Fast Adaptive Blue Noise on Polygonal Surfaces

Esdras Medeiros\textsuperscript{a,}\textsuperscript{*}, Lis Ingrid\textsuperscript{b}, Sinésio Pesco\textsuperscript{b}, Claudio Silva\textsuperscript{c}

\textsuperscript{a}Departamento de Matemática, Fortaleza, CE, Brazil
\textsuperscript{b}Departamento de Matemática, PUC-Rio de Janeiro, Brazil
\textsuperscript{c}Polytechnic Institute of NYU, Brooklyn, NY, USA

Abstract

This paper proposes a novel method for the computation of hierarchical Poisson disk samplings on polygonal surfaces. The algorithm generates a pointerless hierarchical structure such that each level is a uniform Poisson disk sampling and a subset of the next level. As the main result, given a dynamically-varying importance sampling function defined over a surface, the hierarchy is capable of generating adaptive samplings with blue noise characteristics, temporal-coherence and real-time computation. Classical algorithms produce hierarchies in tight ratios, which is a serious bottleneck specially for a large number of samples. Instead, our method uses sparse ratios and decreases the adaptation error of the hierarchy through a fast optimization process. Therefore, we save a considerable amount of time (up to 74\% in our experiments) while preserving the good blue noise properties. We present applications on Non Photo Realistic rendering (NPR), more specifically, on surface stippling effects. First, we apply our method by taking illumination to be the importance sampling to shade the surface, and second, we dynamically deform a surface with a predefined stippled texture.

Keywords: Poisson Disk Sampling, Hierarchy, Adaptation, Importance Sampling.

1. Introduction

For a variety of fundamental problems in computer graphics areas, including rendering, imaging, and geometry processing, point sampling is an important step. Over the past decade, many different sampling patterns have been studied and analyzed. Among them, blue noise was identified as one of the best sampling patterns because it has unique spatial and spectrum property. In essence, blue noise produces better results by replacing low frequency aliasing with high frequency noise. The studies on blue noise are inspired in the works of Yellott [31, 32], who found that the photo-receptors in the retina of the eye follow the blue noise distribution. Because this indicates that this sampling pattern is effective for imaging applications, synthesizing blue noise becomes very attractive.

A large volume of work investigating the generation of uniformly distributed blue noise sample patterns on the plane and non-plane domains exists. See Lagae and Dutrée [14] for a good introduction to the topic of plane domains. However, the problem of efficiently generating adaptive blue noise samplings on surfaces when the importance sampling varies dynamically is still unsolved. To our knowledge, this is the first work to focus on this issue.

Poisson disk distributions were introduced to solve the aliasing problem in computer graphics [9] and they have excellent blue noise characteristics. This paper describes a method for generating a hierarchy of uniform Poisson disk samplings on polygonal surfaces. The method also produces adaptive samplings having blue noise properties (see Figure 1).

Related methods that construct a hierarchy of points use additional pointers to store dependencies between the points [23]. However, using our method, each point in our hierarchy stores only the local geometric radius. Hence, there is no need for additional pointers. Then, to construct an adaptive sampling, we sweep linearly an array of points and perform a simple comparison between the geometric radius and the importance sampling function (see Section 4.5). This is done very efficiently via shaders which allow the precomputed samples to be repeatedly used for a variety of importance sampling functions.

For a large number of samples, generating the hierarchy requires a considerable amount of time especially when the radii ratios between successive levels is tight (close to 1). To alleviate this problem, our method employs sparse ratios and decreases the adaptation error of the hierarchy through a fast optimization process (Section 4.3).

The paper is organized as follows. In Section 2, we present related work and state our main contributions. In Section 3 we define Poisson disk samplings and hierarchical samplings on surfaces. In Section 4, the algorithms and implementation details are discussed. Section 5 shows the applications. In Section 6 we discuss the performance of the algorithms and applications. In addition, we analyze the quality of the adaptive samplings using differential domains. We conclude the paper in Section 7.
2. Related Work and Contribution

We briefly review related work on blue noise distributions. We state our contributions and compare them with previous methods.

2.1. Related work on Blue Noise

**Planar domains:** The most natural way for generating a Poisson disk distribution is the dart throwing technique [7]. Basically, this strategy consists in shooting random points on the domain and reject those which do not satisfy the minimum distance with the already validated points. Traditional dart throwing algorithms have been considered inefficient until recent studies [4, 11, 12]. Because of the localized property of the minimum distance between sample points, the main approach to overcome this problem explores spatial data structures. For example, Bridson [4] constructs a uniform grid in which each cell can contain at most one sample. Then, the conflicting test of the minimum distance is accelerated because this constraint can happen only between neighboring cells.

