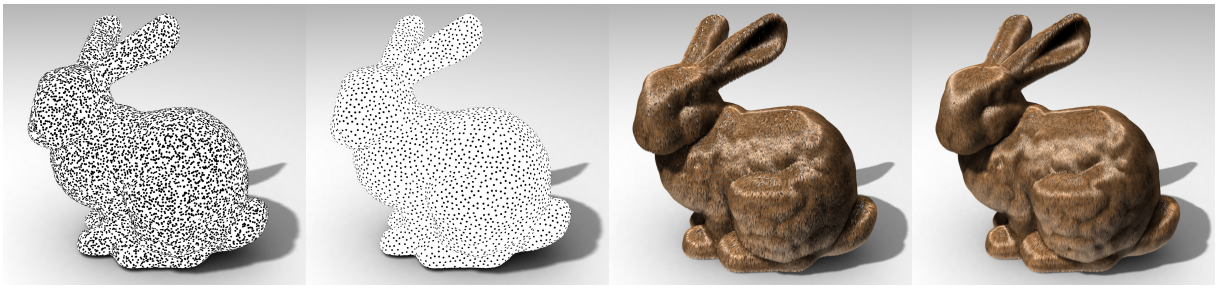
EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

Sample Elimination for Generating Poisson Disk Sample Sets

Cem Yuksel

University of Utah



(a) 15K random samples uniformly distributed

(b) 5K Poisson disk samples selected from 15K random samples

(c) Random distribution of 100K fur strands

(d) Poisson disk distribution of 100K strands selected from 300K

Figure 1: *Given an input set of samples, sample elimination selects a subset with Poisson disk property: (a) 15K input samples, (b) 5K selected samples from this input, (c) 100K strands randomly placed, and (d) 100K strands selected from 300K samples.*

Abstract

In this paper we describe sample elimination for generating Poisson disk sample sets with a desired size. We introduce a greedy sample elimination algorithm that assigns a weight to each sample in a given set and eliminates the ones with greater weights in order to pick a subset of a desired size with Poisson disk property without having to specify a Poisson disk radius. This new algorithm is simple, computationally efficient, and it can work in any sampling domain, producing sample sets with more pronounced blue noise characteristics than dart throwing. Most importantly, it allows unbiased progressive (adaptive) sampling and it scales better to high dimensions than previous methods. However, it cannot guarantee maximal coverage. We provide a statistical analysis of our algorithm in 2D and higher dimensions as well as results from our tests with different example applications.

1. Introduction

Sampling is commonplace in computer graphics and Poisson disk sample sets are often desired due to their statistical (blue noise) properties as well as the fact that no two samples are placed too close together. Most Poisson disk sample set generation methods are based on dart throwing [DW85, Coo86] that aims to generate as many samples as necessary to cover the sampling domain based on the given Poisson disk radius. In this paper we define *sample elimination* by approaching the Poisson disk sample set generation problem differently. Given a set of samples, we select a subset of *exactly* the desired size with a Poisson disk property without explicitly providing a Poisson disk radius. The sample elimination approach is especially useful for numerous sampling problems in which the quality/performance of sampling is directly associated with the sample count. Hence, controlling the sample count directly is more desirable than specifying a Poisson disk radius. Our approach also permits progressive sampling such that when samples in the final set are introduced one by one in a particular order, each subset in the sequence exhibits blue noise characteristics.

We describe an efficient greedy algorithm for sample elimination that relies upon common data structures, which makes it extremely easy to implement. Sample sets generated with this algorithm have improved blue noise characteristics over sample sets generated using dart throwing. Unlike previous work, our method scales well with high dimensions and it can produce unbiased progressive sample sets. Moreover, it allows varying density and it is suitable for any sampling domain including arbitrary manifolds (Figure 1).

2. Background

Poisson disk sample sets are typically characterized by the Poisson disk radius r, the half distance between the closest pair of samples, or the fraction ρ of the maximum possible distance r_{max} for N samples to cover the sampling domain, such that $r = \rho\, r_{max}$.

Perhaps the simplest way of generating Poisson disk sample sets is dart throwing [DW85, Coo86]. Given r, dart throwing generates random samples and rejects or accepts each sample based on its distance to the previously accepted samples. Its main shortcoming is the termination condition, since the algorithm does not know if the domain is fully covered. Hence, the algorithm has poor convergence, which makes it computationally expensive. Also, generating sample sets with a desired size is difficult, because fewer than desired samples can fully cover the domain when ρ is large ($\rho > 1/\sqrt{3} \approx 0.58$ in 2D). On the other hand, dart throwing is easy to implement and applicable to any sampling domain. As for its results, they are often used as the ground truth for evaluating other algorithms. Therefore, in spite of its shortcomings, dart throwing is still used in practice.

For faster sample set generation some researchers proposed approximate techniques that produce sample sets with similar spectral characteristics to Poisson disk. The earlier ones [Mit91, MF92] are relatively simple and they can be used for a wide range of sampling domains, but the gain in computational performance is limited. Methods that partition the space into grid cells can achieve linear time [Bri07] and parallel sample generation [Wei08].

