Eurographics Symposium on Geometry Processing 2012
Eitan Grinspun and Niloy Mitra
(Guest Editors)

Volume 31 (2012), Number 5

Parallel Blue-noise Sampling by Constrained Farthest Point Optimization

Renjie Chen† and Craig Gotsman‡

Technion – Israel Institute of Technology

Abstract

We describe a fast sampling algorithm for generating uniformly-distributed point patterns with good blue noise characteristics. The method, based on constrained farthest point optimization*, is provably optimal and may be easily parallelized, resulting in an algorithm whose performance/quality tradeoff is superior to other state-of-the-art approaches.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture-Sampling—

1. Introduction

Many computer graphics applications such as rendering, imaging and importance sampling require uniform but not regular point samplings of the plane, i.e. a distribution of points whose density is almost the same everywhere, yet the points not follow any obvious visual patterns. Ulichney [Uli87] provided the so-called "blue-noise characterization" of these point distributions in the Fourier domain, namely that the *power density spectrum* of a blue noise distribution has no low frequency energy or structural bias. Essentially, it avoids visually objectionable artifacts by replacing low frequency aliasing with high frequency noise. For easier analysis of the power density spectrum, Ulichney [Uli87] derived two useful one-dimensional statistics from the power spectrum of point distributions, i.e. the *radially averaged power spectrum* and the *anisotropy*. The power spectrum of a good blue-noise distribution has a structural residual peak which decays as slow as possible, while the anisotropy remains flat in the frequency domain.

In the last two decades, many algorithms have been developed to generate point sets possessing a blue-noise spectrum, but superior spectral properties seem to be accompanied by lengthy computation times. One of the most recent algorithms, based on Farthest Point Optimization (**FPO**) [SHD11], produces point distributions with excellent spatial statistics and blue-noise spectra. In a nutshell, it produces point distributions by starting with an initial distribution and then improves the positions of the points by iteratively moving them to globally optimal positions, in the sense that they are as far away as possible from all other points. The process terminates when no more improvement is possible. Unfortunately, **FPO** is relatively slow and not easily parallelizable because of its reliance on global data structures. In this paper we present a *localized* version of **FPO**, based on *constrained* farthest point optimization (**CFPO**). It uses a methodology very similar to that of **FPO**, except that it is localized. We prove that despite its local nature, it is *equivalent* to **FPO**, in the sense that it generates point distributions which **FPO** cannot improve further. It is difficult to over-emphasize the importance of this locality property. Not only does it simplify the algorithm in terms of the sophistication of the data structures employed, but it is also faster, without compromising at all the quality of the results. As we will see later, **CFPO** is 8× faster than **FPO**. Most important, it is easily parallelized with almost perfect speedup, resulting in a parallel version on the GPU which is 75× faster than **FPO**.

2. Related Work

Within the family of blue noise distributions, the Poisson disk distribution is the most popular. In these pseudo-random patterns, each sample is separated from all other samples by a minimal distance, the distribution does not exhibit any visual artifacts and may be shown to possess good blue-noise spectra. Many algorithms have been proposed to generate

† renjie.c@gmail.com
‡ gotsman@cs.technion.ac.il

Poisson disk distributions. We mention here just the most important variants, but an extensive survey of these methods can be found in [LD08].

Cook [Coo86] first proposed the famous "dart throwing" method for generating Poisson disk distributions: Given the radius d of the Poisson disk, the goal is to position as many points as possible in the unit square such that the disks of radius d centered at each point are all empty. At each step, a candidate point (the dart) is randomly generated and added to the point set only if it does not conflict with the existing points, i.e. overlap with any of the disks centered at existing points. However, as the number of points increases, the chance that a dart is accepted decreases dramatically, making the dart throwing algorithm very expensive and slow. Several improvements have been developed, and a parallelized variant taking advantage of graphics hardware was proposed by Wei [Wei08]. Ebeida *et al*. [EDP*11] proposed the *maximal* Poisson disk sampling algorithm where provably no more samples can be added into the set without violating the empty Poisson disk condition. (Note that the maximality property does not necessarily imply that the point set is the largest possible with Poisson disk radius d). The same paper provides a parallel implementation and, very recently, Ebeida *et al*. [EMP*12] generalized the maximal Poisson disk sampling algorithm to higher dimensions.

Another type of algorithm to generate blue-noise distributions is Lloyd's relaxation scheme [Llo82]. McCool and Fiume [MF92] first used it to optimize the point set generated by the dart throwing approach by iteratively moving closely-spaced points apart and widely-spaced points closer. Lloyd's algorithm eventually converges to a Centroidal Voronoi Tessellation (**CVT**), where each point resides at the centroid of its Voronoi cell. Unfortunately, **CVT** usually closely resembles a regular hexagonal grid, exhibiting high regularity with no blue noise characteristics, so the process must be stopped before it converges. Balzer *et al*. [BSD09] proposed the Capacity-Constrained Voronoi Tessellation (**CCVT**), a modification of the **CVT**, by imposing an extra capacity constraint - that all generalized Voronoi cells should have the same area. They demonstrate that the resulting point distributions possess high-quality blue noise spectra and the distribution does not converge to the hexagonal grid due to the random initialization in their pixel-based algorithm. Furthermore, given a density function, the area of a Voronoi cell can be generalized to its capacity, the total density contained in the cell. By equalizing the capacity of the cells, a non-uniform distribution conforming to the density function can be obtained. However, **CCVT** is extremely slow due to the pixel-based nature of the algorithm. Li *et al*. [LNW*10] proposed the fast **CCVT** algorithm, which achieves orders of magnitude acceleration over **CCVT** without compromising the blue noise characteristics. More recently Xu *et al*. [XLGG11] proposed the Capacity-Constrained Delaunay Triangulation (**CCDT**), which can be thought of as the dual of **CCVT**, where a Delaunay triangulation is optimized such that each triangle has the same capacity. Thanks to its use of a simple triangulation structure, **CCDT** was shown to produce point distributions comparable to those of **CCVT**, but at a fraction of the cost in runtime (and still faster than the accelerated algorithm of Li *et al*. [LNW*10]).

A third type of algorithm for generating blue-noise distributions, first introduced by Shade *et al*. [SCM00], is based on precomputed tiles of point sets. They use a dart throwing algorithm to generate a Poisson disk distribution over a set of Wang tiles in a preprocessing stage. Then during runtime, the tiles can be randomly and seamlessly assembled to obtain a Poisson disk distribution of arbitrary size. Since then several improved algorithms have been proposed. Lagae and Dutre [LD08] conclude that tile-based approaches are the only practical option for real-time applications and applications requiring large sample sets, and among all these approaches, corner-based Poisson disk tiles [LD06] has the best blue noise spectrum, yet due to the limited number of patterns obtainable from the precomputed tiles, the spectrum is far from optimal.

