Eurographics Symposium on Rendering 2022
A. Ghosh and L.-Y. Wei
(Guest Editors)

Volume 41 (2022), Number 4

Single-pass stratified importance resampling

Ege Ciklabakkal[1] Adrien Gruson[2] Iliyan Georgiev[3] Derek Nowrouzezahrai[4] Toshiya Hachisuka[1]

[1]University of Waterloo [2]École de Technologie Supérieure [3]Autodesk [4]McGill University

Abstract

Resampling is the process of selecting from a set of candidate samples to achieve a distribution (approximately) proportional to a desired target. Recent work has revisited its application to Monte Carlo integration, yielding powerful and practical importance resampling methods. One drawback of these methods is that they cannot generate stratified samples. We propose a method to achieve efficient stratification. We first introduce a discrete sampling algorithm which yields the same result as conventional inverse CDF sampling but in a single pass over the candidates, similarly to reservoir sampling. The algorithm traverses the candidate list adaptively from both ends, without needing to store them. We order the candidates along a space-filling curve to ensure that stratified CDF sampling of candidate indices yields stratified samples in the integration domain. We showcase our method on various resampling-based rendering problems.

CCS Concepts

• ***Computing methodologies*** → ***Rendering;***

1. Introduction

Resampling is a powerful method for approximately sampling from distributions that are difficult to handle directly. The basic approach is to stochastically select from a list of candidate samples (coming from another, known distribution), where the selection probability of each candidate is proportional to an associated weight. Judicious setting of these weights makes the resulting samples drawn approximately proportionally to a chosen target distribution. Talbot et al. [TCE05] showed how resampling can efficiently generate product-sampled ray directions for complex BSDFs and incident-radiance distributions. More recently, Bitterli et al. [BWP*20] applied such resampling to spatio-temporal reuse of path samples.

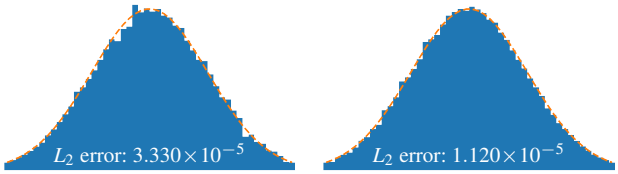
Two common algorithms for resampling are inverse cumulative distribution function (CDF) sampling and reservoir sampling [Cha82, Efr10]. The former precomputes a CDF from the candidate weights and inverts it to select a sample. The latter is a precomputation-free alternative that can select a sample according to the desired distribution in a single pass over all candidates.

To date, the key difference between the aforementioned two algorithms is understood as the need or absence of a precomputed CDF. We show that another significant difference lies in their *stratification* abilities. Put briefly, inverse CDF sampling can potentially transfer the stratification of the input canonical samples (e.g., coming from a low-discrepancy sequence) to the output samples, whereas reservoir sampling cannot. We also show that neither approach retains stratification when the sampling domain has more than one dimension. This latter limitation is due to the fact that resampling is always performed on the 1D space of candidate indices, thus the ordering of candidates impacts the output samples. As such, no efficient approaches exist to produce stratified samples with resampling.

We propose a solution to this problem. Our *bidirectional CDF sampling* algorithm combines the strengths of inverse CDF sampling and reservoir sampling, without their limitations. Specifically, it yields the same output as inverse CDF sampling (i.e., can retain input stratification), but in a single pass over the candidates, like reservoir sampling. This is achieved via an adaptive traversal of the candidate list from both ends. We augment this algorithm with *candidate ordering* along a space-filling curve to form a bijective map that preserves the locality of high-dimensional samples in their 1D index space. A similar idea has been used successfully in the context of importance sampling [SM03], and we show how it can be used for resampling.

Combining bidirectional CDF sampling with candidate ordering enables the first single-pass resampling method that achieves output stratification. An additional benefit of our method is that it can be easily combined with techniques that impose a blue-noise error distribution in image space [GF16], further improving the visual result. We demonstrate the benefits of our method across various light-transport simulation settings. Our novel contributions include:

- A resampling algorithm that yields identical results to classical inverse CDF sampling but in a single pass over the candidates;
- Ordering of candidates along a space-filling curve; and
- A practical stratified resampling-based rendering algorithm that exhibits blue-noise distribution of image error.



(a) Reservoir + 1D QMC + rescaling (b) Reservoir + MD QMC

(c) CDF + independent samples (d) CDF + 1D QMC

Figure 1: *Histogram comparison between reservoir sampling with rescaled 1D Sobol (i.e., van der Corput) samples (a) and M-dimensional Sobol samples (b), and inverse CDF sampling with 1D independent samples (c) and 1D Sobol samples (d). Sampling is done proportionally to Gaussian weights. Neither reservoir sampling variant retains the input's stratification; inverse CDF sampling does, producing a substantially lower L_2 histogram error.*

2. Motivation

We begin by reviewing the problem of stratification in importance resampling. We show two experiments to motivate our contributions.

Importance resampling [Rub87] aims to sample proportionally to a target function q by selecting from M candidate samples $\{y_k\}_{k=1}^{M}$ drawn from some probability density function (PDF) p. A weight $w_k = q(y_k)/p(y_k)$ is associated with each candidate, and an index j is sampled with probability proportional to the weight: $P(j) = w_j/\sum_{k=1}^{M} w_k$. The PDF of the output sample $x = y_j$ approaches proportionality to the target q as $M \to \infty$. Talbot et al. [TCE05] showed how to construct an unbiased Monte Carlo (MC) estimator for an integral (refer to Talbot [Tal05] for in-depth discussions):

$$\int_{\mathcal{H}} f(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{q(x_i)}\left(\frac{1}{M}\sum_{k=1}^{M} w_k\right), \tag{1}$$

which averages over N samples drawn from M candidates. In this work we consider integration over a unit hypercube $\mathcal{H}$ of problem-dependent dimension, a.k.a. primary sample space. Sampling in this space is traditionally uniform; the integrand f may internally warp x to the native integration domain (e.g., unit sphere, or path space, with an appropriate Jacobian). In our case, the candidates y_k are uniform (with $p(y_k) = 1$) but the final samples x_i generally are not.