Tile based methods are proposed for generating a large number of Poisson disk samples in 2D. Some of these methods generate a relatively small set of samples using another Poisson disk sampling method and tile those samples [HDK01, CSHD03, LD05, KCODL06, KS12, WPC*14], while others use a regular tile structure for placing each sample [ODJ04, Ost07]. As a result, they can generate a large number of samples efficiently at a cost of sampling quality. Some of them are shown to support varying sampling density [ODJ04, LD05, Ost07], but none of them can be used for sampling higher dimensions or arbitrary manifolds.

More recently, many researchers explored the idea of partitioning the sampling space to avoid generating new samples that will be ultimately rejected by dart throwing. Some of these methods only work in 2D [Jon06, DH06, WCE07, JK11, EDP*11, KS11] and others can be used for higher dimensions [GM09, EMP*12], but their performance drops exponentially with increasing dimensions. The methods that can handle 3D surfaces either repeatedly split the surface definition into smaller fragments [CJW*09, GZWW13] or completely replace the surface with a large number of random samples [BWWM10, CCS12, YXSH13]. Many of them are shown to work with varying sample densities [Jon06, CJW*09, BWWM10, KS11, YW13], but few of them can handle all these sampling domains simultaneously. The results of all these algorithms can mimic dart throwing and the most efficient ones can even achieve linear time performance [DH06, JK11, EDP*11, EMP*12], but they all have an overhead for storing and updating the space partitioning, and the algorithms are significantly more complicated than dart throwing.

Relaxation methods can iteratively increase the Poisson disk radius of a sample set [MF92, HDK01] by repositioning the samples, but they can also converge to regular patterns with tight packing unless randomness is explicitly enforced [BSD09, Fat11, SHD11, XLGG11]. Iterative techniques are also useful for reducing the sample count of an existing sample set while preserving the Poisson disk radius [EMA*13] or generating other sample distributions with different spectral properties in addition to Poisson disk [ZHWW12]. There are also other relaxation methods that can be used for sampling smooth surfaces in 3D [Tur91, WH94].

Progressive sampling in previous work is achieved only by reducing r progressively [MF92, MREB12]. The performance and quality of this approach depend on how r is updated and it is prone to introducing sampling bias.

In comparison, the weighted sample elimination method described in the next section:

- Allows generating sample sets with a desired count without having to specify a Poisson disk radius,
- Supports unbiased progressive sampling without any parameter tuning,
- Produces sample sets with more pronounced blue noise characteristics as compared to dart throwing,
- Scales well with high dimensions enabling Poisson disk sampling in extremely high dimensions,
- Supports any sampling domain including varying densities and arbitrary manifolds,
- Has $O(N \log N)$ computational complexity and $O(N)$ memory complexity, and
- Relies upon common data structures, which makes it extremely easy to implement.

3. Sample Elimination

We use the term sample elimination for algorithms that take a set of samples as input and select a subset as output. For example, the dart throwing method (with an upper bound on the number of samples to be tested) can be considered a sample elimination algorithm. Given the desired Poisson disk radius, dart throwing tests each sample in the order of increasing index, and decides whether to accept or *eliminate* the sample based on its distance to the previously accepted samples. Note that not all Poisson disk sampling algorithms can be formulated as sample elimination, since some of them move existing samples and others dictate where the next test sample can appear. When two samples are close together, dart throwing always eliminates the sample with the greater

```
function WeightedSampleElim( samples )
   Build a kd-tree for samples
   Assign weights w_i to each sample s_i
   Build a heap for s_i using weights w_i
   while number of samples > desired
      s_j ← pull the top sample from heap
      for each sample s_i around s_j
         Remove w_ij from w_i
         update the heap position of s_i
```

Figure 2: *The pseudo code of weighted sample elimination.*

index. Hence, given a set of random samples in a particular order, dart throwing picks an arbitrary subset with an arbitrary size. Since the index of a sample carries no information on the sample's contribution to the statistical properties of the selected subset, dart throwing is not expected to pick the subset with the largest Poisson disk radius or the one with the best blue noise characteristics.

For numerous sampling problems it is more desirable to specify the number of samples rather than the Poisson disk radius. Fortunately, the sample elimination problem for Poisson disk sample set generation can be defined as picking a desired number of samples from a given sample set. Given a set of M samples, finding the subset of N samples with the largest Poisson disk radius is an NP-Complete problem. However, an approximate solution can be obtained using a greedy algorithm in $M-N$ steps.

3.1. Weighted Sample Elimination

We introduce weighted sample elimination as a simple greedy algorithm for picking a subset with a reasonably large Poisson disk radius from a given set of input samples. We begin with assigning a weight to each sample based on its distance to its neighbors. At each step we eliminate the sample with the highest weight and adjust the weights of the remaining samples around it. Therefore, an efficient implementation of this algorithm merely needs two relatively common data structures: a spatial partitioning structure for quickly finding the neighboring samples and a priority queue for picking the sample with the highest weight. In our implementation we used a kd-tree and a heap respectively, resulting in an extremely simple algorithm shown in Figure 2.