3. Parallel constrained farthest point optimization

3.1. Farthest point distribution

Let X be a set of points in the plane. For any $x \in X$, denote by $\delta(x)$ the minimal distance between x and any other point in X. Denote by $\delta(X)$ the minimum of $\delta(x)$ among all $x \in X$. Point sets X having large values of $\delta(X)$ are desirable and tend to have good blue noise properties. We say that a point $x \in X$ has the *global farthest point property* if it is a point inside the convex hull of X farthest away from all points of $X \backslash \{x\}$. This means that $\delta(x)$ cannot be increased by moving x within the convex hull. If all interior points in X (i.e. all points except the ones on the convex hull of X), have the global farthest point property, we say that X has the global farthest point property and is a *Farthest Point Distribution* (**FPD**) or that X is a *FPD with distance* $\delta(X)$. This means that $\delta(X)$ cannot be increased by moving any *single* interior point of X within the convex hull of X. In the following, we restrict ourselves to periodic point sets, i.e. point sets on the unit torus. Thus all the points can be treated in the same way, and the convex hull constraint for each point can be ignored. It is well known [SHD11] that if X contains n points, then $\delta(X) \leq \Delta(n)$, where $\Delta(n) = \sqrt{\frac{2}{\sqrt{3}n}}$. The bound is tight, achieved when the points of X are positioned on the hexagonal grid.

Schlömer *et al*. [SHD11] proposed to achieve **FPD**'s by a (greedy) Farthest Point Optimization (**FPO**) procedure: starting off with an essentially random point set X, and repeatedly moving each point $x \in X$ to the location which is farthest from all the other points in $X \backslash \{x\}$. The farthest location of a point set (inside the domain) is at the center of its largest empty circle, which is found using the Delaunay

triangulation of X. Each iteration of the algorithm performs a pass over all points. Since each iteration cannot decrease $\delta(X)$, **FPO** will converge to a local maximum of this quantity. Typically $50-100$ iterations are required until convergence. By definition, the limit point set X is a *FPD with distance* $\delta(X)$.

To locate the largest empty circle efficiently, **FPO** maintains a dynamic global Delaunay triangulation $DT(X)$ and a priority queue of the triangles in $DT(X)$, where the priority of a triangle is the radius of its circumcircle. **FPO** proceeds by removing a point from the set, updating the triangulation and inserting the point back into the triangulation at the center of the largest circumcircle. The triangulation is kept Delaunay all the time using a suitable update algorithm.

Querying the priority queue for the largest empty circle requires $O(\log n)$ time, thus the time complexity of a full **FPO** iteration is at least $O(n \log n)$, where n is number of points. To speed it up, Schlömer *et al.* [SHD11] proposed a local version of the **FPO** algorithm (**LFPO**) computing the farthest point locally, for example, only inside the 2-ring neighborhood of the current vertex. This way only $O(g^2)$ time is required to find the largest empty circle, where g is the average number of neighbors in $DT(X)$. As g is essentially independent of n, a full iteration requires only $O(n)$ time.

However, the true bottleneck of the **FPO/LFPO** procedure is the insertion of the point back at a new location. This can require $O(n)$ time in the worst case (to update the Delaunay triangulation), so in practice, one iteration costs more time than simply rebuilding the entire Delaunay triangulation from scratch, which costs only $O(n \log n)$ time [SD95]. Ironically, in the experiments we performed, the **LFPO** method actually runs a little slower than the global **FPO** method, as shown in Fig. 8. Furthermore, as the global Delaunay triangulation is required to identify the 2-ring neighborhood, the **LFPO** method is not truly a local algorithm.

On the positive side, the point set that **FPO** converges to exhibits extremely good blue noise properties. More precisely, the power spectrum of the point sets that **FPO** converges to have radially averaged power spectrum with structured residual peaks which decays much slower than the other state-of-the-art algorithms, while the anisotropy still remains flat. Beyond these two quantitative measures, Schlömer *et al.* [SHD11] measure $\delta(X)$ and $\bar{\delta}(X)$ - the minimal and average distance, respectively, between a point and its closest neighbor, normalized by the best possible value $\Delta(n)$, thus in the range $[0,1]$. They show that their algorithm achieves values larger than 0.9 for all values of n, compared to other state-of-the-art algorithms, which seem to achieve much worse values. Thus this would seem to be the algorithm with best quality results. The aim of this paper is to localize the **FPO** algorithm in order to accelerate it and make it easier to parallelize, thus potentially speeding it up indefinitely. For this we propose the Constrained Farthest Point Optimization (**CFPO**) algorithm. We use both the traditional one-dimensional power spectrum statistics and the minimal distance quantities to measure the quality of the resulting blue-noise distributions.

3.2. Constrained farthest point optimization

FPD's are very desirable, but we would like to generate them using only *local* properties. We say a point $x \in X$ has the *local farthest point property* relative to the set $Y \subset X$ if it is the point inside the convex hull of Y farthest from all points of $Y \backslash \{x\}$. Y is typically some small subset of X in the vicinity of x, as illustrated in Fig. 1. Now, as Fig. 1 also shows, if a point x is a local farthest point, it is not necessarily also a global farthest point, and vice versa.

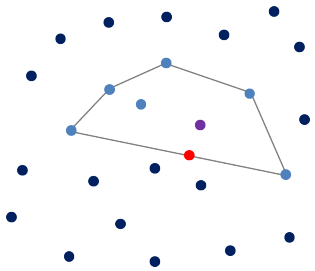


Figure 1: *Global farthest point and local farthest point properties in a set of (light and dark blue) points X: When added to this set, the purple point has the global farthest point property. When added to this set, the red point has the local farthest point property relative to Y - the set of light blue points.*

Fortunately, it is possible to relate the two concepts by adding an additional concept, the *constraint region* in which x is allowed to be, to the definition of the local farthest point property. With this, we can characterize a **FPD** locally:

A point set X is a **FPD** if and only if each point in the set satisfies two local properties, namely, for each point $x \in X$, there exists a local neighborhood D such that:

1. S is fully covered by the disks centered at the points of X in D (including x) having radius δ.
2. x is the farthest point of Y - the points in D (excluding x) - constrained to lie inside S.

where S is D shrunk by δ.

Formally, we have the following theorem which provides a purely local characterization of an **FPD**.

