First publ. in: Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics : HPG '11 / Stephen N. Spencer (ed.). - New York : ACM, 2011. - pp. 135-142. - ISBN 978-1-4503-0896-0

http://dx.doi.org/10.1145/2018323.2018345

Farthest-Point Optimized Point Sets with Maximized Minimum Distance

Thomas Schlömer* University of Konstanz

Daniel Heck† University of Konstanz

Oliver Deussen‡ University of Konstanz

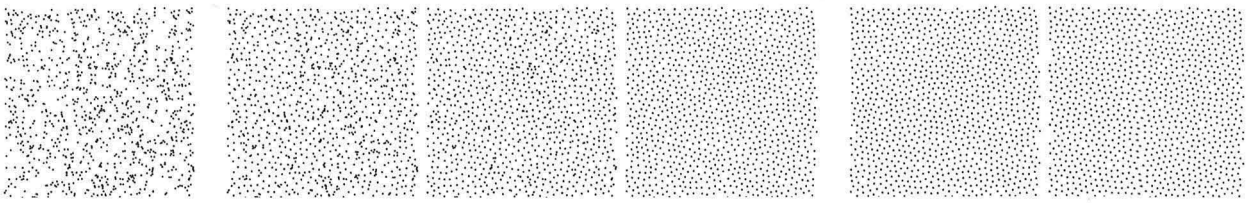


Figure 1: *Farthest-point optimization of a random point set with 1024 points. Both the global minimum distance δ_X and the average minimum distance $\bar{\delta}_X$ increase rapidly using our optimization technique. After one iteration the point set is already well-distributed.*

Abstract

Efficient sampling often relies on irregular point sets that uniformly cover the sample space. We present a flexible and simple optimization strategy for such point sets. It is based on the idea of increasing the mutual distances by successively moving each point to the "farthest point," i.e., the location that has the maximum distance from the rest of the point set. We present two iterative algorithms based on this strategy. The first is our main algorithm which distributes points in the plane. Our experimental results show that the resulting distributions have almost optimal blue noise properties and are highly suitable for image plane sampling. The second is a variant of the main algorithm that partitions any point set into equally sized subsets, each with large mutual distances; the resulting partitionings yield improved results in more general integration problems such as those occurring in physically based rendering.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Antialiasing; I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Sampling

Keywords: sampling, anti-aliasing, blue noise, Poisson-disk, maximized minimum distance, Delaunay triangulations, numerical integration, trajectory splitting

1 Introduction

Point distributions that are uniform but irregular have found many applications in computer graphics. Here, uniform means that the average point density is approximately constant and there are no "holes" or "clusters"; irregular means that there are no symmetries or regular structures that could lead to moiré patterns or other visually distracting artifacts. The *blue noise* criterion [Ulichney 1988] characterizes such point distributions in the Fourier domain.

In this paper, we describe a new optimization procedure that iteratively enlarges the minimum distance between points and thereby improves the blue noise characteristics of the point set. The main algorithm moves each point in such a way that the point spacing increases monotonically until convergence. Since it can be interpreted as an iterative version of the farthest point strategy introduced by Eldar et al. [1997], we call the point sets *farthest-point optimized* and our method *farthest-point optimization* (FPO). The resulting point sets have excellent blue noise properties and a significantly higher minimum distance than previous methods. Unlike other iterative methods that have been proposed for point distribution, our procedure does not converge towards (locally) regular patterns. The close connection between farthest points and Delaunay triangulations permits a very efficient implementation that requires only $\mathcal{O}(n \log n)$ per full iteration. We discuss our method and this implementation in Section 3.

In Section 4 we extend our main idea to the problem of partitioning a given point set such that each subset is well-distributed and of high minimum distance. This yields improvements in sampling applications other than image plane sampling such as numerical integration problems in physically based rendering.

We demonstrate the benefits of both methods to sampling and integration problems in Section 5 and conclude with a few ideas for further research in Section 6.

2 Background

In this section we briefly review the basics of Poisson-disk distributions, a few numerical measures for characterizing point sets, and survey current algorithms for generating such distributions.

2.1 Poisson-Disk Patterns

Point sets with a uniform but irregular distribution have a characteristic energy distribution in the Fourier domain: if the points are widely spaced, the spectral energy is low in a circular disc around the origin, and if they are irregularly distributed, the energy varies

*e-mail: thomas.schloemer@uni-konstanz.de
†e-mail: daniel.heck@uni-konstanz.de
‡e-mail: oliver.deussen@uni-konstanz.de

smoothly outside this empty inner ring. Point sets with such a spectrum are known as *blue noise* patterns in computer graphics. Many methods for constructing blue noise patterns have been proposed over the last 20 years. Most of them are not based on the blue noise definition itself but on geometric constraints in the spatial domain. The most popular constraint is the *Poisson-disk* criterion, which demands that no two points are closer than a certain minimal distance.

We will be mostly interested in points distributed in the 2D unit torus (i.e., the unit square with periodic boundary conditions), in which the distance between two points x, y is measured using the toroidal metric $d_T(x, y)$. For a set of points X containing $n := |X|$ points we define the following three measures:

$$d_x := \min_{y \in X \setminus \{x\}} d_T(x, y) \qquad \text{local mindist,}$$

$$d_X := \min_{x,y \in X, x \neq y} d_T(x, y) \qquad \text{global mindist,}$$

$$\bar{d}_X := \frac{1}{n} \sum_{x \in X} d_x \qquad \text{average mindist.}$$

The *local mindist* d_x is the distance from a point x to its nearest neighbor in X; the *global mindist* d_X is the smallest separation between any two points in X; and the *average mindist* measures the overall spacing of the point set. Obviously, every irregular point set with $d_X > 0$ is a Poisson-disk set with radius $d_X/2$.