Relaxation-based methods introduce additional topological information into the point pattern. Starting with some initial sampling, these algorithms iteratively move the point positions so that the new distribution minimizes a given energy. One of the simplest approach is the Lloyd’s relaxation [18], which converges to the well-known Centroidal Voronoi Tessellation. The main limitation of this method is the significant regularity of artifacts without blue noise characteristics. Recently, variants of this method [2, 10, 29] were proposed to overcome this issue.

**Surface Domains:** Turk [25] uses a variant of Lloyd’s relaxation defined for polygon meshes. Similar to the planar domain this method was also extended [5, 30] using a different formulation of the energy. Cline et al. [6] presents a dart throwing algorithm to generate maximal Poisson disk point sets directly on 3D surfaces. It optimizes dart throwing by excluding areas of the domain that are already covered by existing darts. Bowers et al. [3] propose a GPU based parallel dart throwing algorithm that generates high-quality surface samples at interactive rates. Corsini et al. [8] propose a new constrained Poisson-disk sampling scheme for polygonal meshes, which generates a customized set of points with generic constraints. Li et al. [15] introduce the Dual Poisson-Disk Tiling scheme to stochastically distribute features on surfaces of 3D models. The generated distribution pattern follows the Poisson-disk distribution and the method can work on parameterized surfaces of arbitrary topology. Li et al. [16] present the anisotropic blue noise sampling on the plane. Then, they convert the problem of isotropic surface sampling into anisotropic sampling of the 2D parameter domain.

**Hierarchical Blue Noise:** McCool and Fiume [19] construct hierarchical Poisson disk sampling on the plane. Ostromoukhov et al. [21] propose another hierarchical sampling on the plane by means of the Penrose’s tiling. Both hierarchical structures can produce adaptive samplings efficiently.

2.2. Contributions

In table 1, we compare our contributions by considering three desirable properties regarding dynamically-varying adaptive samplings on surfaces: blue noise characteristics, temporal coherence, and real-time computation (at least 24 fps). Here, temporal coherence means that the changes from one sampling to the next should add or remove just a few points.

Bowers et al. [3] and Corsini et al. [8] report that an adaptive sampling takes 1 – 2 seconds to generate 10,000 points. In terms of practical applications in graphics, when an adaptive function in the surface domain varies dynamically over time, the computation of a new sampling at each time step could be a serious bottleneck. Temporal coherence cannot also be achieved because, at each frame, new seeds are replaced by the old ones. Pastor et al. [22] construct a hierarchy of points by means of edge-collapse simplification. However, the sampling
distribution is far from the blue noise characteristics. Vanderhaeghe et al. [26] proposes a hierarchy of Poisson disk samplings on the screen and project them to the surface. One drawback is the substantial amount of computation required to maintain temporal coherence. As a consequence, an interactive manipulation on 100 points produces frame rates that are between 5 and 10 per second. Chen et al. [5] compute adaptive blue noise on deformable triangular surfaces by taking the result of the previous frame as the initialization of the next frame, to generate a blue noise in a few interactions. Again, frames at interactive rates are not achieved.

Taking advantage of several techniques found in [3], our approach extends to surface domains the hierarchical Poisson disk sampling algorithm designed to planes [19]. Because the construction of the hierarchical sampling is a pre-processing step, given an importance sampling function and a local radius assigned to each sample point, the hierarchy is capable of constructing adaptive samplings with blue noise characteristics in the order of milliseconds. In addition, it preserves temporal coherence while importance sampling varies in each frame.

To construct the hierarchy, McCool and Fiume [19] relax the minimum distance criteria using tight ratios, in other words, \( |\text{ratio} - 1| \sim 0 \). There can be a serious bottleneck, especially when a large number of samples is demanded. To reduce the pre-processing time, we allow sparse ratios. Then, we apply a novel optimization algorithm to decrease the adaptation error of the hierarchy (see Figure 2).

3. Preliminaries

In this section, we provide formal definitions that are associated with Poisson disk samplings, hierarchical samplings and adaptive samplings. Then, we define the adaptation error and state the quantization problem.

3.1. Poisson Disk Sampling

Definition 1. Let \( A \) be a topological space with a metric \( d \). We say that the set of sample points \( S^p = \{s_1, s_2, \ldots, s_n\} \subset A \) is a Poisson disk sampling of a fixed radius \( \alpha \) if:

\[
\text{(a) tight: } |\text{ratio} - 1| \sim 0
\]

\[
\text{(b) sparse: } |\text{ratio} - 1| \gg 0
\]

\[
\text{(c) Optimization: sparse to tight}
\]

Figure 2: Sampling the ramp function. Instead of using tight ratios (a), our method generates a hierarchy in sparse ratios (b) and decreases its adaptation error (c).

Figure 3: Nested levels of a \( p \)-hierarchy: \( S_i \subset S_{i+1} \subset S_{i+2} \).

