社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2008−CG−133　(17)
2008／11／8

ドロネー三角形分割を用いた高品質の ブルーノイズ特性を持った点分布の生成

Zoltan Szego[†]　　金森　由博[†]　　西田　友是[†]

Blue noise 特性を持つサンプリングパターンは CG 分野で応用範囲が広いが、それを効率的に生成するのは難しい。本研究では blue noise 特性を持った点集合を前処理なしで生成する決定論的な手法を提案する。提案手法は点集合のドロネー三角形分割を行い、三角形の外接円の中で最大のものを見つけ、その中心点に点を加えることを繰り返してサンプリングを行う。また、サンプル数の事前指定、密度分布に応じた適応的サンプリング、3 次元曲面上でのサンプリングが可能である。

Generation of High-Quality Blue Noise Distributions based on Delaunay Triangulation

Zoltan Szego,[†] Yoshihiro Kanamori[†] and Tomoyuki Nishita[†]

Sampling patterns with a blue noise distribution are widely used in many areas of computer graphics, yet their efficient generation remains a difficult problem. We propose a method to generate point sets with a blue noise distribution using a deterministic algorithm with no preprocessing. We insert each new sample at the center of the largest empty circle in the point set, which is obtained by calculating the Delaunay-triangulation of the set and finding the triangle with the largest circumcircle. Our method supports adaptive sampling according to a user-specified density function, as well as specifying the exact number of required samples. It can also be extended to perform sampling on a three-dimensional curved surface.

1. Introduction

Sampling is a fundamental part of many graphics applications such as image processing or rendering. With the number of samples allowed often being limited in these applications, getting the best-quality results out of as few samples as possible is essential. For this purpose, one of the most often used sampling patterns are *Poisson Disk* patterns, defined by a minimum distance between all neighboring sample points. These have the spectral properties of *blue noise*, defined as having a spectrum with low energy in the low frequencies. This property has been shown to be very effective at providing good sampling with a low amount of noise or aliasing, as evidenced by the fact that it also occurs naturally, for example, in the placement of photoreceptor cells in the eye[11].

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo

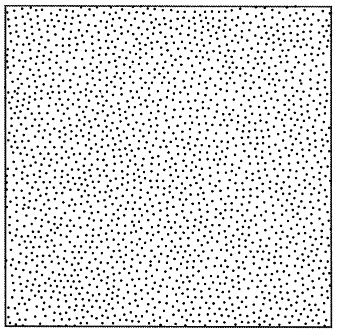

Fig. 1　2,000 sample points with a blue noise distribution

The traditional approach to generating blue noise sampling patterns involves random dart throwing, which is a time consuming process. There are many alternative approaches and ways to accelerate the process, some of which we will introduce in Section 2. The general trend, however, is that of a tradeoff between speed and quality, as well as precomputing large chunks of data in advance. Our method, while slower than some of the alternatives,

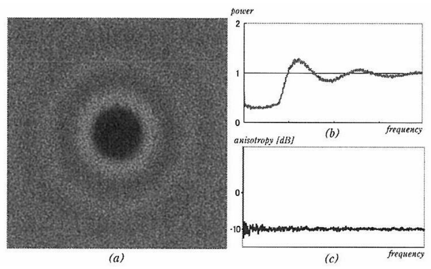

Fig. 2 (a) A typical blue noise power spectrum. The graphs on the right represent (b) the radially averaged power and (c) anisotropy.

produces point sets with high-quality blue noise distributions without any precomputation, using a geometrically-based algorithm to determine the exact position for each new sample.

For many applications, the requirement to perform importance sampling often arises. For example, a renderer that uses an environment map as a light source will need to sample the map for light colors at relevant points - the brighter areas of the map, requiring an algorithm that adaptively samples the map based on some importance function, in this case, brightness. Our method supports adaptive sampling with a simple modification to the algorithm.

2. Related Work

2.1 Quality evaluation

Before introducing other related methods, we will first explain how the quality of a blue noise distributed point set is evaluated, as detailed in[6]. We assume the samples are generated on a two-dimensional domain $[0,1)^2$. We can consider N point samples x_i as a signal given by the sum of Dirac-deltas (δ) located at the position of each point:

$$\sum_{i=1}^{N} \delta(x - x_i). \tag{1}$$

Taking the Fourier-transform of this signal gives the spectrum for the point set, and averaging many of these gives the characteristic power spectrum $P(f)$ for the method used, with f being the frequency. The typical blue noise spectrum (as shown in Figure 2) has a low-frequency band surrounding the center, followed by a sharp transition to higher energies, finally flattening out for high frequencies. The spectrum is radially symmetric, therefore it can be represented by a one-dimensional *radially averaged* power spectrum by averaging $P(f)$ in concentric rings corresponding to frequencies:

$$P_r(f_r) = \frac{1}{N(f_r)} \sum_{i=1}^{N(f_r)} P(C(i)), \tag{2}$$

where $N(f_r)$ is the number of samples that are taken from the spectrum in each ring with radius f_r. $C(i)$ gives the position for the i^{th} sample in the ring. The variance of the power for each frequency, $s^2(f_r)$, is used to derive the *anisotropy* of the spectrum:

$$A_r(f_r) = \frac{s^2(f_r)}{P_r^2(f_r)}, \tag{3}$$

where

$$s^2(f_r) = \frac{1}{N(f_r)} \sum_{i=1}^{N(f_r)} (P(C(i)) - P_r(f_r))^2. \tag{4}$$

High-quality blue noise spectra are characterized by a wide central low-energy ring and low anisotropy (around -10dB, like in Figure 2).

2.2 Methods based on dart throwing

Dart throwing, introduced by Cook[2)], generates Poisson disk distributions by first generating uniformly distributed points, then rejecting points that are too close to each other given a minimum separation distance. While easy to implement, this algorithm is slow and difficult to control since a minimum distance needs to be given instead of the required number of samples. Dart throwing has been optimized and extended in a number of ways, one of which is called *relaxation dart throwing*[8)], where the minimum separation distance starts out large and is gradually reduced, producing hierarchical samples somewhat faster than traditional dart throwing. A method called *Lloyd's relaxation*[7)] is often applied to these results as a post-process, which is a rather costly iterative process that constructs the Voronoi-diagram for the entire point set repeatedly.

In 2008, Wei[10)] proposed a method to generate blue noise distributed samples at high speed on highly parallel hardware, i.e. programmable GPUs, based on the observation that areas of the sampling domain that are far enough apart do not affect each

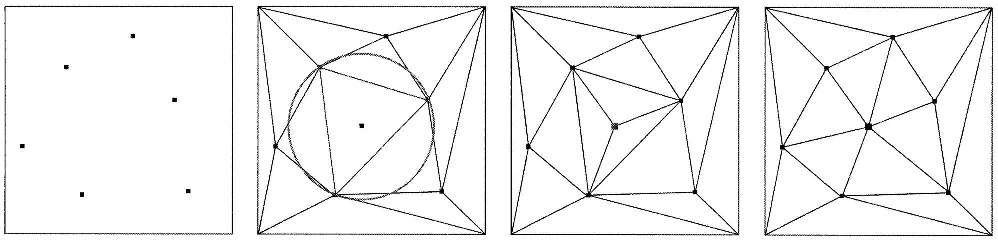
Fig. 3 Overview of a step of our algorithm. Left to right: an initial set of points, their Delaunay triangulation with the largest empty circle, the new sample point immediately after being added, and the updated triangulation.

other, and can therefore be handled independently in parallel. The method produces good quality results at high speed thanks to the high performance provided by GPUs, however, it is difficult to control the exact number of produced samples due to the random nature of the algorithm. Our method can deterministically generate any given number of samples.

2.3 Tile-based methods

Methods based on various tiling schemes use a different approach than dart throwing. Instead of relying on randomness, they generally precalculate distributions for a given number of predefined tiles, and then arrange those on the sampling domain, allowing for better performance at the expense of quality and precomputation time.

For example, *Wang tiles* are used by various methods. Hiller et al[4] introduced the use of Wang tiles for generating point sets, however, the power spectrum of the method is of low quality. Kopf et al[5] used Wang tiles recursively to produce hierarchical sampling patterns at high speed from a large precomputed dataset, but the spectrum still has unwanted spikes and noise.

Ostromoukhov[9] proposed a different tiling scheme based on hierarchically subdivided polyominoes. The method produces reasonably good results at high speed, however, it still requires a rather complicated and expensive precomputing step. The variety and randomness of the results are limited by the number of pregenerated tiling variations.

Our method produces point sets with high-quality spectra, and is not limited in the variety of outcomes by any kind of precomputed data. Note that our method does not fit into the two categories presented in this section, it should instead be classified as a geometrically-based method that selects the exact position for each sample point.