To ensure that the points in X are uniformly distributed, we would like both the global and the average mindist to be as large as possible: a high d_X means that the points do not cluster anywhere, and a high $\bar{d}_X$ that the points are evenly spaced. The largest mindist is obtained if the points form a hexagonal lattice [Tóth 1951]:

$$d_{\max} = \left(2/\sqrt{3}n\right)^{1/2}.$$

We generally report the mindist relative to this maximum value

$$\delta_x := d_x/d_{\max}, \quad \delta_X := d_X/d_{\max}, \quad \bar{\delta}_X := \bar{d}_X/d_{\max}.$$

2.2 Constructing Poisson-Disk Patterns

In the following we will briefly review the most important algorithms for generating Poisson-disk patterns; for a more in-depth survey refer to the article by Lagae and Dutré [2008]. The relative mindist δ_X is a good measure to compare different Poisson-disk patterns. Lagae and Dutré recommend $\delta_X \geq 0.65$ for well-distributed point sets but conjecture that $\delta_X \geq 0.85$ leads to regular configurations; we will see later that this conjecture is not correct.

We distinguish two main categories of algorithms: non-iterative algorithms that generate point sets in one pass and iterative algorithms that improve the arrangement of points in multiple passes. Table 1 compares several algorithms with respect to their mindist.

The classical non-iterative method for generating Poisson-disk patterns is the *dart throwing* algorithm proposed by Cook [1986]. This method takes the desired Poisson-disk radius R as its input and randomly generates candidate points. A candidate point is rejected if it lies closer than $2R$ to any of the existing points and accepted otherwise. Achieving a high mindist is very difficult with dart throwing since, in later stages of the algorithm, an excessive number of candidates must be generated. Several improvements have been suggested [Dunbar and Humphreys 2006; Wei 2008; Gamito and Maddock 2009] which execute faster and yield similar distributions. The maximum mindist that is achievable with dart throwing and related methods is $\delta_X \approx 0.75$.

A non-iterative algorithm that doesn't use stochastic sampling was introduced by Eldar et al. [1997]. Given a few randomly distributed seed points, the algorithm deterministically adds points according to the "farthest point strategy," which chooses the location with maximum distance from all current points. Voronoi diagrams can be used to obtain a $\mathcal{O}(n \log n)$ implementation of this algorithm. In general, the results are comparable to dart throwing, and the mindist is also $\delta_X \approx 0.75$. The best candidate algorithm by Mitchell [1991] can be considered an approximative version of this algorithm and Kanamori et al. [2011] presented an equivalent formulation based on Delaunay triangulations.

Generation Method	δ_X	$\bar{\delta}_X$	*Note*
Jittered Grid[a]	0.049	0.586	
Electrostatic Halftoning[b] ($\varrho = 0.002$)	0.741	0.882	
Best Candidate[c] and FPS[d]	0.751	0.839	
Dart throwing[e] and variants[f]	0.765	0.808	
CCCVT Centroids[g]	0.778	0.896	I
CVT Centroids[h] and methods using Lloyd's algorithm	0.795	0.939	I, R
Electrostatic Halftoning[b]	0.826	0.952	I, R
Boundary Sampling[i]	0.829	0.862	
Low discrepancy[j]	0.903	0.920	D, R
Farthest-Point Optimization	*0.930*	*0.932*	I

[a] [Cook 1986] [b] [Schmaltz et al. 2010] [c] [Mitchell 1991]
[d] [Eldar et al. 1997] [e] [Cook 1986] [f] [Lagae and Dutré 2008]
[g] [Balzer et al. 2009] [h] [Du et al. 1999] [i] [Dunbar and Humphreys 2006]
[j] [Grünschloß and Keller 2009]

Table 1: *Relative minimum distances for common methods that generate irregular, well-distributed point sets. The last column marks (D)eterministic and (I)terative methods, and methods that converge towards (R)egular arrangements. For all non-deterministic methods, the results were obtained by averaging δ_X and $\bar{\delta}_X$ over 10 output point sets with 4096 points each.*

Most non-iterative methods have difficulty achieving a mindist larger than ≈ 0.75. The reason is that they cannot move points after they have been placed, which can cause later points to be placed in suboptimal positions. The main exception is the boundary algorithm by Dunbar and Humphreys [2006] with $\delta_X \approx 0.83$.

The standard iteration scheme for improving the distribution of points is Lloyd's method [1982]. Unfortunately, the results are often a little "too good" since Lloyd's method converges towards hexagonal arrangements; the point sets show strong spikes in the Fourier domain and therefore fail to be blue noise patterns. Stopping the iteration before the arrangement becomes too regular is sometimes possible but in general too unreliable. Lloyd's method achieves a mindist of $\delta_X \approx 0.8$.

Two alternative iterative methods have been proposed in recent years. Both optimization techniques are rather costly and show $\mathcal{O}(n^2)$ time complexity per iteration.

The algorithm by Balzer et al. [2009] is conceptually similar to Lloyd's method but places a constraint on the area of the resulting Voronoi regions. This leads to very uniform point distributions, but since the constraint is on areas not distances, the mindist is not significantly higher than for non-iterative methods ($\delta_X \approx 0.78$).

Finally, Schmaltz et al. [2010] model the points as charged particles with repelling forces. Simulating the movement of these particles results in uniform point distributions with a Poisson-disk radius comparable to Lloyd's method ($\delta_X \approx 0.83$), but is also prone to producing locally hexagonal structures. A variant proposed by the authors breaks up these regularities by incorporating random movements, but it also lowers the Poisson-disk radius.

3 Farthest-Point Optimization

In this section we present a new iterative algorithm for generating Poisson-disk patterns that have a high minimum distance ($\delta_X \approx 0.93$) but do not suffer from regular structures. We show that the general algorithm runs in $\mathcal{O}(n \log n)$ per full iteration and always converges. We also discuss a variant of the main algorithm that runs in $\mathcal{O}(n)$ per full iteration and gives results of the same quality.

3.1 Main Algorithm

The basic algorithm is very simple: each step takes a single point from a set of points X and attempts to move it to a new position that is as far away from the remaining points as possible, i.e., the *farthest point*. One full iteration consists of moving each point in X once. As we will see, this iteration scheme converges, and each full iteration increases the average mindist $\bar{\delta}_X$.

In general, the farthest point f_Y of a set of points Y is the center of the largest circle that can be placed in the domain under consideration without covering any of the points in Y. This largest empty circle can be computed efficiently using the Delaunay triangulation $\mathcal{D}(Y)$: it corresponds to the largest circumcircle of the triangles in $\mathcal{D}(Y)$. An equivalent formulation in terms of the Voronoi diagram of Y was used by Eldar et al.