1. \( A \subset \bigcup B_\alpha(s) \);
2. \( (B_\alpha(s_i) \cap (S^p - \{s_i\})) = \emptyset \), \forall i ;

where \( B_\alpha(s_i) = \{ s \in A \mid d(s, s_i) < \alpha \} \) is the topological ball centered at the point \( s_i \).

3.2. Hierarchical Sampling

Let \( \mathcal{H} = \{ S_i \}_{i \in \{1, 2, \ldots, n\}} \) be a family of samplings of a given topological space \( A \). We say that \( \mathcal{H} \) is a hierarchy if \( S_1 \subset S_2 \subset \ldots \subset S_n \). The family of samplings \( \mathcal{H} = \{ S_i \} \) is a \( p \)-hierarchy if each level \( S_i \) is a Poisson disk sampling of radius \( \alpha_i \) (see Figure 3). Unless explicitly stated otherwise, \( \mathcal{H} \) will stand for a \( p \)-hierarchy. For each sample \( s \in \mathcal{H} \), we define

\[
\text{level}(s) = \min\{ i \mid s \in S_i \}
\]

A special case of hierarchy takes place when successive levels differs by exactly one point. Let \( \mathcal{H} = \{ S_i \} \) be a hierarchy.
We say that $\mathcal{H}$ is graded if $\#(S_{i+1} - S_i) = 1$. When the hierarchy is graded we will use the notation $\mathcal{H}^g$.

3.3. Adaptive Samplings and Adaptation Error

Let $\mathcal{H} = \{S_i\}_{i=1,2,..}$ be a hierarchy of the topological space $A$. Let $I : A \rightarrow [0, 1]$ be a smooth function. Here, this function is playing the role of importance sampling. An adaptive sampling $\mathcal{A}_I$ subject to the function $I$ is defined as the set,

$$\mathcal{A}_I = \{ s \in \mathcal{H} \mid \frac{\text{level}(s)}{m} \leq I(s) \}$$  (1)

As we can see, the adaptive sampling $\mathcal{A}_I$ selects the most representative samples in the hierarchy $\mathcal{H}$ by taking the function $I$ as threshold. For example, let $I(\cdot) = c$ be a constant importance sampling function. Let $l_c$ be the largest integer such that $\frac{l_c}{m} \leq c$. Thus, Equation 1 returns the uniform sampling $S_{l_c}$. On the other hand, if $I(\cdot)$ is spatially variant, then each sample element $s \in \mathcal{A}_I$ locally represents an uniform sampling $S_{l(s)}$.

The error associated with the adaptive sampling $\mathcal{A}_I$ is calculated in the following way (assuming that each level resembles an uniform distribution):

$$E(\mathcal{A}_I) = \sum_{s \in \mathcal{A}_I} \left( I(s) - \frac{[ml(s)]}{m} \right)^2$$  (2)

where $\lfloor \cdot \rfloor$ is the nearest integer function. By fixing the number of samples of $\mathcal{H}$ to $n$ and increasing the number of levels $m$, the error $E$ tends to zero. As a consequence, less adaptation error of the importance sampling $I$ is accomplished by the hierarchical sampling $\mathcal{H}$. The minimum adaptation error given by equation 2 is obtained when the number of levels and the number of samples are equal ($m = n$), or equivalently, $\mathcal{H}$ is graded. On the other hand, if the adaptation error is high, in other words, $m \ll n$, quantization artifacts are present in the adaptive samplings (see Figure 2(b)). P-hierarchies having tight/sparse ratios are equivalent to $p$-hierarchies with low/high adaptation error.

Now, consider the following problem regarding the optimization of the adaptation error: is it possible to construct a graded $p$-hierarchy $\mathcal{H}^g$ for any given $p$-hierarchy $\mathcal{H} = \{S_i\}$? More specifically, is there a natural ordering of the last level, $S_m$, such that it becomes a graded $p$-hierarchy?

4. Algorithms

We discuss three algorithms for solving the problems presented in the previous section: how can we generate a single Poisson disk sampling? How can we compute a $p$-hierarchy? How can we grade a $p$-hierarchy? Because the considerations considered in this paper are restricted to piecewise linear manifolds as topological spaces, the implementation details are limited to this set.

First, we consider the Euclidean distance to describe the algorithms. In Section 4.4 we discuss a more accurate approximation of the geodesic distance.