We calculate the total weight w_i of sample i as a sum of all w_{ij}, the weight contribution of sample j on sample i, for all samples j ($i \neq j$) that are within $2\,r_{\max}$ distance of sample i. The weight function w_{ij} we use goes to 1 when the distance between the two samples d_{ij} goes to zero, and it drops to zero as d_{ij} increases up to $2\,r_{\max}$, such that

$$w_{ij} = \left(1 - \frac{\hat{d}_{ij}}{2\,r_{\max}}\right)^{\alpha} \qquad (1)$$

with $\hat{d}_{ij} = \min\left(d_{ij}, 2\,r_{\max}\right)$. In our implementation we use

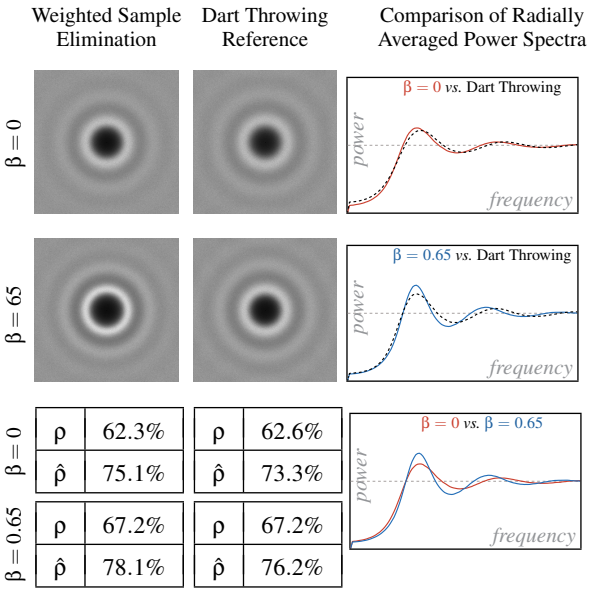


Figure 3: *Comparison of dart throwing to weighted sample elimination with and without weight limiting (i.e. $\beta = 0$ and $\beta = 1$) using 1,000 samples generated from 4,000 random input samples. For a fair comparison, the resulting r value of weighted sample elimination is given as input to dart throwing and dart throwing is given additional random samples until it produces 1,000 samples. The instances where dart throwing fails to generate 1,000 samples are ignored.*

$\alpha = 8$, so that smaller d_{ij} values have sufficiently greater weight contributions as compared to a collection of greater d_{ij} values, though smaller or larger α values produced similar results in our tests. Note that the distance measure for d_{ij} can be the Euclidian distance as well as the geodesic distance on a surface or any other function. Moreover, the distance metric does not have to be symmetric, such that d_{ij} can be different than d_{ji}.

The value of r_{max} depends on the sampling domain. In 2D and 3D,

$$r_{\max,2} = \sqrt{\frac{A_2}{2\sqrt{3}N}} \quad \text{and} \quad r_{\max,3} = \sqrt[3]{\frac{A_3}{4\sqrt{2}N}}\,, \qquad (2)$$

where A_2 and A_3 are the area and volume of the sampling domain. In higher dimensions we use a conservative estimate for $r_{\max,d}$ with $d > 3$, assuming that the hypervolume of the domain A_d can be *completely* filled with hyperspheres with no overlap. Note that this assumption causes overestimation of the $r_{\max}$ values. The hypervolume V_d of a hypersphere with radius r is $V_d = C_d\,r^d$, where C_d is a constant such that $C_d = C_{d-2}\frac{2\pi}{d}$ with $C_1 = 2$ and $C_2 = \pi$, resulting

$$r_{\max,d} \simeq \sqrt[d]{\frac{A_d}{C_d N}}\,. \qquad (3)$$

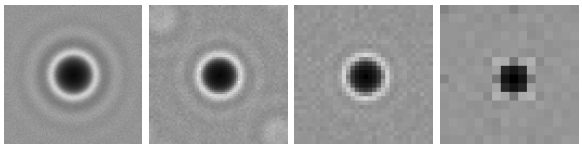


Figure 4: *Cross-sections of the high dimensional power spectra for weighted sample elimination showing blue noise characteristics. We reduce the power spectrum resolution (and the sample count accordingly) in higher dimensions, simply because the power spectrum computation time increases exponentially with increasing dimension.*

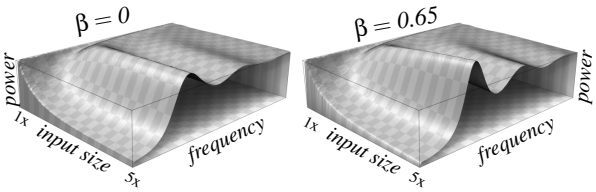


Figure 5: *Radially averaged power spectrum of 1,000 samples generated with weighted sample elimination from varying input set size between 1,000 and 5,000 samples using weight functions with and without weight limiting: (left)* $\beta = 0$ *and (right)* $\beta = 0.65$.

