23-PEIA-0004 (PDE-AI), and donations from Adobe Inc.

REFERENCES

Abdalla G.M. Ahmed, Hélène Perrier, David Coeurjolly, Victor Ostromoukhov, Jianwei Guo, Dongming Yan, Hui Huang, and Oliver Deussen. 2016. Low-Discrepancy Blue Noise Sampling. *ACM Transactions on Graphics* 35, 6 (2016). https://doi.org/10.1145/2980179.2980218

Abdalla G.M. Ahmed, Matt Pharr, and Peter Wonka. 2023a. ART-Owen Scrambling. *ACM Transactions on Graphics* 42, 6 (2023), 1–11. https://doi.org/10.1145/3618307

Abdalla G.M. Ahmed, Jing Ren, and Peter Wonka. 2022. Gaussian Blue Noise. *ACM Transactions on Graphics* 41, 6, Article 260 (Nov. 2022), 15 pages. https://doi.org/10.1145/3550454.3555519

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

Abdalla G.M. Ahmed and Peter Wonka. 2021. Optimizing Dyadic Nets. *ACM Transactions on Graphics* 40, 4, Article 141 (jul 2021), 17 pages. https://doi.org/10.1145/3450626.3459880

Michael Balzer, Thomas Schlömer, and Oliver Deussen. 2009. Capacity-constrained Point Distributions: a Variant of Lloyd's Method. *ACM Transactions on Graphics* 28, 3, Article 86 (jul 2009), 8 pages. https://doi.org/10.1145/1531326.1531392

Paul Bratley and Bennett L. Fox. 1988. Algorithm 659: Implementing Sobol's Quasi-random Sequence Generator. *ACM Trans. Math. Softw.* 14, 1 (mar 1988), 88–100. https://doi.org/10.1145/42288.214372

Samuel Burer and Adam N Letchford. 2012. Non-convex Mixed-integer Nonlinear Programming: A Survey. *Surveys in Operations Research and Management Science* 17, 2 (2012), 97–106. https://doi.org/10.1016/j.sorms.2012.08.001

Brent Burley. 2020. Practical Hash-based Owen Scrambling. *Journal of Computer Graphics Techniques (JCGT)* 10, 4 (December 2020), 1–20. http://jcgt.org/published/0009/04/01/

Stanley H Chan, Todd Zickler, and Yue M Lu. 2014. Monte Carlo Non-local Means: Random Sampling for Large-scale Image Filtering. *IEEE Transactions on Image Processing* 23, 8 (2014), 3711–3725. https://doi.org/10.1109/TIP.2014.2327813

Zhonggui Chen, Zhan Yuan, Yi-King Choi, Ligang Liu, and Wenping Wang. 2012. Variational Blue Noise Sampling. *IEEE Transactions on Visualization and Computer Graphics* 18 (03 2012). https://doi.org/10.1109/TVCG.2012.94

Per Christensen, Andrew Kensler, and Charlie Kilpatrick. 2018. Progressive Multi-Jittered Sample Sequences. *Computer Graphics Forum* 37, 4 (2018), 21–33. https://doi.org/10.1111/cgf.13472

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

Roy Cranley and Thomas NL Patterson. 1976. Randomization of Number Theoretic Methods for Multiple Integration. *SIAM J. Numer. Anal.* 13, 6 (1976), 904–914. https://doi.org/10.1137/0713071

Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue Noise Through Optimal Transport. *ACM Transactions on Graphics* 31, 6, Article 171 (nov 2012), 11 pages. https://doi.org/10.1145/2366145.2366190

Qiang Du, Vance Faber, and Max Gunzburger. 1999. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Rev.* 41, 4 (1999), 637–676. https://doi.org/10.1137/S0036144599352836

Frédo Durand. 2011. A Frequency Analysis of Monte-Carlo and other Numerical Integration Schemes. *MIT CSAIL Technical report TR-2011-052* (2011). http://hdl.handle.net/1721.1/67677

Raanan Fattal. 2011. Blue-Noise Point Sampling using Kernel Density Model. *ACM SIGGRAPH 2011 papers* 28, 3 (2011), 1–10. https://doi.org/10.1145/1531326.1531328