1. function $S \leftarrow$ PoissonDiskSampling$(O, \alpha, k)$
2. //S: Poisson disk sampling
3. //O: oversampling
4. //\alpha: radius
5. //k: maximum number of trials per grid cell
6. \mu \leftarrow \frac{\alpha^2}{\pi} ;
7. S \leftarrow \emptyset ;
8. box \leftarrow \text{BoundBox}(O) ;
9. $C[\cdot] \leftarrow [ \text{bboxmax-bboxmin} ] ;$
10. hashtable \leftarrow InitializeHashtable(O, C) ;
11. for each valid cell id $c \in C$ do
12. cell \leftarrow hashtable.search(c) ;
13. for each trial $t$ from 1 to $k$ do
14. if $O[\text{cell.first index} + t].\text{cellid} \neq \text{cell.id}$ then
15. break ;
16. $s \leftarrow O[\text{cell.first index} + t]$ ;
17. conflict \leftarrow false ;
18. for each neighboring cell id $d$ do
19. if $c \leftarrow \text{hashtable.search(c)}$ is not null then
20. if $d(s, cell_\text{sample}) \geq \alpha$ then
21. conflict \leftarrow true ;
22. break ;
23. if conflict = false then
24. cell.sample \leftarrow s ;
25. $S \leftarrow S \cup \{s\}$ ;
26. break ;
27. end ;
28. end ;
29. end ;
30. end ;

Algorithm 1: Algorithm for uniform surface sampling.

4.1. Generating a Single Poisson disk Sampling

Basically we use the dart throwing algorithm described in [3]. For simplicity, we omit implementation details that regard to GPU acceleration.

Oversampling: Initially, we generate a set of random points $|O| = \delta \cdot N$ by uniformly sampling the entire surface. Here, $N$ is the number of expected points for a given radius $\alpha$, and $\delta$ is a density parameter that represents the number of oversamples per sample. Following Lagae and Dutr´e [14], we estimate $N$ by solving the equation

$$\alpha = \rho \cdot \alpha_{\text{max}},$$  (3)

where $\alpha_{\text{max}} = 2 \cdot \sqrt[3]{\text{area}(A)} \cdot \sqrt[3]{2 \cdot \sqrt[3]{3N} / V}$ is the maximum packing density, $A$ is the polygonal mesh and $\rho \in [0.65, 0.85]$ is a fraction of the maximum radius for Poisson disk distributions. The algorithm repeatedly selects a triangle with probability that is proportional to its area and, then, uniformly samples the selected triangle using barycentric coordinates, which are computed as $u = 1 - \psi_2 \sqrt[3]{V_1} \cdot \psi_2$, where $\psi_1, \psi_2 \in [0, 1]$ are two uniform random numbers.

Data structure: We build a 3D grid around the bounding box of $O$, using $\frac{\Delta}{V}^\frac{1}{3}$ as the grid cell size. Then, we partition the
oversampling \( O \) into the grid cells. Because valid cells (cells containing at least one sample) can be sparse in the case of 3D surfaces, we use a hash table to store them. We take the global cell id as a hash key, and a modulo operator as the hash function: hash idx = key % hash_table.size. The size of the hash table is 4 times the number of non-empty cells. To handle hash collision, we allocate \( \frac{\alpha}{\delta} \) times the number of oversampling initializations.

**Algorithm outline:** Given the minimum distance requirement, each cell can contain at most one Poisson disk sample. For each cell, the algorithm makes up to \( k \) trials to accept a random point that satisfies the minimum distance requirement with existing samples in the neighboring cells. If all \( k \) trials are rejected, then the cell is left empty. The process uses a random ordering of the cells to reduce bias. Algorithm 1 shows the pseudocode summarized in this section.

4.2. Generating a p-hierarchy

Let \( \{\alpha_i\} \) be a set of positive real numbers \( \alpha_1 > \alpha_2 > \ldots > \alpha_m \). To construct the \( p \)-hierarchy, we apply Algorithm 1 to generate each level. We optimize the oversampling initialization by using a hierarchy \( \{O_l\}_{l=1,m} \) of uniform oversamplings, where \( |O_l| = \delta N_l \), and \( N_l \) is computed through Equation 3. For this goal, we build a large set of uniform random samples \( O_m \) stored in an array. For each level \( i \), we set its respective initial oversampling as the first \( \delta N_l \) samples. Next, suppose we are at level \( i \); we apply dart throwing (algorithm 1) over the set \( Q = O_i \cup \bigcup_{n_{i,l}} B_{\alpha_i}(s) \) to obtain the sampling \( S_i \) (see figure 4). To compute \( Q \), we run an advancing front algorithm by taking the set \( S_{i-1} \) as seed points and continuing to remove any conflicting point with the topological balls having radii \( \alpha_i \). Algorithm 2 summarizes this process.

**Implementation**

We generate \( |O_m| \sim 4 \cdot 10^6 \) millions of initial oversampling points. The number of trials \( k \) is 30 and the number of oversamples per sample \( \delta \) is 100. Although the quality of the sampling can always be improved by increasing these parameters, in practice, these settings give us good blue noise distributions (see Section 6.2). In algorithm 2, if \( \delta N_l > |O_m| \) from some level \( i \), then we set \( O_i = O_m \). Finally, we set \( \rho = 0.75 \), a commonly used fraction of the maximum radius for Poisson disk distributions [14].

