A Simple Push-Pull Algorithm for Blue-Noise Sampling

Abdalla G. M. Ahmed*, Jianwei Guo*, Dong-Ming Yan, Jean-Yves Franceschi, Xiaopeng Zhang, Oliver Deussen

Abstract—We describe a simple push-pull optimization (PPO) algorithm for blue-noise sampling by enforcing spatial constraints on given point sets. Constraints can be a minimum distance between samples, a maximum distance between an arbitrary point and the nearest sample, and a maximum deviation of a sample's capacity (area of Voronoi cell) from the mean capacity. All of these constraints are based on the topology emerging from Delaunay triangulation, and they can be combined for improved sampling quality and efficiency. In addition, our algorithm offers flexibility for trading-off between different targets, such as noise and aliasing. We present several applications of the proposed algorithm, including anti-aliasing, stippling, and non-obtuse remeshing. Our experimental results illustrate the efficiency and the robustness of the proposed approach. Moreover, we demonstrate that our remeshing quality is superior to the current state-of-the-art approaches.

1 Introduction

Point sets are ubiquitous in computer graphics. While the desirable features in a point set possibly vary from one application to another, isotropic point sets with blue noise spectrum [1] are universal and suit many applications, including dithering and stippling, object distribution, meshing, and remeshing. The term "blue noise" is characterized by low energy in low frequencies, a sharp transition towards a peak frequency (corresponding to the average distance between neighbor points), followed by a flat spectrum over higher frequencies.

The spectral properties of blue noise are very important for rendering applications. In image sampling, point sets with a "Poisson-disk" property, that is, a large minimum separation between samples, are an excellent choice [2], [3]. The blue noise spectrum of such sets trades low-frequency aliasing for broadband noise, but avoids the excessive noise levels of white noise. In light transport sampling [4], blue noise properties present certain advantages for Monte Carlo integration, as recently reported by Pilleboue et al. [5].

From the geometric point of view, the corresponding point set for blue noise spectrum is "even but not regular". Evenness implies a uniform density of points, with no noticeable clusters or holes (gaps) in the point set since the presence of either suggests[1] low-frequency energy. Irregularity implies no noticeable "harmony" between points, hence a flat high-frequency spectrum (quickly decaying harmonics of the primary peak). The presence of local clusters is detected by a small minimum nearest neighbor distance ($d_{\min}$), also known as conflict radius (r_{f}), or Poisson-disk radius. The local presence of holes is reflected in a large coverage radius (r_{c}), which is the largest distance between a location and the nearest point. These measures are usually presented relative to the measures of a triangular grid under the same point density [7]. Finally, for an irregular point set, equal area (capacity) of Voronoi cells implies global evenness.

The classic approach in generating Poisson-disk point sets is dart throwing, which was suggested by Dippé and Wold [2] and Cook [8], and then greatly improved thereafter [9], [10], [11], [12], [13], [14], [15], [16], [17]. While perfectly isotropic, point sets obtained by dart throwing are "noisy", that is, they have large holes and relatively small $d_{\min}$, leading to relatively narrow low-energy bands. To improve the Poisson-disk radius, McCool and Fiume [18] proposed the use of Lloyd's algorithm [19] to "relax" the point set towards a *centroidal Voronoi tessellation* (CVT). The problem of CVT is that it is considerably regular: energy is concentrated in a few harmonic frequencies, which leads to aliasing. Over the last few years, many optimization algorithms emerged as alternatives to Lloyd's method, and they offer various advantages; these algorithms include *Capacity Constrained Voronoi Tessellations* (CCVT) [20], *Farthest Point Optimization* (FPO) [21], *Capacity Constrained Delaunay Triangulation* (CCDT) [22], *Kernel Density Model (KDM) [23]*, *Blue Noise through Optimal Transport* (BNOT) [24], and *Smoothed Particle Hydrodynamics* (SPH) [25].

While many blue noise algorithms exist, each algo-

- *Joint first authors with equal contributions.
- A. G. M. Ahmed and O. Deussen are with the University of Konstanz, 78457 Konstanz, Germany.
- J. Guo, D.-M. Yan and X. Zhang are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.
- J. Y. Franceschi is with Ecole Normale Supérieure de Lyon, France.

