Vision, Modeling, and Visualization (2018)
F. Beck, C. Dachsbacher, and F. Sadlo (Eds.)

# Hierarchical additive poisson disk sampling

Alexander Dieckmann and Reinhard Klein[†]

University of Bonn

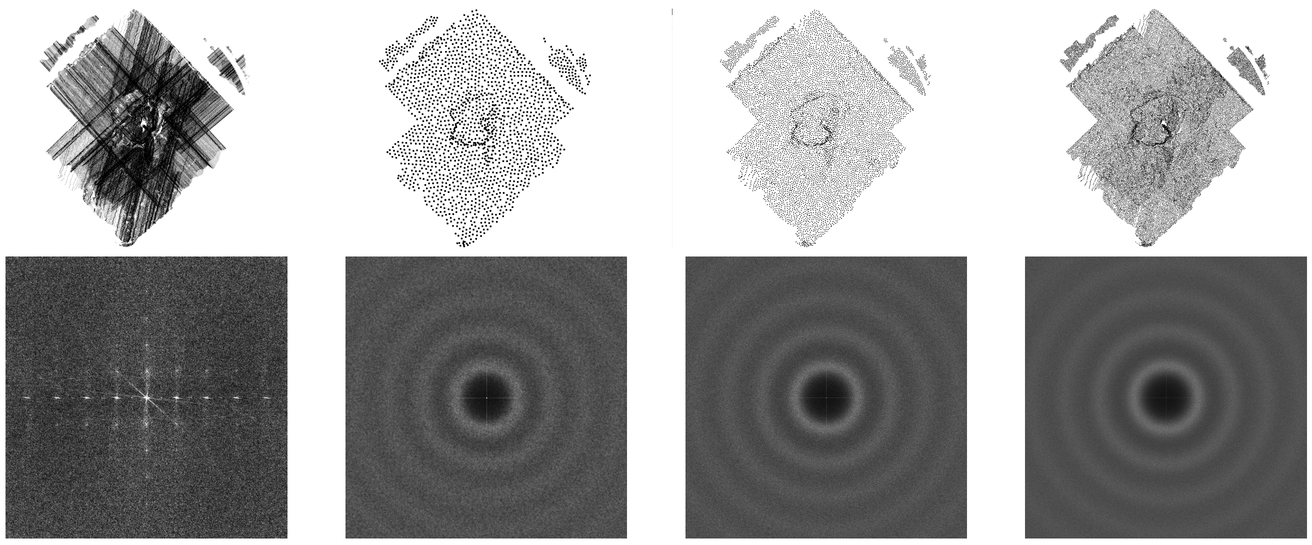

**Figure 1:** *Additive hierarchical poisson disk sampling of the top view of a tomato plant scan.* ***Top left:*** *Original point set with scan-artifacts.* ***To right****: Increasing resolution of poisson disk samplings with octree depth $D = 0$ to $D = 3$. The scan-line and scan-edge artifacts vanish in the lower resolution poisson disk samplings.* ***Bottom:*** *Respective power spectrums of the samplings.*

**Abstract**

*Generating samples of point clouds and meshes with blue noise characteristics is desirable for many applications in rendering and geometry processing. Working with laser-scanned or lidar point clouds, we usually find region with artifacts called scan-lines and scan-edges. These regions are problematic for geometry processing applications, since it is not clear how many points should be selected to define a proper neighborhood. We present a method to construct a hierarchical additive poisson disk sampling from densely sampled point sets, which yield better point neighborhoods. It can be easily implemented using an octree data structure where each octree node contains a grid, called Modifiable Nested Octree [Sch14]. The generation of the sampling amounts to distributing the points over a hierarchy (octree) of resolution levels (grids) in a greedy manner. Propagating the distance constraint r through the hierarchy while drawing samples from the point set leads to a hierarchy of well distributed, random samplings. This ensures that in a disk with radius r, around a point, no other point upwards in the hierarchy is found. The sampling is additive in the sense that the union of points sets up to a certain hierarchy depth D is a poisson disk sampling. This makes it easy to select a resolution where the scan-artifacts have a lower impact on the processing result. The generated sampling can be made sensitive to surface features by a simple preprocessing step, yielding high quality low resolution poisson samplings of point clouds.*

**CCS Concepts**

• ***Computing methodologies*** $\rightarrow$ ***Point-based models;***

[†] {dieckman, rk}@cs.uni-bonn.de

DOI: 10.2312/vmv.20181256

## 1. Introduction

Sampling methods are an essential part of computer graphics. For example, laser scanned point sets are the result of a sampling process of a surface. The laser scanner shoots rays into uniformly sampled scan directions and measuring the distance of a point into that direction. The produced point clouds have inhomogeneous local point distributions, due to the irregular sampling rate of the scanned surface. When multiple scans are combined to produce a more complete representation of the scanned surface, the inhomogeneity is increased and the point cloud will contain over- and under-sampled regions, visible as scan-lines and scan-edges, see Figure 1 and 2. Applications which process point sets locally are sensitive to the point distribution. An example is the normal estimation based on k-nearest-neighbors which gets instable in case that all neighbors belong to the same scan-line, see Figure 3. Similar problems occur in other contexts like surface reconstruction [KH13], object/mosaic-tile placement, point splatting [BHZK05], etc.

One way to overcome this problem is to subsample the point set and to adjust the distribution in such a way that local point neighborhoods better represent the local surface. Unfortunately, the chosen subsampling heavily influences the resulting neighborhoods. For example, a grid-based random sampling, used in many out-of-core rendering contexts [Sch14], [WS06], where each grid cell contains at most one point, results in samplings where some neighboring points can be arbitrarily close. This introduces artificial noise into the the sampling and leads to neighborhoods which don't represent the local surface well. Choosing the closest point to the cell's center or the center it self as a representative yields higher quality neighborhoods but introduces bias into the sampling, see Figure 5. While the first approach introduces noise, this approach leads to structural aliasing, see Figure 8.