As usual in MC integration, variance is reduced when the samples x_i are stratified. Achieving this efficiently is the goal of our work. Next we show that it requires a suitable resampling algorithm and careful candidate ordering, and why existing methods struggle.

2.1. Stratification of sampled indices

Consider the problem of sampling a candidate index j. The conventional approach is to select it as $j = F^{-1}(u)$, where $u \in [0,1)$ is a canonical uniformly distributed input sample and F^{-1} is the inverse CDF of P. The mapping of the unit line to the index space via

(a) 2D candidates (b) QMC order (c) Our method

Figure 2: *The ordering of resampling candidates can substantially affect the stratification of the output samples. (a) We generate $M = 8,192$ Sobol samples on the unit square and weight them by a 2D Gaussian target function. We then select $N = 256$ of them via stratified inverse CDF sampling. (b) Processing the candidates in their Sobol-sequence generation order produces no stratification in the output. (c) Ordering the candidates using our proposed method (Section 4) yields stratified unit-square samples.*

F^{-1} typically preserves the locality, thus a set of N well-distributed canonical input samples yield N well-distributed indices.

As an alternative to inverse CDF sampling, reservoir sampling avoids any CDF computation, and keeps track of only one selected candidate as it sweeps over the candidates once. For each observed candidate, a canonical sample is consumed to decide whether to replace the candidate in the reservoir. Reservoir sampling thus requires M canonical samples to sample a single index j. In an attempt to obtain N stratified output indices, we consider two options. One is to use a set of N M-dimensional stratified canonical samples and consume one dimension of each for every reservoir decision. A low-discrepancy sequence like Sobol's [Sob67] can produce such an input. The second option is to use only N canonical random numbers but repeatedly shift and scale each for every reservoir decision, much like probability-tree traversal [MH97, Oga21].

Figure 1 compares histograms of inverse CDF sampling, with and without a stratified input, and reservoir sampling using the two aforementioned techniques. There are $M = 50$ candidates with Gaussian weights. We repeat the resampling process N times before estimating the samples' distribution. Both reservoir variants fail to preserve input stratification, producing histograms resembling that of inverse CDF sampling with independent input samples. In contrast, feeding stratified samples to inverse CDF sampling produces a high-quality distribution, but at the cost of precomputing and storing a CDF.

2.2. Impact of candidate order

One may intuitively expect a stratified resampling algorithm to yield stratified samples whenever the candidates themselves are stratified. However, this is not always true, as we illustrate in Fig. 2. We generate M stratified candidates on the unit square using a 2D Sobol sequence (Fig. 2a), which we then resample to produce $N < M$ samples distributed approximately according to a 2D Gaussian function. Even though we employ stratified inverse CDF resampling (using a 1D Sobol sequence), the resulting samples in Fig. 2b are not well stratified. Contrast this result to that in Fig. 2c which exhibits improved stratification; the only difference is that we have applied our proposed candidate ordering scheme. The takeaway here is that

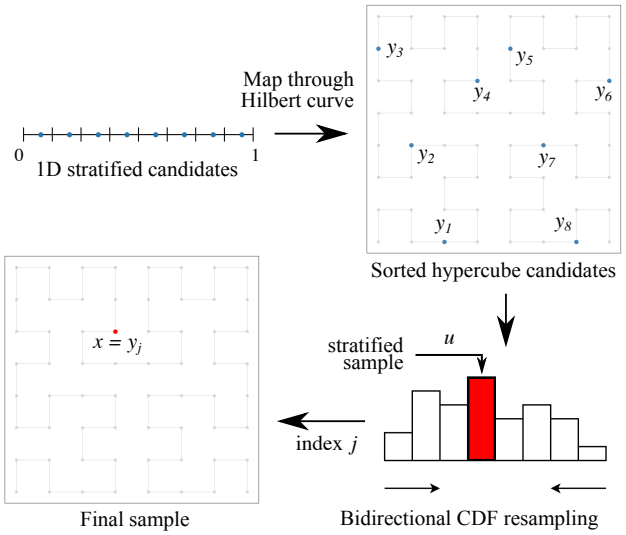


Figure 3: *Overview of our algorithm. Sorted 1D stratified samples are mapped onto a space-filling curve to produce stratified candidates y_k on the unit hypercube. We resample these candidates along the curve proportionally to their associated weights using our on-the-fly bidirectional CDF sampling. Stratifying the resampling input u yields stratified output integration samples x.*

resampling operates only on the candidate indices, oblivious to the fact that they may correspond to points in a high-dimensional integration space. Stratification in the 1D index space thus may not correspond to stratification in the integration space, unless care is taken when mapping samples to indices.

2.3. Problem statement and overview

The above two experiments reveal two problems that need to be solved in order to achieve stratified resampling. The first is that stratification in the input canonical samples should be preserved by the index sampling algorithm. While inverse CDF sampling has the potential to achieve this, it may not be a viable option as it requires precomputation. As Bitterli et al. [BWP*20] pointed out, when resampling at every image pixel in parallel, it is desirable to use a method that avoids precomputation, as reservoir sampling does. The second problem is that a stratified resampling algorithm must be paired with an appropriate candidate ordering that retains locality between the sampling domain and the index space, so that stratified index selection translates to stratified output samples.

To solve the first problem, in Section 3 we propose a new algorithm that returns the same output as inverse CDF sampling but in a single pass, without precomputation or candidate storage, like reservoir sampling. In Section 4 we show that the second problem can be solved by ordering the candidates along a space filling curve in the hypercube. Figure 3 illustrates our overall algorithm.

Algorithm 1: Our bidirectional CDF sampling algorithm.