In our case, to move a point x, we need to inspect the Delaunay triangulation (DT) of the remaining points $X\backslash\{x\}$. Instead of calculating the full DT for each point x, we build a full DT once and update it dynamically during the iteration: before we move x, we remove it from the DT, inspect the remaining triangles to find the farthest point f, and finally reinsert f as a new point into the DT. The full algorithm can be formulated as follows.

FARTHEST-POINT-OPTIMIZATION(X)

```
1  D = DELAUNAY(X)
2  repeat
3      foreach vertex x in D
4          (f, r_max) = (x, d_x)
5          DELAUNAY-REMOVE(D, x)
6          foreach t in D
7              (c, r) = center and radius of t's circumcircle
8              if r > r_max
9                  (f, r_max) = (c, r)
10         DELAUNAY-INSERT(D, f)
11 until converged
12 return vertices of D
```

We make sure that a point is only moved to a new position if its new local mindist, namely $r_{\max}$, would be larger than its old local mindist d_x; otherwise, we simply reinsert it at its old position.

Figure 2 illustrates how the method successively distributes five points $X = \{x_1, \ldots, x_5\}$ in the unit torus. Panels 1a and 1b show how the target position for the first point x_1 is chosen: we search for the triangle in $\mathcal{D}(X\backslash\{x_1\})$ that has the largest circumcircle and move x_1 to the circle's center. The distance map in the background indicates that this is indeed the farthest point. We proceed in the same way for $x_2, \ldots, x_5$, as shown in the remaining panels.

It is easy to see that this farthest-point optimization always converges and yields arrangements with a high average mindist. The key observation is that moving a point x to the farthest point of $X\backslash\{x\}$ maximizes, by definition, its local mindist δ_x. In the worst case, no better position can be found and x remains at its old position. Because $\bar{\delta}_X \propto \sum \delta_x$, the average mindist must increase during a full iteration, so the optimization can never return to a previous point distribution or get stuck in cyclic configurations. We

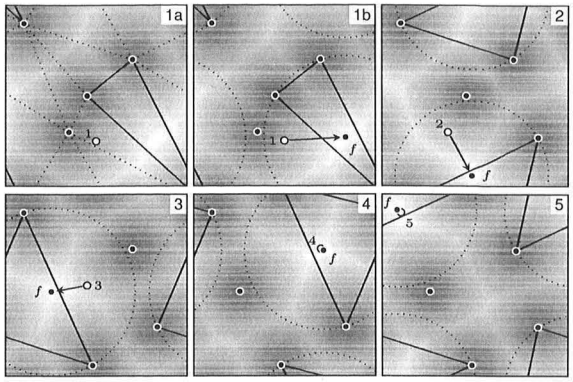
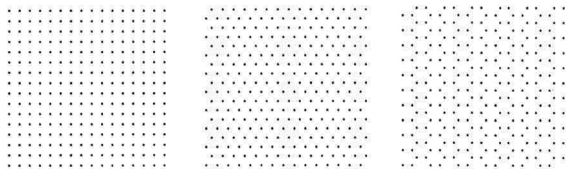


Figure 2: *Geometrical illustration of one full iteration applied to 5 points in the unit torus. Each point is successively moved to the center of the largest empty circle of the remaining points. The grayscale image in the background represents the distance map of $X\backslash\{x_i\}$ and reflects the toroidal metric. The dotted circle is the largest empty circle, and the highlighted triangle the corresponding face in the Delaunay triangulation of $X\backslash\{x_i\}$.*

stop the iteration as soon as the increase of $\bar{\delta}_X$ falls below a threshold ϵ, i.e., as soon as $\bar{\delta}_X^{\text{new}} - \bar{\delta}_X^{\text{old}} < \epsilon$; this must happen eventually since $\bar{\delta}_X$ is bounded for points in the unit torus. Convergence is fast enough that we can use the machine precision for ϵ.

For the global mindist we have $\delta_X = \min \delta_x$, so we are only guaranteed that it is non-decreasing. In fact, it is easy to construct point sets where δ_X remains constant for several iterations. But δ_X is strictly increasing as long as all points are still moving. For randomly distributed point sets we found this to be always the case.

In this case of random seed points, farthest-point optimization converges towards distributions with a mindist $\delta_X \approx 0.93$; a few intermediate steps during the optimization of 1024 points are shown in Figure 1. Since convergence becomes slower as we approach the maximum, we have found it useful to stop the iteration earlier. In our experience, a threshold of $\delta_X = 0.925$ is a good compromise between high-quality results and reasonable computation times. We will study the convergence empirically in Section 3.4.



Even though most input point sets converge towards irregular arrangements, some stable configurations are regular. In the three examples above, no point can be moved to a position that is "farther" away from the remaining points. If there are defects in the regular arrangements, however, FPO quickly breaks up the regularity. In this sense, FPO doesn't actively randomize its input, but it amplifies irregularities; this intuitively explains why the algorithm doesn't converge towards regular arrangements.

3.2 Runtime Complexity

Let us consider the runtime complexity of the inner loop in FARTHEST-POINT-OPTIMIZATION. We denote the average *degree* of a point (i.e., its average number of neighbors in the Delaunay triangulation) by g and the number of points by $n := |X|$. The

runtime of lines 4–10 can now be broken down as follows:

4: $\mathcal{O}(g)$ since we have to inspect the Delaunay neighbors of x to determine d_x.

5: between $\mathcal{O}(g)$ and $\mathcal{O}(g^2)$, depending on the algorithm used [Devillers 2002].

6–9: $\mathcal{O}(n)$ since there are $\mathcal{O}(n)$ triangles in $\mathcal{D}(X)$.

10: $\mathcal{O}(g)$ if we already know the triangle that contains the point; otherwise, between $\mathcal{O}(\sqrt{n})$ and $\mathcal{O}(\log n)$, depending on the algorithm used to locate the triangle [Devroye et al. 2004].

We assume that $g = \mathcal{O}(1)$ which is true or conjectured to be true for large classes of well-distributed point sets [Erickson 2005]. In this case, the overall runtime is $\mathcal{O}(n)$ for a single movement and $\mathcal{O}(n^2)$ for a full iteration. Two algorithmic improvements allow us to push this down to approximately $\mathcal{O}(n \log n)$ per full iteration.