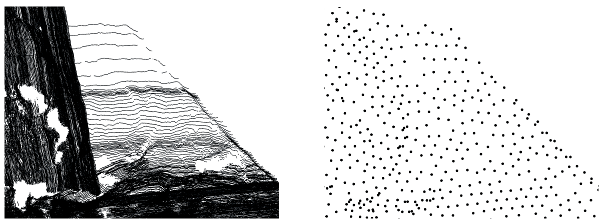
**Figure 2:** *Top view of a part of a tomato plant laser scan.* ***Left:*** *Scan-lines are the horizontal and vertical lines. Scan-edges come from overlaying different scans, visible in the left dark region .* ***Right:*** *Poisson disk sampling of the left scan. The distances between the points are more uniform (the close points are due to the 2D view of the 3D Scan).*

In contrast, better random but more uniform point sets can be generated by blue noise sampling which produces poisson disk point sets. These point sets are, due to their statistical characteristics, ideal candidates in various geometry processing and rendering applications. Points of a poisson disk point set are randomly distributed with a minimum distance $r$ constraint between the points. These point sets exhibit two spectral characteristics which can be used to identify random and well-spaced samplings. First, the energy is equally distributed without spikes. Second, the spectrum has a deficiency of low-frequency energy. These conditions are called *blue-noise criteria* [Mit87], [Uli88]. The PSA tool [SD11] can be used to analyze these 2D point distributions by their 2D spectrum. Additionally two 1D statistics of the sampling are computed. The first is the radial distance function, which is a measure of the probability to find a particle at distance $r$. The second is the radially averaged power spectrum which shows the direction independent energy distribution of the spectrum, see Figure 5 and 4. While

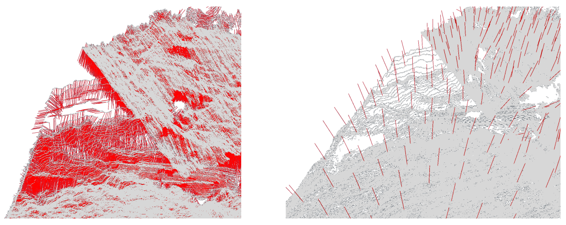
**Figure 3:** ***Left:*** *Degenerated neighborhoods in scan-line-areas the lead to arbitrary normal directions (red lines). In addition orienting normals using [HDD*92] cannot guarantee correct oriented normals.* ***Right:*** *Smooth normals (red lines), computed on a lower resolution and shown on the full point set.*

blue noise sampling drastically improves the computation of local surface features there still remains the question how to choose the minimal point distance $r$. While $r$ must be chosen such that the local neighborhoods do not degenerate, e.g. due to a single scan-line, choosing $r$ too large might lead to over-smoothing. In high density areas $r$ should be smaller, in low density areas respectfully larger. Adapting $r$ in different regions of the point set can be done with Yuksel's [Yuk15] progressive sample elimination, which generates a poisson disk point set with a user-specified number of points and very good blue noise characteristics. Due to the progressive nature of this approach, points can be added to the sampling to increase the resolution and decrease $r$. Unfortunately, this can be done only globally, i.e. if higher resolution is necessary in some areas, all points have to be inserted into the point set.

Using a hierarchy of scales enables easy switching between different levels to adapt the resolution. In this paper we therefore describe a sampling method, to produce poisson multi-resolution samplings of large point sets, which are additive in the poisson disk sampling sense. The union of points in different resolution levels is guaranteed to have empty neighborhoods up to the radius corresponding to the finest resolution level of the union. This way the minimum sampling distance $r$ is achieved by simply choosing locally the appropriate level of detail. These samplings can be used to decrease the influence of data related artifacts in point sets, i.e. scanned point sets, on the results of various geometry processing methods as well as on the rendering quality of these point sets. Furthermore, we describe a greedy approach which makes the multi-resolution sampling feature-sensitive, such that feature points are always contained in coarse resolutions. While being reasonably fast, see Table 1, our method cannot guarantee maximal coverage. We focus on the efficient processing of large point clouds stemming from unprocessed laser scanned surfaces and show some additional results for triangle meshes.

## 2. Related Work

Generating a sampled point set with blue noise characteristics has many applications in rendering [Coo86], [CGW*13], [SZG*13], texturing [LD05a], stippling [BSD09], [DGBOD12], [Fat11], animation [SB12], visualization [LWSF10] and numerical optimization [EPM*14], [MEA*18] and geometry processing [ACSD*03], [ÖAG10], [CGW*13], [GYJZ15], [YGW*15], [AGY*17], [NBH18].

### 2.1. Poisson sampling

A simple way of generating poisson disk sample sets in arbitrary dimensions is dart throwing [DW85], [Coo86]. Given the desired poisson disk radius $r$, dart throwing rejects or accepts randomly generated points based on its distance to previously accepted points. Due to difficulties in generating very large, well distributed point sets, a data structure that partition the sampling domain into grid cells is introduced in [Bri07], which achieves linear time and can be extended to parallel sample generation [Wei08].

Tile based methods are used to efficiently generate a large number of poisson disk samples in 2D by reusing another poisson disk sampling method [HDK01], [CSHD03], [LD05b], [KCODL06], [KS12], [WPC*14]. The tiling of small generated samplings produces very large samplings but is restricted to the 2D sampling domain.

### 2.2. Progressive sampling

Progressive sampling methods reduce the poisson disk radius $r$ progressively [MF92], [MREB12], but they are prone to introducing sampling bias and their quality and performance depend on how $r$ is updated. Other progressive approaches require mesh surfaces [YW13] or are restricted to the 2D domain. A different approach of generating a progressive poisson disk sampling in higher dimensions, called sample elimination, is described by Yuksel [Yuk15]. This method produces a progressive poisson disk sampling with a user-specified number of points $N$ from a given set of $M$ samples. Finding such a subset with the largest poisson disk radius is an NP-complete problem, but an approximate solution can be found using a greedy algorithm in $M-N$ steps. Each sample is assigned a weight based on its distance to its neighbors. At each step, the sample with the highest weight is eliminated and the weights of the remaining points are adjusted. The inverse elimination order gives a hierarchical representation of the point set. While sample elimination produces point sets with excellent blue noise characteristics, we found it was not possible to adapt the radius $r$ locally and to use this method in the context of out-of-core capable data structures on point clouds with 10-63 million points in a reasonable amount of time.