3.2. Weight Limiting

An approximate greedy algorithm cannot guarantee to pick the subset with the largest Poisson disk radius, and the order of elimination is important for selecting a subset with a reasonably large radius. We found that weighted sample elimination generates sets with a larger Poisson disk radius when samples with many relatively close neighbors are removed earlier than samples with fewer but very close neighbors. We facilitate this using *weight limiting* to assign the same w_{ij} value to all pairs of samples with $d_{ij} < 2\,r_{\min}$ using

$$\hat{d}_{ij} = \begin{cases} \min\left(d_{ij}, 2\,r_{\max}\right), & \text{if } d_{ij} > 2\,r_{\min} \\ 2\,r_{\min}, & \text{if } d_{ij} \leq 2\,r_{\min} \end{cases}, \qquad (4)$$

where $r_{\min}$ is a fraction of $r_{\max}$. The resulting Poisson disk radius r of the final sample set improves with increasing $r_{\min}$ until $r_{\min}$ gets close to r, at which point the output includes pairs of adjacent samples, resulting in an extremely small Poisson disk radius. Unfortunately, the largest possible r as well as the optimal value of $r_{\min}$ depend on the sample positions in the input and there is no closed form solution for accurately calculating them. However, it is possible to iteratively search for the optimal value of $r_{\min}$ for a given input sample set by running the weighted sample elimination algorithm multiple times on the same input samples with different $r_{\min}$ values. Such an algorithm can locate the optimal r_{min} value with $1/2^s$ precision in s steps. In our implementation, instead of searching for the optimal r_{min} value, we opted to use a conservative estimate for $r_{\min}$ that we formulated based on our experiments with different N, M, and $r_{\min}$ values in different dimensions, such that

$$r_{\min} = r_{\max}\left(1 - \left(\frac{N}{M}\right)^{\gamma}\right)\beta\,, \qquad (5)$$

where $\beta = 0.65$ and $\gamma = 1.5$. In this formulation $r_{\min}$ is set to zero when $N = M$, and $r_{\min}$ approaches $\beta\,r_{\max}$ as M/N goes to infinity. We found that this formulation produces sets with greater r values for any combination of N and M values.

4. Statistical Analysis

We use three metrics for statistical analysis: power spectrum (as described by Schlömer and Deussen [SD11]), Poisson disk radius ratio $\rho = r/r_{\max}$, and the mean radius ratio $\hat{\rho} = \hat{r}/r_{\max}$, where $\hat{r}$ is the average half distance of each sample to its closest neighbor. All values are the averages of 1000 tests and the power spectrum intensities are scaled by 2 for better visualization

Figure 3 is a comparison between sample sets generated using dart throwing and weighted sample elimination with and without weight limiting (i.e. $\beta = 0$ and $\beta = 0.65$). The power spectra show more pronounced blue noise characteristics for weighted sample elimination as compared to dart throwing with the same ρ threshold and the same sample count. Even though dart throwing uses the same ρ values, it produces about 2% smaller $\hat{\rho}$ values as compared to weighted sample elimination. This is an expected result, since dart throwing has no mechanism to favor results with larger $\hat{\rho}$. Figure 3 also shows that the blue noise characteristics are amplified with weight limiting in 2D.

Cross-sections of the high dimensional power spectra are presented in Figure 4, showing clearly visible blue noise characteristics in all dimensions.

The progression of the radially averaged power spectrum with increasing input size (M/N) is shown in Figure 5. Notice that the power spectrum starts flat when $N = M$, showing white noise, since no sample is eliminated in that case. As M/N increases, the blue noise characteristics quickly emerge and get more pronounced with increasing input size.

We show the typical ρ and $\hat{\rho}$ values for different input set size ratios (M/N) in Figure 6. Notice that our weight limiting formulation with $\beta = 0.65$ makes a significant difference when $M/N > 2$. Increasing the input size ratio M/N beyond 4 makes only a minor improvement on ρ and $\hat{\rho}$ without weight limiting ($\beta = 0$), while weight limiting logarithmically improves ρ and $\hat{\rho}$ with increasing input size ratio in 2D.

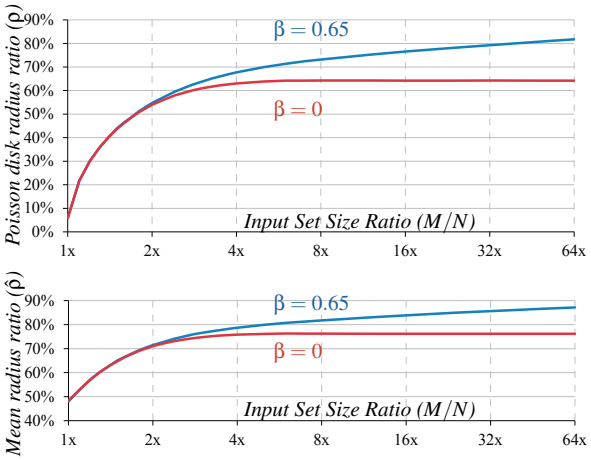


Figure 6: *Average Poisson disk radius ratios (ρ) of 128 samples generated using weighted sample elimination in 2D for different input set sizes (M/N).*