Input : $\{w_1, \dots, w_M\}$: weights, $u \in [0,1)$: random number
Output : Sampled index
Function `BidirectionalCDF` ($\{w_1, \dots, w_M\}$, u) :
  $front \leftarrow 1$
  $back \leftarrow M$
  $\overline{w}_{front} \leftarrow w_{front}$
  $\overline{w}_{back} \leftarrow w_{back}$
  while $front \neq back$ **do**
    if $\overline{w}_{front} \leq u \cdot (\overline{w}_{front} + \overline{w}_{back})$ **then**
      $front \leftarrow front + 1$
      $\overline{w}_{front} \leftarrow \overline{w}_{front} + w_{front}$
    else
      $back \leftarrow back - 1$
      $\overline{w}_{back} \leftarrow \overline{w}_{back} + w_{back}$
  return $front$

3. Single-pass bidirectional CDF sampling

We propose a new discrete sampling algorithm that bands together the advantages of inverse CDF sampling and reservoir sampling. It scans the list of candidates only once, processing them one by one without keeping any in memory, similarly to reservoir sampling. Crucially, it can produce stratified output like inverse CDF sampling.

Reservoir sampling processes a list of candidates in a streaming manner. We additionally assume that it is possible to also read the list backward from its tail. As we will discuss later (Section 6), this additional constraint is typically not a limitation in practical rendering applications.

We process the list of candidates one by one, from both ends. We keep track of a *front* index and a *back* index, respectively, and the corresponding running sums of weights $\overline{w}_{front}$ and $\overline{w}_{back}$. Given an input sample $u \in [0,1)$, at each step we increment *front* if $\overline{w}_{front} \leq u \cdot (\overline{w}_{front} + \overline{w}_{back})$, or decrement *back* otherwise, and update the corresponding running weight sum. We terminate when $front = back$ which we take as the sampled index j. We provide pseudocode in Alg. 1. Note that the algorithm does not require any precomputation and has a constant storage cost, just like reservoir sampling.

In Appendix A we prove that the result of our algorithm is *exactly the same* as that of inverse CDF sampling. To that end, we show that in both algorithms the output index j satisfies

$$\sum_{i=1}^{j-1} w_i \leq u \cdot \overline{w} < \sum_{i=1}^{j} w_i, \tag{2}$$

where $\overline{w} = \sum_{i=1}^{M} w_i$ is the sum of all weights. That is, both inverse CDF sampling and our bidirectional CDF sampling output the same index j given the same input sample u.

3.1. Taking multiple samples

When doing importance resampling, we often need to generate multiple samples ($N > 1$) from a given list of candidates. This is straightforward to do with inverse CDF sampling, once the CDF has been computed and stored in memory. Reservoir sampling requires maintaining N reservoirs, feeding every candidate to each. On the

other hand, our bidirectional CDF sampling from Alg. 1 can generate only a single sample in one pass. This is because it traverses the candidates in an order that depends on the input sample u. The naive approach of doing N full passes for as many samples can be inefficient. To offer a better solution, we analyze the variance of resampling-based multi-sample MC integration.

The variance of the estimator in Eq. (1), taking N samples from M candidates, is [Tal05]

$$\frac{1}{M}\operatorname{Var}\left(\frac{f}{p}\right)+\left(1-\frac{1}{M}\right)\frac{1}{N}\operatorname{Var}\left(\frac{f}{q}\right). \tag{3}$$

Recall that such sampling is not efficient to do with Alg. 1 as it would require N passes over all candidates. We therefore consider a variant of the estimator which takes one sample out of M/N candidates and we repeat this process N times to generate N samples. The total number of candidates is still $M/N \times N = M$. In other words, we split the set of M candidates into N subsets, take one sample from each subset, and average the N estimates. The variance of this alternative estimator is obtained from Eq. (3) by substituting $N \to 1$ and $M \to M/N$, and dividing the entire expression by N (i.e., the average over N repetitions). This yields

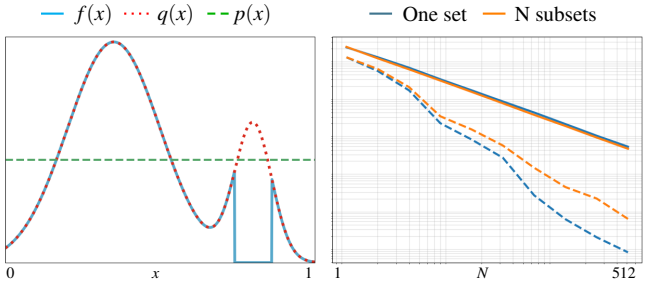
$$\frac{1}{M}\operatorname{Var}\left(\frac{f}{p}\right)+\left(1-\frac{N}{M}\right)\frac{1}{N}\operatorname{Var}\left(\frac{f}{q}\right). \tag{4}$$

Interestingly, this variance is *smaller* than the variance (3) of the estimator which takes N samples out of the entire set of M candidates.

The above analysis shows that we never need to consider generating multiple samples from a single candidate set at once: taking one sample from each of multiple subsets is always better. It also outlines a simple recipe for taking multiple samples using our algorithm by sweeping over all candidates exactly once. Note that this recipe is, in fact, a *requirement* when using Alg. 1. In contrast, reservoir sampling does not require major changes to generate more than one sample in a single pass. Nevertheless, our variance analysis suggests that it too should benefit from candidate splitting.

In Fig. 4 we numerically reproduce the results of the above variance analysis. We compare two variants of N-sample generation using our method: with repeated (One set) and split (N subsets) candidates. The former produces the same result as multi-sample CDF sampling. The experiment is thus effectively as a comparison between inverse CDF sampling (One set) and our proposed single-pass method (N subsets). The integrand f is zero in part of the domain, and the target function q mostly follows the integrand except for a small peak when f is zero. This setting corresponds to having two light sources where one is occluded and the target importance function excludes visibility. The candidate density p is uniform.

Using uncorrelated candidates and canonical input samples, subset resampling (solid orange curve in Fig. 4b) yields slightly lower variance, as the theory predicts. Stratifying candidates and samples (dashed curves) brings even more improvement, which is larger when not splitting the candidates. This result does not contradict the variance analysis which assumes fully independent sampling. Since repeating the same candidates may not be practical as we discussed above, we use the splitting approach in our experiments, even if it yields slightly higher variance.