3. Algorithm

3.1 Overview

Our method generates samples deterministically and sequentially. Each new sample is placed at the center of the *largest empty circle* in the already existing set of samples. The intuitive reasoning for the algorithm used is that given a set of samples, the next one should go in the most sparsely sampled area so far, and should be equidistant from the other samples surrounding it. Finding the largest empty circle is equivalent to finding the triangle with the largest circumcircle in the *Delaunay triangulation* of the point set, a triangulation which guarantees by definition that no other point lies within any triangle's circumcircle. The process is repeated, gradually filling up the sampling domain with points.

See Figure 3 for a graphical overview of the algorithm. Initially, the input point set consists of a few *seed points*, which serve as a means of providing deterministic control over the pseudorandom outcome of the process. The procedure to generate new points are as follows:

```
// N : the total number of samples required
// Points : the vector of sample points
// (initialized with a small arbitrary set of seed points)
Delaunay := InitializeDelaunayTriangulation(Points);
while (Points.length < N)
  C := Delaunay.findLargestCircumcircle();
  Points.append(C.center);
  Delaunay.insertPointUpdateTriangulation(C.center)
end
```

In order to find the largest circumcircle quickly, in addition to the Delaunay triangulation, we maintain a heap data structure containing all of the

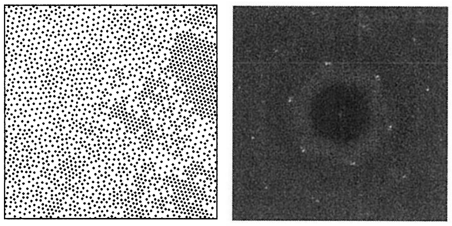
Fig. 4 Example of the appearance of undesired regular patterns mentioned in Section 3.2. The pattern in the generated points (left) causes hexagonally arranged spikes in the spectrum (right).

triangles, sorted by the radius of their circumcircle. The largest triangle can therefore be obtained in constant time, and updates to the triangulation (such as inserting and modifying triangles) happen in $O(\log N)$. Triangles with a circumcircle whose center lies outside of the domain $[0,1)^2$ are excluded from the search.

3.2 Avoiding regular patterns

Running the above algorithm as-is, it produces results that are often uneven, with noticeable large areas of regular patterns that show up as spikes on the power spectrum. See Figure 4 for an example. The reason these patterns emerge is because of the way the algorithm behaves for equilateral triangles. If the Delaunay triangulation happens to include a cluster of equilateral triangles sometime during the process, and one of them is selected as the one with the largest empty circle, the new point will form smaller equilateral triangles, with further subdivision of the area only preserving the pattern. The result is a regular triangular grid, obviously unsuitable for blue noise sampling.

To prevent this, an additional step is included in the algorithm: when inserting a new point into the Delaunay triangulation, we first check if the triangle containing the point is close to being equilateral up to a certain threshold. If it is, the new point is first offset by a small amount, such that it stays within its triangle, in a pseudorandom direction (depending only on the point's original position, in this case, to keep our algorithm deterministic). This modification eliminates the formation of regular patterns and fixes the occasional spikes in the power spectrum.

3.3 Adaptive sampling

With a simple modification, the proposed method can also perform adaptive sampling based on a user-specified importance function $f : [0,1)^2 \rightarrow [0,1]$, that specifies how dense the samples should be in an area. For all of the triangles in the heap data structure mentioned in Section 3.1, the radius of the circumcircle that acts as the sort key is weighted by a sample from the importance function, taken at the center of the circumcircle. This way, triangles that lie in areas requiring more thorough sampling are given more priority, resulting in sample points that are distributed according to the importance function.

3.4 Sampling on 3D curved surfaces

Our method can be extended to generate samples on three-dimensional surfaces. The basic process is mostly the same: starting from a sparse set of points and their Delaunay triangulation on the surface, new sample points are added in triangles with the largest circumcircle. The new points are projected onto the original surface and the triangulation is updated.

In case the surface is specified by a complex polygonal mesh, the initial set of points and their connectivity can be obtained with a mesh simplification method such as edge decimation[3]. Adaptive sampling is also possible if an importance function is defined on the surface.

4. Results

The environment used for our experiments was a desktop PC with an Intel Core 2 X6800 CPU running at 2.9 GHz and 2GB of memory.

We compared the quality and speed of our implementation to two other techniques: a tile-based technique (recursive Wang tiles[5]) and a dart throwing-based one (Curl noise[1]). See Figures 7 and 8 for the spectral analysis of the results for the case of 20,000 and 50,000 samples. Table 1 shows our speed measurements.