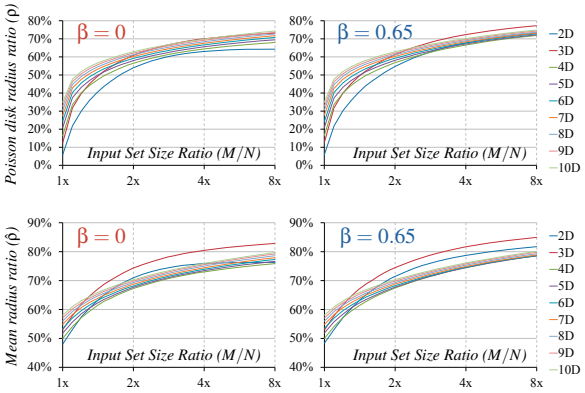
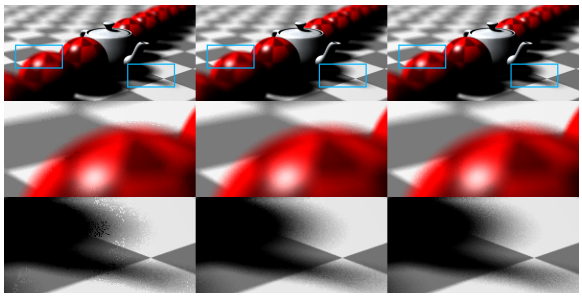


Figure 7: *Mean radius ratios and Poisson disk radius ratios of 128 samples generated using weighted sample elimination in 2 to 10 dimensions for different input set sizes.*

The progression of the ρ and ρ̂ with increasing input size (M/N) in 2 to 10 dimensions is shown in Figure 7. Note that ρ and ρ̂ values in high dimensions beyond 3D are computed using the overestimated $r_{\max}$ values, which make them appear lower. In 3D and higher dimensions ρ and ρ̂ continue to grow with increasing input size even without weight limiting, but our weight limiting formulation in Equation 5 still provides a moderate improvement.

5. Results

We experimented with weighted sample elimination using it in various sampling problems. Figure 1 shows an example of distributing samples over a surface in 3D, which is used for placing fur roots as an example application.



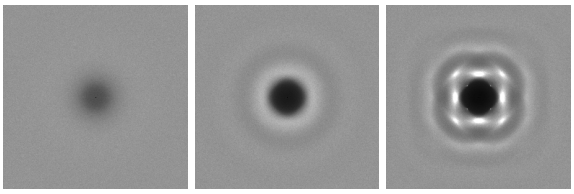
16 to 64 unordered samples | 16 to 64 ordered samples | 64 samples

Figure 8: *Progressive sampling for rendering using the same Poisson disk sample set showing the importance of ordering.*

5.1. Progressive Sampling

One of the most important properties of weighted sample elimination is its ability to facilitate progressive sampling. We achieve this by ordering the samples in the final set, such that when the samples are introduced in this order, each subset in the sequence exhibits blue noise characteristics. For efficiently ordering the samples, we continue the elimination process with target set sizes $N/2^k$ where $k = 1,2,3...$ until $N/2^k < 1$, and we set the order of samples in the inverse order of elimination by placing each eliminated sample to the end of the current set.

An example rendering application using ray tracing with progressive (adaptive) sampling is presented in Figure 8. A set of 64 samples is generated in 6D as a union of three 2D sets for antialiasing, depth of field, and soft shadows. Ordering the samples for progressive sampling allows us to adaptively adjust the number of rays generated per pixel, which in turn makes it possible to render this scene approximately twice as fast with a visually similar result as compared to using all of the samples at each pixel. Notice that when the samples are not ordered, the error estimation fails to accurately predict when more samples are needed.



$\delta = 1 \times 10^{-3}$, $\rho = 26\%$ | $\delta = 1 \times 10^{-4}$, $\rho = 55\%$ | $\delta = 1 \times 10^{-5}$, $\rho = 66\%$

Figure 9: *Power spectra of sample sets generated with progressively reducing the Poisson disk radius using $r_{i+1} = r_i(1-\delta)$ at each step i with three different δ parameters. The power spectra and ρ values are computed for the full sample sets, rather than the intermediate subsets in the sequences.*

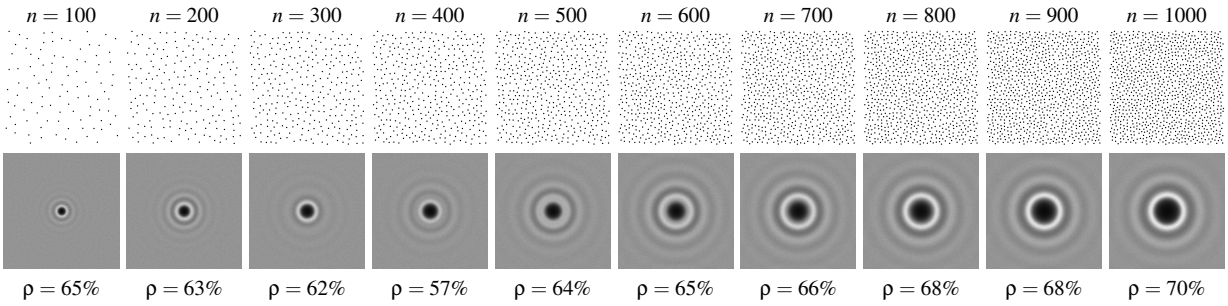


Figure 10: *A sequence of 2D sample sets generated using weighted sample elimination from a set of 3000 random samples.*

