DOI: 10.1111/cgf.12442
Eurographics Symposium on Geometry Processing 2014
Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

*Volume 33 (2014), Number 5*

# Blue-Noise Remeshing with Farthest Point Optimization

Dong-Ming Yan[1,2†], Jianwei Guo[2], Xiaohong Jia[3], Xiaopeng Zhang[2†], Peter Wonka[1,4]

[1]KAUST [2]NLPR, Institute of Automation, CAS [3]KLMM, AMSS, CAS [4]Arizona State Univ.

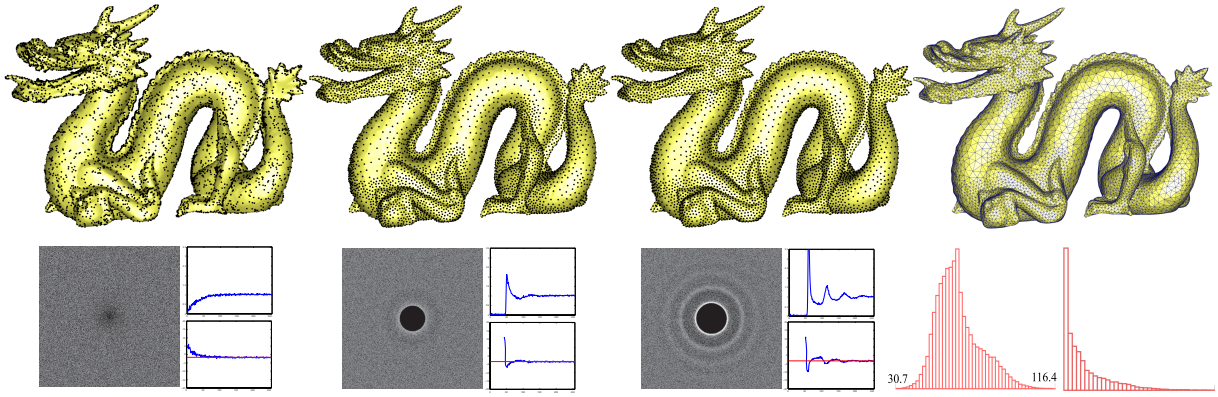


**Figure 1:** *Farthest point optimization on the Dragon model (*15*k samples). From left to right: initial sampling, the result of the initial optimization by shortest edge removal (Section 4.3), the final result and the remeshing. Bottom row left: spectral analysis of the corresponding sampled point set. Bottom row right: angle and edge length distribution of the remeshing.*

**Abstract**

*In this paper, we present a novel method for surface sampling and remeshing with good blue-noise properties. Our approach is based on the farthest point optimization (FPO), a relaxation technique that generates high quality blue-noise point sets in 2D. We propose two important generalizations of the original FPO framework: adaptive sampling and sampling on surfaces. A simple and efficient algorithm for accelerating the FPO framework is also proposed. Experimental results show that the generalized FPO generates point sets with excellent blue-noise properties for adaptive and surface sampling. Furthermore, we demonstrate that our remeshing quality is superior to the current state-of-the-art approaches.*

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Blue-noise sampling and remeshing

## 1. Introduction

Generating uniformly distributed random point sets on surfaces is essential for many applications in computer graphics, such as rendering, geometry processing, and physical simulation. Among all the existing sampling techniques, blue-noise sampling [Uli87] has received most of the attention.

Most previous blue-noise sampling approaches only handle the 2D Euclidean space. More recently, several classic 2D blue-noise sampling methods have been generalized to surfaces, such as Poisson-disk sampling [CJR*09, BWWM10, CCS12, YW13] and Lloyd relaxation [CYC*12, XHGL12]. Furthermore, Wei and Wang [WW11] propose a new technique, called *Differential Domain Analysis*, to evaluate the Fourier power spectrum of non-uniformly sampled point sets on surfaces. This work provides an evaluation tool of the blue noise property and hence facilitates the comparison of different approaches.

In rendering applications, blue-noise sampling is used for

† Corresponding authors: X. Zhang (xpzhang@ia.ac.cn), D.-M. Yan (yandongming@gmail.com)

anti-aliasing [Mit87] and raytracing [Mit91]. In the context of surface sampling, recent work explained why blue-noise properties are an important factor for solving certain PDEs [SB12] (like water animation), objects placement [CYC*12], and stippling [MIPS14], on surfaces. For these applications, blue-noise properties of the point distributions should therefore be considered as additional quality criterion in an addition to traditional mesh quality metrics [FB97], e.g., maximal smallest mesh angle.

In this paper, we propose a new method for surface sampling and remeshing with high blue-noise properties based on the farthest point optimization. We make two important extensions of the FPO algorithm introduced by Schlömer et al. [SHD11]. First, we generalize the concept of FPO to non-uniform sampling according to a density function defined over the sampling domain. Second, we generalize FPO to mesh surfaces, e.g., see Figure 1. In Section 6, we will demonstrate that the sampled point sets result in high quality remeshings, which are superior to the current state-of-the-art approaches. The main contributions of this paper include:

- A generalized version of the farthest point optimization for non-uniform sampling.
- A complete framework for blue-noise sampling and remeshing on surfaces using non-uniform FPO.
- A simple and efficient algorithm for acceleration of the FPO computation.

### 1.1. Related work

We briefly review techniques for the generation of random point sets in 2D and on surfaces.

**2D Sampling:** There are a large number of sampling algorithms that produce point sets with different attributes [Llo82, Coo86, Mit87, Mit91, MF92]. These techniques can be classified into three types: 1) dart-throwing and its variants, 2) iterative optimization and 3) patch-based or rule-based synthesis. Here, we review only the iterative optimization based approach, which is most related to our approach. A more extensive survey of 2D blue-noise sampling techniques is presented by Lagae and Dutré [LD08].

Iterative relaxation is one of the most important techniques for point set sampling, and our approach falls into this category. Lloyd [Llo82] first proposes such an algorithm for signal quantization. Balzer et al. [BSD09] introduce the *capacity constrained Voronoi tessellation* (CCVT), which iteratively equalizes the area of the Voronoi cells of a point set. The generated point sets exhibit great blue-noise properties. Schmaltz et al. [SGBW10] formulate the point sampling problem by simulating electrostatic forces. The stable status is reached when the forces between particles are balanced. Fattal [Fat11] presents an adaptive sampling algorithm based on kernel density estimation. de Goes et al. [dGBOD12] generate the blue-noise point sets using optimal transport. Schlömer et al. [SHD11] propose to maximize the minimal distance between points by Delaunay removal and insertion, called *farthest point optimization*. The resulting point set exhibits excellent blue-noise properties. Chen and Gotsman [CG12] parallelize the FPO framework of [SHD11] via local Delaunay triangulation. However, the above FPO approaches can handle only 2D uniform sampling.

**Surfaces Sampling:** The classic dart-throwing algorithm [Coo86] has been generalized to surfaces. Cline et al. [CJR*09] maintain a hierarchical primitive list for placing Poisson-disks on surfaces. Corsini et al. [CCS12] first generate a dense point set by Monte-Carlo sampling, and then compute a Poisson-disk set by hierarchical cell-based subdivision. Yan and Wonka [YW13] propose a framework for maximal Poisson-disk sampling and remeshing on surfaces based on gap processing. Then, Yan et al. [YWW14]generalize the MPS framework for unbiased isosurface sampling and meshing. A typical drawback of the dart-throwing based approach is that one cannot explicitly control the number of sampled points, which might be important for certain applications. Öztireli et al. [OAG10] solve the problem of finding optimal sampling conditions based on the spectral analysis of manifolds. Bowers et al. [BWWM10] extend the parallel Poisson-disk sampling technique [Wei08] to surfaces. They first propose a spectral analysis method for surfaces sampling, but this method can only be used for analyzing uniform sampling. Wei and Wang introduce the *Differential Domain Analysis* (DDA) technique for analyzing the spectral properties of non-uniformly sampled point sets, as well as for surface sampling. With this tool, one is able to analyze the blue-noise properties of various methods on surfaces.