Local Characterization Theorem
Let X be a point set on the unit torus R, and $C(z,r)$ be the *closed* disk centered at z with radius r, then X is a **FPD** with distance $\delta = \delta(X)$ if and only if $\forall x \in X, \exists D = D(x) \subseteq R$, such that

Property 1 (*Local covering property*):

$$\bigcup_{z\in D\cap X} C(z,\delta) \supseteq S$$

Property 2 (*Local farthest point property*): x is the farthest point of $Y(x)$ in S

where $Y(x) = D \cap (X \backslash \{x\})$ is the subset of $X \backslash \{x\}$ inside D, $S = S(x) = D(x) \ominus \delta$, and $\ominus$ is the erosion operator. □

The proof of the Local Characterization Theorem is given in Appendix A. In the proof for the necessary condition, we take $D(x)$ to be $\hat{C}(x, 2\delta)$ - the *open* disk centered at x with radius $2d$. However, it is easy to see that D may be taken to be any region of the plane containing $\hat{C}(x,\delta)$. See an example in Fig. 2.

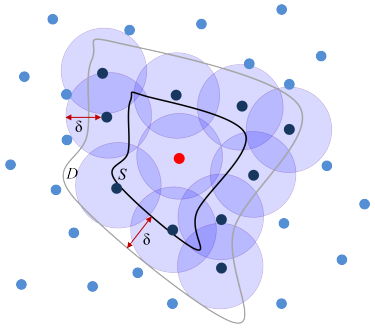


Figure 2: *Local Characterization Theorem. Each (red) point x in a **FPD** is a farthest point of a local point set $Y(x)$ (the points in $D(x)$ - the interior of gray curve) constrained to some region $S(x)$ (the interior of the dark curve). The transparent disks show the local covering property.*

The Local Characterization Theorem gives a completely local procedure to check whether a given point set is a **FPD**. Note that *both* local properties are necessary to obtain a **FPD**. Fig. 3 shows an example of a point set that satisfies *only* the local farthest point property, thus is not a **FPD**.

It is possible to take advantage of the Local Characterization Theorem to produce a **FPD** using a local version of the algorithm described by Schlömer *et al*. [SHD11], i.e. given an initial point set, we can improve it by repeatedly moving each point to a constrained *local* farthest point. So if and when the algorithm converges, the point set will satisfy the local farthest point property (property 2 of the theorem). To show that the limit is also an **FPD**, we need to make sure that this limit also satisfies the local covering property. We take care of this by generating the initial point set properly, as will be discussed in Section 3.5.

Applying the Local Characterization Theorem requires the definition of the neighborhood $D(x)$ and knowledge of the distance $\delta(X)$. We address this by partitioning the point set into cells of a uniform square grid, and define $D(x)$ to be the 3×3 set of grid cells centered at the grid cell containing

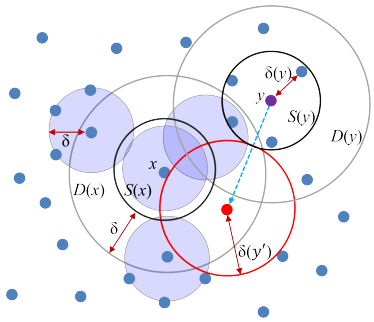


Figure 3: *A point set satisfying only the local farthest point property in the Local Characterization Theorem. Point x violates the local covering property. The point set is not a **FPD**, as the purple point y can move to the red position and increase $\delta(y)$ to $\delta(y')$.*

x (thus $Y(x)$ is the subset of X contained in $D(x)$, excluding x itself).

The full **CFPO** algorithm can now be formulated as in Algorithm 1.

Algorithm 1: Algorithm **CFPO**

```
1  Initialize a point set X with n points
2  Partition the domain with a uniform square grid, and
   store X in a bucket-like data structure G conforming to
   the grid. Denote by G(x) the grid cell containing x.
3  foreach point x ∈ X do
4      Construct D(x) := the 3 × 3 block of grid cells
       centered at G(x) and Y(x) := the local neighbor list
       - all other points of X in D(x).
5      Define the constraint square S(x) = D(x) ⊖ δ
6      x ← LocalCFPO(Y(x), S(x)) // see Section 3.4
7      Update G to reflect the new position of x.
8  if no point x moved (up to a tolerance) then
9      the algorithm has converged. return X
10 else
11     goto Step 3
```

In the following theorem, we show that $\delta(X)$ never decreases during algorithm **CFPO**, thus **CFPO** will always converge to a point set having the local farthest point property. Proof of the theorem appears in Appendix B.

Convergence Theorem
Let X be a planar point set with minimal distance $\delta = \delta(X)$. Partition the domain into a uniform grid G of square cells of size $L > \delta$. Denote by $G(x)$ the cell containing x. If we replace any point $x \in X$ with another point x' inside or no further than $L-\delta$ away from $G(x)$ in both axis-aligned directions, such that the minimal distance from x' to all the other points inside the block

of 3×3 cells centered at $G(x)$ does not decrease, then $\delta(X)$ does not decrease. □

3.3. Parallel CFPO

The greatest advantage of providing a local characterization of a **FPD** is that it paves the way to easy parallelization. According to the Convergence Theorem, we should choose the size of the local neighborhood S to be smaller than $2(L-\delta)+L = 3L-2\delta$ (L is the grid cell size). However, this size is dependent on $\delta(X)$, requiring keeping track of this global quantity throughout the algorithm, which could be difficult in a parallel implementation. Luckily, this can be avoided due to the following theorem, whose proof appears in Appendix C.

Parallelization Theorem
Let X be a set of n of points located on the unit torus. Partition the domain into a uniform grid G of square cells of size L. If L is such that $L > \Delta(n)$ and a subset P of points of X are (sequentially) moved to new positions while satisfying the conditions of the Convergence Theorem taking $\delta = \Delta(n)$, then $\delta(X)$ cannot decrease as a result. If P is such that all points in P are at least three cells distant from each other (in each of the axis directions), then they can move to their new positions in parallel without decreasing $\delta(X)$. □

Thanks to this theorem, we can parallelize **CFPO** easily. Given n, G is taken to be the grid whose cell size is $L > \Delta(n)$, and $S(x)$ is taken to be the square of constant size $3L-2\Delta(n)$. We then partition G into $9 = 3 \times 3$ disjoint subsets of cells, each subset formed by taking every third cell in each of the x- and y- directions (see Fig. 4), with the appropriate shifts. Then we optimize the points in 9 separate phases, where in each phase the points contained in any of the 9 subsets may be optimized in parallel using **CFPO**. Note that the bucket data structure conforming to the grid G must be updated after each phase.