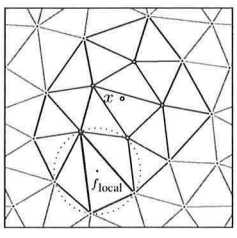
First, we can speed up the process of inserting the farthest point f into the triangulation. In our experience, f almost always lies either inside the triangle t corresponding to the largest empty circle, or at least close to it; this can already be seen in Figure 2. Since we know t from lines 6–9, locating and inserting f can be done in approximately constant time.

Second, we can speed up the search for the farthest point by using a binary search tree to keep track of the largest empty circle. This lets us find the farthest point in $\mathcal{O}(\log n)$, but increases the time required for lines 4 and 6 also to $\mathcal{O}(\log n)$ since structural changes to the Delaunay triangulation must be reflected in the tree. Taken together, this means that the running time is dominated by the tree operations, and the time required for a full iteration is $\mathcal{O}(n \log n)$.

3.3 Local Farthest-Point Optimization

This final $\mathcal{O}(n \log n)$ algorithm from the previous section is efficient, but since the tree operations must be intertwined with the update operations of the Delaunay triangulation, its implementation is a little involved. As an alternative we can use the following variant that only requires $\mathcal{O}(n)$ per iteration but converges more slowly.

The idea behind this modified algorithm is to simplify the search for the farthest point. When moving a point x, we do not attempt to determine the *largest* empty circle but contend ourselves with a *large* empty circle in the neighborhood of x. In other words, instead of checking the circumcircle of all triangles in $\mathcal{D}(X\backslash\{x\})$, we restrict the search to a subset $T \subset \mathcal{D}(X\backslash\{x\})$ that is in some sense "close" to x. If the expected size of T is independent of n, each point can be moved in $\mathcal{O}(1)$.



There are many strategies for choosing T. In our experience, the choice does not influence the quality of the resulting point sets, only the number of iterations. Here, we discuss the one that has proven to be a good compromise between iteration and convergence speed: we include in T all triangles that are incident with the neighbors of x in $\mathcal{D}(X)$ (see embedded figure). Since there are $\mathcal{O}(g^2)$ such triangles, moving a single point can indeed be done in constant time. We will refer to this variant as *local FPO*, in contrast to the *global FPO* from Section 3.1.

Since convergence guarantee from Section 3.1 only relied on the fact that the local mindist doesn't decrease, it remains valid in the case of the local FPO. However, since the local FPO moves points only locally, the mindist increases more slowly. Nevertheless, both methods converge towards point sets that are indistinguishable. In fact, once the points are sufficiently well distributed, local and global FPO are equivalent, since the farthest point of $X\backslash\{x\}$ is almost always located inside the hole that resulted from removing x.

Method	$\delta_X =$	0.75	0.775	0.8	0.825	0.85	0.875	0.9	0.925
[Lloyd 1982]		70	113	425*	-	-	-	-	-
[Balzer et al. 2009]		111	357*	-	-	-	-	-	-
Local FPO		3	4	6	8	14	27	64	352
Global FPO		1	2	2	3	4	6	13	118
	$\bar{\delta}_X =$	0.75	0.775	0.8	0.825	0.85	0.875	0.9	0.925
[Lloyd 1982]		2	3	4	5	8	13	29	122
[Balzer et al. 2009]		2	2	2	4	10	50	414*	-
Local FPO		1	1	1	1	1	2	3	10
Global FPO		1	1	1	1	1	2	2	6

Table 2: *Number of iterations needed to achieve a certain minimum distance δ_X* (top) *and average minimum distance $\bar{\delta}_X$* (bottom). *The results are averaged values from optimizing 10 sets of 4096 random points. (*) indicates that the mindist could not always be achieved.*

This suggests a hybrid algorithm that uses the global $\mathcal{O}(n \log n)$ algorithm for the first few iterations and then switches to the more efficient $\mathcal{O}(n)$ algorithm. In practice, this has turned out to be the fastest variant of farthest-point optimization, but for this paper, we will keep the discussion of the two algorithms separate.

3.4 Discussion and Evaluation

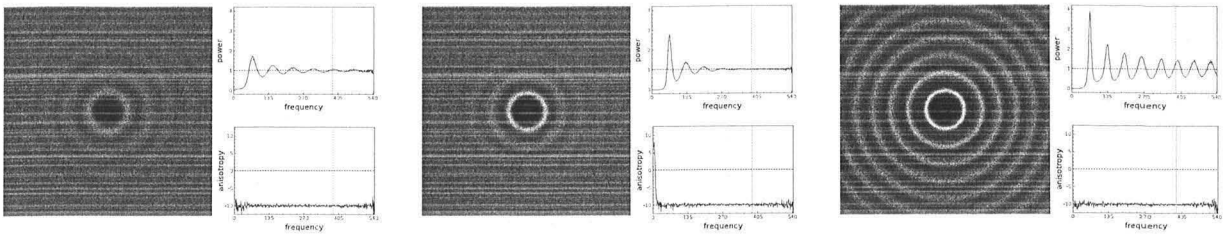
In this section we empirically study the main properties of the proposed optimization scheme and the point sets it generates. We will be primarily concerned with general observations and defer practical applications until Section 5.

We implemented the global and local FPO using the dynamic Delaunay triangulations from CGAL [CGAL], which we extended to handle toroidal boundary conditions. Despite their iterative nature, both algorithms are reasonably fast. For 4096 points, one iteration takes an average 39 ms for the global FPO and 25 ms for the local FPO.[1] Starting with a random point distribution, the full optimization until $\delta_X \geq 0.925$ takes on average 4.7 s (122 iterations) using the global FPO and 8.8 s using the local FPO (348 iterations).

Both algorithms consistently converge towards point sets with excellent blue noise properties. Figure 3 shows a representative example of the standard spectral measures—power spectrum, radially averaged power spectrum, and anisotropy—based on ten FPO point sets with 4096 points (for each set $\delta_X \approx 0.93$). We compare the results to pure dart throwing ($\delta_X \approx 0.75$) and one state-of-the-art method [Balzer et al. 2009]. The spectral properties of the method by Schmaltz et al. [2010] are very similar, but we only had access to non-toroidal point sets, which would have skewed the analysis.