**Algorithm 2:** Generating a \( p \)-hierarchy.

4.3. Grading a \( p \)-Hierarchy

**Grading intuition.** Let \( S_1 \subset S_2 \subset \ldots \subset S_m \) be a \( p \)-hierarchy. To grade this family of samplings, we must sort the set \( D_i = S_i - S_{i-1} = \{s_{i_1}, s_{i_2}, \ldots, s_{i_n}\} \), for each level \( i \). Let us follow an intuitive idea: given the set of topological balls \( B_{\alpha_i}(s), s \in S_i \), increase their radius continuously. At a certain time, a point \( q \in D_i \) will be touched by a topological ball, for example, \( B_{\alpha_i}(p) \), such that \( q \in \partial B_{\alpha_i}(p) \) and \( \alpha' = d(p, q) \). Then, the set \( S_{k-1} \cup q \) is a new “Poisson disk sampling” of radius \( d(p, q) \). Note that the covering condition (see condition 1 in definition 1) is sometimes violated. In this case, we have only an approximation. Continue this process until every sample point \( s \in D_i \) has been touched. In the end, we will obtain the sorted set \( D_i = \{s_{i_1}, s_{i_2}, \ldots, s_{i_{n_i}}\} \), where \( \sigma \) is a permutation of \( \{1, 2, \ldots, n_i\} \).

**k-neighbors construction.** Because we must determine the closest point \( q \), we must build a \( k \)-neighbors data structure. For each sample point \( s \in (S_{i-1} - S_i) \), we take the \( k \)-nearest samples that have a distance that is not greater than \( 2\delta \). In our experiments we set \( k = 10 \). We call \( G \) the resulting directed graph.

**Vertex removal.** When a sample \( q \) is touched, we must update \( G \) by removing the corresponding vertex \( v_q \). Let \( V_{in}^q \) be the set of vertices that point to \( v_q \) and \( V_{out}^q \) the set of points that are pointed by \( v_q \). Then, for each vertex \( v_r \in V_{in}^q \) we choose the closest vertex \( v_p \in (V_{out}^q - \{v_r\}) \) and create a new arc \( v_r v_p^q \).

**Algorithm outline.** We initialize the priority queue \( P \) that contains all arcs of \( G \). At the top of \( P \) lies the closest pair of points of the graph \( G \). Let \( e(v_p, v_q) \) denote the corresponding arc of this pair. We take the level number as the first criterion for choosing which vertex should be removed. In other words, if \( level(v_q) > level(v_p) \) then we remove \( v_q \). Similarly, if \( level(v_p) > level(v_q) \) we remove \( v_p \). In case of equality \( level(v_p) = level(v_q) \), we remove the vertex having the shortest distance to its surrounding neighbors, discarding the arcs.
In this configuration we decide to remove \( v_p \). It has the shortest distance \( d(v_p, v_r) \) to its neighbors.

\[
\begin{align*}
&\text{procedure } \mathcal{H} \leftarrow \text{SmoothHierarchy}(\mathcal{H}) \\
&\mathcal{G} \leftarrow \text{Graph}(S_{1\rightarrow 0} - S_{1\rightarrow 0}'); \\
&P \leftarrow \text{PriorityQueue}(G); \\
\text{while numVert}(G) > 1 \text{ do} \phantom{\text{if level}(v_q) \geq \text{level}(v_p) \text{ then} \text{VertexToRemove} \leftarrow v_q;} \\
&\text{if level}(v_q) > \text{level}(v_p) \text{ then} \phantom{\text{VertexToRemove} \leftarrow v_q;} \\
&\text{VertexToRemove} \leftarrow v_q; \\
&\text{if level}(v_q) = \text{level}(v_p) \text{ then} \\
&\quad d' \leftarrow \min\{d(v_q, v_r) \mid v_r \in V_{\text{out}}, v_r \neq v_p\}; \\
&\quad d'' \leftarrow \min\{d(v_p, v_{r'}) \mid v_{r'} \in V_{\text{out}}, v_{r'} \neq v_q\}; \\
&\quad \text{if } d' < d'' \text{ then} \phantom{\text{VertexToRemove} \leftarrow v_q;} \\
&\quad \text{VertexToRemove} \leftarrow v_q; \\
&\text{else} \phantom{\text{VertexToRemove} \leftarrow v_q;} \\
&\phantom{\text{VertexToRemove} \leftarrow v_q;} \\
&\phantom{\text{VertexToRemove} \leftarrow v_q;} \\
&\text{remove}(G, \text{VertexToRemove}); \\
&\end{align*}
\]

Algorithm 3: Smoothing a \( p \)-hierarchy.

\( e(v_p, v_q) \) and \( e(v_p, v_p) \) (see Figure 5). Algorithm 3 summarizes this grading process.