More recently, Xu et al. [XHGL12] extend the CCVT [BSD09] to surfaces, and Chen et al. [CYC*12] combine the *Centroidal Voronoi Tessellation* (CVT) [YLL*09] and the CCVT [BSD09] for blue-noise sampling on surfaces. Both approaches are based on iterative optimization. Chen et al. [CGW*13] present a blue noise sampling method considering both spatial and non-spatial (feature) properties of the sampling domain. The key idea is to modulate the spatial sample position with a per sample attributes measurement. Medeiros et al. [MIPS14] propose an algorithm for fast blue-noise sampling on polygonal models. Applications such as surface stippling and non-photo realistic rendering are demonstrated. However, these approaches do not consider remeshing as an application.

Once a point set is generated, a mesh can be extracted using a mesh extraction algorithm. There are many possible choices, e.g., Delaunay mesh generation [CDS12]. For remeshing without blue-noise properties, we refer the reader to a survey paper by Alliez et al. [AUGA08]. In Section 6, we make a detailed comparison of the blue-noise properties and the remeshing quality of the current state-of-the-art approaches.

## 2. Overview

In this paper, we present a new method for blue-noise surface sampling and remeshing. Our approach is a generalization of the farthest point optimization [SHD11] to non-uniform and surface sampling. In this section, we first briefly review the FPO framework, and then present our key ideas.

### 2.1. Farthest point optimization

The input is a sampling domain $\Omega$ (2D plane for the FPO algorithm) and an initially sampled point set $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. We denote by $d_x$ the minimum distance for any other point in $\mathbf{X}$ to the point $\mathbf{x}$, $d_{\mathbf{X}}$ the global minimum distance for any pair of points, and $\bar{d}_{\mathbf{X}}$ the average of all the minimum distances $d_x$. Theoretically, the largest minimum distance between any two points is $d_{max} = \sqrt{(2/\sqrt{3}n)}$ if the points in $\mathbf{X}$ form a hexagonal lattice [LD08, SHD11]. As a result, we can normalize $d_{\mathbf{X}}$ and $\bar{d}_{\mathbf{X}}$ by $d_{max}$ to get $\delta_{\mathbf{X}} = \frac{d_{\mathbf{X}}}{d_{max}}$ and $\bar{\delta}_{\mathbf{X}} = \frac{\bar{d}_{\mathbf{X}}}{d_{max}}$, which are used to measure the distribution quality of a point set.

Starting from a random point set $\mathbf{X}$, the FPO algorithm iteratively optimizes the positions of the points by maximizing the minimal distance $d_x$. In each iteration, this algorithm proceeds by repeatedly removing each point from $\mathbf{X}$ and reinserting it back to a new position (called "farthest point") that is as far away from the remaining points $\mathbf{X}\backslash\{\mathbf{x}\}$ as possible. Here a full iteration means that every point in $\mathbf{X}$ has been moved once. In general, the farthest point is the circumscribed center of the largest empty circle in the domain, which can be computed efficiently using the Delaunay triangulation and the Voronoi diagram of the points. Since $\delta_{\mathbf{X}}$ does not decrease in each iteration, the FPO always converges to get a well-distributed point set with a large global minimum distance ($\delta_{\mathbf{X}} > 0.9$ ). The main steps of the FPO algorithm are given in Algorithm 1.

**Algorithm 1:** Farthest point optimization algorithm

```
1  Initial point set X with n points;
2  repeat
3      foreach x_i ∈ X do
4          Compute its local minimum distance
           d_{x_i} = min_{∀x_j ∈ X\{x_i}} distance(x_i, x_j) ;
5          Remove x_i from X, then find the largest empty
           circle cir(c, r) with center c and radius r;
6          if r > d_{x_i} then
7              Insert x_i at the new position c;
8          else                    /* x_i is not moved */
9              Re-insert x_i back to X at its old position;
10 until no point x_i moved, the algorithm has converged;
```

### 2.2. Generalizations

The main contribution of this paper is to generalize the FPO to non-uniform sampling and surface sampling. In our approach, the sampling domain $\Omega$ is either a 2D plane or a 3D surface. A sizing function $\rho(\mathbf{x})$ is defined over the domain to guide the sampling density. We iteratively optimize the point positions to maximize the minimal distance between samples according to this sizing function.

The optimization framework for non-uniform sampling and surface sampling remains similar to [SHD11]. In non-uniform sampling, each sampled point is equipped with a weight. We use the weighted Delaunay triangulation (regular triangulation) and the power diagram instead of the ordinary Delaunay triangulation and the Voronoi diagram for computation. In the following, we will explain the details of non-uniform sampling (Section 3) and surface sampling (Section 4).

## 3. Non-uniform FPO in 2D

In this section, we present the algorithm for non-uniform sampling with the farthest point optimization in the 2D plane. Our approach is based on the concept of the regular triangulation and the power diagram [Aur87]. In the following, we first give the definition of the regular triangulation and the power diagram, and then explain the key ideas for non-uniform sampling.

### 3.1. Regular triangulation and power diagram

The regular triangulation and power diagram are a generalization of the ordinary Delaunay triangulation and Voronoi diagram. They are constructed based on a weighted point set $\mathbf{P}_w = \{\mathbf{p}_i, w_i\}_{i=1}^n$ in domain $\Omega$, where each $\mathbf{p}_i$ is a point and each $w_i$ is a scalar value called the weight of point $\mathbf{p}_i$. Alternatively, each weighted point $(\mathbf{p}_i, w_i)$ can be regarded as a disk (e.g., a circle in 2D or a sphere in 3D) with center $\mathbf{p}_i$ and radius $r_i = \sqrt{w_i}$. To compute the distance between two weighted points $\mathbf{p}_i$ and $\mathbf{p}_j$, we use the power distance defined by Equation 1 instead of the Euclidean distance.

$$\Pi(\mathbf{p}_i, w_i; \mathbf{p}_j, w_j) = \|\mathbf{p}_i - \mathbf{p}_j\|^2 - w_i - w_j \qquad (1)$$

The power diagram of $\mathbf{P}_w$, consisting of a collection of power cells $\{V_i\}_{i=1}^n$, is a weighted Voronoi tessellation of the domain $\Omega$, where the power cell $V_i$ of a weighted point $\mathbf{p}_i$ is given by

$$V_i = \{\mathbf{x} \in \Omega \mid \Pi(\mathbf{p}_i, w_i; \mathbf{x}, 0) < \Pi(\mathbf{p}_j, w_j; \mathbf{x}, 0), \forall j \neq i\}. \quad (2)$$

The dual structure of the power diagram is the regular triangulation. Figure 2 shows an example of the power diagram and regular triangulation. If the weights of all points are the same, the power diagram and regular triangulation are equivalent to Voronoi diagram and Delaunay triangulation.