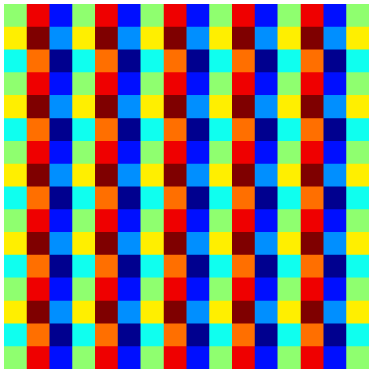


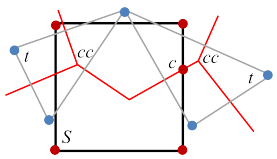
Figure 4: *Partitioning the grid G into $9 = 3 \times 3$ disjoint color-coded subsets of cells.*

3.4. Local constrained farthest point

The core of Algorithm **CFPO** is the local optimization for each point, **LocalCFPO**($Y(x)$,$S(x)$), i.e. find the point inside the constraint square $S(x)$ such that the minimal distance from it to $Y(x)$ is maximized. This is essentially the well-known largest empty circle problem [Tou83] and this optimal point is either a vertex of the Voronoi diagram of $Y(x)$ (i.e. the circumcenter of some triangle in the Delaunay triangulation of $Y(x)$), or the intersection of an edge of the Voronoi diagram of $Y(x)$ with $S(x)$, or some corner of $S(x)$. The complexity of finding the constrained farthest point is $O(m \log m + k \log m)$, where m is the size of $Y(x)$ and k is the number of intersections between $S(x)$ and the edges of the Voronoi diagram, based on the fact that it takes $O(m \log m)$ time to compute the Voronoi diagram or its dual - the Delaunay triangulation. Thanks to the simplicity of the square constraints, we can easily compute the intersection between the Voronoi edges and P, thus reduce the complexity to $O(m \log m)$. To avoid the overhead of maintaining a complex data structure, we use the giftwrapping algorithm of Chand and Kapur [CK70] to compute the local Delaunay triangulation of $Y(x)$. Although its complexity is $O(m^2)$, as opposed to the $O(m \log m)$ complexity of the more modern algorithms, it has a very simple implementation which makes it preferable when the size of the input point set m is very small (our scenario). It is easy to see that for each Delaunay triangle of $Y(x)$, we must check the following conditions on its circumcircle CC: 1. The center of CC is contained in $S(x)$ or it encloses a circle C whose center is on the boundary of $S(x)$, 2. The radius of CC or C increases. Based on these observations, we have the following algorithm:

Algorithm 2: Algorithm LocalCFPO(Y, S)

```
1  rmax ← −1
2  Compute T: the Delaunay triangulation of Y
3  foreach triangle t ∈ T with vertices p1, p2, p3 do
4      (cc, cr) = center and radius of t's circumcircle
5      if cc ∉ S then
6          c ← intersect(dual(t), S)
           // dual(t) is defined as the dual Voronoi edges of
              t's 3 edges. Among the corners of S and the
              intersection points between dual(t) and S, use
              the one closest to cc. (see inset figure)
7          cc ← c
8          r ← min_{i=1,2,3} ||p_i − c||
9          if r < cr then
10             cr ← r
11         else
12             cr ← 0
13     if cr > rmax then
14         rmax ← cr, cfp ← cc
15 return cfp
```



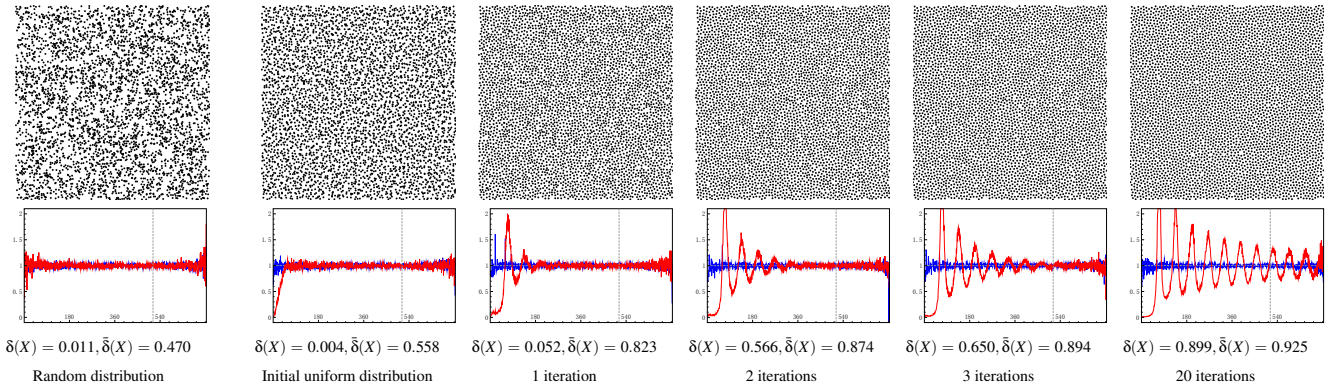


Figure 5: *Constrained farthest point optimization (**CFPO**) of a set of* $4,096$ *points. The second row shows the evolution of the spectrum of the point set, the red curve and the blue curve correspond to the radially averaged power spectra and anisotropy respectively.*

3.5. Point set initialization

It remains to show how to generate an initial point set for Algorithm **CFPO**. The easiest way is simply randomly generate n points in the domain (the unit torus). However in practice, a randomly generated point set is always non-uniformly distributed, some regions are overcrowded while some are too sparse, as shown in Fig. 5. This is bad for a number of reasons. First, it causes **CFPO** to converge slower, since a large proportion of the points need to move a longer distance from their initial positions to their final (limit) positions. Second, it damages the load balance of parallel **CFPO**, since some processors need to optimize many points inside their cells while some processors are idle because of empty cells. Last, but most importantly, this initial point set may lead the algorithm to point distributions violating the local covering property. To avoid all these problems, using the fact that the points in a **FPD** are always uniformly distributed anyway, we exploit the uniform grid cell partition G of the domain for this purpose and randomly generate a very small number of points in each cell. We achieve this by defining G to have approximately $m = 0.8n$ cells in total. One point is randomly located within each cell. The remaining $n-m$ points are randomly located in $n-m$ random cells of G. Thus every cell contains either one or two points. Since L - the size of the grid cell - is chosen to be slightly larger than $\Delta(n)$ (the upper bound on $\delta(X)$, the local covering property will be satisfied in the limit. Fig. 7 shows that this uniform random initialization causes **CFPO** to converge to a **FPD** no slower than **FPO**. Fig. 5 shows the evolution of the **CFPO** algorithm on $4,096$ points.