1. While this is the common situation, it is not absolutely necessary that clusters and holes imply low frequency energy (e.g. [6]).

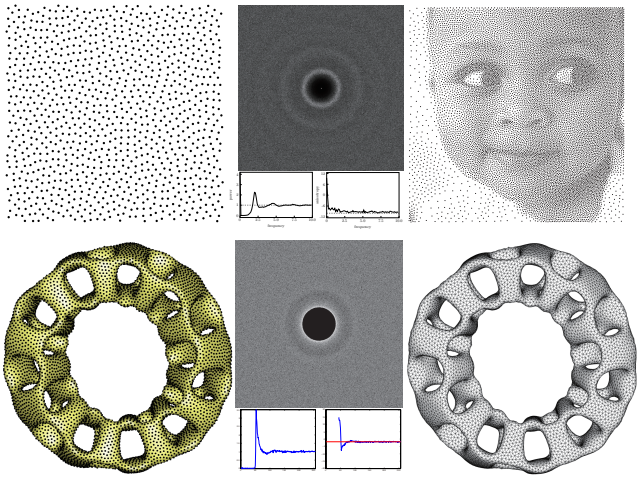


Fig. 1. Top: Results for blue noise sampling with isotropic spectrum and stipple drawing using a density function. Bottom: our algorithm is applied in surface sampling and non-obtuse remeshing.

rithm has its shortcomings in terms of speed, memory footprint, coding complexity, and control of sample count, among others. To our best knowledge, no single algorithm can be considered ideal for all purposes. Moreover, the corresponding point set profiles include trade-offs in their properties. For example, FPO reaches the highest known Poisson-disk radius (no clustering whatsoever) but it leaves holes; CCDT eliminates holes but leaves clusters; and CCVT and BNOT are more even but their Poisson-disk radii are not high, and so on.

Furthermore, the optimization targets of existing blue noise algorithms are quite limited; for instance, they only consider capacity constraints or maximized minimum distances. Thus, the set of possible spatial constraints remains incompletely covered, and finding new blue noise algorithms displaying improved properties remains possible. One example of such algorithm is presented in this paper.

1.1 Contributions

We describe a simple and intuitive algorithm, called push-pull optimization (PPO), to enforce spatial constraints, including a prescribed conflict radius, a prescribed coverage radius, and equal capacity of Voronoi cells. Apart from offering a practical and efficient blue noise algorithm, our method supplements several important contributions to research on blue noise:

1) In many spatial and spectral measures, e.g. conflict radius, coverage radius, and effective Nyquist frequency ν_{eff} [6], our algorithm surpasses the best outcomes of known algorithms; hence pushing the limits of realizable blue noise.
2) Enriching the set of blue noise profiles is useful in investigating the correlation between measurable properties and the actual performance of point sets [5]. Our method involves three tunable parameters, leading to a large ensemble of profiles; cf. a single parameter for each of [26], [23], and [25].
3) When applied in surface remeshing, our approach can effectively remove obtuse triangles, and exhibits significantly better minimum and maximum angles $(\theta_{\min}, \theta_{\max})$ in remeshing.

2 Related Work

We briefly review the most related literature, which falls into two categories.

2.1 Blue-Noise Sampling

The irregularity of blue noise makes algorithmic generation of blue noise difficult, and this phenomenon explains the large number of published methods for such purpose [7], [27]. A common principle, however, underlies most of these methods. They typically start from a random point set (flat spectrum), and energy is attenuated in low frequencies by posing one or another local constraint that leads to some correlation between the close neighborhoods of points. Such constraints vary between a minimum separation distance between points (dart throwing [8], sample elimination [28]), regular distribution of neighbors (CVT [18]), equal area of Voronoi or power cells [20], [24], [29], [30], equal area of Delaunay triangles (CCDT [22]), maximized separation between points (FPO [21]), or maximized aspect ratio of Voronoi cells [31]. These algorithms generate similar but not identical point sets, leading to different blue noise profiles usually named with acronyms (e.g. CCVT, FPO, or BNOT profile).