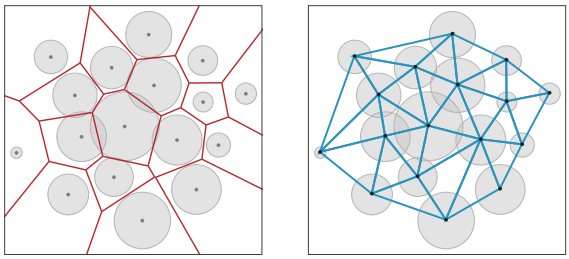

**Figure 2:** *An example of the power diagram (left) and the regular triangulation (right) of a weighted point set.*

### 3.2. Implementation

Initially, we sample a random weighted point set $\mathbf{P}_w$ with respect to a certain sizing function $\rho(\mathbf{x})$, which indicates the local edge length. In our 2D implementation, we use the local feature size (lfs) introduced in [ABK98] as the sizing function, as shown in Figure 4(a). For each point $\mathbf{p}_i$, we define its weight $w_i = r_i^2$, where $r_i = k\rho(\mathbf{p}_i)$ is proportional to its local edge length ($k$ is a constant). Then our non-uniform FPO algorithm iteratively optimizes the point positions. To speed up the algorithm, we maintain a dynamic global regular triangulation $RT(\mathbf{P}_w)$, in which each triangle keeps its dual power vertex of the power diagram. In addition, a priority queue $PQ$ of the triangles in $RT(\mathbf{P}_w)$ is also maintained, where the priority of a triangle $t$ is the radius of its corresponding circumcircle. Note that the radius of the circumcircle is computed by the power distance between one vertex of $t$ and its dual power vertex.

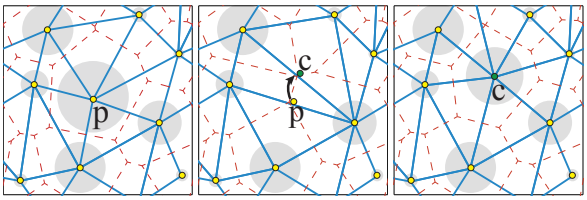


**Figure 3:** *(Left) An initial point set, and* $\mathbf{p}$ *is to be deleted. (Middle) Delete* $\mathbf{p}$ *from RT then find the farthest point at* $\mathbf{c}$*. (Right) Inset* $\mathbf{c}$ *back to RT. This illustration uses periodic boundary conditions.*

In each iteration, non-uniform FPO removes each point from $RT(\mathbf{P}_w)$, finds the farthest point (the center of the first circumscribed circle in $PQ$) from the remaining triangles and finally reinserts the new point back to the $RT(\mathbf{P}_w)$. Figure 3 shows the process of deleting one point and inserting the farthest point. The algorithm converges when no point is moved during one iteration. In our experience, we can stop the iteration as soon as the global minimum distance $\delta_{\mathbf{X}}$ is larger than a threshold (we set it to 0.926). Similar to [SHD11], the data structure changes of $RT(\mathbf{P}_w)$ and $PQ$ caused by removal or insertion operations are updated dynamically and

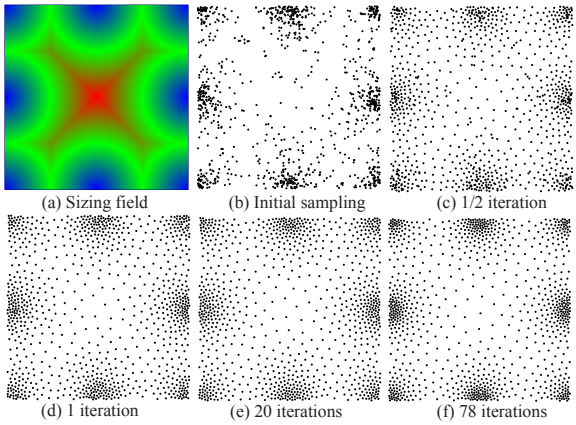


**Figure 4:** *2D non-uniform FPO of 1024 random points. In (a), red color indicates a higher value, and blue color indicates a lower value of the sizing function.*

locally. Since on average the local updating of $RT(\mathbf{P}_w)$ requires $O(1)$ time and maintaining $PQ$ for the largest empty circle requires $O(\log n)$ time, the runtime complexity for each iteration is $O(n \log n)$, where $n$ is the number of the sample points. Though in the worst case, the local updating of $RT(\mathbf{P}_w)$ could requires $O(n)$ time, our approach performs well in practice. Figure 4 shows one optimization result of a random point set with 1024 points. After one iteration the point set is already well-distributed.

## 4. FPO on Surfaces

In this section, we describe our approach for farthest point optimization on surfaces. We assume a 2-manifold input domain $\Omega$ that consists of a set of triangles $\{t_i\}_{i=1}^m$. The sizing function $\rho(\mathbf{x})$ is typically defined by the curvature, or the local feature size.

The main challenge of computing FPO on surfaces is to find the largest empty circle around each point in the domain and to insert new samples in the right location. Generally, it might seem natural to use the geodesic power diagram for optimization. However, the computation of geodesic power diagram is time consuming and not robust if the input domain contains badly-shaped triangles. Hence, we opt for the restricted power diagram on surfaces, which is a good approximation of the geodesic power diagram when the sampling density is high. More importantly, the state of the art remeshing algorithms are based on Euclidean distances and not geodesic distances.

### 4.1. Restricted power diagram

The restricted power diagram (RPD) is defined as the intersection of the 3D power diagram $\{V_i\}$ and the input mesh $\Omega$.

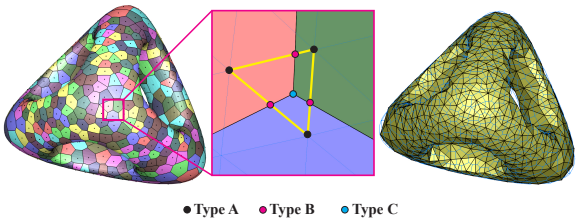


**Figure 5:** *Illustration of the RPD (left), three types of intersecting vertices (middle), and RRT (right) on a mesh surface. The yellow triangle is a triangle of the input surface.*

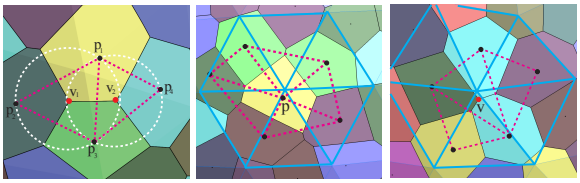


**Figure 6:** *(a) Illustration of the circumcircles of two restricted regular triangles $t_{p_1,p_2,p_3}$ and $t_{p_3,p_4,p_1}$ on a surface. The dashed lines indicate that the edges or the circles may not be on the surface exactly. (b) and (c) shows the input triangles (solid blue) and restricted regular triangles (dashed rose-red) that will be influenced by deleting one point* **p** *or inserting one point at position* **v**.

The dual of the RPD is called the restricted regular triangulation (RRT). The concepts of RPD and RRT are the generalizations of the restricted Voronoi diagram and the restricted Delaunay triangulation [ES97, YLL*09]. Figure 5 illustrates the RPD and the RRT on a mesh surface. Each cell of the RPD is called a restricted power cell and has three types of vertices.

- Type A: the original vertex of the input mesh.
- Type B: the intersection of a bisecting plane of the power diagram and an edge of the input mesh.
- Type C: the intersection of an edge of the power diagram and a mesh triangle. We call this type of vertex a restricted power vertex.

We use the implementation of [YLL*09] for the symbolic representation of the vertices of the RPD, so that each vertex of the RPD is represented as the intersection of three planes.

### 4.2. Point insertion and deletion

We use the RPD and RRT for computing FPO on surfaces. We first illustrate the general method directly derived from Algorithm 1, which requires $O(n^2)$ time for each iteration, where $n$ is the number of sampling points. Then we present a method to update the RPD and RRT locally to speed up this process.