We see in Figure 3 that there is almost no energy around the origin and no discernible anisotropy for FPO points. By maximizing the mindist, our method pushes low energy as far as possible towards higher frequencies.We also see that the amplitude of the radial power falls off very slowly. This reflects another kind of uniformity: for FPO points, the variance of the local mindists is very small, i.e., $\bar{\delta}_X \approx \delta_X$ (see also Table 1). As a consequence, not only the distance of each point to its direct neighbors is relatively constant, but the distance to its second and higher-order neighbors becomes very similar too.

[1] Performance measurements were obtained using a single core of a Xeon processor with 2.8 GHz using gcc.



Dart Throwing, $\delta_X \approx 0.75$ [Balzer et al. 2009], $\delta_X \approx 0.78$ *Our Method*, $\delta_X \approx 0.93$

Figure 3: *Analyzing common spectral properties for point sets generated by pure dart throwing* (left), *a state-of-the-art method* (center), *and our optimization method* (right). *Both of our algorithms consistently converge towards point sets with excellent blue properties. There is almost no energy around the origin and no discernible anisotropy. Maximizing the minimum distance by our method pushes low energy as far as possible towards higher frequencies as indicated by the strong peak at the spatial frequency* $1/d_X$.

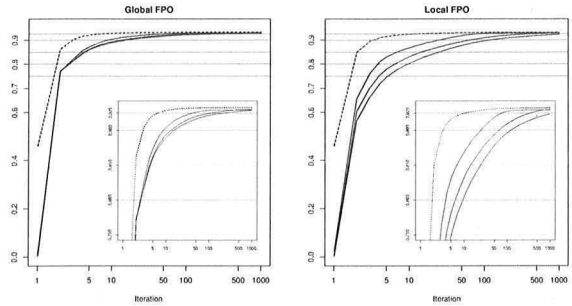


Figure 4: *Average convergence of* δ_X *(solid lines) and* $\bar{\delta}_X$ *(dashed lines) for random sets of 512, 4096, and 32768 points, from left to right. The inset magnifies the region* $0.75 \leq \delta_X \leq 0.95$.

The convergence speed of the global and local FPO is illustrated in Figure 4. Both δ_X and $\bar{\delta}_X$ increase rapidly at first and then converge more slowly towards a maximum around 0.932. The achieved maximum isn't the same for each set but consistently falls between 0.93 and 0.933. For both algorithms, the three curves for the average mindist (dashed lines) lie almost on top of each other. This means that convergence of $\bar{\delta}_X$ is mostly independent of the number of points, which underlines how effectively FPO distributes the points. The convergence of the mindist (solid lines) depends more strongly on the input size, especially for the local variant.

Finally, Table 2 compares the number of iterations required to obtain well-distributed point sets with Lloyd's method and the algorithm by Balzer et al. It is obvious that both FPO variants are far more effective than the other methods at spreading out the points: a handful of iterations are typically sufficient to obtain point sets with excellent blue noise properties. These improvements are even more significant considering that state-of-the-art techniques [Balzer et al. 2009; Schmaltz et al. 2010] require $\mathcal{O}(n^2)$ per iteration.

4 Extension: Partitioning a Point Set

We saw that farthest-point optimizing a point set is a simple strategy to increase the minimum distance without introducing regularity. But in sampling scenarios other than image plane sampling, this is often not enough: we also need the *union* of several sets of samples to be well-distributed. A prominent example is direct light estimation or BSDF sampling via trajectory splitting [Arvo and Kirk 1990]: for each ray traced through the image plane, trace multiple rays towards an area light source or evaluate a BSDF multiple times. The overall coverage of the sample space is much better if the sample points for each ray *and* their union is well-distributed.

Since a high minimum distance remains desirable for such integration scenarios [Grünschloß and Keller 2009], we propose an extension of our optimization strategy that partitions our optimized point sets from the main section into equally sized subsets, each with optimized minimum distance. In contrast to existing techniques, such as the multi-class algorithm by Wei [2010] which directly generates partitioned point sets by dart throwing, we can in fact partition any given point set such that each subset is farthest-point optimized.

4.1 Partition Algorithm

Let us formulate the problem more formally. We want to partition a set X of input points into m subsets Y_i of size $n := |X|/m$. The points in each subset should be spread out as far as possible, i.e., we need to find n points in X that have maximized minimum distance. This can be solved by a process similar to our main algorithm: we start with a random subset of n points from X and successively move each point to a better position. The main difference is that we cannot move points arbitrarily to increase the mindist but are restricted to the positions of the points in the base set X.

To partition the full base set X so that $X = \bigcup_i Y_i$, we construct the subsets Y_i sequentially by first optimizing Y_1 considering all points in X, then Y_2 considering the points in $X \backslash Y_1$, and so on. The full algorithm can be formulated as follows.

FARTHEST-POINT-OPTIMIZED-PARTITION$(X, \{Y_i\})$

```
1  D_X = DELAUNAY(X)
2  foreach i in m − 1
3      Y_i = n random vertices of D_X
4      D_Y = DELAUNAY(Y_i)
5      DELAUNAY-REMOVE(D_X, Y_i)
6      repeat
7          foreach vertex y in D_Y
8              (f, d_f) = (y, d_y)
9              remove y from D_Y and insert into D_X
10             foreach vertex z in D_X
11                 d_z = mindist to any vertex in D_Y
12                 if d_z > d_f
13                     (f, d_f) = (z, d_z)
14             remove f from D_X and insert into D_Y
15         Y_i = vertices of D_Y
16     until Y_i did not change
17 Y_m = vertices of D_X
18 return {Y_i}
```

Since the location of the farthest point f is now restricted to an unused point $x \in X$, we maintain two Delaunay triangulations: D_X for the remaining points in X, and D_Y for the subset Y_i currently being optimized (lines 1, 4–5). To identify the global farthest point,