Instead of posing explicit spatial constraints, some algorithms employ a simulation that implies similar constraints; e.g. electrostatic equilibrium between charged particles representing the points [26], statistical-mechanical interaction between kernels representing the points (KDM [23]), mechanical equilibrium between springs representing the edges between adjacent points ([32], DistMesh [33]), or interaction of particles representing the points in fluid hydrodynamics simulations (SPH [25]).

In addition to their correlation enforcement mechanism, blue noise algorithms also explicitly or implicitly incorporate a randomization mechanism to prevent this local correlation from creeping to larger neighborhoods and manifesting as regular structures in a point set, or as strong harmonics in its spectrum.

Recently, [6], [34], [35], [36], [37] proposed sampling methods that allow for better spectral control of the resulting blue noise distributions.

2.2 Sampling-Based Remeshing

In this section, we briefly discuss the most relevant isotropic surface remeshing approaches based on sampling techniques. More details can be found in the survey paper of Alliez et al. [38].

Owing to the good distribution of a blue noise point set, a new high-quality mesh (e.g. angle bounds or edge-length bounds) can be extracted from the samples. Maximal Poisson-disk Sampling (MPS) has been demonstrated to be able to generate high-quality 2D triangulations [39], [40], surface remeshing [16], [17], [41], and 3D tetrahedral meshes [42]. A typical drawback of such dart-throwing-based approaches is that one cannot explicitly control the number of the sampled points, which is possibly important in certain applications.

Another well-known remeshing approach is based on CVT [43]. To compute CVTs on mesh surfaces, one needs to compute the Voronoi diagram on surfaces, called *Restricted Voronoi Diagram* (RVD). Early methods either use parameterization [44] or discrete clustering [45] to approximate the RVD. The quality of the consequent remeshing results is not very high because of distortion or inexact computation. Recent works of Yan et al. [46], [47] improved remeshing quality and efficiency by using exact computation of RVDs and a quasi-Newton solver [48] to compute CVTs.

Most remeshing approaches cannot generate meshes without obtuse triangles. To our best knowledge, Li and Zhang [49] were the first (within the context of graphics resampling) to propose an algorithm that can generate non-obtuse meshes; however, this algorithm could not simultaneously eliminate small angles. More recent works [41], [50] present non-obtuse remeshing results by using either CVT or MPS. In this paper, we show how a simple push-pull approach can further improve remeshing quality compared with the state-of-the-art approaches.

3 METHODOLOGY

In this section, we illustrate our core idea of generating blue-noise sampling sets, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$, with a desired number of points, n, in a given sampling domain, $\mathcal{D}$. We discuss our algorithm in the 2D case, where $\mathcal{D}$ is a unit square, $[0,1)^2$, with periodic boundary conditions; however, this concept can be applied directly to higher dimensions. Subsequently, we extend this idea into 3D mesh surfaces.

3.1 Formal Definitions

As presented in the introduction, the description of a blue-noise pattern is reflected directly in three spatial measures: a large conflict radius, a small coverage radius, and an equal area (capacity) of Voronoi cells. For a given point set, $\mathbf{X}$, in a toroidal domain, $\mathcal{D}$, these measures are formally defined as follows:

Conflict Radius: The smallest distance between any pair of points in $\mathbf{X}$:

$$r_{\mathrm{f}} = \min_{\mathbf{x}_i,\mathbf{x}_j \in \mathbf{X}, i \neq j} d_T(\mathbf{x}_i, \mathbf{x}_j) \tag{1}$$

where d_T is the toroidal distance between two locations.

Coverage Radius: The largest distance between any location in $\mathcal{D}$ and the nearest point in $\mathbf{X}$:

$$r_{\mathrm{c}} = \max_{\mathbf{y} \in \mathcal{D}, \mathbf{x}_i \in \mathbf{X}} d_T(\mathbf{y}, \mathbf{x}_i) \tag{2}$$

A set of disks with radius r_{c}, centered at each $\mathbf{x}_i$, *covers* the entire domain; hence the name "coverage radius". Technically, the coverage radius is the largest distance from a point in the set to the farthest corner of its Voronoi cell.