First, we generate an initial point set by randomly sampling $n$ points on the input triangles, where $n$ is the user specified number of points. In the uniform case, to generate a random point on the input mesh, we randomly select a triangle with the probability proportional to the triangle area. Then we randomly generate a point inside the selected triangle. In the non-uniform case, the probability of the triangle selection is proportional to the weighted area of each triangle. Different from the 2D case, the farthest point is no longer the center of the circumcircle of a restricted regular triangle, because the center may not be located on the surface. In this paper, we redefine the circumcircle of a restricted regular triangle $t$ as $cir_t(\mathbf{v}, r)$ where $\mathbf{v}$ is the corresponding restricted power vertex and $r$ is the power distance between any vertex of $t$ and $\mathbf{v}$. Figure 6 (a) shows an example of circumcircles. The key insight for point insertion is that the farthest point is always one of the restricted power vertices. Now given an initial sampling point set, we can iteratively optimize it just as in the 2D domain. However, when removing one point from the point set, we should compute the RPD and RRT from scratch and traverse all the restricted power vertices again to find the farthest point. As a result, the algorithm requires $O(n^2)$ time for each iteration.

To compute the RPD, Yan and Wonka [YW13] start from clipping one input triangle with its incident cells (the sampling points) and computing the three types of vertices. Then they use a propagation method to clip other triangles. Based on this observation, we present a new method to modify the RPD and RRT locally and apply this method to speed up FPO. In the initial stage, we build a global RPD and RRT of the point set, and use a priority queue to maintain the restricted regular triangles (the priority is set to the radius of its circumcircle). When deleting or inserting one point, we can update the RPD and RRT locally. As shown in Figure 6(b), when deleting one point **p**, we first delete all the restricted regular triangles (dashed rose-red ones in this figure) that contain **p** from the RRT. Then we find the input triangles (solid blue) that are affected by **p** and re-clip them with their other incident cells. Similarly, when inserting one point at position **v** (Figure 6(c)), we first remove the affected restricted regular triangles and then recompute the local RPD and RRT in the neighborhood of **v**. The priority queue is also updated in this process.

We analyze the runtime complexity of the proposed algorithm. First, the local updating of RPD and RRT can be done in constant time by using proper data structures. In our implementation, we maintain a list of cell labels for each input triangle. Each cell in this list will cover a part of this triangle. For each cell, we record the input triangles that are fully or partially covered by it and the restricted regular triangles that connect to it. When deleting or inserting one point, we just inspect the affected input or restricted regular triangles in the neighborhood of that point. Thus, we can recompute

the RPD and the RRT locally in $O(1)$ time. Second, we use a max-heap to implement the priority queue, and deleting or inserting one element costs $O(\log n)$ time. Overall, the time required for one iteration is $O(n \log n)$.

### 4.3. Acceleration

Although FPO can generate point sets with good blue-noise properties, the performance can be improved since it usually takes $60-100$ iterations to converge. In this section, we present an efficient approach to accelerate the sampling speed. This algorithm can be regarded as a preprocessing step of the point set. After the preprocessing, we need fewer FPO iterations to reach the convergence criteria.

Since the key point of FPO is to maximize the global minimum distance $\delta_{\mathbf{X}}$, the shortest edge between all pairs of points must be removed until the algorithm converges. This fact motivates us to present the shortest edge removal (SER) algorithm. Starting from the initially sampled point set, we select the shortest edge every time and remove one of its endpoint. Then we search for the farthest point on the input mesh to insert the point back into the point set. The strategy of point insertion and deletion is the same as in FPO. This process is performed repeatedly until no point is moved (meaning that we insert the farthest point to the original position). As we will see in Section 6, a point set generated by this algorithm already has good blue-noise properties.

In our experiments, there are about 15% to 20% points in the initial set that have never been moved. The shortest edge removal algorithm converges to achieve a global minimum distance of $\delta_{\mathbf{X}} \approx 0.78$, while $\delta_{\mathbf{X}}$ in FPO is larger than 0.9. Fortunately, if we perform our FPO algorithm after shortest edge removal, almost all the points (the percentage is larger than 99%) will be moved locally around their original positions. Thus instead of maintaining a priority queue for all empty circles, we can search for the farthest point locally, for example only inside the 2-ring neighborhood of the current point. In this way, finding the largest empty circle only requires $O(g^2)$ time, where $g$ is the average number of neighbors in the RRT. As $g$ is usually independent of $n$, one iteration requires only $O(n)$ time. This idea is similar to the local variant of FPO in [SHD11], but the convergence speed of our new method is much faster.

## 5. Surface Remeshing

In our implementation, we use the lfs [ABK98] as the sizing function to guide the sampling density. If the sampling density satisfies the requirement of the $\varepsilon$-sampling property [ABK98] and the topological ball property [ES97], then each restricted power cell is a single connected component and the RRT is homomorphic to the input mesh. After optimizing the point locations, we apply the mesh extraction algorithm [YLL*09] to compute the remeshing.

**Angle optimization:** Our uniform FPO sampling is always a maximal Poisson-disk sampling (shown later in Section 6), which exhibits an angle bound $[30^o, 120^o]$. However, our adaptive sampling does not have this property, and some angles are smaller than $30^o$ (termed as bad angles), although the percentage is usually lower than 0.1%. To improve the remeshing quality, we propose a simple method for angle optimization. For each bad triangle in the output remeshing, we find the vertex that corresponds to a bad angle and set the weights of the other two vertices to smaller values ($cw_i$, we set $c$ to be $0.4 \sim 0.8$ empirically). Then we simply resample these vertices with the algorithm, respectively. The vertices with bad angles should be handled first. The reason that we decrease the weights of the vertices is that it can make the bad samples move closer to them, so as to make the bad angles larger. The optimization terminates when there are no bad angles.

**Feature preservation:** To preserve the features or boundaries, we implement a feature-aware sampling step. We first identify the features of the input mesh, including boundary edges, creases and corners, which are organized in the form of 1D curved skeletons. Then we use the method introduced in [AVDI05] to determine how many points will be sampled on the features. The corners are inserted directly as sample points and we perform 1D FPO on the feature curves to generate other feature samples. Finally the feature samples remain fixed in the later FPO optimization to sample the interior of the surface.

## 6. Experimental Results

We first present the experimental results to demonstrate that the proposed algorithm is able to generate high quality sample distributions and remeshing results. Then we evaluate the convergence and performance of this algorithm. We use the CGAL [cga] library for computing the regular triangulation and the power diagram. Our tests are conducted on a PC with Intel i7-3770, 3.40 GHz CPU, 16GB memory and a 64-bit Windows 7 operating system.

**Spectral analysis:** First we apply the differential domain analysis [WW11] to analyze the spectral properties of the point sets sampled on surfaces, including the power spectrum, radially average and anisotropy. We show that our sampling have better blue-noise properties compared with previous work. Figure 7 and Figure 8 illustrate the comparison results for uniform and adaptive sampling respectively. Table 1 lists the acronyms of different sampling methods used for comparison. From these results, we can see that except CVT, all the other algorithms can produce blue-noise distributions, while the spectra of our FPO algorithm exhibit the best blue-noise properties with low anisotropy and lack of low frequencies around the origin. In addition, the power spectrum of our algorithm has a residual peak decaying slower than all the other methods. The point sets of CVT show strong spikes in the power spectrum and fail to exhibit

| Acro. | Ref. | Full Name |
|---|---|---|
| CVT | [YLL*09] | Centroidal Voronoi Tessellation |
| MPS | [YW13] | Maximal Poisson-disk Sampling |
| VBS | [CYC*12] | Variational Blue Noise Sampling |
| PPS | [BWWM10] | Parallel Poisson-disk Sampling |
| CCST | [XHGL12] | Capacity-Constrained Surf. Triangulation |
| CPS | [CCS12] | Constrained Poisson-disk Sampling |
| HPS | [MIPS14] | Hierarchical Poisson-disk Sampling |
| SER | Ours | Shortest Edge Removal |
| FPO | Ours | Farthest Point Optimization |

**Table 1:** *Acronyms of different sampling methods.*

blue-noise patterns. Figure 9 shows the spectral analysis of our feature-aware samplings.