Henri Faure and Christiane Lemieux. 2016. Irreducible Sobol' Sequences in Prime Power Bases. *Acta Arithmetica* 173, 1 (2016), 59–80. https://doi.org/10.4064/aa8226-1-2016

Leonhard Grünschloß, Johannes Hanika, Ronnie Schwede, and Alexander Keller. 2008. *(t, m, s)-Nets and Maximized Minimum Distance.* Springer Berlin Heidelberg, 397–412. https://doi.org/10.1007/978-3-540-74496-2_23

Daniel Heck, Thomas Schlömer, and Oliver Deussen. 2013. Blue noise sampling with controlled aliasing. *ACM Transactions on Graphics* 32, 3 (2013), 25:1–25:12. https://doi.org/10.1145/2487228.2487233

Andrew Helmer, Per Christensen, and Andrew Kensler. 2021. Stochastic Generation of (t, s) Sample Sequences. In *Eurographics Symposium on Rendering - DL-only Track*, Adrien Bousseau and Morgan McGuire (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20211287

Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. 2018. Monte carlo Convolution for Learning on Non-uniformly Sampled Point Clouds. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–12. https://doi.org/10.1145/3272127.3275110

Fred J. Hickernell. 1996. The Mean Square Discrepancy of Randomized Nets. *ACM Trans. Model. Comput. Simul.* 6, 4 (oct 1996), 274–296. https://doi.org/10.1145/240896.240909

James T. Kajiya. 1986. The Rendering Equation. *Computer Graphics* 20, 4 (1986), 143–150. https://doi.org/10.1145/15886.15902

Alexander Keller. 2013. Quasi-Monte Carlo Image Synthesis in a Nutshell. In *Monte Carlo and Quasi-Monte Carlo Methods 2012.* Springer, 213–249. https://doi.org/10.1007/978-3-642-41095-6_8

Pierre L'Ecuyer and David Munger. 2016. Algorithm 958: Lattice Builder: A General Software Tool for Constructing Rank-1 Lattice Rules. *ACM Trans. Math. Softw.* 42, 2, Article 15 (may 2016), 30 pages. https://doi.org/10.1145/2754929

Thomas Leimkühler, Gurprit Singh, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. 2019. Deep Point Correlation Design. *ACM Transactions on Graphics* 38, 6 (2019). https://doi.org/10.1145/3355089.3356562

Bruno Lévy and Erica L Schwindt. 2018. Notions of optimal transport theory and how to implement them on a computer. *Computers & Graphics* 72 (2018), 135–148.

Hongwei Li, Diego Nehab, Li-Yi Wei, Pedro V. Sander, and Chi-Wing Fu. 2010. Fast Capacity Constrained Voronoi Tessellation. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Washington, D.C.) *(I3D '10)*. ACM, Article 13, 1 pages. https://doi.org/10.1145/1730804.1730985

S. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

Jiří Matoušek. 1998. On the L2-discrepancy for Anchored Boxes. *J. Complex.* 14, 4 (dec 1998), 527–556. https://doi.org/10.1006/jcom.1998.0489

Gonzalo Mena, David Belanger, Gonzalo Munoz, and Jasper Snoek. 2017. Sinkhorn Networks: Using Optimal Transport Techniques to Learn Permutations. In *NIPS Workshop in Optimal Transport and Machine Learning*, Vol. 3.

Harald Niederreiter. 1992. *Random Number Generation and Quasi-Monte Carlo Methods.* Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611970081

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)*. ACM, New York, NY, USA, Article 72, 9 pages. https://doi.org/10.1145/3641519.3657431

Art B. Owen. 1997a. Monte Carlo Variance of Scrambled Net Quadrature. *SIAM J. Numer. Anal.* 34, 5 (1997), 1884–1910. https://doi.org/10.1137/S0036142994277468

Art B. Owen. 1997b. Scrambled Net Variance for Integrals of Smooth Functions. *Ann. Statist.* 25, 6 (1997), 1541–1562. http://dml.mathdoc.fr/item/1031594731