To summarize, the Convergence Theorem guarantees that **CFPO** converges to a limit. The Local Characterization Theorem then guarantees that this limit is a **FPD**. The Parallelization Theorem furthermore guarantees that the parallel version of **CFPO** will also result in a **FPD**.

4. Experimental Results

In this section, we demonstrate the efficiency of our **CFPO** algorithm and the quality of its results. We compare **CFPO** with **FPO** [SHD11], brute-force dart throwing (**DT**), boundary sampling (**BS**) - a dart throwing variant proposed by Dunbar and Humphreys [DH06], **CCDT** [XLGG11] and **CCVT** [BSD09], whose C++ implementations were kindly provided by the respective authors. Our experiments were run on a PC with an Intel i7-i2720QM @ 2.2GHz 4-core CPU with 8GB RAM and an NVIDIA Quadro 2000M GPU.

As guaranteed by our theorems, the **CFPO** algorithm eventually converges to point sets with the same excellent blue noise properties (**FPD**), exactly as the **FPO** algorithm does. Fig. 6 shows a comparison of the standard spectral measures - power spectrum, radially averaged power spectrum and anisotropy - and the relative minimum distance measures $\delta(X)$ and $\bar{\delta}(X)$ averaged over ten sets of $4,096$ points each. From the results, we can see that **FPO** and **CFPO** produce the best blue-noise distributions among all algorithms as their power spectrum have residual peak decaying slower than all the other algorithms, and **BS** also produces blue-noise distribution with more structured power spectra than brute-force dart throwing. The point distributions produced by **CCDT** and **CCVT** are inferior to **DT** as the residual peaks in their power spectrum decays much faster than that of **DT**. Note that in Step 5 of Algorithm **CFPO**, we can take the containing grid cell as the constraint squares, so that Step 7 can be skipped and the number of phases in parallel **CFPO** can be reduced to 4, thereby reducing the complexity of the algorithm. Since this simplification violates the conditions of the Local Characterization Theorem, the resulting point set is not guaranteed to be a **FPD**, but the last row of Fig. 6 shows that the result of *simplified* **CFPO** still possesses good blue noise properties except for a small peak in the anisotropy at a low frequency.

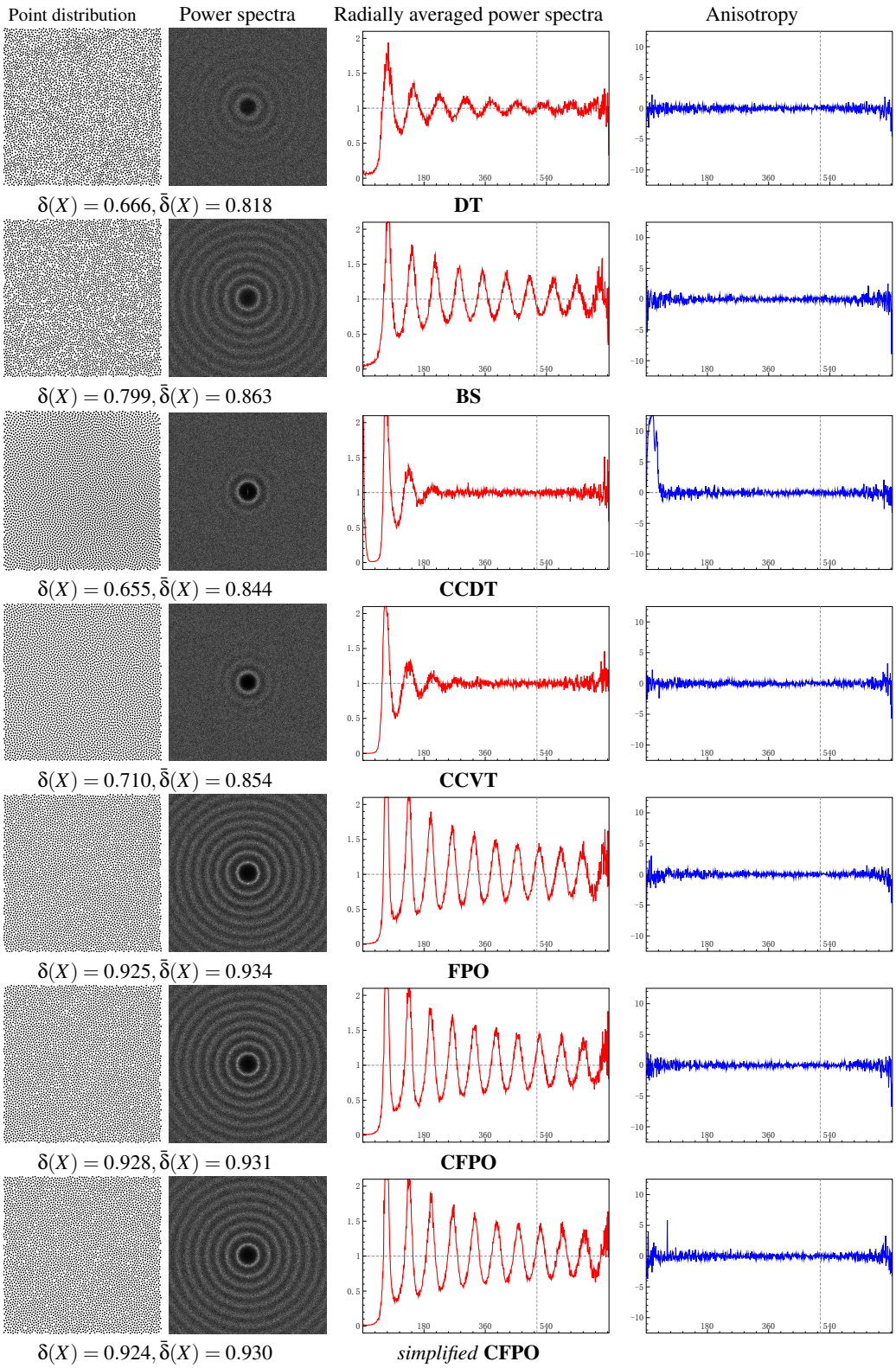


Figure 6: *Spectral analysis point distributions generated with different algorithms. The graphs were averaged over* 10 *sets of* $4,096$ *points.*