(a) Experiment setup (b) Variance analysis

Figure 4: *Variance analysis of 1D importance resampling using our bidirectional CDF method (Alg. 1). (a) Candidates are generated with density p and resampled according to q which better matches the integrand f. (b) Plots of integral-estimation variance as a function of sample count N, with $M = 4N$ candidates. When the candidates and input samples are independent, subset-based resampling (solid orange) yields lower variance than always resampling from the entire candidate set (solid blue), in line with the theory in Section 3.1. Stratifying the candidates and the input samples reduces variance further (dashed curves), though slightly less with subset-based resampling.*

4. Ordering candidates on a space-filling curve

Our bidirectional CDF sampling algorithm can produce stratified indices, but this alone is not sufficient to achieve stratification after resampling. We need to ensure these indices map to stratified samples in the integration space. Typical candidate-generation orders, e.g., random, or the QMC order in Fig. 2b, do not provide a mapping with such correlation in the two different spaces. To that end, we propose to order the candidates along a space-filling curve.

Suppose we discretize the n-dimensional unit hybercube into a uniform grid with resolution 2^m along each axis and a total of 2^{nm} points. We can map a Hilbert curve onto that grid, which provides a bijective mapping between the index space $[1..2^{nm}]$ and the grid, allowing us to enumerate the grid points. Importantly, this mapping has the desirable property that any two points that are nearby along the curve are also nearby in the hypercube [Sag12]. Steigleder and McCool [SM03] used this property to map 1D stratified samples to approximately stratify high-dimensional samples. We use it to generate stratified resampling candidates

$$y_k = \text{Hilbert}\left(\left\lfloor \frac{k}{M} 2^{nm} \right\rceil\right), \quad \text{where} \quad \text{Hilbert} : [1..2^{nm}] \to [0,1]^n \tag{5}$$

and where $\lfloor\cdot\rceil$ denotes rounding to the nearest integer. This analytic mapping allows enumerating candidates on-the-fly in the order along the curve. Since the Hilbert curve preserves proximity, stratified indices j map to stratified candidates y_j. We set the grid resolution to $m = 32$ to represent 32-bit floating-point coordinates.

We generate candidates directly on the Hilbert curve through Eq. (5), but this is not the only option. An alternative is to use a stratified hypercube sampling method (e.g., a Sobol sequence) to generate $M = 2^{nm}$ candidates, one per stratum, and enumerate the strata along a Hilbert curve. On-the-fly resampling then requires the ability to compute the sample location inside a given stratum [GRK10].

5. Stratified resampling algorithm

The discrete sampling algorithm from Section 3 and the candidate ordering scheme from Section 4 can be combined to achieve efficient single-pass importance resampling that produces stratified samples.

Given a uniform sample $u \in [0,1)$, the resampling step returns an index $j = R(u)$ by going over the candidates once. The final output sample is thus $x = y_{R(u)} \in [0,1)^n$. Feeding N stratified canonical samples u_i yields N stratified hypercube samples x_i. However, as discussed in Section 3.1, to avoid iterating over all M candidates for every sample, in practice we split the candidates into N subsets and resample from each. The splitting is done by interleaving the indices, i.e., the candidates for sample $i \in [1..N]$ are $\{y_{i+kN}\}_{k=0}^{M/N-1}$. In conjunction with the Hilbert-curve ordering, this interleaving effectively bins the candidates into M/N hypercube strata and puts one candidate from each into every subset. And since the N input canonical samples are stratified, the resampling selects N well-distributed strata (one per subset), producing N stratified output samples x_i.

Blue-noise error distribution. Apart from stratification, another way to effectively improve rendering quality is to distribute pixel error as blue noise over the image. Since our approach is based on 1D sampling, it can be easily combined with the dithering method of Georgiev and Fajardo [GF16]. Using the same set of N stratified canonical samples u_i for the entire image, the samples for pixel p are obtained through offsetting: $u_{p,i} = \mathrm{mod}(u_i + o_p/N, 1)$, where the offset $o_p \in [0,1)$ is taken from a dither mask [GF16].

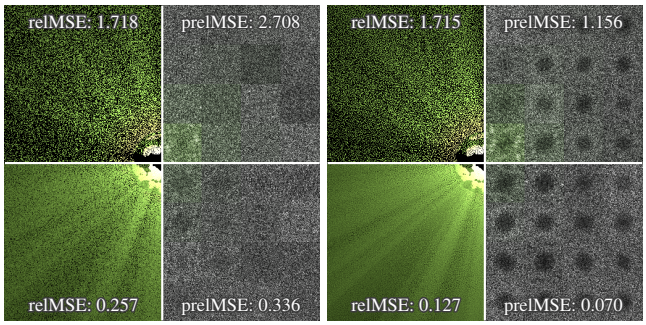
Note that this sample dithering produces blue-noise error distribution only if there is a chain of correlation between u_i, x_i, and $f(x_i)$ [HB19]. Our stratified resampling method specifically aims to maintain high correlation between u_i and x_i, and we assume x_i and $f(x_i)$ are correlated (see discussion in Section 4). In reservoir-based resampling, the correlation between u_i and x_i is low, which inhibits both stratification and blue-noise error distribution.

If the same set of M candidates used for the entire image, aliasing may occur. To avoid it, we also dither the candidates along the Hilbert curve across pixels, by applying a fractional offset to the index k in Eq. (5). We then require a dither mask with two offsets per pixel [GF16], one for the candidates and one for the samples.

6. Results

We implemented our method and other resampling strategies in the PBRT renderer [PJH16]. Being agnostic to the rendering process, our method can be easily integrated into other rendering systems.

We show equal-sample-count comparisons since the overhead of our method is insignificant. We measure the relative mean squared error (relMSE): $E = \sum_{i=1}^{n} \frac{(e_i - r_i)^2}{(r_i^2 + \varepsilon)}$, where r_i and e_i are respectively the reference and estimated values for the i-th pixel, and $\varepsilon = 0.001$. We also show "perceptual" relMSE (prelMSE), which is the same metric but with the images first blurred with a 2D Gaussian kernel of standard deviation 2.1 pixels [CGMS22]. We additionally include tiled error power spectra to demonstrate that our method can generate a blue-noise error distribution when dithering is applied, similarly to Chizhov et al. [CGMS22]. We do not compare against inverse CDF sampling; while it can also produce stratified samples, it requires precomputation, and our goal is to stick to single-pass resampling.