Table 1 Time to generate a given number of samples

Method	Number of samples		
	20,000	50,000	100,000
Proposed	348.25 ms	1057.98 ms	2716.16 ms
Curl noise	182.25 ms	403.69 ms	810.12 ms
Wang tiles	2.01 ms	5.14 ms	10.28 ms

Recursive Wang tiles performed several orders of

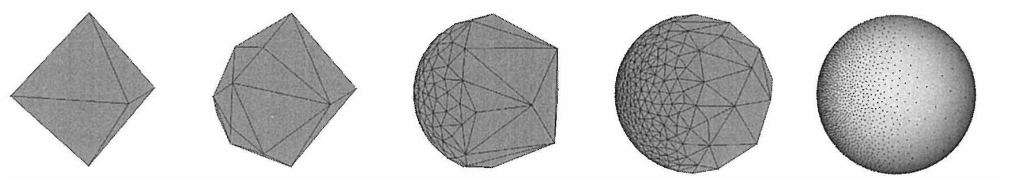

Fig. 5 Adaptive sampling on the surface of a sphere. From left to right: the initial sample points and their connectivity, the 3D Delaunay mesh after adding a few samples, two additional intermediate states, and the final result of 2,500 samples shown as points on the sphere. The importance function used concentrates samples on one side, demonstrating, for example, the sampling of an environment map with one bright light only.

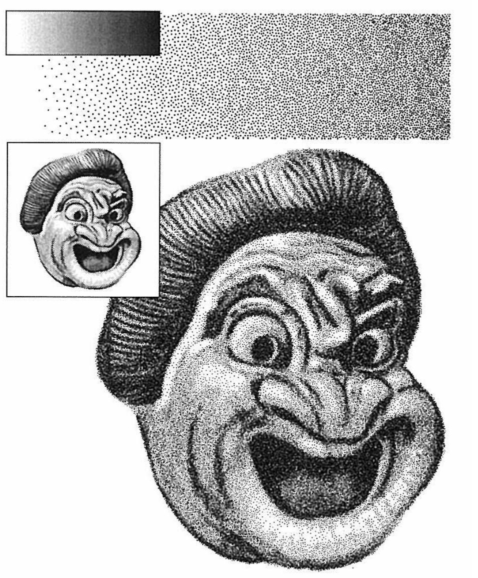

Fig. 6 Halftoning: a linear gradient drawn with 4,000 points, and a greyscale image drawn with 100,000 points. The source images are shown in boxes.

magnitude faster than the alternatives, however, at the expense of quality. The larger the number of samples required, the noisier the output, as seen in the spectrum and anisotropy graphs in Figure 8. Our method, while slower than Curl noise, produced slightly better quality results, based on the wider inner ring of the power spectrum.

Figure 5 shows the process of sampling on a 3D surface, in this case, a sphere. The starting points for the algorithm are the vertices of an octahedron. To demonstrate adaptive sampling as well, as a simple example, the cosine of the central angle on the sphere between sample points and a fixed point is used as the importance function to increase the density of samples on one side of the sphere.

Finally, Figure 6 shows two examples of adaptive sampling on a 2D domain, demonstrated by point-based halftoning of grayscale input images. A simple linear gradient and a more complex bitmap image are used as the importance functions.

5. Conclusion and Future Work

We described a method to generate sampling patterns with a high quality blue noise distribution. Our method is deterministic, produces the exact number of required samples, requires no precomputation, and handles adaptive sampling based on a user-specified importance function.

For future work, we would like to speed up the proposed method via parallelization: after a certain density of samples has been reached, smaller regions of the domain can be updated without affecting each other, so the process should be able to run in parallel. We would also like to try extending the sampling domain to three dimensions, using Delaunay-tetrahedrization to find largest empty spheres in $\mathbf{R}^3$.

References

1) R. Bridson, J. Houriham, and M. Nordenstam. Curl-noise for procedural fluid flow. *ACM Trans. Graph.*, 26(3):46, 2007.
2) R. L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986.
3) M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97*, pp. 209–216, 1997.
4) S. Hiller, O. Deussen, and A. Keller. Tiled blue noise samples. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pp. 265–272. Aka GmbH, 2001.
5) J. Kopf, D. Cohen-Or, O. Deussen, and D. Lischinski. Recursive wang tiles for real-time blue noise. In *SIGGRAPH '06*, pp. 509–518, 2006.
6) A. Lagae and P. Dutré. A comparison of methods for generating Poisson disk distributions. Report CW 459, Department of Computer Science, K.U.Leuven,