A progressive solution to generate an approximate poisson sampling was proposed in the context of efficient point cloud rendering [Sch16](Potree). This approach built on a data structure introduced by Scheiblauer [Sch14], called "Modifiable Nested Octree". This data structure consists of an octree storing a uniform grid in each node and can be used efficiently for dynamic frustum culling of large parts of the surface, while keeping a certain resolution level with respect to the viewer. The octree is filled with points from a scan in a greedy fashion. Each grid cell is allowed to contain only one point. Starting at the root node of the octree, the node's grid occupancy is queried. If a grid cell already contains a point, then all points falling into the same cell are rejected to the children of the octree node. Choosing a random point from the point set for each grid cell leads to a hierarchy of samplings where neighboring points and points in consecutive octree levels can be arbitrary close. Incorporating a distance constraint, to generate an approximate poisson sampling per grid, still produces a hierarchy of samplings in which points in neighboring grids and in consecutive octree levels can be arbitrarily close. While this improves the rendering quality of the scans, further processing of the scanned geometry is still biased to the used data structure. In our approach we overcome this drawback and generate unbiased hierarchical additive poisson disk samplings using a similar data structure.

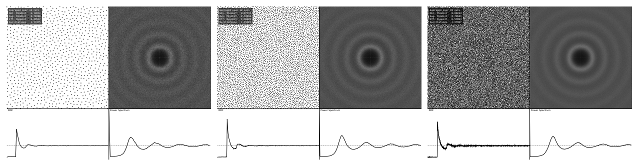
**Figure 4:** *A hierarchical sampling of the 2D plane. The resolution increases from* ***left*** *to* ***right****.* ***Top row:*** *The respective periodograms on the right side of each sampling show the blue noise characteristics of the samplings though the hierarchy. The sampling is additive in the sense that the points from the left sampling are included in the right one.* ***Bottom row:*** *The radial distribution function (left) and the power spectrum (right). The images are generated with the PSA tool [SD11].*

## 3. Method Overview

Based on a simple C++/OpenGL rendering framework we implemented a sparse octree data structure and a sparse grid data structure using generic hash data structures. While an octree can be used to represent a grid, making this distinction makes implementation and maintenance much easier than, for example, an octree of octrees. The grids store indices into the original point set, while the octree is used to traverse the hierarchy of grids. The general idea of our method is to use the extended dart-throwing method [Bri07] by drawing points from the point scan and propagate the distance constraints through the hierarchy by traversing the octree.

Starting at the root node, the octree is build up layer by layer, where a layer contains all active octree nodes of the same depth. For each point in a scan we maintain a pair of its point set index and a pointer to the node in which the point should be inserted. Initially the node pointers are set to the root node. These pairs are stored in a list, which we will call the active point list. For each point in the list, we try to insert its point set index into the octree node, see Figure 6. We query the grid inside the node and decide if we accept or reject that point by making a neighborhood query

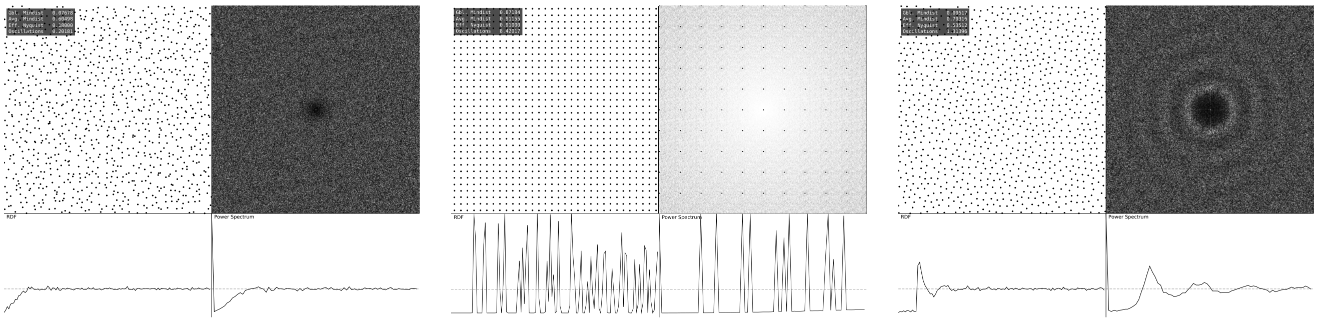

**Figure 5:** *The frequency analysis for comparison of grid random sampling, grid closest to center sampling and poisson sampling of the 2D plane. The colors in the frequency plot for the grid closest to center sampling are inverted for better visibility. The results are generated with PSA tool [SD11].*

to all nodes from the root node to the current node with the current poisson radius $r_D$ suitable for the current resolution $D$.

When the point is accepted it is deleted from the active point list and only considered for the rejection of following points. If the point is rejected we push the point further down to the respective child node and update the node pointer in the pair, yielding the next active point list. We iterate this procedure until the next active point list is empty or stop at a depth of 21 and reject all points from this depth on due to the small resolution at this level. The former active point list is freed from memory when it has been fully processed.

Unifying the samples in octree nodes of all nodes from the root node to the current depth yields a poisson sampling of the point set. Adding another layer, doubles the resolution of our poisson disk sampling. The sampling is additive in the sense, that the union of samplings is a poisson sampling, Figure 4.