(a) Reservoir sampling (b) **Our method**

Figure 5: *Importance resampled single scattering in a medium with $N = 1$ (top row) and $N = 8$ (bottom row) samples. Our method yields lower pixel and perceptual error than reservoir-based resampling, thanks to its effective sample stratification and blue-noise dithering. prelMSE values are scaled by* $100\times$.

Single scattering in media. Our first experiment renders single scattering from a point light in a homogeneous participating medium with isotropic phase function. This is a 1D integration problem per pixel where we can readily define 1D ordered candidates. We use $M = 32$ stratified dithered candidates, each representing a distance along a ray, distributed proportionally to transmittance. We resample them with weight equal to the unoccluded path contribution, a target function similar to existing resampling applications [TCE05, BWP*20].

Figure 5 compares our method and weighted reservoir sampling [Cha82] with $N = 1$ and $N = 8$ final samples. Both methods use the same candidate sets and dithered stratified input samples. The tiled power spectra show that only our method produces a blue-noise error distribution. Also thanks to the stratification, our method has lower numerical and perceptual errors.

Direct illumination. We also apply our resampling method to direct illumination from area emitters, where a candidate on the unit square maps to an point on an emitter [PJH16]. The first dimension of the 2D candidate is used to choose an emitter. It is then rescaled so that the candidate can be used to sample the 2D point on the emitter. For the resampling target function, we again chose the unoccluded path contribution function. Figure 6 shows results with $N = 1$ sample from $M = 32$ candidates.

We use a 2D Halton sequence to generate candidates which we dither per pixel via 2D offsetting. We consider reservoir sampling with a stratified input (Fig. 6a) as a reasonable baseline. This method cannot produce blue-noise image-error distribution. We also show two other results where the candidates are reordered along a 2D Hilbert curve. Note that it is not the exact algorithm we propose since we actually store all the candidates and sort them afterward. This is done only to demonstrate the importance of candidate ordering. Again, reservoir sampling fails to produce a blue noise error distribution (Fig. 6b) as its mapping of canonical samples to indices

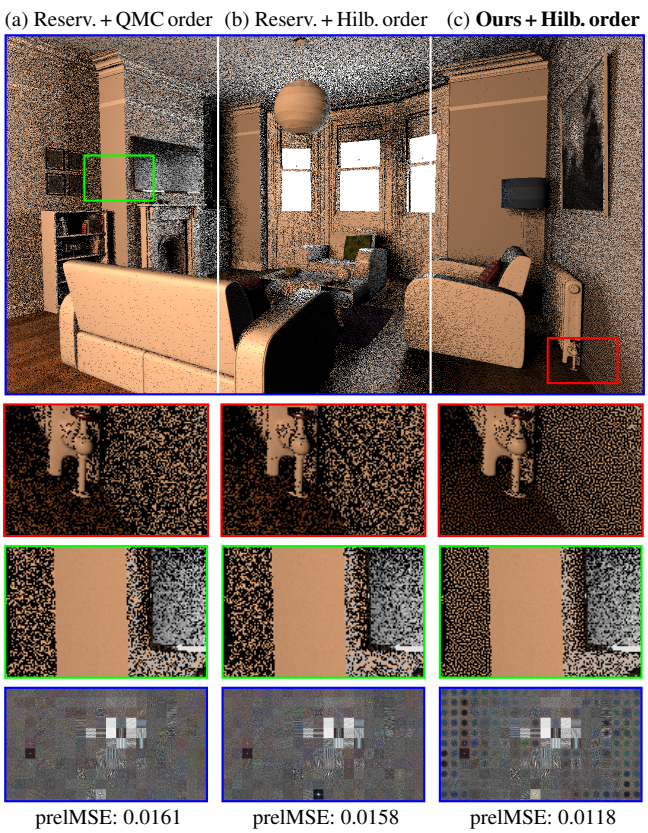


Figure 6: *Comparison of our bidirectional CDF resampling against reservoir baselines on area-light illumination with $N = 1$ sample from $M = 32$ Halton-sequence candidates. Dithered reservoir resampling fails to produce a pleasing error distribution, whether using the candidate generation order (a) or along a Hilbert curve (b). In contrast, our dithered bidirectional CDF resampling with Hilbert-curve ordering (c) produces a high-quality blue-noise distribution.*

does not preserve locality. In contrast, our technique produces the desired error distribution from the same inputs (Fig. 6c).

Figure 7 shows another scene where we resample on the fly from $M = 32$ candidates coming from an (unsorted) Halton sequence or our Hilbert-curve distribution (Section 4). We apply dithering to both the candidates and the resampling ($N = 1$). Our ordered candidate generation and resampling together produce a blue-noise error distribution (Fig. 7a). However, we are unable to obtain blue noise if one of these components is missing, i.e., using unordered Halton-sequence candidates (Fig. 7b) or reservoir sampling (Fig. 7c).

Resampled multiple importance sampling. Since we operate in primary sample space, to generate M MIS candidates from T techniques we can use the same set of M/T hypercube candidates, each producing one candidate per technique. This intentional correlation between the techniques facilitates stratification without adding bias. To draw N samples, we split the hypercube candidates into N subsets. The effective candidate count per subset is still M/N, as in non-MIS applications.