**Remeshing results:** Next, we list the statistics of remeshing quality compared with previous methods. Table 2 is for uniform remeshing and Table 3 is for adaptive remeshing. The comparison contains various criteria that are used in recent remehsing papers. The quality of a triangle is measured by $Q_t = \frac{6}{\sqrt{3}} \frac{s_t}{p_t h_t}$, where $s_t$ is the area of $t$, $p_t$ is the half-perimeter of $t$ and $h_t$ is the longest edge length of $t$ [FB97]. Here $Q_{min}$ and $Q_{avg}$ are the minimal and average triangle quality; $\theta_{min}$ and $\theta_{max}$ are the minimal and maximal angle, and $\bar{\theta}_{min}$ is the average of minimal angles of all triangles; $\theta_{<30^o}$ is the ratio of angles smaller than $30^o$; $V_{567}$ is the percent of vertices with valences 5, 6 and 7; $d_H$ is the Hausdorff distance (divided by the diagonal length of the input mesh bounding box) between input mesh and remeshing result, measured with the Metro tool [CRS98]. Both of these tables suggest that our approach exhibits better $Q$, $\theta$ and $V_{567}$ than previous methods with blue-noise properties. Our $d_H$ is not the best but it is still competitive with all the other methods. The best results without blue-noise property are highlighted in gray, while the best results with blue-noise property are highlighted in bold font. For the uniform remeshing, CVT always has the best meshing quality since it tends to generate a point distribution with regular patterns which is undesirable for applications such as animating fluids [SB12]. Table 4 lists the results of boundary and feature handling. We only compare with CVT and MPS since the boundary and feature preservation are not addressed by the other methods. The remeshing results are provided in the supplemental materials.

**Convergence:** Figure 10 (left) illustrates the convergence speed of our FPO algorithm. We compute the minimum ($\delta_{\mathbf{X}}$) and average minimum ($\bar{\delta}_{\mathbf{X}}$) edge length from the remeshing result after each iteration. The values are averaged over different models, including Venus, Eight, Homer and Bunny. From this figure we can see that $\delta_{\mathbf{X}}$ and $\bar{\delta}_{\mathbf{X}}$ increase rapidly at the first few iterations and then converge more slowly. On average, after 60-100 iterations, they converge toward the maximum values (0.926 for $\delta_{\mathbf{X}}$ and 0.933 for $\bar{\delta}_{\mathbf{X}}$). It is also interesting to note that after one or two iterations, our algorithm has produced well-distributed point sets with very similar power spectrum to that of Poisson-disk sampling such as MPS, as shown in Figure 1, 7 and 8.

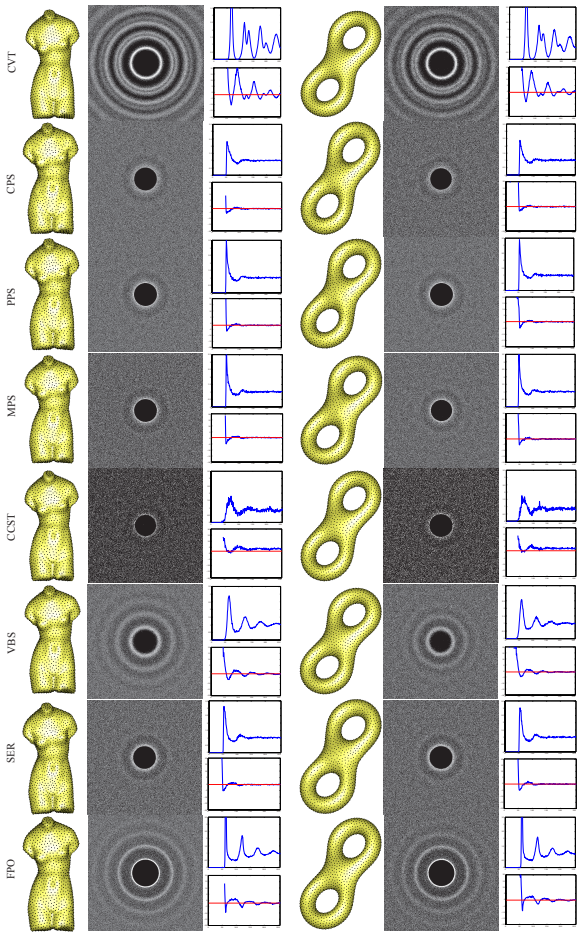


**Figure 7:** *Comparison of spectrum analysis for uniform sampling on surfaces of Venus and Eight. Left: sampling results; middle: power spectrum, and right: radial average (top) and anisotropy (bottom) of the power spectrum. Each case of both models is produced from 8 sets with about* 1700 *samples each.*

**Performance:** We now evaluate the performance of our presented algorithm. For the 2D adaptive sampling, we test our FPO algorithm in the unit square with periodic boundaries. For the uniform/adaptive sampling on surface, we test our algorithm with different number of samples generated on the surface of the Homer model, in which there are $12k$ input faces. We also provide the runtime of 2D uniform FPO algorithm [SHD11] for ground truth.

Figure 10 (right) shows the timing curve of our approach. It shows that our algorithms are reasonably fast. The main difference between 2D_AFPO and the 2D_UFPO is that we

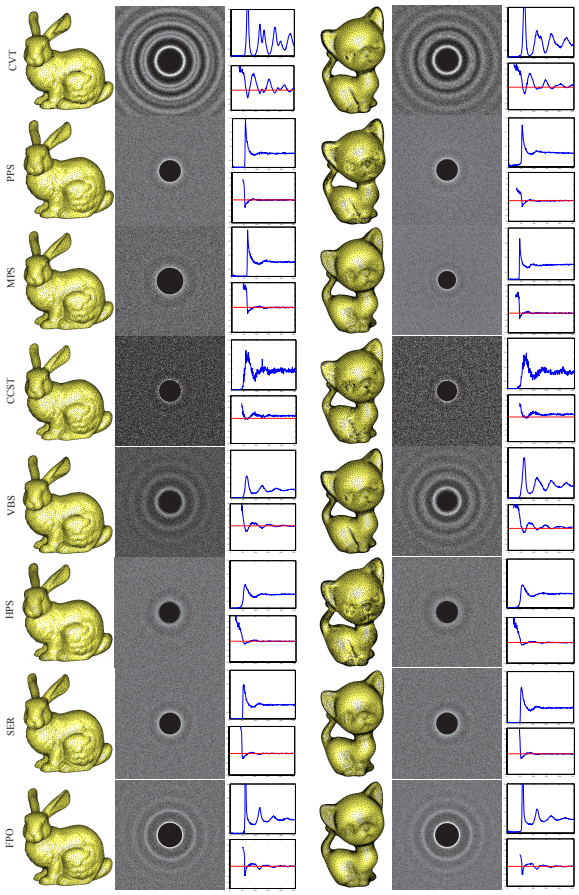


**Figure 8:** *Comparison of spectrum analysis for Adaptive sampling on surfaces of Bunny (with* $\sim$ 6300 *samples) and Kitten (with* $\sim$ 9700 *samples).*