Note that this grading algorithm can also be applied to other hierarchical constructions. For example, the hierarchy proposed by Wei [27] uses an octree-based data structure, in other words, the ratios between successive levels are dyadic.

In the Section 6.2, we show that our grading algorithm produces a better result than a random sorting method.

4.4. Using the Approximation of Geodesic Distance

So far, we were using the Euclidean distance as a rough approximation of the geodesic distance. Although the Euclidean distance demonstrates easy implementation and fast performance, it can lead to undesirable distributions when the surface has high curvatures and small tubular regions (thin features).

This is because Euclidean distance measures the shortest distance in space as opposed to surface.

Ideally, we should use an accurate geodesic distance [20, 24], but they are often expensive to compute, especially for generating millions of points. A simple and more accurate approximation of the geodesic that only relies on the points and not on the mesh connectivity or parametrization is introduced in [3]. Let \( p, q \) be two points on the surface and \( \overrightarrow{pq}, \overrightarrow{q} \) be their respective normals. The Euclidean distance between the two points is \( d_e = \|q - p\| \) and the normalized vector is \( \overrightarrow{pq}/d_e \). Assuming that there is a smooth curve on a surface that passes through two points, \( p \) and \( q \), the geodesic distance between the two points is estimated as the length of this curve:

\[
d_g = \arcsin\left(\frac{\overrightarrow{pq} \cdot \overrightarrow{q}}{d_e} \right) - \arcsin\left(\frac{\overrightarrow{pq} \cdot \overrightarrow{q}}{d_e} \right) \cdot d_e 
\]

The main advantage of this approximation is that it is fast and can easily fit into the previous algorithms. On the other hand, because it ignores the mesh connectivity, it is only accurate when the surface changes smoothly from point \( p \) to point \( q \).

In the Euclidean setting, we assume that each grid cell contains at most one sample. However, this is not true for the geodesic setting, because there can be two points in the cell such that \( d_e > \alpha \). To handle this case we allow each cell accept more than one sample. This increases the running time but in practice it is rarely observed any cell that contains more than one sample. Figure 6 compares the dancer model with two different uniform Poisson disk samplings. In Figure 6.(a) we construct the sampling using the Euclidean distance and it produces 88 samples. On the other hand, when we use geodesic distance, we obtain 113 points (see Figure 6.(b)). This means an increase of 28% in the number of samples.

4.5. Fast Construction of Adaptive Sampling

Let \( F : M \rightarrow \mathbb{R}^+ \) be an adaption function over the surface \( M \) and let \( \mathcal{H} \) be a hierarchical Poisson disk sampling of \( M \). Let \( R : \mathcal{H} \subset M \rightarrow \mathbb{R}^+ \) be the local radius defined over the samples. In the case of a \( p \)-hierarchy, this function corresponds to the radius at the level of the Poisson disk sampling. If the \( p \)-hierarchy is graded by Algorithm 3, we set the radius of the sample to be the size of its corresponding closest pair.

The query for construction of an adaptive blue noise is performed by sweeping the samples and making a simple comparison: if \( R(s) \leq F(s) \), then draw the sample; otherwise, discard it. This query is very efficient since it linearly sweeps the sample points.
Note that the query resembles Equation (1). Here, we substitute \( \text{level}(s) \) and the importance sampling function \( I(s) \) to the local radius \( R(s) \) and the adaptive function \( F(s) \), respectively. Because we are using geometric information to make the comparison, the latter formulation leads to more accurate distributions.

If the input is an importance sampling function, it is necessary to perform the conversion \( F(s) = \frac{r_{\min}}{\sqrt{I(s)}} \), where \( r_{\min} \) is the minimum value of the function \( R \).

5. Applications

To explore the potential of the hierarchy, we show applications in the area of NPR. In the first application, we apply our method by taking the illumination as importance sampling and shade the surface, creating a stippling effect. In the other application, we dynamically deform a surface with a predefined stippled texture.

5.1. Shading

Our system is very simple and is separated into two steps. First we render the polygonal surface with the background color. Afterward, we plot the points with a user predefined size. We use the video memory to store arrays that correspond to position, normal and radius. For maximum efficiency, both steps are implemented using shader programs.

In order to achieve different sampling densities, the system allows the user control the importance sampling according to a given attribute. For example, tone depiction can be simulated by scaling the importance sampling proportionally to the light intensity (Figure 9). Similarly, zooming is performed by scaling the importance sampling proportionally to the distance from the viewer (Figure 7).

We compare our result with Pastor et al. [22] (Figure 8). Our adaptive blue noise scheme has the best compromise to distribute the points evenly along the surface.