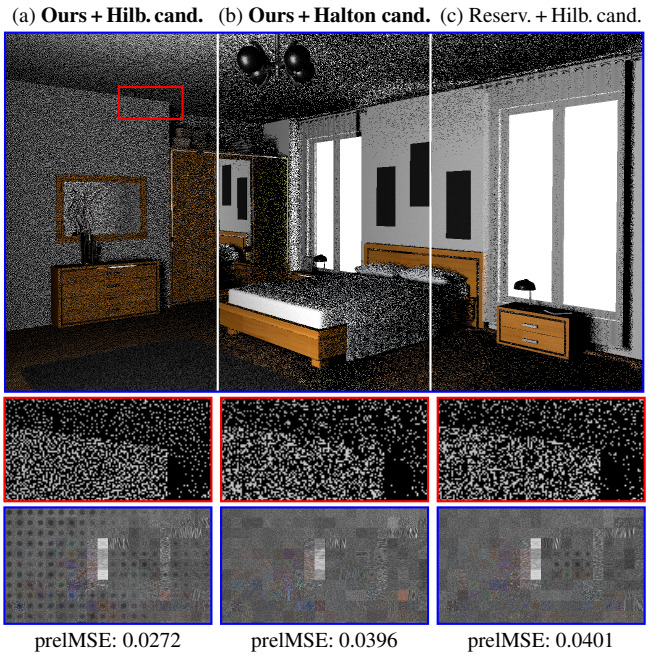


Figure 7: *Our candidate generation along a Hilbert curve, combined with our bidirectional CDF resampling, yields blue-noise error distribution (a). Using unsorted Halton-sequence candidates (b), or reservoir sampling (c), fails to attain such distribution.*

Figure 8 shows rendering of direct illumination for a scene lit with a complex environment map. We feed Halton-sequence and Hilbert-curve candidates to reservoir and bidirectional CDF resampling respectively. We combine emitter and BSDF techniques, and resample from $M = 32$ candidates (i.e., 16 hypercube candidates). Thanks to the careful stratification, our method produces lower error.

Higher-dimensional integration. Our method easily scales to higher dimensions, by using a Hilbert curve of corresponding dimension. Figure 9 shows a 3D integration problem of computing single scattering from multiple area emitters inside a homogeneous medium. The scene contains difficult visibility as the cage surrounding the main object heavily occludes light sources. We compare our method with 3D Hilbert-curve candidates to reservoir-based resampling of 3D Halton-sequence candidates. Here we do not observe blue noise even with our method. We believe this is due to the lack of sufficient correlation between resampling input and output. It still exhibits lower error with $N > 1$ samples thanks to their stratification.

Convergence. Figure 10 shows how error plots as functions of the number of final samples N and candidates M, with $M = 8N$. We compare our bidirectional CDF resampling and reservoir sampling with Hilbert-curve candidates to a reservoir baseline with Halton-sequence candidates, with dithering enabled for all. All three methods perform resampling in a single pass and have the same memory complexity. Our method performs best in all cases. Note that when the dimension of the integration problem increases, sample stratification becomes less effective and so does our method.

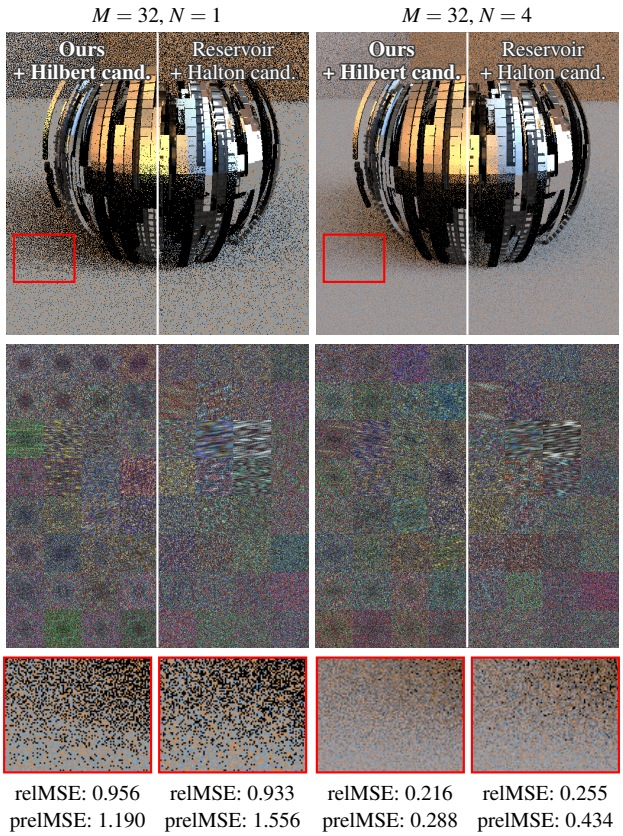


Figure 8: *Resampling an MIS mixture of direct-illumination candidates. 16 unit-square candidates produce $M = 32$ hemispherical candidates via environment-map and BSDF warping. For both $N = 1$ and $N = 4$ samples, our technique produces a blue-noise error distribution and reduces the overall error over reservoir-based resampling. prelMSE values are scaled by $100\times$.*

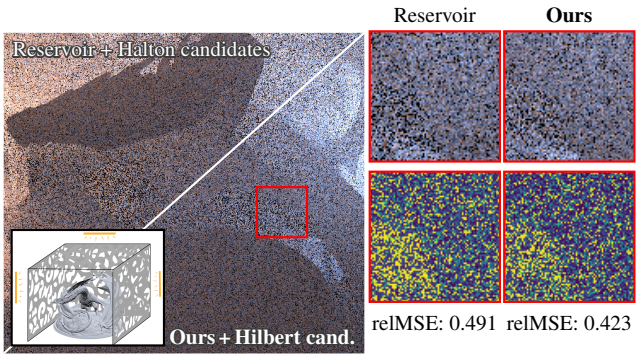


Figure 9: *Single scattering from multiple area lights in a medium. We compare our bidirectional CDF resampling of Hilbert-curve candidates to reservoir resampling of 3D Halton-sequence candidates ($M = 32$, $N = 8$). We include false-color error zoom-ins.*