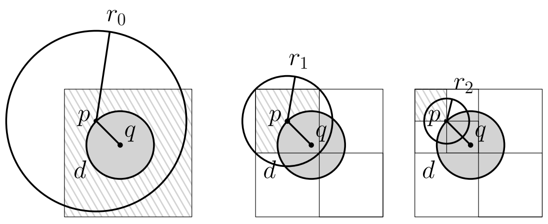

**Figure 6:** *A grid cell is occupied (hatched) by point $p$ and rejects a point $q$ if it falls into the same cell or their distance $d$ is smaller than the poisson disk radius $r_D$. Starting in hierarchy depth $D = 0$. The point $q$ is pushed to the child in the next hierarchy level and will be rejected again, $d < r_1$, until a free cell is reached and no conflict can be found in its neighborhood $d > r_2$.*

## 4. Data structure

### 4.1. Grid Neighborhood Queries

We use a 3-dimensional grid to store the sample indices and the occupancy in each octree node. A grid is defined by an axis aligned bounding box which is then regularly subdivided into $width \cdot height \cdot depth$ cells. We take the cubed bounding box with $width = height = depth$, to exploit the axis-symmetry with respect to the center of the box, which we need for fast conflict checking. For a conflict check with radius $r$ and query point $q$ we first compute the morton index [Bae18] of the cell containing the query point $q$. If this cell is free, we test all adjacent cells for conflicts. If any point inside the query ball is found, we reject the point and suspend further grid cell tests.

### 4.2. Octree Neighborhood Queries

The octree stores nodes, which we identify by their 64-bit location code. The location code of a node's children are computed by shifting the node's location code three bits to the left, followed by a bit-wise *OR* operation with the octant number of the child. For example, the root node has a binary location code 1. After the shift operation its binary representation is 1000. For the child octant number 2, in binary 0010, we get it's location code in binary as

$$1 \ll 3 \mid 0010 = 1000 \mid 0010 = 1010$$

, which is in decimal 10. The location code of a node's parent can be computed by shifting its location code three bits to the right i.e. for the location code of child octant 2 of the root node:

$$1010 \gg 3 = 1$$

With this indexing based on the location codes we can pull the hierarchy information out of the octree nodes and we don't have to store pointers inside the nodes. Each octree node stores only its location code, a grid with point indices and an existence flag for each child node. We can easily traverse the octree in Breadth-First-Order and stop on different resolution levels. This will be useful for processing a scan and computing various quantities from local point neighborhoods.

### 4.3. Accelerated search

Each point in the active point list is associated to a node in the octree which we have to check next. Instead of walking downwards from the root node, we can accelerate the search by considering the last layer of the octree nodes, which changed in the last run, first.

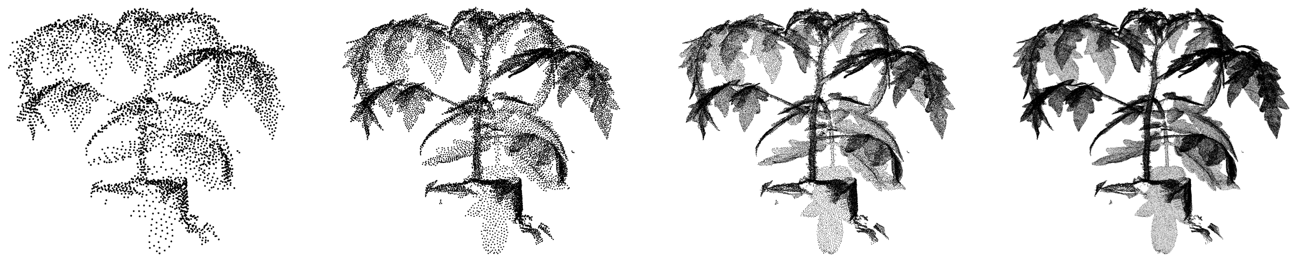

**Figure 7:** *Increasing the resolution from* ***left*** *to* ***right****. The resolution levels contain 4.282, 16.499, 59.509, 219.393 points. The scan contains 6.364.517 points in total.* ***Total time:*** *25.5 seconds.*

For checking conflicts of octree nodes with the query ball we use the inside and overlap test presented in Behley et al. [BSC15]:

If a the query ball is located completely inside the current octree node, it will also be located completely inside the parent node. We can thus query the current node and then proceed with its parent, if it exists. The inside test checks for a point $q$ with query ball $B(q,r)$ of radius $r$ and a node with center $c$ and extent $e$, if any of its transformed coordinates are outside of the node. The query point $q$ is transformed to the local coordinate system of the node $q' = |q-c|$, where $|\cdot|$ denotes the component wise absolute value. The transformed query ball $B(q',r)$ is inside the node if

$$q'^{(j)} + r < e$$

is true $\forall j$.

Otherwise, the transformed query ball can overlap neighboring octree nodes. The transformed query ball $B(q',r)$ overlaps with the a node if the maximum coordinate $q'^{(j)}$ of its midpoint $q'$ fulfills

$$\max_j q'^{(j)} < e + r$$

and at least one of the following two conditions

$$\min_j q'^{(j)} < e \quad \text{or} \quad \|q' - 1 \cdot e\| < r$$

where 1 is the vector $(1,\dots,1)^T$ containing only ones. We first descend from the node's parent. If still no conflict is found, we just descend from the root node. If the query ball and a node do not overlap, the node can be disregarded for the current conflict check.

## 5. Poisson disk sampling Implementation

Each sparse grid has an *std::unordered_map* to store point set indices and a *std::bitset* for fast occupancy queries. We use these data structures as a compromise between fast queries and memory efficiency. Each grid-cell is indexed by morton index [Bae18]. We subdivide the cubed ($width = height = depth$) axis aligned bounding box with extent $e_D$ in hierarchy level $D$ into $width^3$ cells, yielding a cell extent of $e_c = e_D/width$, from which the poisson radius $r_D$ in is computed as

$$r_D = \frac{e_c}{2} \cdot \sqrt{3}$$

Each grid-cell is empty at first. For all points inside the active point list, we start at the first point and then proceed to the next. Given a point we want to insert, we compute its morton index [Bae18] and make an occupancy test in the *std::bitset*. If the cell is not occupied by another point, we query the cell's direct neighborhood. We then test whether the points inside the cell's neighborhood are outside the query ball.

When all tests pass, the point is accepted. We store the point's point set index in the *std::unordered_map* using its morton index and mark the cell occupied in the *std::bitset*. When one test fails, the point is rejected to the respective child node, which is inserted to the next active point list, see Figure 6. We repeat the insertion process after the current active point list is fully processed. We stop if the next active point list is empty. Using 64 bits to represent the location code of an octree node, the octree can reach a maximum depth of 21. Assuming a side length of 1 meter for an object, subdividing 21 times yield lengths in the 100 nanometer range. At this scale any point is as good as another, so we ether reject or replace all samples afterwards.