5.2. Texturing and Shading under Deformations

Because of the hierarchical structure, our method can be easily integrated with mesh deformation techniques in such a way that only the regions of the model affected by the deformation process can change. Given a triangular mesh \( M \), for each triangle \( t_i \) of \( M \) we analyze the relationship of its area before and its area after the deformation. This relationship is expressed by the following constant:

\[
k_i = \frac{\text{Area}(t_i)}{\text{Area}(\tilde{t}_i)},
\]

where \( \tilde{t}_i \) is the deformed area of the triangle.
Figure 10: (a) Checkerboard with stippled texture, (b) Deformation without adaptation, (c) Deformation using new importance sampling \( \tilde{I} \).

Figure 11: (Left) \( t_i \) before deformation. (Center) \( I \) over \( \tilde{t}_i \). (Right) \( \tilde{I} \) over \( \tilde{t}_i \).

For each sample \( s \), we use its barycentric coordinates related to the vertices of \( t_i \) to calculate the respective positions on \( \tilde{t}_i \). In Figure 11, we depict, in a single triangle, the steps for the placement of the samples under the deformation process.

Observe that, if \( t_i \) is not affected by the deformation, the constant \( k_i \) is equal to 1. In this case, the temporal coherence is maximal; in other words, the number of samples in \( t_i \) does not change.

For their simplicity, flexibility and speed, we use a cage-based deformation technique in the deformation process, although the steps described above can be integrated into any mesh deformation technique. Cage-based deformations express the vertices of the mesh through affine sums of the cage vertices weighted by functions called coordinates. The cage is a simpler mesh with the same shape of the model. Once we have the vertices of the mesh as a function of the cage, we can control the deformation of the mesh by interactively moving the position of the cage vertices (see Figure 12).

In our experiments, we use Mean Value Coordinates [13] and Green Coordinates [17] as weight functions to express the vertices of the mesh. These coordinates have different derivations and their deformations will present different characteristics as well. For our purposes, the main difference is that, unlike Mean Value Coordinates, cage-based deformation using Green Coordinates generates conformal mapping in 2D deformations and quasi-conformal mapping in 3D deformations, which means that the triangle shapes are better preserved and the sampling on the mesh after the deformation also maintains blue noise characteristics. To illustrate this effect, we deform a square with a checker-board pattern using the Green Coordinates method (see Figure 10).

Deformations using Mean Value Coordinates present a higher flexibility degree. For surfaces in \( \mathbb{R}^3 \), we combine Mean Value deformation with the stippling technique described in Section 5.1. If the displacement of the cage vertices is small,
then we preserve the shading effects and the blue noise characteristics as well (see Figure 13). However, for large displacements, the blue noise properties are lost in triangles that have a high anisotropic deformation.

**Implementation:** Deformation and rendering are performed separately. Because each vertex can be processed independently, the deformation is computed in the CPU and accelerated using the multi-processing software OpenMP [1]. Once we have the new mesh, we send its vertices as a texture to the GPU. Then, in the shader program, we perform two steps. First, the new sample positions are computed using their barycentric coordinates. Second, using an appropriate importance sampling function as shown in equation (6), we render the points.

6. Results

We show the performances of the hierarchical sampling algorithm and the grading process. In all experiments, the computer has an Intel® Core™ i5 430M (2.26GHz) processor and 4GB RAM of memory. Then, we analyze the quality of the adaptive sampling using **differential domains**.

6.1. Performance

We benchmark the hierarchical sampling and the grading algorithms with the heart model (fig 7) which has 14,646 faces. On the top of the Figure 14, we obtain the running times of the hierarchical algorithm for different ratios. In the the graph corresponding to the dyadic ratio, only 7 levels are required. For the ratios $\sqrt{2}$, 1/0.90 and 1/0.95, the hierarchical sampling algorithm generated 14, 48 and 98 levels, respectively. For each graph, the time required to construct the hierarchy increases exponentially with the number of points. Thus, because of this exponential behaviour, tight ratios are not feasible for a large number of points. On the other hand, for sparse ratios, we obtain high adaptation error. See for example, the quantization artifacts shown in the Figure 2(b).

On the bottom of the Figure 14, we show running times of the grading algorithm for dyadic and $\sqrt{2}$ ratios, which are summed up with the hierarchical sampling time. Note that running both algorithms in sparse ratios (dyadic and $\sqrt{2}$) still require much less time than running only the hierarchical sampling algorithm with tight ratios (1/0.90 and 1/0.95). For 240,000 points, running sequentially the hierarchical sampling algorithm with dyadic ratio and the grading algorithm is 74% faster than running only the hierarchical sampling with ratio 1/0.95.
In table 2 we have the results of some examples discussed in this paper. The first 5 columns show the running timings of the pre-processing algorithms (hierarchy and grading). The ratio between successive levels is fixed to $\sqrt{2}$. The last column expresses the frame rate as the user interacts with the object. Because we are computing deformations, the last two rows show a great decay in the frame rate.