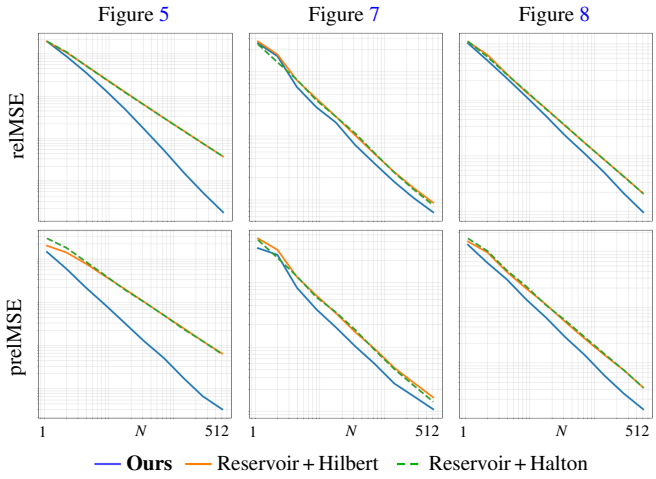


Figure 10: *Error plots as functions of the sample count N for different scenes, with $M = 8N$ candidates. Our technique always outperforms reservoir sampling variants, achieving a better convergence rate. All of the methods shown are single pass.*

7. Discussion and future work

7.1. Related work

Importance resampling was introduced to computer graphics by Talbot et al. [TCE05], showing how it can be a powerful and simple alternative when conventional importance sampling is not feasible. Recently, ReSTIR [BWP*20] used cascaded reservoir sampling [ES06] to combine samples across pixels and frames for direct lighting. That approach has been extended to indirect illumination by reusing subpaths after the first camera vertex [OLK*21], similarly to virtual-point-light approaches. Lin et al. [LWY21] extended the technique to heterogeneous participating media by using different transmittance approximations during the different resampling steps. Orthogonally to these works, Ogaki [Oga21] vectorized reservoir sampling to better utilize CPU parallelism. Our bidirectional CDF sampling has similar properties as reservoir sampling since both methods do not need to store any candidates and avoid precomputation. We thus expect that our technique can replace reservoir sampling in many existing applications when stratification is desired.

Our work enables a combination of resampling and blue-noise error distribution for the first time. To that end, we rely on dithered sampling [GF16]. Other approaches also exist [AW20, HBO*19, BH21], and it would be interesting to investigate their utility. Since Monte Carlo denoising can benefit from blue-noise error distribution [HB19], our approach could also enable an efficient application of denoising to resampling.

7.2. Limitations

While bidirectional CDF resampling achieves stratification in a single pass, it requires the ability to traverse the list of candidates from both ends, inlike reservoir resampling. Even though this requirement is typically not limiting in rendering applications, some applications may be incompatible with bidirectional CDF sampling.

Our method stratifies samples in the primary sample space, not the native integration domain. We assume that the mapping between

the two preserves the stratification, as in quasi-Monte Carlo literature [Kel13]. This assumption is empirically known to be violated in higher dimensions [HB19]. For example, while it is conceivable to perform stratification over the 8D primary space of four consecutive direction-sampling decisions along a path, the resulting paths are unlikely to be stratified. This is not a problem for any method that operates in primary sample space.

We generate and resample candidates along a Hilbert curve with resolution 2^{nm} which is the number of points on the n-dimensional grid it traverses. In our experiments we have $m = 32$ and $n = 1..3$, but always use 32-bit numbers to offset and sample along the curve. This means that not all grid points might be sampled. At the tested moderate sample counts N this does not seem to be an issue, though ideally numbers with precision at least 2^{nm} should be used.

7.3. Future work

Uniform sampling along a Hilbert does not yield very high-quality stratification in the hypercube. The point sets lack some desirable properties such as well-stratified lower-dimensional projections. Devising a better ordering that achieves such properties is an important direction for future work.

Recent work has shown that reservoir resampling in combination with spatio-temporal sample reuse can be a very effective approach for rendering complex illumination [BWP*20]. It would be interesting to explore extending our approach to maintain stratification with spatio-temporal reuse, which our preliminary experiments show is not straightforward.

Acknowledgements

This work has been partially funded by a grant from Autodesk. We thank Benedikt Bitterli for the scenes in Figs. 6 and 7 [Bit16]. The scene in Fig. 8 comes from PBRT's resources. The models used in Fig. 9 were obtained from Artec 3D.

References

[AW20] AHMED A. G., WONKA P.: Screen-space blue-noise diffusion of Monte Carlo sampling error via hierarchical ordering of pixels. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–15. doi:10.1145/3414685.3417881. 7

[BH21] BELCOUR L., HEITZ E.: Lessons learned and improvements when building screen-space samplers with blue-noise error distribution. In *ACM SIGGRAPH Talks* (2021). doi:10.1145/3450623.3464645. 7

[Bit16] BITTERLI B.: Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 8

[BWP*20] BITTERLI B., WYMAN C., PHARR M., SHIRLEY P., LEFOHN A., JAROSZ W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 39*, 4 (2020). doi:10/gg8xc7. 1, 3, 5, 7, 8

[CGMS22] CHIZHOV V., GEORGIEV I., MYSZKOWSKI K., SINGH G.: Perceptual error optimization for Monte Carlo rendering. *ACM Transactions on Graphics 41*, 3 (2022). doi:10.1145/3504002. 5

[Cha82] CHAO M.-T.: A general purpose unequal probability sampling plan. *Biometrika 69*, 3 (12 1982), 653–656. doi:10.1093/biomet/69.3.653. 1, 5

[Efr10] EFRAIMIDIS P. S.: Weighted random sampling over data streams, 2010. doi:10.48550/arXiv.1012.0256. 1

[ES06] EFRAIMIDIS P. S., SPIRAKIS P. G.: Weighted random sampling with a reservoir. *Information Processing Letters 97*, 5 (2006), 181–185. doi:10/cw2qc4. 7

[GF16] GEORGIEV I., FAJARDO M.: Blue-noise dithered sampling. In *ACM SIGGRAPH Talks* (2016), ACM Press, pp. 35:1–35:1. doi:10/gfznbx. 1, 5, 7

[GRK10] GRÜNSCHLOSS L., RAAB M., KELLER A.: Enumerating quasi-monte carlo point sequences in elementary intervals. *Monte Carlo and Quasi-Monte Carlo Methods* (2010). 4