The octree has an *std::unordered_map* to store the octree nodes by their location code and a pointer to a point set container used to store points. We use this data structure to avoid the construction of a full linear octree and store only active octree nodes in memory. We use Breadth-First-Order traversal to collect all points up to octree depth $D$ to get a low resolution subsampling of the whole point cloud.

## 6. Results

### 6.1. Estimation of Normals

We compare three different sampling strategies, random, closest-to-center and poisson disk sampling. On low resolution levels the random and closest-to-center sampling yield a point set where local surface details are not as visible as in the poisson disk sampling, see Figure 8. The distance constraint in Poisson disk sampling ensures that local point neighborhoods represent the local surface better than the other approaches. The random sampling introduces noise into the sampling, by considering any point in a grid cell. The local neighborhoods thus only approximate a noisy surface. The medioid sampling is biased to the data structure, since it selects the point closest to the grid-cell center and thus cannot select the points responsible for the high frequencies in the normals. The poisson disk sampling selects points, such that local neighborhoods

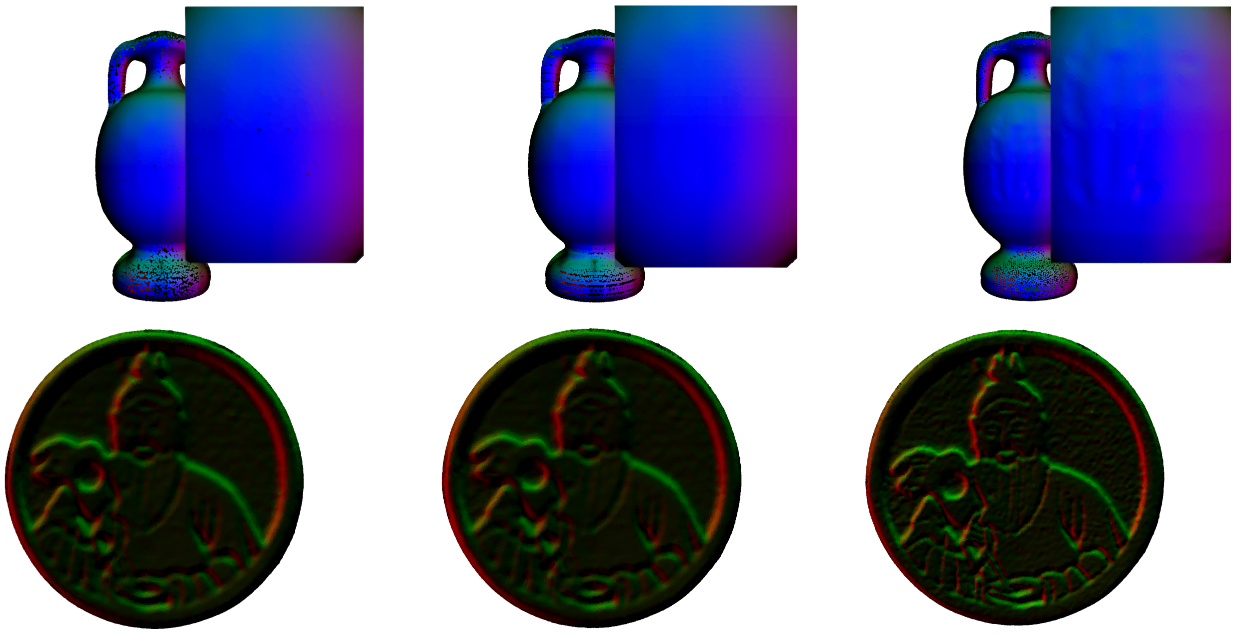

**Figure 8:** ***Top row:*** *Closeup showing the area with the surface details. All samplings of the vase have close to 30.000 points.*
***Bottom row:*** *Samplings of the chinese coin [Art18] contain close to 100.000 points in the same sampling order.*
***Left:*** *Random sampling, surface details are not visible.* ***Middle****: Closest-to-center, details are not surface visible,* ***Right:*** *Poisson disk sampling, surface details are visible.*

represent the local surface much better. For visualization of the normals we plot the Least squares normals as rgb values. We used the splatting scheme from [SW15] to visualize the point clouds as approximately closed surfaces. Using our data structure we can easily switch between resolutions by traversing the octree up- and downwards. Figure 7 shows this resolution traversal on a point cloud of a Laser-scanned tomato plant of approximately 6 million points. For the meshes shown in Figure 8, Figure 9 and Figure 11, we subsample the triangles to generate a point cloud of 300.000 points, which are then inserted into the data structure to generate the hierarchy.

| $N$ | Total time | Total time | Samples per second |
|---|---|---|---|
| | [Yuk15] | our | our |
| 100K | 1.4s | 0.23s | 434K |
| 1M | 14.9s | 2.99s | 334K |
| 10M | 168s | 43s | 232K |
| 20M | 347s | 92s | 217K |
| 30M | 523s | 144s | 208K |
| 40M | 699s | 200s | 200K |
| 50M | 878s | 255s | 196K |
| 60M | - | 339s | 176K |

**Table 1:** *Timings using a single thread on an Intel Core i7-5820K @ 3.30Ghz. The first column shows the point set size. The second shows, the total time for N samples progressive sample elimination. The third shows, the total insertion time for N samples of our approach. The last column shows the number of samples per second of our approach.*

## 6.2. Feature sensitive additive poisson disk sampling

When points are inserted into our data structure, the lower resolution levels are filled first. The points can thus be reordered before the insertion by some precomputed point quantity. We precompute a curvature like measure from the eigenvalues of a local principal component analysis (pca). Given a point we query the octree for all points in a radius $r$. Then we compute the sorted eigenvalues from the local covariance matrices by computing a pca on the points in the $r-$neighborhood. From these eigenvalues $\lambda_i$ we can derive point quantities like sphericity

$$sp = \frac{\lambda_{min}}{\lambda_{max}} \quad \in \{0 \dots 1\}$$

or surface variation

$$sv = \frac{\lambda_{min}}{\sum_i \lambda_i} \quad \in \{0 \dots 1\}.$$

For example, sphericity $sp$ is low if the shape of the neighborhood is flat or line-like and high if the neighborhood is sphere-like. Sorting the points by this quantity in descending order will force the lower resolution grids of the octree to accept sphere-like points first and others only later. This can be seen in the Figures 9 and 11 for the sphericity scalar field. Comparing the dragon in Figure 10 and Figure 11, we can see that the feature aware sampling contains points on the dragon scales, claws, tooths and horns while the poisson sampling in Figure 10 does only contain some of these samples.