We now discuss the relative minimum distance measures proposed by Schlömer *et al* [SHD11]. Like their **LFPO** algorithm, our **CFPO** algorithm moves points only locally, therefore the relative global minimum distance $\delta(X)$ increases slower than in **FPO** in the first few iterations. However, due to the uniform initialization, **CFPO** achieves the same $\delta(X)$ as **FPO** soon after this. And using the same convergence criteria for both algorithms, **CFPO** requires more or less the same number of iterations as **FPO**, albeit with a $8\times$ increase in speed. Fig. 7 shows the convergence of **CFPO** compared to that of **CCDT**, **FPO** and **LFPO**. The results are visually indistinguishable from **FPO**, as shown in Fig. 6. In practice both **CFPO** and **FPO** converge to a *rigid* **FPD** - where each point is a unique farthest point from the remaining point set, although the Local Characterization Theorem does not guarantee this rigidity property. It is easy to see that a *rigid* **FPD** is also a maximal Poisson-disk sample [EDP*11, EMP*12] with radius $\delta(X)$, however the latter possess blue noise spectra of quality similar to dart throwing, which is far inferior to **FPD**. It is interesting to note that after very few, e.g. 2 or 3 iterations, **CFPO** produces distributions with very similar power spectrum to that of dart throwing. Fig. 5 shows the evolution of the power spectrum of the point set when **CFPO** is running.

Fig. 8 shows the runtime of **CFPO** in different configurations, compared to **CCDT**, **FPO** and **LFPO**. It is evident that **CFPO** has linear complexity, which can be derived directly from the fact that the complexity of the constrained farthest point optimization for each point is a constant independent of the size of the entire point set. The graph also shows that **LFPO** actually runs a little slower than **FPO**, and **CFPO** runs $8\times$ times faster than **FPO**.

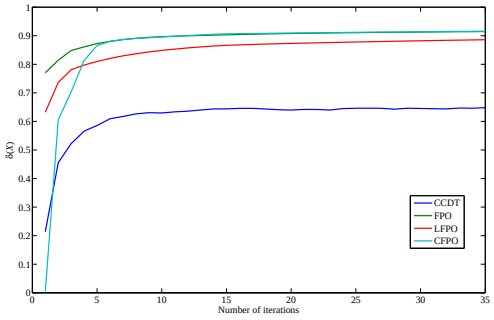


Figure 7: *Convergence of different blue noise sampling algorithms, averaged over* 10 *sets of* $4,096$ *points.*

We parallelized the **CFPO** algorithm on a multi-core CPU using OpenMP [OPE11]. An atomic directive is used to build and update the uniform grid data structure for partitioning the point set. Among all the threads, only the point set X and the grid data structure G is shared. Fig. 8 shows the runtime of **CFPO** using 1 and 4 cores, and Fig. 9 shows the speedup on a different machine containing two Intel Xeon E5420@2.5 GHz 4-core CPUs, for point sets of different

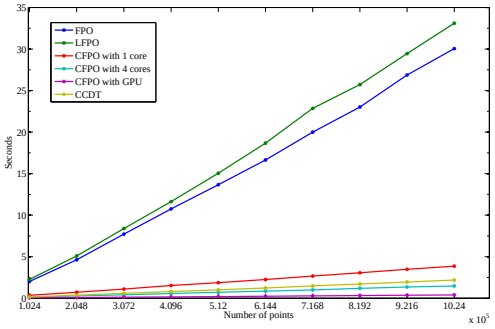


Figure 8: *Runtime of different sampling algorithms.*

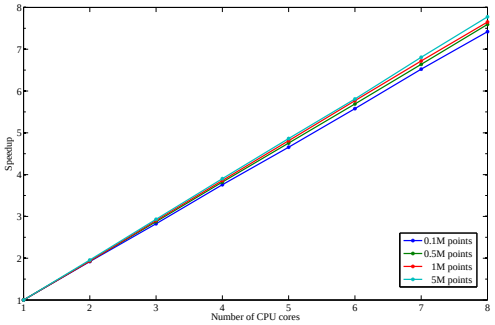


Figure 9: *Speedup of parallel* ***CFPO*** *using multi-core CPU.*

sizes. As can be seen, parallel **CFPO** gives an almost perfect speedup over the sequential version.

Since the **CFPO** algorithm is extremely simple, we implemented a parallel version also on the GPU using CUDA [NVI11]. Fig. 8 shows that parallel **CFPO** on our GPU runs $75\times$ faster than **FPO**. Each iteration of parallel **CFPO** on our NVIDIA Quadro 2000M GPU takes 0.39 seconds for 1M points, while Wei [Wei08] reports 0.25 seconds for the parallel dart-throwing algorithm on the NVIDIA Geforce 8800 GTX and Ebeida *et al*. [EMP*12] report 1.0 second for the parallel maximal Poisson-disk sampling algorithm on the NVIDIA GTX 460. Note that it takes tens of iterations for **CFPO** to converge to **FPD**. However, both parallel dart-throwing [Wei08] and parallel maximal Poisson-disk sampling [EMP*12] produce results with similar quality to that of brute-force dart throwing, which are far inferior to **CFPO**. In situations where speed is critical or a more flat high frequency in the power spectrum, like that produced by dart throwing, is preferred, we can stop **CFPO** after very few (2 or 3) iterations. In this configuration, **CFPO** is 3-5 times slower than parallel dart throwing [Wei08], and is close to parallel maximal Poisson-disk sampling [EMP*12] .

Like **FPO**, there are some stable but regular configurations for **CFPO**, such as the square grid and regular hexagonal grid. In these configurations, each point is a farthest point

from the remaining points. However, since **FPO** and **CFPO** start from an essentially random initial distribution and use a similar optimization scheme for the objective function $\delta(X)$, where each time $\delta(X)$ is maximized by optimizing two variables (the coordinates of the active point) while keeping all the other variables (the remaining points) fixed, **FPO** and **CFPO** in practice always converge to irregular local maximums of $\delta(X)$.

5. Conclusion

Sampling plays a fundamental role in many computer graphics applications. In this paper, we proposed a very simple, yet efficient, method for generating point distributions with excellent blue noise spectra. The method is based on constrained farthest point optimization, where each point is iteratively moved to the constrained local farthest point. This is shown to be equivalent to an algorithm which does exactly the same, but globally, thus is optimal. The locality allows for very easy parallelization.

The main limitation of the **FPD** framework is that it is restricted to uniform point densities. Therefore neither **FPO** nor our **CFPO** algorithms can be directly extended to non-uniform point distributions.

Future work includes generalization to higher dimensions and to non-Euclidean metric spaces such as manifold surfaces.