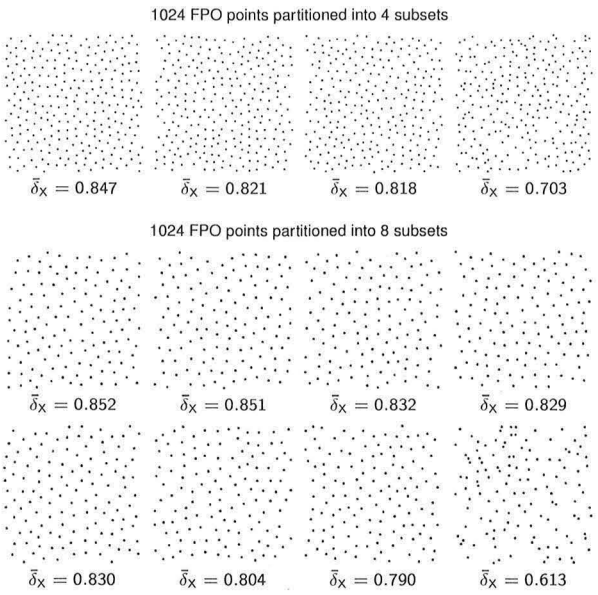


Figure 5: *Partitioning the 1024 points from Figure 1 into four* (top) *and eight* (bottom) *subsets of equal sizes. Despite the greedy characteristic of the partition algorithm, most subsets show average minimum distances* $\bar{\delta}_X > 0.8$.

we loop over all available points $z \in X \backslash Y_i$ and pick the one that is farthest from the current selection Y_i (lines 7, 10–13). Keeping the intersection $X \cap Y_i$ empty allows us to swap the old point y and the farthest point f between the corresponding Delaunay triangulations, i.e., remove it from the first and insert it into the second (lines 9, 14). Repeating this procedure for all points in Y_i concludes a full iteration. Once we have optimized a subset, we repeat the process for the other subsets using only the remaining points.

The optimization of a subset converges when no farther point $f \in X \backslash Y_i$ can be found for any of the $y \in Y_i$; this can be easily detected e.g. by setting a flag in line 13. Since $\bigcup_i Y_i = X$, the last subset Y_m is fully determined by the first $m-1$ subsets (line 17) and therefore does not need to be optimized.

Analogous to the continuous FPO, the computational complexity of a naive implementation of this discrete space algorithm is roughly quadratic in the number of points per subset. (The real complexity is $\mathcal{O}(n|X|)$ per iteration per subset, where $|X|$ denotes the number of remaining points after removing each Y_i in line 5.) Similar to the continuous variant, this can be sped up to $\mathcal{O}(n \log n)$ per iteration (per subset) if we utilize a binary tree that tracks the global farthest point and is updated after each insert and remove operation.

4.2 Discussion

This FPO-based partition algorithm works as a post process for arbitrary input point sets. Similar to the continuous variant, it is guaranteed to converge since eventually the current selection Y_i cannot be improved by finding a farther point in $X \backslash Y_i$.

Figure 5 shows the result when partitioning the 1024 points from Figure 1 into four *(top)* and eight *(bottom)* subsets. It can be seen that the algorithm's greedy characteristic mostly affects the last subset which is not able to optimize its selection of points. Although this is not ideal, the greedy approach still generated the overall best results among other strategies we experimented with. The first $m-1$ subsets are all very uniform and show high average minimum distances with most $\bar{\delta}_{Y_i} > 0.8$. Our non-optimized implementation of the partition algorithm took 97 ms to perform the partition in Figure 5 into 8 subsets. Partitioning 16384 points into 16 subsets of 1024 points takes an average 3.3 s.

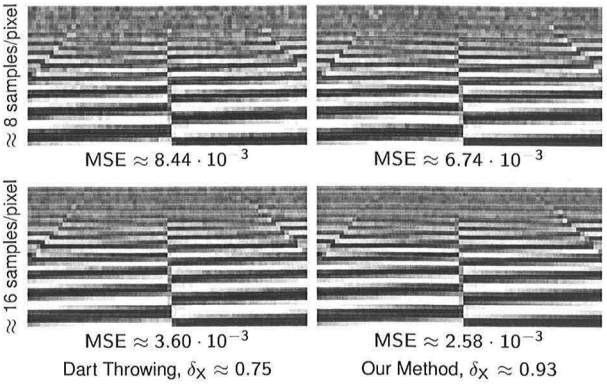


Figure 6: *Sampling an infinite checkerboard. Although the differences are subtle, notice the better edge anti-aliasing* (bottom of the close-ups) *and reduced noise in regions beyond the Nyquist frequency* (top of the close-ups) *for our point sets.*

5 Applications

We already evaluated the general properties of farthest-point optimized point sets in the main section. We now analyze their quality in two important practical applications: image plane sampling and numerical integration for physically based rendering.

5.1 Image Plane Sampling

We saw that FPO points show excellent blue noise properties. This makes them especially suitable for image plane sampling in graphics, where we want the samples to be both irregular (so that aliasing is mapped to noise) and of high minimum distance (so that aliasing is mapped to high frequencies). We show that this is indeed the case using three image plane sampling scenarios shown in Figures 6, 7, and 8. In each scenario, reconstruction was performed using a Lanczos-2 filter and mean square errors were obtained in relation to a reference image using 4096 random samples/pixel.

Fig. 6 shows the result of sampling the infamous checkerboard. The close-ups were chosen so that content both below and above the Nyquist frequency is visible in the same image. Although the differences are subtle, note the better anti-aliasing along edges and the reduced noise in the top of the images.

The improvement is even more obvious in the two other figures which show dedicated scenarios for edge anti-aliasing and high frequency sampling. The higher uniformity of FPO point sets noticeably improves the rendering of edges in Figure 7, and when sampling the common 2D chirp $f(x,y) = (\cos(\alpha x^2 + \alpha y^2) + 1)/2$ in Figure 8, our method already produces a very good solution at approximately 2 samples per pixel. This is mainly due to a good trade-off between noise and moiré artifacts since maximizing the minimum distance yields both a high effective Nyquist frequency (which lessens aliasing) and very uniform point sets (which yields less noise).