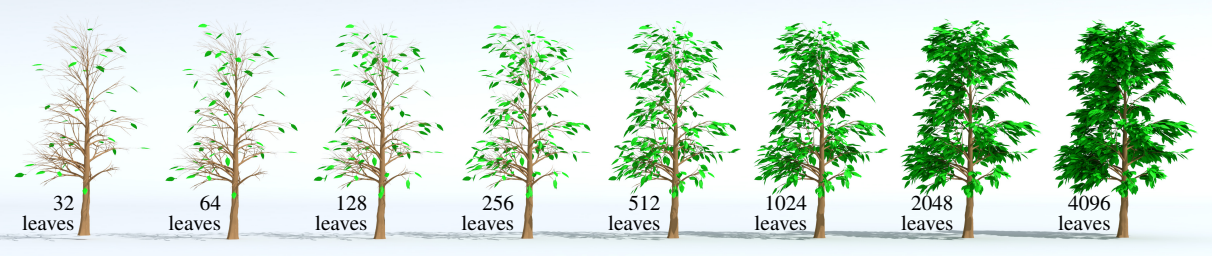


Figure 11: *A sequence of tree leaves placed on the surfaces of the branches using weighted sample elimination.*

Generating sample sets by progressively reducing r employed in previous work [MF92, MREB12] is prone to sampling bias as shown in Figure 9. Notice that depending on how fast r is reduced, the resulting sample set either gets poor blue noise characteristics or causes sampling bias, caused by the shape of the sampling medium.

In comparison, Figure 10 shows frames from an example sequence. Notice that the blue noise properties are preserved throughout the sequence with reasonable Poisson radius ratios ρ.

Weighted sample elimination enables progressive sampling in all sampling domains and it is not limited to 2D sampling problems. An example application is shown in Figure 11 that uses weighted sample elimination for progressively distributing tree leaves on the surfaces of the branches.

5.2. Other Example Applications

An example importance sampling application is presented in Figure 12 using an image as the density maps with $r_{\max} = 3\,r_{\max,2}$ and $\hat{d}_{ij} = d_{ij}(3-2I_i)$, where I_i is the density map intensity at the position of sample i. Note that in this formulation the boundary conditions between low-density and high-density regions are handled automatically, since $\hat{d}_{ij}$ is not necessarily equal to $\hat{d}_{ji}$. While this example clearly shows that weighted sample elimination can be used for importance sampling, it also demonstrates one of the shortcomings of our approach. Since weighted sample elimination does not introduce new samples, it cannot plug the holes that are apparent in the input. Therefore, providing an input that is more suitable for the desired output improves the quality of the selected sample set.

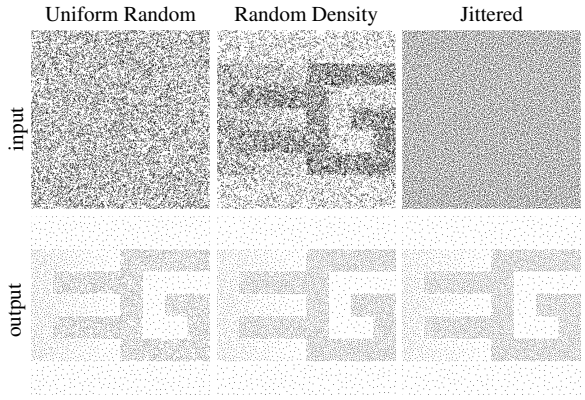


Figure 12: *Importance sampling example of 4,000 samples selected using weighted sample elimination with* $\beta = 0$ *from 20,000 input samples generated using: (left) uniform random distribution, (middle) random rejection sampling using the target density map, and (right) jittered samples.*

Weighted sample elimination can also be used for generating different noise types by merely changing weight function. Figure 13 shows magenta noise generated using $w_{ij} = 2\,r_{\max}/d_{ij} - 3$ for $d_{ij} \le 2\,r_{\max}$. This weight function produces negative values when $2\,r_{\max}/3 < d_{ij} < 2\,r_{\max}$, thereby favoring clusters of samples that are still separated by some distance.

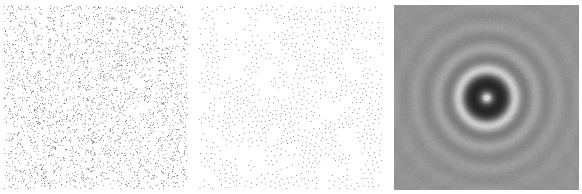


Figure 13: *(left) 3,000 input random samples are used for generating (middle) Magenta noise of 1,000 samples with weighted sample elimination using a different weight function and (right) its average power spectrum.*

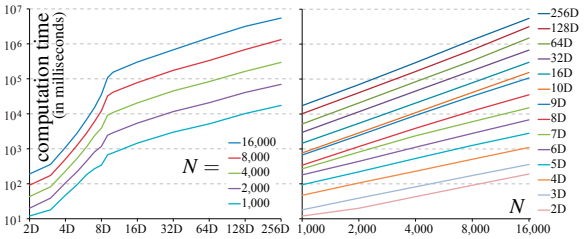


Figure 14: *Computation times of weighted sample elimination in milliseconds in different dimensions and N with $M = 3N$.*

5.3. Computation Time and Empirical Complexity

The average computation times of our single threaded application for different dimensions are provided in Figure 14 and Table 1. We present timing results from a single-threaded CPU implementation. For higher efficiency with large sample sets in low dimensions parallel processing can be easily employed by splitting the sampling domain into tiles [Wei08].