Art B. Owen. 2003. Variance with Alternative Scramblings of Digital Nets. *ACM Trans. Model. Comput. Simul.* 13, 4 (oct 2003), 363–378. https://doi.org/10.1145/945511.945518

A. Cengiz Öztireli, Marc Alexa, and Markus Gross. 2010. Spectral Sampling of Manifolds. *ACM Transactions on Graphics* 29, 6, Article 168 (dec 2010), 8 pages. https://doi.org/10.1145/1882261.1866190

A. Cengiz Öztireli and Markus Gross. 2012. Analysis and Synthesis of Point Distributions based on Pair Correlation. *ACM Transactions on Graphics* 31, 6 (2012), 174:1–174:6. https://doi.org/10.1145/2366145.2366189

Loïs Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Alexander Keller, and Victor Ostromoukhov. 2022. MatBuilder: Mastering Sampling Uniformity over Projections. *ACM Transactions on Graphics* 41, 4 (Aug. 2022). https://doi.org/10.1145/3528223.3530063

Lois Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Antoine Webanck, Mathieu Desbrun, and Victor Ostromoukhov. 2020. Sliced optimal transport sampling. *ACM Transactions on Graphics* 39, 4, Article 99 (aug 2020), 17 pages. https://doi.org/10.1145/3386569.3392395

Loïs Paulin, David Coeurjolly, Jean-Claude Iehl, Nicolas Bonneel, Alexander Keller, and Victor Ostromoukhov. 2021. Cascaded Sobol Sampling. *ACM Transactions on Graphics* 40, 6 (Dec. 2021), 274:1–274:13. https://doi.org/10.1145/3478513.3480482

Hélène Perrier. 2018. *Anti-Aliased Low Discrepancy Samplers for Monte Carlo Estimators in Physically Based Rendering.* Ph. D. Dissertation. Université Claude Bernard Lyon 1. http://www.theses.fr/2018LYSE1040/document

Hélène Perrier, David Coeurjolly, Feng Xie, Matt Pharr, Pat Hanrahan, and Victor Ostromoukhov. 2018. Sequences with Low-Discrepancy Blue-Noise 2-D Projections. *Computer Graphics Forum* 37, 2 (2018), 339–353. https://hal.science/hal-01717945

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc. https://doi.org/10.1016/C2013-0-15557-2

Adrien Pilleboue, Gurprit Singh, David Coeurjolly, Michael Kazhdan, and Victor Ostromoukhov. 2015. Variance Analysis for Monte Carlo Integration. *ACM Transactions on Graphics* 34, 4 (2015), 124:1–124:14. https://doi.org/10.1145/2766930

Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022. Scalable Multi-class Sampling via Filtered Sliced Optimal Transport. *ACM Transactions on Graphics* 41, 6 (2022). https://doi.org/10.1145/3550454.3555484

John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. 2011. Parallel Random Numbers: As Easy as 1, 2, 3. In *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12. https://doi.org/10.1145/2063384.2063405

Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-free Approach to PDE-based Methods on Volumetric Domains. *ACM Transactions on Graphics* 39, 4 (2020). https://doi.org/10.1145/3386569.3392374

Ilya M. Sobol. 1967. On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals. *USSR Computational mathematics and mathematical physics* 7 (1967), 86–112. https://doi.org/10.1016/0041-5553(67)90144-9

Kartic Subr and Jan Kautz. 2013. Fourier analysis of stochastic sampling strategies for assessing bias and variance in integration. *ACM Trans. Graph.* 32, 4 (2013), 128:1–128:12. https://doi.org/10.1145/2461912.2462013

UTK. 2018. Uniform Tool Kit. https://utk-team.github.io/utk/.

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph. D. Dissertation. Stanford University. https://graphics.stanford.edu/papers/veach_thesis/thesis-bw.pdf

Yin Xu, Ligang Liu, Craig Gotsman, and Steven Gortler. 2011. Capacity-Constrained Delaunay Triangulation for point distributions. *Computers & Graphics* 35 (06 2011), 510–516. https://doi.org/10.1016/j.cag.2011.03.031