5.2 Numerical Integration

We now investigate the applicability of FPO points to numerical integration problems occurring in physically based rendering [Pharr

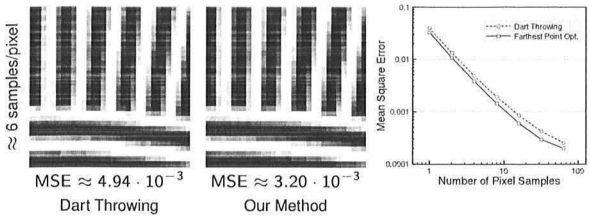


Figure 7: *Sampling a series of circles and lines of different orientation. Increasing the uniformity of point sets via FPO noticeably enhances edge anti-aliasing, both visually and numerically.*

and Humphreys 2010]. Due to the curse of dimension it is likely that point sets with maximized mindist do not improve upon random point sets for higher dimensional problems, but for one- and two-dimensional problems they promise a noticeable increase in convergence speed. As aforementioned, a prominent example is direct light estimation by trajectory splitting. The common method to generate the corresponding sample points is the Sobol' $(0, 2)$-sequence [Sobol 1967] which has the powerful property that each successive set of power-of-2 points is well-distributed (a $(0, m, 2)$-net of low-discrepancy) while their union is also well-distributed. We can mimic this property using our partition algorithm from Section 4. To this end, we compute an optimized partition of an FPO point set. This ensures that each subset is well-distributed and their union is a FPO point set with maximized minimum distance. For performance reasons we do not perform the optimization procedure online but precompute a point set and the necessary partitions. In order to ensure that the corresponding integral estimators remain unbiased, we can randomly shift the sets on the unit torus (Cranley-Patterson rotation) which preserves the minimum distance.

Figures 9 and 10 show results for this procedure which we integrated into PBRT [Pharr and Humphreys 2010]. The simple scene is lit by two light sources—one circular area light source and one infinite light source in form of a HDR environment image—to verify that our samples retain their good distribution even after mapping them from the unit square to e.g. a disk. The scene is untextured so that the remaining approximation error isn't masked. The close-ups in Figure 9 show the result when utilizing 4×4 samples, i.e., 4 pixel samples and for each of those 4 secondary samples (*integrator samples*). The MSE confirms that FPO points partitioned into optimized subsets outperform the low-discrepancy sequence independent of the number of samples. For comparison, the plot also shows results for two naive variants where either the subsets are optimized but not their union ("no partition"), or where the union is a FPO point set but where subsets were chosen randomly ("random partition"). Obviously, the best result is obtained if both the union and the partition are optimized, but overall an optimized total set is more important than an optimized partition.

In Fig. 9 we increased both pixel and integrator samples at the same time, whereas we kept their product constant at 16 combined samples in Fig. 10. Using partitioned FPO samples yields a lower error for every combination of pixel and integrator samples. In particular, the MSE remains roughly constant despite the varying numbers of pixel vs. integrator samples. The plot again underlines the importance of the partition procedure. Just assigning good integrator samples to each pixel sample quickly becomes inferior to the partitioned results, even at twice the number of combined samples.

6 Conclusion

We have presented a new iterative method for optimizing the distribution of points in the plane. The main feature of the resulting point sets is that they are practically optimal blue noise samples under the assumption that such point sets should be both irregular and of high minimum distance. This re-raises the question of ideal image plane sample points as we suspect that it will be hard to increase the minimum distance further without introducing regular structures.

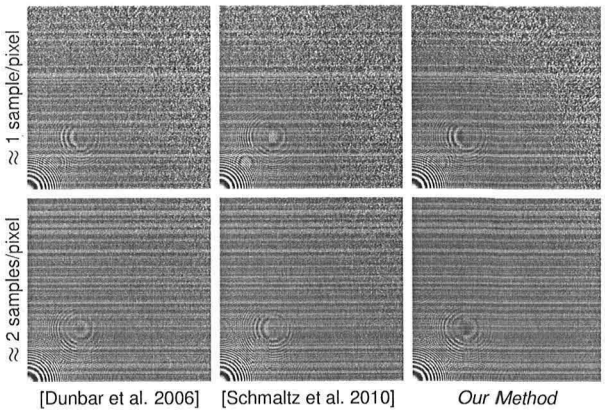


Figure 8: *Sampling a common 2D chirp. In the top row we used 512^2 sample points to produce images of 512×512 pixels, and in the bottom row $2 \cdot 512^2$ samples for the same output resolution. Maximizing the minimum distance increases the Nyquist frequency so that aliasing is shifted to noise of higher frequency.*

We introduced a variant of this algorithm that allows to partition a given point set such that each subset is well-distributed. This enables the use of our optimized point in a broad range of numerical integration problems occurring in physically based rendering. Even though the partition algorithm yields good results, we think there is still room for improvement and are investigating other strategies than the greedy approach currently used.

We have only considered the problem of distributing points in the unit torus, but the general algorithm directly extends to other geometric arrangements, such as higher dimensions, points on bounded surfaces, triangulated domains, or non-Euclidean metrics. Some work in this area has been done in the context of non-uniform sampling and remeshing [Moenning and Dodgson 2003; Peyré and Cohen 2006] using the strategy by Eldar et al., but our iterative method may prove advantageous for these applications as well.

Acknowledgments We thank the anonymous reviewers for their in-depth reviews and valuable suggestions for improvement.

References

ARVO, J., AND KIRK, D. 1990. Particle transport and image synthesis. *Computer Graphics (Proc. of SIGGRAPH 90)*, 63–66.

BALZER, M., SCHLÖMER, T., AND DEUSSEN, O. 2009. Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Trans. Graph. 28*, 3, 86:1–8.

CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

COOK, R. L. 1986. Stochastic sampling in computer graphics. *Computer Graphics (Proc. of SIGGRAPH 86) 5*, 1, 51–72.

DEVILLERS, O. 2002. On deletion in Delaunay triangulations. *Int. J. Comput. Geometry Appl. 12*, 3, 193–205.

DEVROYE, L., LEMAIRE, C., AND MOREAU, J.-M. 2004. Expected time analysis for Delaunay point location. *Comput. Geom. 29*, 2 (Oct.), 61–89.