Notice that the timing results of weighted sample elimination reflect the $O(N \log N)$ complexity in 2D and 3D. In higher dimensions the empirical complexity increases rapidly with increasing dimensions up to 10D, beyond which there is only a minor increase in the empirical complexity. This is due to the fact that the kd-tree representation does not perform as efficiently in high dimensions and the benefits of the kd-tree are almost completely lost beyond 10D. Note that when no acceleration structure (such as a kd-tree) is used, the theoretical complexity of weighted sample elimination becomes $O(N^2)$. Therefore, in extremely high dimensions the empirical complexity of weighted sample elimination approaches $O(N^2)$, as expected.

For constant N, the computation times of weighted sample elimination increase linearly beyond 10D. This linear increase appears simply because each additional dimension increases the data size linearly. Hence, weighted sample elimination is not affected at all by the "curse of dimensionality" aside from the inefficiencies of the acceleration structure. Therefore, we can provide results from much higher dimensions than reported by the previous methods [GM09, EMP*12], since their computation times increase exponentially with increasing dimensions.

6. Discussion and Conclusion

We introduced a new approach for Poisson disk sample set generation using sample elimination. We also presented a greedy sample elimination algorithm that is easy to implement, computationally efficient, and general enough to handle any sampling domain. This algorithm allows generating sample sets in higher dimensions than previously reported, and supports progressive sampling.

The main drawback of weighted sample elimination, however, is that it cannot guarantee maximal coverage. Therefore, it is not suitable for applications where maximal coverage is required. Nonetheless, when maximal coverage is not necessary, weighted sample elimination is a good choice for any sampling application due to its efficiency, implementation simplicity, and ability to handle any sampling domain.

Weighted sample elimination excels in three areas in particular as compared to previous methods:

1. It produces exactly as many samples as desired, which is only possible with relaxation methods in previous work.
2. It is ideal for sampling in high dimensions, as it does not suffer from the "curse of dimensionality" like prior methods and its memory consumption does not depend on the number of dimensions (aside from the storage needed for the input and output sample sets).
3. It can produce unbiased progressive sample sets starting with a single sample in all sampling domains.

Prior methods that support surfaces in 3D by generating a large number of initial random samples [BWWM10, CCS12, YXSH13] can be considered sample elimination. However, these methods do not permit directly controlling the output size and they aim to mimic dart throwing. Therefore, they require a significantly larger input set for producing the same ρ as (with lower $\hat{\rho}$ than) weighted sample elimination.

While we show that the statistical properties of weighted sample elimination are superior to dart throwing, relaxation methods can produce more pronounced blue noise characteristics. Hence, a relaxation step could be employed to further improve the sampling quality when the sampling domain permits relaxation.

Table 1: *Samples per second with $M = 3N$.*

N	**2D**	**3D**	**4D**	**5D**	**6D**	**7D**	**8D**
1K	100K	55K	23K	11K	5.9K	3.8K	3.3K
10K	91K	46K	16K	6.4K	2.7K	1.3K	590
100K	75K	38K	23K	4.4K	1.5K	560	230
1M	43K	21K	6.5K	2.1K	640	210	80

Timings using a single thread on an Intel Xeon X5690 @ 3.46 GHz.

References

[Bri07] BRIDSON R.: Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH'07 sketches* (2007). 2

[BSD09] BALZER M., SCHLÖMER T., DEUSSEN O.: Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Trans. on Graph. (SIGGRAPH'09) 28*, 3 (2009), 86:1–8. 2

[BWWM10] BOWERS J., WANG R., WEI L.-Y., MALETZ D.: Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph. (SIGGRAPH ASIA'10) 29*, 6 (2010), 166. 2, 7

[CCS12] CORSINI M., CIGNONI P., SCOPIGNO R.: Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. on Vis. & Comp. Graph. 18*, 6 (2012), 914–924. 2, 7

[CJW*09] CLINE D., JESCHKE S., WHITE K., RAZDAN A., WONKA P.: Dart throwing on surfaces. *Computer Graphics Forum (EGSR'09)* (2009), 1217–1226. 2

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1 (1986), 51–72. 1, 2

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. *ACM Trans. Graph. (SIGGRAPH'03) 22*, 3 (2003), 287–294. 2

[DH06] DUNBAR D., HUMPHREYS G.: A spatial data structure for fast poisson-disk sample generation. *ACM Transactions on Graphics (SIGGRAPH'06) 25*, 3 (2006), 503–508. 2

[DW85] DIPPÉ M. A. Z., WOLD E. H.: Antialiasing through stochastic sampling. *SIGGRAPH'85 19*, 3 (1985), 69–78. 1, 2

[EDP*11] EBEIDA M. S., DAVIDSON A. A., PATNEY A., KNUPP P. M., MITCHELL S. A., OWENS J. D.: Efficient maximal poisson-disk sampling. *ACM ToG (SIGGRAPH'11) 30*, 4 (2011), 49. 2

[EMA*13] EBEIDA M. S., MAHMOUD A. H., AWAD M. A., MOHAMMED M. A., MITCHELL S. A., RAND A., OWENS J. D.: Sifted disks. *Comp. Graphics Forum 32*, 2 (2013), 509–518. 2