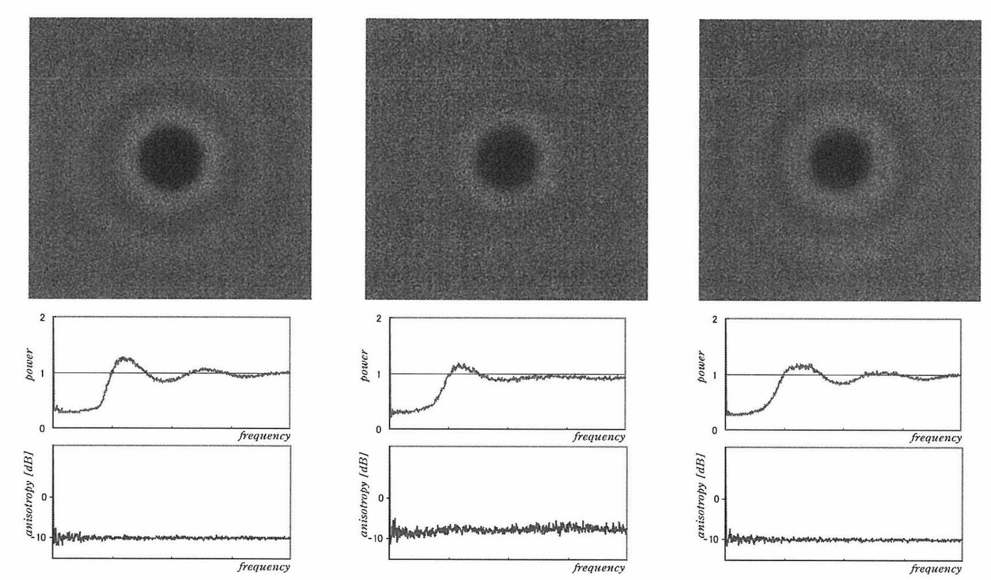

Fig. 7 Comparison with other techniques for 20,000 sample points. From left to right: the present method, recursive Wang tiles, and Curl Noise. Each column contains, from top to bottom: the power spectrum, the radially averaged power graph and the anisotropy graph.

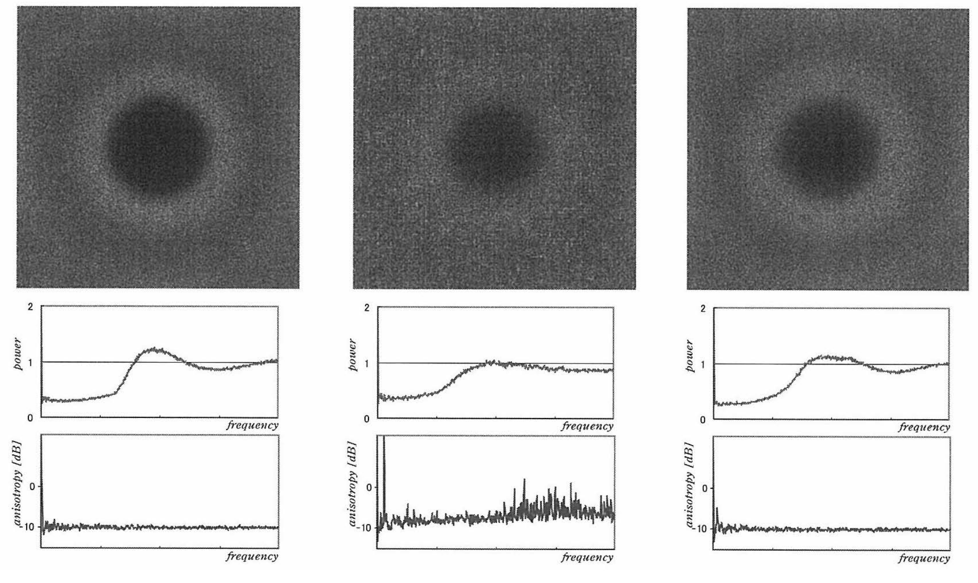

Fig. 8 Comparison with other techniques for 50,000 sample points. From left to right: the present method, recursive Wang tiles, and Curl Noise. The graphs are laid out as in Figure 7

Leuven, Belgium, August 2006.

7) S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982.

8) M. McCool and E. Fiume. Hierarchical poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface '92*, pp. 94–105, 1992.

9) V. Ostromoukhov. Sampling with polyominoes. *ACM Trans. Graph.*, 26(3):78, 2007.

10) L.-Y. Wei. Parallel poisson disk sampling. *ACM Trans. Graph.*, 27(3):1–9, 2008.

11) J. Yellott, JI. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science*, 221(4608):382–385, 1983.