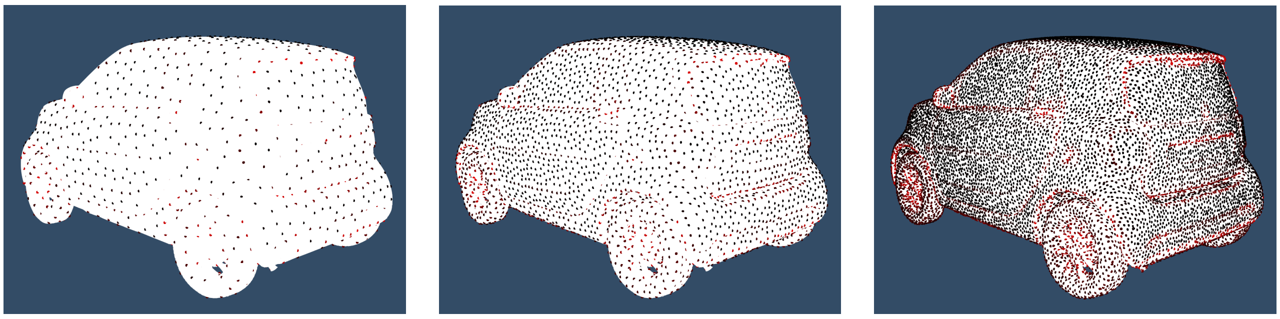

**Figure 9:** *Results on the Smart Car mesh [Art18]. Increasing the resolution from* ***left*** *to* ***right****. The color values go from* ***red/high*** *to* ***black/low****. We can see how the low resolutions contain the desired points.*

### 6.3. Computation Time and theoretical Complexity

Table 1 shows the computation time on synthetic data sets. We randomly generated $N$ samples on triangle-meshes and then inserted the points into our data structure. The first column shows the number of points inserted into the hierarchy. The second show the progressive sample elimination [Yuk15] time in seconds, the third column show the total insertion time into our hierarchy in seconds, while the last column shows the average number of samples processed per second. These results confirm our expected theoretical complexity of $T(N) \in O(N\log^2(N))$ which comes from building the octree and descending the respective depth for each point. It can be calculated as follows:

We have denote by $N$ the total number of points, $k_j$ are the points inserted into layer $j$ and $D$ is the worst case lookup for a conflict. The doubling of the resolution from one layer to the next leads to quadrupling the number of points $k_j = k_0 \cdot 4^j$. We denote the total generation time of the hierarchy by $T(N)$, which gives:

$$\begin{aligned}
T(N) &= N + \sum_{i=1}^{D}(N - \sum_{j=0}^{i-1} k_j)D \\
&= N + \sum_{i=1}^{D}(N - k_0 \sum_{j=0}^{i-1} 4^j)D \\
&= N + \sum_{i=1}^{D}(N - \frac{k_0}{3}(4^i - 1))D \\
&= N + ND^2 - \frac{k_0}{3}D^2 - \frac{k_0}{3}D\sum_{i=1}^{D} 4^i \\
&= N + ND^2 - \frac{k_0}{3}D^2 - \frac{k_0}{3}D\sum_{i=0}^{D-1} 4^{i+1} \\
&= N + ND^2 - \frac{k_0}{3}D^2 - \frac{4k_0}{9}D(4^D - 1) \\
&= N + ND^2 - \frac{k_0}{3}D^2 + \frac{4k_0}{9}D - \frac{4k_0}{9}D4^D
\end{aligned}$$

Since we have $D = \log_8(N)$ we get $\frac{4k_0}{9}\log_8(N)4^{\log_8(N)} = \frac{4k_0}{9}\log_8(N)4^{\frac{2}{3}\log_4(N)} = \frac{4k_0}{9}\log_8(N)N^{\frac{2}{3}}$. The second term dominates which gives $T(N) \in O(N\log^2 N)$.

## 7. Conclusions

We introduced a fast hierarchical poisson disk sampling method to eliminate scan artifacts from laser and lidar scanned point clouds. A simple reordering of the points according to some weights, makes the sampling feature aware to better represent surface features at low resolution levels. We also showed on two examples that using a poisson disk sampling, for the computation of normals, shows finer details than a simple random grid or structured grid sampling. Building on a data structure used in several prominent out-of-core rendering pipelines makes our approach directly applicable to these applications.

**Acknowledgements**

This work has partly been supported by the DFG under the grant number FOR 1505: Mapping on Demand. The authors would like to thank the Stanford Computer Graphics Laboratory for providing the dragon, bunny and buddha meshes in the Stanford 3D Scanning Repository http://graphics.stanford.edu/data/3Dscanrep/, 2018.