6. Acknowledgements

This research project was financially supported by the state of Lower-Saxony and the Volkswagen Foundation, Hannover, Germany. R. Chen is partially supported by the Ali Kaufmann postdoctoral fellowship at the Technion.

References

[BSD09] BALZER M., SCHLÖMER T., DEUSSEN O.: Capacity-constrained point distributions: a variant of Lloyd's method. *ACM Trans. Graph. 28* (July 2009), 86:1–86:8. 2, 6

[CK70] CHAND D. R., KAPUR S. S.: An algorithm for convex polytopes. *J. ACM 17*, 1 (Jan. 1970), 78–86. 5

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

[DH06] DUNBAR D., HUMPHREYS G.: A spatial data structure for fast Poisson-disk sample generation. *ACM Trans. Graph. 25*, 3 (July 2006), 503–508. 6

[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 Trans. Graph. 30*, 4 (Aug. 2011), 49:1–49:12. 2, 8

[EMP*12] EBEIDA M. S., MITCHELL S. A., PATNEY A., DAVIDSON A., OWENS J. D.: A simple algorithm for maximal Poisson-disk sampling in high dimensions. *Comput. Graph. Forum 31*, 2 (May 2012). 2, 8

[LD06] LAGAE A., DUTRÉ P.: An alternative for Wang tiles: colored edges versus colored corners. *ACM Trans. Graph. 25*, 4 (2006), 1442–1459. 2

[LD08] LAGAE A., DUTRÉ P.: A comparison of methods for generating Poisson disk distributions. *Comput. Graph. Forum 27*, 1 (2008), 114–129. 2

[Llo82] LLOYD S. P.: Least squares quantization in PCM. *IEEE Trans. on Inf. Theory 28*, 2 (1982), 129–136. 2

[LNW*10] LI H., NEHAB D., WEI L.-Y., SANDER P. V., FU C.-W.: Fast capacity constrained Voronoi tessellation. In *Proc. Symp. on Interactive 3D Graphics and Games* (2010), I3D '10, ACM, pp. 13:1–13:1. 2

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

[NVI11] NVIDIA CORPORATION: *CUDA Programming Guide 4.0*. 2011. 8

[OPE11] OPENMP ARB: *OpenMP API Specifiications for Parallel Programming*. 2011. 8

[SCM00] SHADE J., COHEN M. F., MITCHELL D. P.: Tiling layered depth images. pp. 231–242. 2

[SD95] SU P., DRYSDALE R. L. S.: A comparison of sequential Delaunay triangulation algorithms. In *Proc. Symp. Computational Geometry* (1995), SCG '95, ACM, pp. 61–70. 3

[SHD11] SCHLÖMER T., HECK D., DEUSSEN O.: Farthest-point optimized point sets with maximized minimum distance. In *Proc. SIGGRAPH Symp. High Performance Graphics* (2011), HPG '11, ACM, pp. 135–142. 1, 2, 3, 4, 6, 8, 11

[Tou83] TOUSSAINT G.: Computing largest empty circles with location constraints. *Int. J. Comput. Inf. Sci. 12*, 5 (1983), 347–358. 5

[Uli87] ULICHNEY R.: *Digital halftoning*. MIT Press, 1987. 1

[Wei08] WEI L.-Y.: Parallel Poisson disk sampling. *ACM Trans. Graph. 27* (August 2008), 20:1–20:9. 2, 8

[XLGG11] XU Y., LIU L., GOTSMAN C., GORTLER S. J.: Capacity-constrained Delaunay triangulation for point distributions. *Comput. Graph. 35* (June 2011), 510–516. 2, 6

Appendix A

Local Characterization Theorem. Let X be a point set on the unit torus R, and $C(z,r)$ be the *closed* disk centered at z with radius r, then X is a **FPD** with distance $\delta = \delta(X)$ if and only if $\forall x \in X, \exists D = D(x) \subseteq R$, such that

Property 1 (*Local covering property*):

$$\bigcup_{z \in D \cap X} C(z,\delta) \supseteq S$$

Property 2 (*Local farthest point property*): x is the farthest point of $Y(x)$ in S

where $Y(x) = D \cap (X \backslash \{x\})$ is the subset of $X \backslash \{x\}$ inside D, $S = S(x) = D(x) \ominus \delta$, and $\ominus$ is the erosion operator.

Proof. Let $d(\cdot)$ be the distance function between two point sets.

The "only if" direction:

First we prove the *global covering property* of a **FPD**.

$$R = \bigcup_{x\in X} C(x,\delta)$$

where R is the domain, i.e. the unit torus.

If $\exists p \in R$ such that $p \notin \bigcup_{x\in X} C(x,\delta)$, then $\forall x \in X$, $p \notin C(x,\delta) \Rightarrow \|p-x\| > \delta \Rightarrow d(p,X) > \delta$. Then for any $y \in X$, such that $d(y,X\backslash\{y\}) = \delta$, we have $d(y,X\backslash\{y\}) = \delta < d(p,X) \le d(p,X\backslash\{y\})$. Thus y does not have the farthest point property in X, contradicting the fact that X is a **FPD**.

To prove properties 1 and 2, we define $D = D(x)$ to be $\hat{C}(x,2\delta)$, the *open* disk centered at x with radius 2δ. Now $\forall p \in S$, we have

$$\left.\begin{array}{c} p \notin C(y,\delta), \forall y \in X \cap (R\backslash D) \\ p \in R = \bigcup_{y\in X} C(y,\delta) \end{array}\right\} \Rightarrow p \in \bigcup_{y\in X\cap D} C(y,\delta)$$

This proves property 1.

If X is a **FPD** then $\forall p \in C(x,\delta) \subseteq R, d(p,X\backslash\{x\}) \le d(x,X\backslash\{x\})$, and equality holds when $p = x$.

For any $p \in S$ such that $\|p-x\| > \delta$, according to property 1, $d(p,Y(x)) \le \delta$.

$$\begin{aligned} &\Rightarrow \forall p \in S, d(p,Y(x)) \\ &\le \max\{\max_{p\in S, p-x>\delta} d(p,Y(x)), \max_{p\in S, p-x\le\delta} d(p,Y(x))\} \\ &\le \max\{\delta, \max_{p\in C(x,\delta)} d(p,Y(x))\} \le d(x,X\backslash\{x\}) \end{aligned}$$

and equality holds when $p = x$.

By definition, $x \in D$ is the farthest point of $Y(x)$ among all points of S.

The "if" direction:

Now we assume properties 1 and 2, and prove that X is an **FPD**.