Capacity: Assuming we use the sample set $\mathbf{X}$ to construct a Voronoi diagram in $\mathcal{D}$, then the capacity of a sample point is defined as the area of its corresponding Voronoi cell, optionally weighted by an underlying density function for non-uniform sampling.

It is worth noting that these three geometric constraints have been previously studied individually. Among them, conflict radius was the first addressed in literature [2], [8], and is equivalent to the popular Poisson-disk criterion in dart-throwing algorithms. Capacity optimization has been addressed more recently [20], [24], [29], [51], and has become increasingly adopted in creating high-quality blue noise sets. Most recently, the need to improve coverage has been highlighted in [31], although the algorithm presented by the authors does not provide explicit control over r_{c}. To our knowledge, however, no method so far considers enforcing these three constraints simultaneously.

3.2 Push-Pull Algorithm

We propose a novel algorithm that involves three steps to achieve the prescribed target values of the spatial measures defined above. All steps are based on Delaunay triangulation, and its dual, the Voronoi diagram.

Given that these three constraints are defined for each sample $\mathbf{x}_i$, the underlying mechanism of our approach should be similar to other optimization algorithms (e.g., Lloyd's algorithm, CCVT, FPO), which allow each visited point to choose an optimal placement for itself relative to its neighbors. However, point-by-point[2] capacity enforcement seems impractical by moving the visited points [52]. Therefore, we propose a different approach: a visited point *repositions its neighbors* rather than itself. This "move neighbors" technique turned out to make much of a difference: a full iteration gives each point a turn to optimize its local neighborhood, contributing to global optimization.

2. There are effective ways, though, for capacity enforcement over the whole set [24], [29].

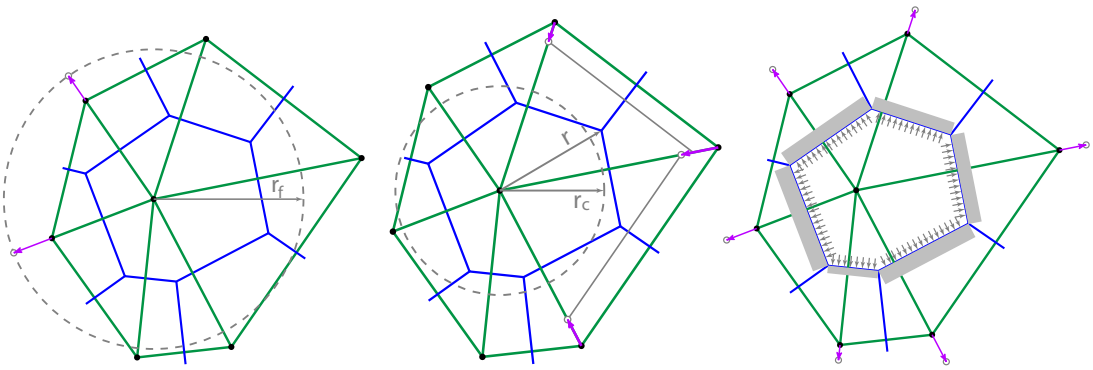


Fig. 2. Schematic of the strategy applied in repositioning neighbor points to enforce conflict, coverage, and capacity constraints.

Optimization 1: Enforcing a conflict radius r_f

```
Initial sample set X with n points;
Construct Delaunay triangulation DT(X);
repeat
    foreach x_i ∈ X do
        foreach neighbor sample x_j of x_i do
            d⃗ = x_j − x_i;
            if |d⃗| < r_f then
                shift = (r_f − |d⃗|) × d⃗/|d⃗|;
                apply shift to x_j: x_j = x_j + shift;
        update DT(X);
until no more points are moved;
```

Optimization 2: Enforcing a coverage radius r_c

```
Initial sample set X with n points;
Construct Delaunay triangulation DT(X);
repeat
    foreach x_i ∈ X do
        foreach incident triangular face t_j of x_i do
            if t_j is obtuse then
                skip t_j;
            else if t_j' circumradius r > r_c then
                set c as the circumcenter of t_j;
                r = |c − x_i|;
                set scale = r_c/r;
                for other two end points x_k of t_j do
                    d⃗ = x_i − x_k;
                    shift = (1.0 − scale) × d⃗;
                    apply shift to x_k: x_k = x_k + shift;
        update DT(X);
until no more points are moved;
```

Thus, after visiting each point, a part of the domain is optimized, allowing higher convergence speed than that when the point itself is simply moved.