## References

[ACSD*03] Alliez P., Cohen-Steiner D., Devillers O., Lévy B., Desbrun M.: Anisotropic polygonal remeshing. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22, ACM, pp. 485–493. 3

[AGY*17] Ahmed A. G., Guo J., Yan D.-M., Franceschia J.-Y., Zhang X., Deussen O.: A simple push-pull algorithm for blue-noise sampling. *IEEE transactions on visualization and computer graphics 23*, 12 (2017), 2496–2508. 3

[Art18] Arttec3D: Smart car. https://www.artec3d.com/3d-models, 2018. 6, 7

[Bae18] Baert J.: Libmorton: C++ morton encoding/decoding library. https://github.com/Forceflow/libmorton, 2018. 4, 5

[BHZK05] Botsch M., Hornung A., Zwicker M., Kobbelt L.: High-quality surface splatting on today's gpus. In *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings* (2005), IEEE, pp. 17–141. 2

[Bri07] Bridson R.: Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH sketches* (2007), p. 22. 3

[BSC15] Behley J., Steinhage V., Cremers A. B.: Efficient Radius Neighbor Seach in Three-dimensional Point Clouds. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)* (2015). 5

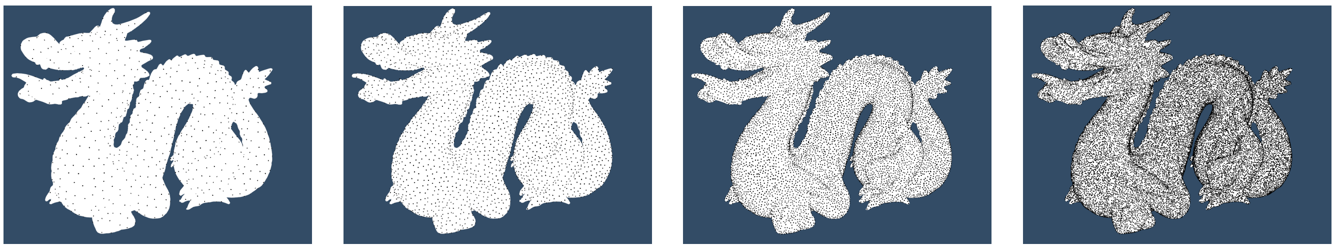

**Figure 10:** *Additive hierarchical poisson disk sampling of the Dragon model. The points in the lower resolution levels are not restricted to features. Increasing the resolution from* ***left*** *to* ***right***

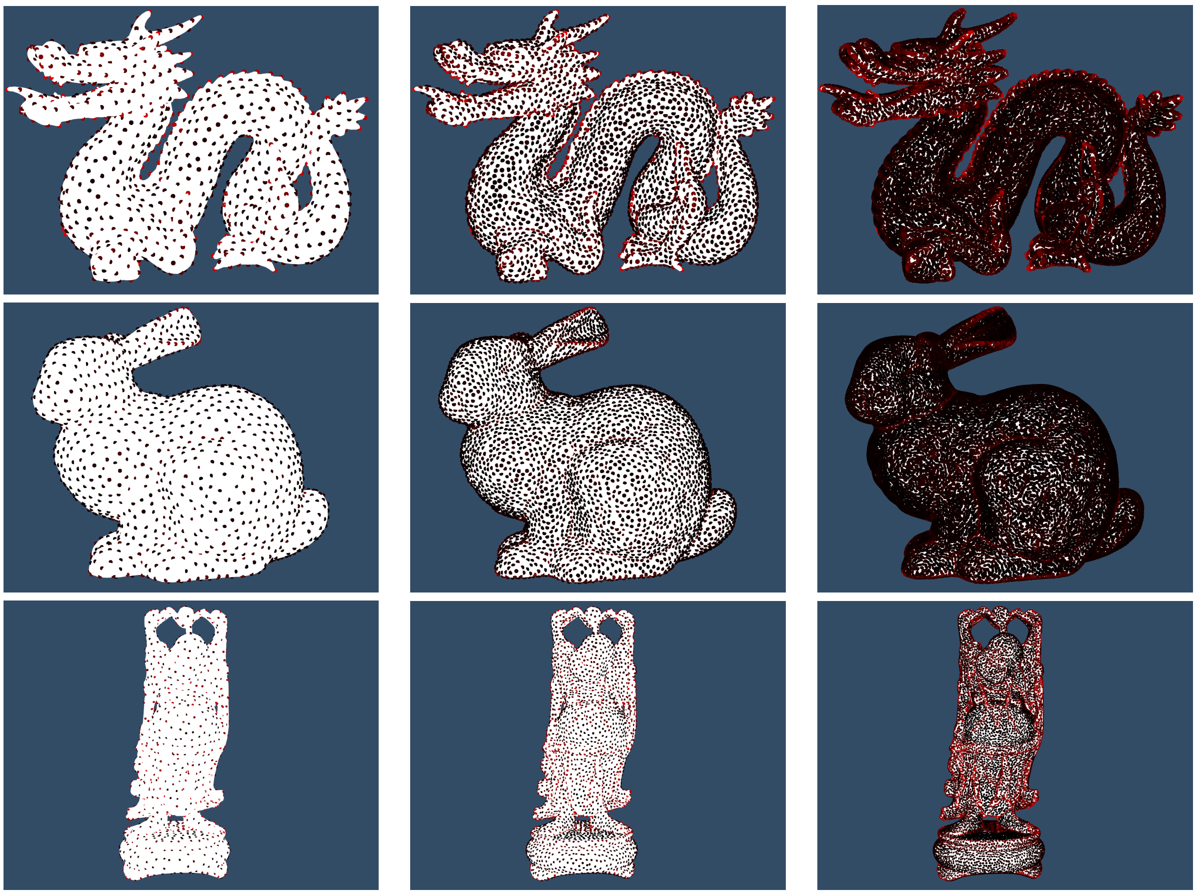

**Figure 11:** *Results on the Stanford meshes. Increasing the resolution from* ***left*** *to* ***right****. The color values go from* ***red/high*** *to* ***black/low****. We can see that the low resolution levels include the points of high sphericity.*

[BSD09] Balzer M., Schlömer T., Deussen O.: *Capacity-constrained point distributions: a variant of Lloyd's method*, vol. 28. ACM, 2009. 3

[CGW*13] Chen J., Ge X., Wei L.-Y., Wang B., Wang Y., Wang H., Fei Y., Qian K.-L., Yong J.-H., Wang W.: Bilateral blue noise sampling. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 216. 3

[Coo86] Cook R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1 (Jan. 1986), 51–72. URL: http://doi.acm.org/10.1145/7529.8927, doi:10.1145/7529.8927. 3