First we prove that property 1 implies the global covering property:

$$R = \bigcup_{x\in X} S(x)$$

Assume $\exists q \in R$ such that $q \notin \bigcup_{x\in X} S(x)$

Let $x \in X$ such that $d(q,X) = \|q-x\|$, and let $p \in \overline{qx} \cap C(x,\delta+\varepsilon)$, where $\varepsilon > 0$. Then on the one hand, using property 1,

$$\begin{aligned} &d(p,Y(x)) \le \delta \\ &\Rightarrow \exists y \in Y(x), \|p-y\| \le \delta \\ &\Rightarrow \|q-y\| \le \|q-p\| + \|p-y\| \\ &\le \|q-x\| - (\delta+\varepsilon) + \delta = \|q-x\| - \varepsilon < \|q-x\| \end{aligned}$$

On the other hand, since $y \in Y(x) \subseteq X$, then $\|q-x\| = d(q,X) \le \|q-y\|$, which is a contradiction.

Now $\forall x \in X$, the global covering property implies

$$\begin{aligned} &\forall p \notin S(x), p \in \bigcup_{y\in X\backslash\{x\}} S(y) \\ &\Rightarrow \exists y \in X\backslash\{x\}, p \in S(y) \xrightarrow{property1} d(p,Y(y)) \le \delta \\ &\Rightarrow d(p,X\backslash\{x\}) \le d(p,Y(y)) \le \delta \end{aligned}$$

$$\forall p \in S(x), \begin{cases} \|p-x\| > \delta \xrightarrow{property1} d(p,Y(x)) \le \delta \\ \|p-x\| \le \delta \xrightarrow{property2} d(p,Y(x)) \le d(x,Y(x)) \end{cases}$$

and $d(x,Y(x)) \ge d(x,X\backslash\{x\}) \ge \delta$, therefore,

$$\begin{aligned} d(p,X\backslash\{x\}) &\le \max\begin{cases} \max_{p\in S(x), \|p-x\|>\delta} d(p,Y(x)) \\ \max_{p\in S(x), \|p-x\|\le\delta} d(p,Y(x)) \end{cases} \\ &\le \max\{\delta, d(x,Y(x))\} = d(x,Y(x)) \end{aligned}$$

Thus $\forall p \in R$,

$$\begin{aligned} d(p,X\backslash\{x\}) &\le \max\begin{cases} \max_{p\notin S(x)} d(p,X\backslash\{x\}) \\ \max_{p\in S(x)} d(p,X\backslash\{x\}) \end{cases} \\ &\le \max\{\delta, d(x,Y(x)\} = d(x,Y(x)) \end{aligned}$$

equality holds when $p = x$, so x is the farthest point of $X\backslash\{x\}$, implying that X is a **FPD**. □

Appendix B

Convergence Theorem. Let X be a planar point set with minimal distance $\delta = \delta(X)$. Partition the domain into a uniform grid G of square cells of size $L > \delta$. Denote by $G(x)$ the cell containing x. If we replace any point $x \in X$ with another point x' inside or no further than $L-\delta$ away from $G(x)$ in both axis-aligned directions, such that the minimal distance from x' to all the other points inside the block of 3×3 cells centered at $G(x)$ does not decrease, then $\delta(X)$ does not decrease.

Proof. Let $Y \subset X$ be the subset of the points, excluding x, contained in the 3×3 cells centered $g = G(x)$. As shown in Fig. 10, we are given that

$$\begin{cases} |x'-g|_x \le L-\delta \\ |x'-g|_y \le L-\delta \end{cases}$$

and

$$d(x',Y) \ge d(x,Y)$$

where the distance between a point and a cell is defined to be 0 if the cell contains the point, otherwise the distance from the point to the cell boundary, and $|\cdot|_x$ and $|\cdot|_y$ denote distance in x and y directions, respectively. Then $\forall p \in X\backslash\{Y\cup\{x\}\}$,

$$|x'-p| \ge \min\{|x'-p|_x, |x'-p|_y\} \ge L-(L-\delta) = \delta$$

therefore

$$\begin{aligned} d(x',X\backslash\{x\}) &= \min\{d(x',Y), d(x',X\backslash(Y\cup\{x\}))\} \\ &\ge \min\{d(x,Y),\delta\} = \delta \end{aligned}$$

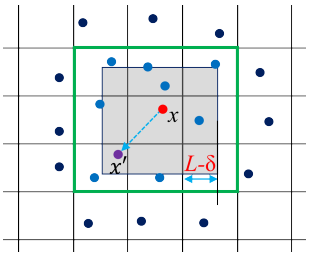


Figure 10: *Proof of Convergence Theorem.*

Let X' be the new point set after x' replaces x, and $\delta' = \delta(X')$. Then,

$$\begin{aligned}\delta' &= \min_{p,q\in X', p\neq q} |p-q| \\ &= \min\{\min_{p,q\in X'\setminus\{x'\}, p\neq q} |p-q|, \min_{p\in X'\setminus\{x'\}} |x'-p|\} \\ &= \min\{\min_{p,q\in X\setminus\{x\}, p\neq q} |p-q|, \min_{p\in X\setminus\{x\}} |x'-p|\} \\ &\geq \min\{\delta,\delta\} = \delta\end{aligned}$$

□

Appendix C

Parallelization Theorem. Let X be a set of n of points located on the unit torus. Partition the domain into a uniform grid G of square cells of size L. If L is such that $L > \Delta(n)$ and a subset P of points of X are (sequentially) moved to new positions while satisfying the conditions of the Convergence Theorem taking $\delta = \Delta(n)$, then $\delta(X)$ cannot decrease as a result. If P is such that all points in P are at least three cells distant from each other (in each of the axis directions), then they can move to their new positions in parallel without increasing $\delta(X)$.

Proof. It is well known [SHD11] that for any point set X containing n points, $\delta = \delta(X) < \Delta(n) = \Delta$. Thus choosing L such that $L > \Delta$ and using a strip of width $L - \Delta$ instead of $L - \delta$ around the cell guarantees that $L > \delta$ and $|x' - g| \leq L - \delta$, no matter what δ is in practice. So applying this reasoning to each point replacement in turn inductively implies that δ cannot decrease.

For the parallel movement case, since $\max\{|x'-g|_x, |x'-g|_y\} \leq L - \delta \leq L$, each point is constrained to move inside the 3×3 neighboring cells only. If we consider the subsets of points contained in 3×3 neighborhoods centered at cells which are three cells distant from each other (e.g. the monochromatic cells in Fig. 4), then these subsets are all disjoint and each point in their union maintains the same neighborhood set Y despite the movement of any other point in the union. This allows all such points to move to new positions in parallel without affecting each other. □