[HB19] HEITZ E., BELCOUR L.: Distributing Monte Carlo errors as a blue noise in screen space by permuting pixel seeds between frames. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 38*, 4 (2019), 149–158. doi:10/ggjbxw. 5, 7, 8

[HBO*19] HEITZ E., BELCOUR L., OSTROMOUKHOV V., COEURJOLLY D., IEHL J.-C.: A low-discrepancy sampler that distributes Monte Carlo errors as a blue noise in screen space. In *ACM SIGGRAPH Talks* (2019), ACM Press, pp. 1–2. doi:10/ggjbxt. 7

[Kel13] KELLER A.: Quasi-Monte Carlo image synthesis in a nutshell. In *Monte Carlo and Quasi-Monte Carlo Methods* (2013), Dick J., Kuo F. Y., Peters G. W., Sloan I. H., (Eds.), Springer-Verlag, pp. 213–249. doi:10/gfz9gw. 8

[LWY21] LIN D., WYMAN C., YUKSEL C.: Fast volume rendering with spatiotemporal reservoir resampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 40*, 6 (12 2021), 278:1–278:18. doi:10.1145/3478513.3480499. 7

[MH97] MCCOOL M. D., HARWOOD P. K.: Probability trees. In *Graphics Interface* (1997), pp. 37–46. 2

[Oga21] OGAKI S.: Vectorized reservoir sampling. In *ACM SIGGRAPH Asia Technical Briefs* (2021), ACM Press, pp. 1–4. doi:10.1145/3478512.3488602. 2, 7

[OLK*21] OUYANG Y., LIU S., KETTUNEN M., PHARR M., PANTALEONI J.: ReSTIR GI: Path resampling for real-time path tracing. *Computer Graphics Forum 40*, 8 (2021), 17–29. 7

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Morgan Kaufmann, 2016. 5

[Rub87] RUBIN D. B.: The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association 82*, 398 (1987), 543–546. 2

[Sag12] SAGAN H.: *Space-Filling Curves*. Springer Science & Business Media, 2012. 4

[SM03] STEIGLEDER M., MCCOOL M. D.: Generalized stratified sampling using the Hilbert curve. *Journal of Graphics Tools 8*, 3 (2003), 41–47. doi:10.1080/10867651.2003.10487589. 1, 4

[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), 86–112. doi:10/crdj6j. 2

[Tal05] TALBOT J. F.: *Importance resampling for global illumination*. Brigham Young University, 2005. 2, 4

[TCE05] TALBOT J. F., CLINE D., EGBERT P.: Importance resampling for global illumination. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)* (2005), Eurographics Association, pp. 139–146. doi:10/gfzsm2. 1, 2, 5, 7

Appendix A: Equivalence of inverse CDF sampling and our bidirectional CDF sampling

We show that Eq. (2) holds for the output index j. We first prove by induction that the following condition $\mathcal{C}$ is satisfied at any stage of the algorithm for the *front* and *back* indices:

$$\mathcal{C}\ :\ \sum_{i=1}^{front-1} w_i \ \leq\ u\cdot\overline{w}\ <\ \sum_{i=1}^{back} w_i. \tag{6}$$

For the initial state, $front = 1$ and $back = M$, the inequalities hold trivially since $0 \leq u < 1$ and thus $0 \leq u\cdot\overline{w} < \overline{w}$.

Assume that $\mathcal{C}$ holds for indices $front < back$; we need to show that it still holds after one step of the algorithm. Recall that we keep track of the running sums $\overline{w}_{front} = \sum_{i=1}^{front} w_i$ and $\overline{w}_{back} = \sum_{i=back}^{M} w_i$. The algorithm handles the following two cases.

Case 1: If $\overline{w}_{front} \leq u\cdot(\overline{w}_{front}+\overline{w}_{back})$, we increment *front*. We then need to show that $\mathcal{C}$ still holds, i.e., that

$$\sum_{i=1}^{(front+1)-1} w_i \ =\ \sum_{i=1}^{front} w_i \ \leq\ u\cdot\overline{w}\ <\ \sum_{i=1}^{back} w_i. \tag{7}$$

The right inequality $u\cdot\overline{w} < \sum_{i=1}^{back} w_i$ holds by the inductive hypothesis since *back* remains unchanged. For the left inequality, noting that $(\overline{w}_{front}+\overline{w}_{back}) \leq \overline{w}$ for any $front < back$ and that $\overline{w}_{front} \leq u\cdot(\overline{w}_{front}+\overline{w}_{back})$ (the condition for this first case), we have

$$\sum_{i=1}^{front} w_i = \overline{w}_{front}\ \leq\ u\cdot\left(\overline{w}_{front}+\overline{w}_{back}\right)\ \leq\ u\cdot\overline{w}.\quad\square \tag{8}$$

Case 2: If $\overline{w}_{front} > u\cdot(\overline{w}_{front}+\overline{w}_{back})$, we decrement *back*. We then need to show that $\mathcal{C}$ still holds, i.e., that

$$\sum_{i=1}^{front-1} w_i \ \leq\ u\cdot\overline{w}\ <\ \sum_{i=1}^{back-1} w_i. \tag{9}$$

The left inequality $\sum_{i=1}^{front} w_i \leq u\cdot\overline{w}$ holds by the inductive hypothesis since *front* remains unchanged. For the right inequality, using the condition for this case $\overline{w}_{front} > u\cdot(\overline{w}_{front}+\overline{w}_{back})$, we have

$$\begin{aligned}\sum_{i=1}^{back-1} w_i = \overline{w}_{front} + \sum_{i=front+1}^{back-1} w_i\ &>\ u\cdot\left(\overline{w}_{front}+\overline{w}_{back}\right) + \sum_{i=front+1}^{back-1} w_i \\ &>\ u\cdot\left(\overline{w}_{front}+\overline{w}_{back} + \sum_{i=front+1}^{back-1} w_i\right) = u\cdot\overline{w}.\quad\square\end{aligned} \tag{10}$$

Therefore, the condition $\mathcal{C}$ holds at any stage of the algorithm. At the termination point, where the output index is $j = front = back$, that condition is identical to Eq. (2), and holds for j just like in inverse CDF sampling.