[EMP*12] EBEIDA M. S., MITCHELL S. A., PATNEY A., DAVIDSON A. A., OWENS J. D.: A simple algorithm for maximal poisson-disk sampling in high dimensions. *Computer Graphics Forum 31*, 2 (2012), 785–794. 2, 7

[Fat11] FATTAL R.: Blue-noise point sampling using kernel density model. *ACM Trans. Graph. (SIGGRAPH'11) 28*, 3 (2011), 1–10. 2

[GM09] GAMITO M. N., MADDOCK S. C.: Accurate multidimensional poisson-disk sampling. *ACM ToG 29*, 1 (2009), 8:1–8:19. 2, 7

[GZWW13] GENG B., ZHANG H., WANG H., WANG G.: Approximate poisson disk sampling on mesh. *Science China Information Sciences 56*, 9 (2013), 1–12. 2

[HDK01] HILLER S., DEUSSEN O., KELLER A.: Tiled blue noise samples. In *Proceedings of the Vision Modeling and Visualization Conference 2001* (2001), pp. 265–272. 2

[JK11] JONES T. R., KARGER D. R.: Linear-time poisson-disk patterns. *Journal of Graph., GPU, and Game Tools 15* (2011). 2

[Jon06] JONES T. R.: Efficient generation of poisson-disk sampling patterns. *Journal of Graph., GPU, and Game Tools 11*, 2 (2006), 27–36. 2

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive wang tiles for real-time blue noise. *ACM Transactions on Graphics (SIGGRAPH'06) 25*, 3 (2006), 509–518. 2

[KS11] KALANTARI N. K., SEN P.: Efficient Computation of Blue Noise Point Sets through Importance Sampling. *Computer Graphics Forum (EGSR'11) 30*, 4 (2011), 1215 ŰŰ 1221. 2

[KS12] KALANTARI N. K., SEN P.: Fast generation of approximate blue noise point sets. *CG. Forum 31*, 4 (2012), 1529–1535. 2

[LD05] LAGAE A., DUTRÉ P.: A procedural object distribution function. *ACM Transactions on Graphics 24*, 4 (2005), 1442–1461. 2

[MF92] MCCOOL M., FIUME E.: Hierarchical poisson disk sampling distributions. In *Proc. Graphics interface '92* (1992), pp. 94–105. 2, 6

[Mit91] MITCHELL D. P.: Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH'91 25*, 4 (1991), 157–164. 2

[MREB12] MITCHELL S. A., RAND A., EBEIDA M. S., BAJAJ C.: Variable radii Poisson-disk sampling. In *Proc. of the 24th Canadian Conference on Computational Geometry* (2012), pp. 185–190. 2, 6

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast hierarchical importance sampling with blue noise properties. *ACM Trans. on Graphics (SIGGRAPH'04) 23*, 3 (2004), 488–495. 2

[Ost07] OSTROMOUKHOV V.: Sampling with polyominoes. *ACM Trans. Graph. (SIGGRAPH'07) 26*, 3 (2007). 2

[SD11] SCHLÖMER T., DEUSSEN O.: Accurate spectral analysis of two-dimensional point sets. *Journal of Graphics, GPU, and Game Tools 15*, 3 (2011), 152–160. 4

[SHD11] SCHLÖMER T., HECK D., DEUSSEN O.: Farthest-point optimized point sets with maximized minimum distance. In *Proceedings ACM HPG'11* (2011), pp. 135–142. 2

[Tur91] TURK G.: Generating textures on arbitrary surfaces using reaction-diffusion. In *Proc. SIGGRAPH'91* (1991), pp. 289–298. 2

[WCE07] WHITE K. B., CLINE D., EGBERT P. K.: Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the 2007 IEEE Symp. on Interactive Ray Tracing* (2007), pp. 129–132. 2

[Wei08] WEI L.-Y.: Parallel poisson disk sampling. *ACM Transactions on Graphics (SIGGRAPH'08) 27*, 3 (2008), 20:1–20:9. 2, 7

[WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH'94* (1994), pp. 269–277. 2

[WPC*14] WACHTEL F., PILLEBOUE A., COEURJOLLY D., BREEDEN K., SINGH G., CATHELIN G., DE GOES F., DESBRUN M., OSTROMOUKHOV V.: Fast tile-based adaptive sampling with user-specified fourier spectra. *ACM Trans. Graph. 33*, 4 (July 2014), 56:1–56:11. 2

[XLGG11] XU Y., LIU L., GOTSMAN C., GORTLER S. J.: Capacity-constrained delaunay triangulation for point distributions. *Computers & Graphics (Proceedings of Shape Modeling International) 35*, 3 (2011), 510–516. 2

[YW13] YAN D.-M., WONKA P.: Gap processing for adaptive maximal poisson-disk sampling. *ACM Trans. Graph. 32*, 5 (Oct. 2013), 148:1–148:15. 2

[YXSH13] YING X., XIN S.-Q., SUN Q., HE Y.: An intrinsic algorithm for parallel poisson disk sampling on arbitrary surfaces. *IEEE Trans. on Vis. & Comp. Graph. 19*, 9 (2013), 1425–1437. 2, 7

[ZHWW12] ZHOU Y., HUANG H., WEI L.-Y., WANG R.: Point sampling with general noise spectrum. *ACM Transactions on Graphics 31*, 4 (2012), 76:1–76:11. 2