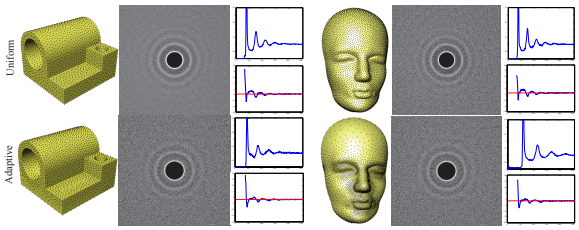


**Figure 9:** *Spectrum analysis of our feature-aware sampling.*

build a global regular triangulation at first instead of a Delaunay triangulation. Though building a regular triangulation is nearly two times slower than building a Delaunay triangulation by CGAL, the difference of updating them locally (removal or insertion of a point) is not so much. Both of them can be done in approximately constant time. Figure 10 (right) shows that the runtime of 2D_AFPO is similar to that of 2D_UFPO. On surfaces, we use the restricted Delaunay triangulation to track the farthest point for uniform sampling,

| Model | Method | $\|\mathbf{X}\|$ | $Q_{min}$ | $Q_{avg}$ | $\theta_{min}$ | $\tilde{\theta}_{min}$ | $\theta_{max}$ | $\theta_{<30^o}$ | $V_{567}$ | $d_H$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Eight | CVT | 1.7K | 0.664 | 0.943 | 38.63 | 55.22 | 95.51 | 0 | 99.9 | 0.16 |
| | CPS | 1.7K | 0.424 | 0.791 | 25.05 | 43.82 | 123.07 | 0.406 | 94.3 | 0.38 |
| | PPS | 1.7K | 0.512 | 0.806 | 29.72 | 45.15 | 113.89 | 0.01 | 95.9 | 0.32 |
| | MPS | 1.7K | 0.478 | 0.805 | 30.24 | 45.15 | 112.23 | **0** | 96.8 | **0.24** |
| | CCST | 1.7K | 0.557 | 0.811 | 24.34 | 44.63 | 107.32 | 0.558 | 97.8 | 0.66 |
| | VBS | 1.7K | 0.415 | 0.812 | 26.54 | 48.33 | 109.43 | 0.019 | 99.5 | 0.29 |
| | SER | 1.7K | 0.517 | 0.814 | 31.49 | 46.17 | 113.50 | **0** | 96.5 | 0.31 |
| | FPO | 1.7K | **0.561** | **0.855** | **34.64** | **50.63** | **105.39** | **0** | **99.7** | 0.33 |
| Venus | CVT | 1.7K | 0.665 | 0.937 | 39.97 | 54.88 | 95.81 | 0 | 100 | 0.67 |
| | CPS | 1.7K | 0.452 | 0.791 | 25.26 | 43.81 | 121.14 | 0.428 | 94.9 | 1.01 |
| | PPS | 1.7K | 0.506 | 0.805 | 27.68 | 45.07 | 114.52 | 0.019 | 96.1 | 0.76 |
| | MPS | 1.7K | 0.478 | 0.808 | 30.29 | 45.40 | 116.04 | **0** | 96.5 | 0.86 |
| | CCST | 1.7K | 0.272 | 0.810 | 9.92 | 44.54 | 115.96 | 0.834 | 98.4 | 2.50 |
| | VBS | 1.7K | 0.464 | 0.830 | 28.89 | 46.92 | 110.41 | 0.019 | **99.7** | 0.86 |
| | SER | 1.7K | 0.525 | 0.816 | 31.31 | 46.29 | 112.53 | **0** | 96.5 | **0.67** |
| | FPO | 1.7K | **0.552** | **0.860** | **33.88** | **51.28** | **109.20** | **0** | **99.7** | 0.85 |
| Homer | CVT | 7.9K | 0.660 | 0.936 | 37.30 | 54.83 | 101.35 | 0 | 99.6 | 0.46 |
| | CPS | 7.9K | 0.431 | 0.792 | 24.61 | 43.96 | 123.92 | 0.523 | 94.7 | 0.55 |
| | PPS | 7.9K | 0.473 | 0.806 | 29.96 | 45.18 | 118.84 | 0.003 | 95.9 | 0.50 |
| | MPS | 7.9K | 0.483 | 0.808 | 30.54 | 45.39 | 117.11 | **0** | 96.0 | 0.59 |
| | VBS | 7.9K | 0.288 | 0.765 | 10.82 | 41.45 | 135.48 | 3.33 | 97.2 | 0.50 |
| | SER | 7.9K | 0.485 | 0.815 | 30.35 | 46.22 | 117.38 | **0** | 96.2 | **0.43** |
| | FPO | 7.9K | **0.528** | **0.855** | **33.04** | **50.85** | **112.17** | **0** | **99.2** | 0.53 |
| Genus | CVT | 6.5K | 0.650 | 0.943 | 39.70 | 55.20 | 96.43 | 0 | 99.8 | 0.39 |
| | CPS | 6.5K | 0.441 | 0.793 | 25.18 | 43.91 | 122.81 | 0.478 | 95.8 | 0.59 |
| | PPS | 6.5K | 0.485 | 0.805 | 24.25 | 45.06 | 117.33 | 0.005 | 96.5 | 0.53 |
| | MPS | 6.5K | 0.469 | 0.807 | 30.32 | 45.35 | 119.27 | **0** | 96.4 | 0.58 |
| | VBS | 6.5K | 0.378 | 0.779 | 16.84 | 42.59 | 130.61 | 1.99 | 98.6 | 0.62 |
| | SER | 6.5K | 0.499 | 0.812 | 30.52 | 46.05 | 115.71 | **0** | 96.4 | **0.52** |
| | FPO | 6.5K | **0.551** | **0.856** | **33.08** | **50.93** | **109.31** | **0** | **99.6** | 0.54 |

**Table 2:** *Comparison of uniform remeshing qualities.* $|\mathbf{X}|$ *is the number of sampled points.*

while we build the RRT and the RPD for adaptive sampling. In our implementation, Surf_UFPO is also a little faster than Surf_AFPO, as shown in Figure 10 (right).

In addition, we accelerate the computation of FPO based on the proposed SER algorithm. After the preprocessing of SER operations, we get a relatively well-distributed point set. Then we perform the FPO locally to get the final result. Figure 11 (left) shows the runtime for the convergence of adaptive SER operations in the 2D domain and on the surface. We can see that the SER algorithm converges as fast as one iteration of FPO. Figure 11 (right) illustrates the time needed to get the final results by performing FPO with or without SER operations. We can observe a nice performance improvement by using the SER acceleration.

**Relationship with MPS:** An interesting question is what is the relationship of FPO and MPS? We can verify that our result is always maximal for uniform sampling, using the global minimal edge length as the sampling radius. However, the adaptive sampling is not always maximal, i.e., if we draw a disk centered at each sample point, using the minimal edge length incident to each point as its radius, the surface cannot be fully covered. We would like to study this problem in our future work.