[CSHD03] Cohen M. F., Shade J., Hiller S., Deussen O.: *Wang tiles for image and texture generation*, vol. 22. ACM, 2003. 3

[DGBOD12] De Goes F., Breeden K., Ostromoukhov V., Desbrun M.: Blue noise through optimal transport. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 171. 3

[DW85] Dippé M. A. Z., Wold E. H.: Antialiasing through stochastic sampling. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 69–78. URL: http://doi.acm.org/10.1145/325165.325182, doi:10.1145/325165.325182. 3

[EPM*14] Ebeida M. S., Patney A., Mitchell S. A., Dalbey K. R., Davidson A. A., Owens J. D.: k-d darts: Sampling by k-dimensional flat searches. *ACM Transactions on Graphics (TOG) 33*, 1 (2014), 3. 3

[Fat11] Fattal R.: Blue-noise point sampling using kernel density model. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 48. 3

[GYJZ15] Guo J., Yan D.-M., Jia X., Zhang X.: Efficient maximal poisson-disk sampling and remeshing on surfaces. *Computers & Graphics 46* (2015), 72–79. 3

[HDD*92] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: *Surface reconstruction from unorganized points*, vol. 26. ACM, 1992. 2

[HDK01] Hiller S., Deussen O., Keller A.: Tiled blue noise samples. In *Vision, Modeling, and Visualization (VMV)* (2001). 3

[KCODL06] Kopf J., Cohen-Or D., Deussen O., Lischinski D.: *Recursive Wang tiles for real-time blue noise*, vol. 25. ACM, 2006. 3

[KH13] Kazhdan M., Hoppe H.: Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG) 32*, 3 (2013), 29. 2

[KS12] Kalantari N. K., Sen P.: Fast generation of approximate blue noise point sets. *Comput. Graph. Forum 31*, 4 (2012), 1529–1535. URL: http://dx.doi.org/10.1111/j.1467-8659.2012.03149.x, doi:10.1111/j.1467-8659.2012.03149.x. 3

[LD05a] Lagae A., Dutré P.: A procedural object distribution function. *ACM transactions on graphics (TOG) 24*, 4 (2005), 1442–1461. 3

[LD05b] Lagae A., Dutré Ph.: A procedural object distribution function. *ACM Transactions on Graphics 24*, 4 (October 2005), 1442–1461. URL: http://doi.acm.org/10.1145/1095878.1095888, doi:10.1145/1095878.1095888. 3

[LWSF10] Li H., Wei L.-Y., Sander P. V., Fu C.-W.: Anisotropic blue noise sampling. *ACM Transactions on Graphics (TOG) 29*, 6 (2010), 167. 3

[MEA*18] Mitchell S. A., Ebeida M. S., Awad M. A., Park C., Patney A., Rushdi A. A., Swiler L. P., Manocha D., Wei L.-Y.: Spoke-darts for high-dimensional blue-noise sampling. *ACM Transactions on Graphics (TOG) 37*, 2 (2018), 22. 3

[MF92] McCool M., Fiume E.: Hierarchical poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface* (1992), vol. 92, pp. 94–105. 3

[Mit87] Mitchell D. P.: Generating antialiased images at low sampling densities. In *ACM SIGGRAPH Computer Graphics* (1987), vol. 21, ACM, pp. 65–72. 2

[MREB12] Mitchell S. A., Rand A., Ebeida M. S., Bajaj C.: Variable radii poisson-disk sampling, extended version. In *Proceedings of the 24th canadian conference on computational geometry* (2012), vol. 5. 3

[NBH18] Nasikun A., Brandt C., Hildebrandt K.: Fast approximation of laplace-beltrami eigenproblems. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 121–134. 3

[ÖAG10] Öztireli A. C., Alexa M., Gross M.: Spectral sampling of manifolds. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 168. 3

[SB12] Schechter H., Bridson R.: Ghost sph for animating water. *ACM Trans. Graph. 31*, 4 (July 2012), 61:1–61:8. URL: http://doi.acm.org/10.1145/2185520.2185557, doi:10.1145/2185520.2185557. 3

[Sch14] Scheiblauer C.: *Interactions with Gigantic Point Clouds*. PhD thesis, 2014. 1, 2, 3

[Sch16] Schütz M.: Potree: Rendering large point clouds in web browsers. 3

[SD11] Schlömer T., Deussen O.: Accurate spectral analysis of two-dimensional point sets. *Journal of Graphics, GPU, and Game Tools 15*, 3 (2011), 152–160. 2, 3, 4

[SW15] Schütz M., Wimmer M.: High-quality point-based rendering using fast single-pass interpolation. In *Digital Heritage, 2015* (2015), vol. 1, IEEE, pp. 369–372. 6

[SZG*13] Sun X., Zhou K., Guo J., Xie G., Pan J., Wang W., Guo B.: Line segment sampling with blue-noise properties. *ACM Trans. Graph. 32*, 4 (2013), 127–1. 3

[Uli88] Ulichney R. A.: Dithering with blue noise. *Proceedings of the IEEE 76*, 1 (1988), 56–79. 2

[Wei08] Wei L.-Y.: Parallel poisson disk sampling. *ACM Transactions on Graphics (TOG) 27*, 3 (2008), 20. 3

[WPC*14] Wachtel F., Pilleboue A., Coeurjolly D., Breeden K., Singh G., Cathelin G., De Goes F., Desbrun M., Ostromoukhov V.: Fast tile-based adaptive sampling with user-specified fourier spectra. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 56. 3

[WS06] Wimmer M., Scheiblauer C.: Instant points: Fast rendering of unprocessed point clouds. In *SPBG* (2006), pp. 129–136. 2

[YGW*15] Yan D.-M., Guo J.-W., Wang B., Zhang X.-P., Wonka P.: A survey of blue-noise sampling and its applications. *Journal of Computer Science and Technology 30*, 3 (2015), 439–452. 3

[Yuk15] Yuksel C.: Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 25–32. 2, 3, 6, 7

[YW13] Yan D.-M., Wonka P.: Gap processing for adaptive maximal poisson-disk sampling. *ACM Transactions on Graphics (TOG) 32*, 5 (2013), 148. 3