<table>
<thead>
<tr>
<th>Model</th>
<th>#faces</th>
<th>#samples</th>
<th>time(s)</th>
<th>samples/s</th>
<th>#levels</th>
<th>fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costa</td>
<td>5,580</td>
<td>34,819</td>
<td>40.08</td>
<td>868</td>
<td>11</td>
<td>&gt; 500</td>
</tr>
<tr>
<td>Teapot</td>
<td>2,256</td>
<td>127,459</td>
<td>155.04</td>
<td>822</td>
<td>13</td>
<td>&gt; 500</td>
</tr>
<tr>
<td>Heart</td>
<td>14,677</td>
<td>436,614</td>
<td>190.01</td>
<td>2,298</td>
<td>15</td>
<td>&gt; 500</td>
</tr>
<tr>
<td>Pelvis</td>
<td>99,846</td>
<td>456,307</td>
<td>154.96</td>
<td>2,945</td>
<td>15</td>
<td>&gt; 500</td>
</tr>
<tr>
<td>Board</td>
<td>32,768</td>
<td>432,038</td>
<td>121.37</td>
<td>3,560</td>
<td>15</td>
<td>63</td>
</tr>
<tr>
<td>Tric.</td>
<td>5,660</td>
<td>156,016</td>
<td>132.50</td>
<td>1,117</td>
<td>13</td>
<td>115</td>
</tr>
</tbody>
</table>

Table 2: Performances of some examples shown in this paper.

6.2. Analysis

Wei and Wang [28] developed the first technique to evaluate non-uniform distributions using differential domains. This technique is analogous to the method that is based on the Fourier transform [14].

In a square, let us consider the ramp linear function varying in the interval $[0, 1]$ as importance sampling. We evaluate our adaptive samplings in four different situations: (a) $\alpha_k = (0.99)^k\alpha_0$, (b) $\alpha_k = (0.5)^k\alpha_0$, (c) $\alpha_k = (0.5)^k\alpha_0$ with a random gradation and (d) $\alpha_k = (0.5)^k\alpha_0$ with our geometric gradation.

Figure 15: Rows from top to bottom: first - samplings of the experiments; second - differential domains; third - power diagrams and last: anisotropy. Columns: (a) $\alpha_k = (0.99)^k\alpha_0$; (b) $\alpha_k = (0.5)^k\alpha_0$; (c) $\alpha_k = (0.5)^k\alpha_0$ with a random gradation and (d) $\alpha_k = (0.5)^k\alpha_0$ with our geometric gradation.

The sampling 15.(b) has a severe artifact because of the quantization. In the naive gradation 15.(c), we observe that the central hole is much smaller than the others. As a consequence, there is a decay in the peak of its power diagram. On the other
hand, our geometric method 15.(d) has the best compromise in preserving the blue noise characteristics. Note that its power diagram best approximates to the reference. In the anisotropy diagrams, all experiments show no structural bias.

In a 3D surface with genus 3, we apply an importance sampling function curvature given by $I(s) = \frac{1}{1 + \kappa^2(s)}$, where $\kappa$ is the mean curvature. We compare the power diagrams and the anisotropy in the following situations: (a) sampling from Bowers et al. [3], (b) $\alpha_k = (0.95)^k \alpha_0$; (c) $\alpha_k = (\sqrt{2}/2)^k \alpha_0$ and (d) $\alpha_k = (0.5)^k \alpha_0$ (see Figure 16). For each experiment we run the sampling algorithm 8 times. All of the adaptive samplings have approximately 3,000 points. To generate the samplings, we used the approximation of the geodesic as distance metric. In the adaptive samplings (c) and (d), we applied our geometric gradation method.

Although the algorithm proposed by Bowers et al. [3] demonstrates superior blue noise characteristics (see experiment 16.(a)), our method also produces remarkable blue noise distributions. From experiment 16.(d) to experiment 16.(c) we see a quality improvement of the power diagram that is higher than the improvement from experiment 16.(c) to 16.(b). This shows that, in order to produce high-quality blue noise at low time consuming, sparse ratios around $\sqrt{2}$ should be enough. Because of the structural artifact present on methods based on grid partitions of the space, the anisotropy diagrams are slightly biased in the experiments 16.(b), 16.(c) and 16.(d). As predicted by Wei [27], this artifact is alleviated as we raise the number of levels of the hierarchy.

7. Conclusion

We have presented a new method to compute a pointerless hierarchy of Poisson disk samplings on surfaces. Using this hierarchy, we can generate adaptive samplings with blue noise characteristics, temporal coherence and real-time computation. Our grading method improves the differential domains of the samplings and preserves the good blue noise properties even for dyadic scales.

The results showed that we achieved real-time rendering in both applications: shading and deformation.
For future work, we plan to generalize the algorithm for generating hierarchical Poisson disk samplings to other surface representations. We also want to generalize it to higher dimensions.

Acknowledgment

References