| Model | Method | $\|\mathbf{X}\|$ | $Q_{min}$ | $Q_{avg}$ | $\theta_{min}$ | $\bar{\theta}_{min}$ | $\theta_{max}$ | $\theta_{<30^o}$ | $V_{567}$ | $d_H$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Bunny | CVT | 6.3K | 0.386 | 0.821 | 18.45 | 45.76 | 126.03 | 0.588 | 96.7 | 0.27 |
| | PPS | 6.3K | 0.216 | 0.769 | 10.28 | 42.34 | 149.91 | 3.068 | 93.1 | **0.30** |
| | MPS | 6.3K | 0.425 | 0.806 | 21.44 | 45.15 | 124.91 | 0.181 | 96.5 | 0.35 |
| | CCST | 6.3K | 0.018 | 0.752 | 0.262 | 40.65 | 150.41 | 4.469 | 92.0 | 0.35 |
| | VBS | 6.3K | 0.421 | 0.822 | 18.03 | 46.03 | 123.35 | 0.559 | 97.7 | **0.30** |
| | HPS | 6.3K | 0.276 | 0.778 | 14.18 | 43.05 | 140.99 | 2.122 | 92.4 | 0.33 |
| | SER | 6.3K | 0.374 | 0.809 | 22.12 | 45.69 | 126.94 | 0.332 | 95.7 | 0.49 |
| | FPO | 6.3K | **0.505** | **0.848** | **31.48** | **49.76** | **114.87** | **0** | **98.9** | 0.43 |
| Kitten | CVT | 9.8K | 0.394 | 0.821 | 20.35 | 46.62 | 125.78 | 0.578 | 95.2 | 0.21 |
| | PPS | 9.8K | 0.275 | 0.791 | 12.64 | 44.02 | 143.96 | 1.183 | 94.6 | 0.26 |
| | MPS | 9.8K | 0.428 | 0.806 | 21.9 | 45.06 | 124.95 | 0.296 | 96.3 | 0.24 |
| | CCST | 9.8K | 0.218 | 0.777 | 11.01 | 41.92 | 150.93 | 3.601 | 96.7 | 0.42 |
| | VBS | 9.8K | 0.387 | 0.817 | 19.76 | 45.13 | 126.81 | 0.546 | 96.8 | **0.19** |
| | HPS | 9.8K | 0.221 | 0.791 | 11.54 | 44.52 | 148.74 | 2.242 | 92.1 | 0.26 |
| | SER | 9.8K | 0.399 | 0.810 | 24.42 | 45.72 | 128.06 | 0.188 | 95.6 | 0.35 |
| | FPO | 9.8K | **0.532** | **0.853** | **32.07** | **50.08** | **111.59** | **0** | **99.4** | 0.21 |
| Elephant | CVT | 11.3K | 0.398 | 0.802 | 20.18 | 44.42 | 125.72 | 0.776 | 93.7 | 0.20 |
| | PPS | 11.3K | 0.343 | 0.794 | 15.73 | 44.14 | 134.53 | 0.987 | 94.4 | 0.28 |
| | MPS | 11.3K | 0.437 | 0.805 | 21.28 | 44.14 | 124.60 | 0.156 | 95.3 | 0.75 |
| | VBS | 11.3K | 0.357 | 0.803 | 20.32 | 44.57 | 133.28 | 0.793 | 94.1 | **0.22** |
| | SER | 11.3K | 0.366 | 0.805 | 20.43 | 44.78 | 125.16 | 0.225 | 95.3 | 0.38 |
| | FPO | 11.3K | **0.491** | **0.847** | **31.01** | **50.48** | **116.59** | **0** | **98.5** | 0.34 |
| Homer | CVT | 7.5K | 0.420 | 0.825 | 19.82 | 46.16 | 127.51 | 0.373 | 96.1 | 0.19 |
| | PPS | 7.5K | 0.257 | 0.794 | 12.55 | 44.18 | 145.96 | 0.883 | 94.7 | 0.25 |
| | MPS | 7.5K | 0.422 | 0.806 | 21.69 | 45.11 | 126.58 | 0.176 | 95.3 | 0.32 |
| | VBS | 7.5K | 0.367 | 0.824 | 21.32 | 46.17 | 131.11 | 0.371 | 95.8 | **0.18** |
| | SER | 7.5K | 0.362 | 0.809 | 20.74 | 45.61 | 132.61 | 0.407 | 95.6 | 0.30 |
| | FPO | 7.5K | **0.494** | **0.849** | **31.12** | **49.62** | **116.35** | **0** | **99.0** | 0.23 |
| Triceratops | CVT | 8.5K | 0.397 | 0.823 | 22.18 | 46.05 | 126.93 | 0.413 | 95.0 | 0.26 |
| | PPS | 8.5K | 0.357 | 0.796 | 16.52 | 44.29 | 128.62 | 0.858 | 92.9 | 0.34 |
| | MPS | 8.5K | 0.433 | 0.806 | 22.27 | 45.18 | 123.74 | 0.192 | 94.8 | 0.38 |
| | VBS | 8.5K | 0.385 | 0.817 | 20.07 | 44.64 | 129.57 | 0.472 | 96.8 | **0.25** |
| | SER | 8.5K | 0.368 | 0.802 | 19.87 | 45.12 | 126.31 | 0.404 | 95.2 | 0.34 |
| | FPO | 8.5K | **0.489** | **0.838** | **30.53** | **48.41** | **116.87** | **0** | **97.8** | **0.25** |

**Table 3:** *Comparison of adaptive remeshing qualities.*

| Model | Method | $\|\mathbf{X}\|$ | $Q_{min}$ | $Q_{avg}$ | $\theta_{min}$ | $\bar{\theta}_{min}$ | $\theta_{max}$ | $\theta_{<30^o}$ | $V_{567}$ | $d_H$ |
|---|---|---|---|---|---|---|---|---|---|---|
| uniform remeshing | | | | | | | | | | |
| Mask | CVT | 6.0K | 0.627 | 0.942 | 38.56 | 55.13 | 100.22 | 0 | 96.2 | 0.34 |
| | MPS | 6.0K | 0.497 | 0.807 | 30.19 | 45.25 | 115.94 | **0** | 92.9 | 0.33 |
| | FPO | 6.0K | **0.549** | **0.856** | **34.06** | **50.83** | **110.75** | **0** | **96.8** | **0.28** |
| Joint | CVT | 3.4K | 0.565 | 0.914 | 32.08 | 53.01 | 107.49 | 0 | 99.3 | 0.28 |
| | MPS | 3.4K | 0.518 | 0.815 | 30.85 | 45.93 | 114.34 | **0** | 98.3 | 0.32 |
| | FPO | 3.4K | **0.539** | **0.851** | **32.39** | **51.22** | **110.77** | **0** | **98.6** | **0.22** |
| adaptive remeshing | | | | | | | | | | |
| Mask | CVT | 3.3K | 0.622 | 0.912 | 32.56 | 52.84 | 98.79 | 0 | 96.8 | 0.23 |
| | MPS | 3.3K | 0.437 | 0.805 | 22.80 | 44.91 | 123.31 | 0.501 | 92.2 | **0.26** |
| | FPO | 3.3K | **0.507** | **0.844** | **31.07** | **49.90** | **114.75** | **0** | **97.1** | 0.33 |
| Joint | CVT | 3.8K | 0.508 | 0.890 | 31.33 | 51.35 | 114.39 | 0 | 98.3 | 0.31 |
| | MPS | 3.8K | 0.507 | 0.809 | 30.04 | 45.33 | 115.67 | **0** | 97.6 | 0.33 |
| | FPO | 3.8K | **0.526** | **0.852** | **31.73** | **49.55** | **111.79** | **0** | **98.8** | **0.27** |

**Table 4:** *Results of feature and boundary handling.*

## 7. Conclusion and Future Work

In this paper, we have presented a new technique for non-uniform blue-noise sampling and remeshing on surfaces. Our approach is based on the FPO framework, using the concept of the regular triangulation and the power diagram.