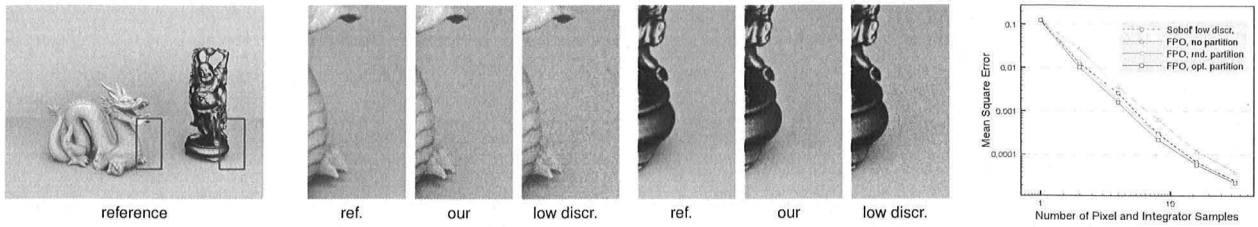


Figure 9: (Left) *Reference image using* 512 *pixel samples and* 16 *integrator samples per pixel sample.* (Center) *Closeups of the reference image, a rendering using* 4×4 *FPO samples partitioned into optimized subsets, and a rendering using* 4×4 *Sobol' low discrepancy samples.* (right) *MSE plot for an increasing number of* $n \times n$ *pixel and integrator samples.*

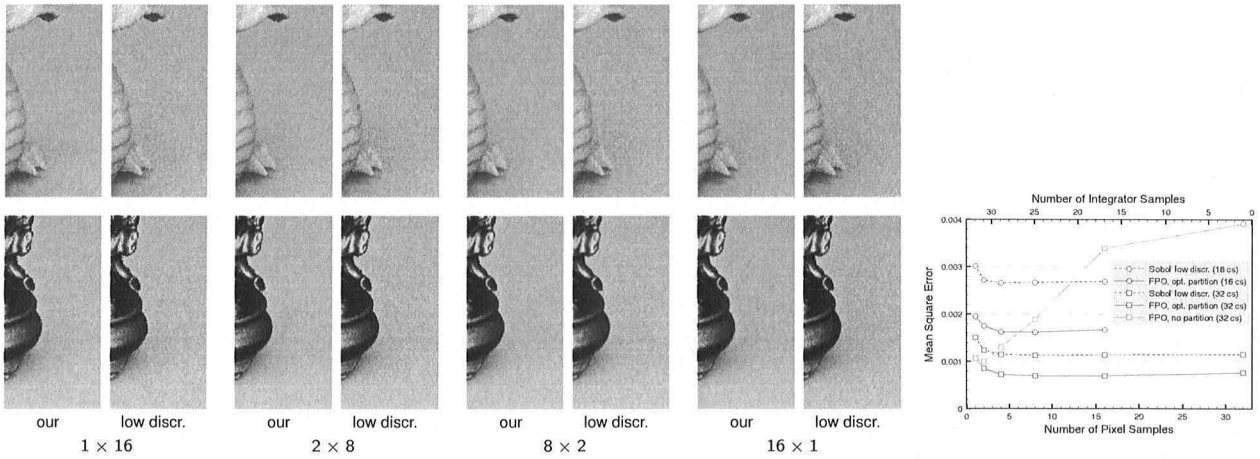


Figure 10: *In this series we kept the number of pixel samples times integrator samples constant at 16 combined samples* (cs)*. Using partitioned FP optimized samples outperforms the low discrepancy sequence for every combination of pixel and integrator samples. It can also be seen that the MSE remains roughly constant despite the varying numbers of pixel vs. integrator samples.*

DU, Q., FABER, V., AND GUNZBURGER, M. 1999. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review 41*, 4, 637–676.

DUNBAR, D., AND HUMPHREYS, G. 2006. A spatial data structure for fast Poisson-disk sample generation. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2006) 25* (July), 503–508.

ELDAR, Y., LINDENBAUM, M., PORAT, M., AND ZEEVI, Y. Y. 1997. The farthest point strategy for progressive image sampling. *IEEE Trans. Image Process. 6*, 9 (Sept.), 1305–1315.

ERICKSON, J. 2005. Dense point sets have sparse Delaunay triangulations. *Discrete Comput. Geom. 33* (Jan.), 83–115.

GAMITO, M. N., AND MADDOCK, S. C. 2009. Accurate multidimensional Poisson-disk sampling. *ACM Trans. Graph. 29* (Dec.), 8:1–8:19.

GRÜNSCHLOSS, L., AND KELLER, A. 2009. (t, m, s)-nets and maximized minimum distance, part II. *Monte Carlo and Quasi-Monte Carlo Methods 2008*, 395–409.

KANAMORI, Y., SZEGO, Z., AND NISHITA, T. 2011. Deterministic blue noise sampling by solving largest empty circle problems. *Journal of IIEEJ 40*, 210.

LAGAE, A., AND DUTRÉ, P. 2008. A comparison of methods for generating Poisson disk distributions. *Computer Graphics Forum 27*, 1, 114–129.

LLOYD, S. P. 1982. Least square quantization in PCM. *IEEE Transactions on Information Theory 28*, 2, 129–137.

MITCHELL, D. P. 1991. Spectrally optimal sampling for distribution ray tracing. *Computer Graphics (Proc. of SIGGRAPH 91)*, 157–164.

MOENNING, C., AND DODGSON, N. A. 2003. Fast marching farthest point sampling. Tech. Rep. UCAM-CL-TR-562, University of Cambridge Computer Laboratory, Apr.

PEYRÉ, G., AND COHEN, L. 2006. Geodesic remeshing using front propagation. *Int. J. Comput. Vision 69*, 1 (Aug.), 145–156.

PHARR, M., AND HUMPHREYS, G. 2010. *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc.

SCHMALTZ, C., GWOSDEK, P., BRUHN, A., AND WEICKERT, J. 2010. Electrostatic halftoning. *Computer Graphics Forum 29*, 8, 2313–2327.

SOBOL, I. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Zh. vychisl. Mat. mat. Fiz. 7*, 4, 784–802.

TÓTH, L. F. 1951. Über gesättigte Kreissysteme. *Mathematische Nachrichten 5*, 3–5, 253–258.

ULICHNEY, R. A. 1988. Dithering with blue noise. *Proc. IEEE 76*, 1 (Jan.), 56–79.

WEI, L.-Y. 2008. Parallel Poisson disk sampling. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2008) 27* (Aug.), 20:1–20:9.

WEI, L.-Y. 2010. Multi-class blue noise sampling. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2010) 29* (July), 79:1–79:8.