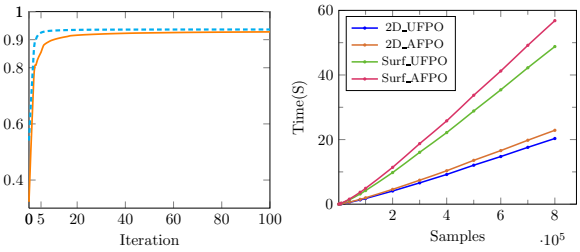


**Figure 10:** *(Left) Convergence of the minimal ($\delta_{\mathbf{X}}$, solid line) and average minimal ($\bar{\delta}_{\mathbf{X}}$, dashed line) edge length. (Right) Runtime of our FPO algorithms for one iteration. 2D_UFPO is the original 2D uniform FPO algorithm, 2D_AFPO, Surf_UFPO and Surf_AFPO represent our 2D or surface uniform/adaptive FPO.*

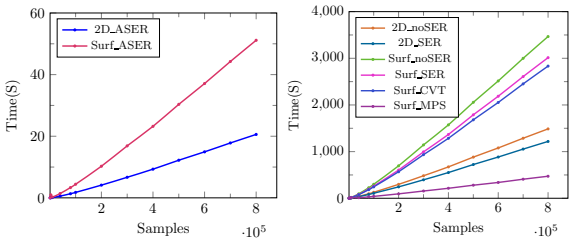


**Figure 11:** *(Left) Runtime for the convergence of the SER algorithms. We test the adaptive sampling in the 2D domain (2D_ASER) and on the surface (Surf_ASER). (Right) Runtime of the full FPO algorithms for adaptive sampling. 2D_SER and 2D_noSER refer to the FPO algorithms with and without SER operations in 2D, while Surf_SER and Surf_noSER refer to the same cases on surfaces. Note that 2D_SER and Surf_SER include the computation time for SER operations.*

Moreover, we propose an efficient acceleration technique for fast computation of a well distributed initial point set. We show that our sampling results exhibit excellent blue-noise properties, and we are able to achieve high quality remeshing superior to the state-of-the-art approaches of blue-noise remeshing.

One main limitation of our work is that we are not able to handle noisy input meshes (especially the inputs with missing geometry), since our approach relies on the exact computation of the RPD to find the farthest point location. Our method is very robust, but it cannot be used for real-time applications. Since FPO is an iterative approach, it is slower than MPS-based approaches, but it is competitive with CVT and the other techniques based on iterative optimization. We plan to look for GPU accelerations in the future. We would also like to explore new applications that could benefit from the meshes with blue-noise properties.

**Acknowledgements**

We are grateful to anonymous reviewers for their suggestive comments. We would like to thank Liyi Wei and Rui Wang for sharing the DDA tool, Zhonggui Chen, Ligang Liu and Esdras Medeiros for providing us with their results for comparison. This work was partially supported by the KAUST Visual Computing Center, the National Natural Science Foundation of China (nos. 61372168, 61331018, 61271431, 11201463), and the U.S. National Science Foundation.

## References

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new Voronoi-based surface reconstruction algorithm. In *Proc. ACM SIGGRAPH* (1998), pp. 415–421. 4, 6

[AUGA08] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring* (2008), pp. 53–82. 2

[Aur87] AURENHAMMER F.: Power diagrams: Properties, algorithms and applications. *SIAM J. Comput. 16* (1987), 78–96. 3

[AVDI05] ALLIEZ P., VERDIÈRE É. C. D., DEVILLERS O., ISENBURG M.: Centroidal voronoi diagrams for isotropic surface remeshing. *Graphical Models 67*, 3 (2005), 204–231. 6

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

[BWWM10] BOWERS J., WANG R., WEI L.-Y., MALETZ D.: Parallel Poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 29*, 6 (2010), 166:1–166:10. 1, 2, 7

[CCS12] CORSINI M., CIGNONI P., SCOPIGNO R.: Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. on Vis. and Comp. Graphics 18*, 6 (2012), 914–924. 1, 2, 7

[CDS12] CHENG S.-W., DEY T. K., SHEWCHUK J. R.: *Delaunay Mesh Generation*. CRC Press, 2012. 2

[CG12] CHEN R., GOTSMAN C.: Parallel blue-noise sampling by constrained farthest point optimization. *Computer Graphics Forum (Proc. SGP) 31*, 5 (2012), 1775–1785. 2

[cga] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. 6

[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 Trans. on Graphics (Proc. SIGGRAPH Asia) 32*, 6 (2013), 216:1–216:11. 2

[CJR*09] CLINE D., JESCHKE S., RAZDAN A., WHITE K., WONKA P.: Dart throwing on surfaces. *Computer Graphics Forum (Proc. EGSR) 28*, 4 (2009), 1217–1226. 1, 2

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. on Graphics 5*, 1 (1986), 69–78. 2

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174. 7

[CYC*12] CHEN Z., YUAN Z., CHOI Y.-K., LIU L., WANG W.: Variational blue noise sampling. *IEEE Trans. on Vis. and Comp. Graphics 18*, 10 (2012), 1784–1796. 1, 2, 7

[dGBOD12] DE GOES F., BREEDEN K., OSTROMOUKHOV V., DESBRUN M.: Blue noise through optimal transport. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 31* (2012), 171:1–171:12. 2

[ES97] EDELSBRUNNER H., SHAH N. R.: Triangulating topological spaces. *IJCGA 7*, 4 (1997), 365–378. 5, 6

[Fat11] FATTAL R.: Blue-noise point sampling using kernel density model. *ACM Trans. on Graphics (Proc. SIGGRAPH) 28*, 3 (2011), 48:1–48:10. 2

[FB97] FREY P., BOROUCHAKI H.: Surface mesh evaluation. In *6th Intl. Meshing Roundtable* (1997), pp. 363–374. 2, 7

[LD08] LAGAE A., DUTRÉ P.: A comparison of methods for generating Poisson disk distributions. *Computer Graphics Forum 27*, 1 (2008), 114–129. 2, 3

[Llo82] LLOYD S. A.: Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137. 2

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

[MIPS14] MEDEIROS E., INGRID L., PESCO S., SILVA C.: Fast adaptive blue noise on polygonal surfaces. *Graphical Models 76*, 1 (2014), 17–29. 2, 7

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *Proc. ACM SIGGRAPH* (1987), pp. 65–72. 2

[Mit91] MITCHELL D. P.: Spectrally optimal sampling for distribution ray tracing. In *Proc. ACM SIGGRAPH* (1991), pp. 157–164. 2

[OAG10] ÖZTIRELI A. C., ALEXA M., GROSS M.: Spectral sampling of manifolds. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 29*, 6 (2010), 168:1–168:8. 2

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Trans. on Graphics (Proc. SIGGRAPH) 31*, 4 (2012), 61:1–61:8. 2, 7

[SGBW10] SCHMALTZ C., GWOSDEK P., BRUHN A., WEICKERT J.: Electrostatic halftoning. *Computer Graphics Forum 29*, 8 (2010), 2313–2327. 2

[SHD11] SCHLÖMER T., HECK D., DEUSSEN O.: Farthest-point optimized point sets with maximized minimum distance. In *High Performance Graphics Proceedings* (2011), pp. 135–142. 2, 3, 4, 6, 7

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

[Wei08] WEI L.-Y.: Parallel Poisson disk sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 27*, 3 (2008), 20:1–20:9. 2

[WW11] WEI L.-Y., WANG R.: Differential domain analysis for non-uniform sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30*, 4 (2011), 50:1–50:8. 1, 6

[XHGL12] XU Y., HU R., GOTSMAN C., LIU L.: Blue noise sampling of surfaces. *Computers & Graphics 36*, 4 (2012), 232–240. 1, 2, 7

[YLL*09] YAN D.-M., LÉVY B., LIU Y., SUN F., WANG W.: Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum 28*, 5 (2009), 1445–1454. 2, 5, 6, 7

[YW13] YAN D.-M., WONKA P.: Gap processing for adaptive maximal Poisson-disk sampling. *ACM Trans. on Graphics 32*, 5 (2013), 148:1–148:15. 1, 2, 5, 7

[YWW14] YAN D.-M., WONKA P., WALLNER J.: Unbiased sampling and meshing of isosurfaces. *IEEE Trans. on Vis. and Comp. Graphics* (May 2014), accepted. 2