Therefore, the basic model is serial: in each iteration, each sample is given a turn to optimize the placement of its immediate neighbors. As Illustrated in Fig. 2, these three spatial constraints are achievable through the following steps.

Conflict Optimization: Conflict optimization is the most straightforward action. For each visited sample, $\mathbf{x}_i$, we iterate through its neighbors, $\{\mathbf{x}_j\}$, (i.e. the incident vertices in the Delaunay triangulation). If $\mathbf{x}_j$ conflicts with $\mathbf{x}_i$ (i.e. closer than the target r_f), then $\mathbf{x}_j$ is pushed by an appropriate offset, $(r_\mathrm{f} - |\mathbf{x}_j - \mathbf{x}_i|)$, beyond the conflict distance. We call this operation a "push". Algorithm "Optimization 1" summarizes this step.

Coverage Optimization: For each sample, $\mathbf{x}_i$, we iterate through its incident facets, $\{t_j\}$, in the Delaunay triangulation. If the circumscribed radius, r, of a facet, t_j, is larger than the target r_c, we pull the two other end-points of t_j to the appropriate offsets, obtained by scaling the edges by a factor of r_c/r. We call this operation a "pull". This step is illustrated in Algorithm "Optimization 2".

Capacity Optimization: This process is slightly more involved. For each sample, $\mathbf{x}_i$, the Voronoi cell is modeled as if it contains a fluid that needs to expand or shrink, by pushing or pulling neighbor cells, in order to attain the optimum volume. The fluid exerts equal pressure, p, on the cell boundary, hence the force, f_j, on each cell edge is proportional to its length, l_j:

$$f_j = p \times l_j \,. \tag{3}$$

However, the gained (or lost) volume in displacing an edge is also proportional to its length, therefore the overall gain is:

$$\Delta = p \sum_j l_j^2 \,. \tag{4}$$

By setting Δ to the required difference in cell volume,

Optimization 3: Enforcing the capacity constraint

Initial sample set $\mathbf{X}$ with n points;
Construct Delaunay triangulation $DT(\mathbf{X})$;
repeat
  foreach $\mathbf{x}_i \in \mathbf{X}$ **do**
    Set A equal to area of Voronoi cell of $\mathbf{x}_i$;
    Compute the deviation from average area: $\Delta = A - \bar{A}$;
    if $|\Delta| < \Delta_{\max}$ **then**
      skip $\mathbf{x}_i$;
    else
      foreach *neighbor sample* $\mathbf{x}_j$ *of* $\mathbf{x}_i$ **do**
        set l_j equal to length of the Voronoi cell edge between $\mathbf{x}_i$ and $\mathbf{x}_j$;
      compute the pressure per unit length on the edges: $p = -\Delta / \sum_j l_j^2$;
      foreach *neighbor sample* $\mathbf{x}_j$ *of* $\mathbf{x}_i$ **do**
        compute the shift direction vector $\vec{d} = \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}$;
        compute the force exerted on $\mathbf{x}_j$: $f = p \times l_j$;
        set $shift = 2 \times f \times \vec{d}$;
        apply shift to $\mathbf{x}_j$: $\mathbf{x}_j = \mathbf{x}_j + shift$;
    update $DT(\mathbf{X})$;
until *no more points are moved*;

we obtain p, hence the required offset of each edge. Finally, the offset of the respective neighbor samples is set to twice the offset of the corresponding Voronoi cell edges. Evidently, there are small errors in these calculations, but the order of magnitude of errors is lower than that of expansions or contractions, hence the algorithm could converge. Although the process can be run to numeric precision, in practice we may wish to set a maximum tolerance, $\Delta_{\max}$, in capacities. This condition represents the parameter that controls this step. The detailed computation process is given in Algorithm "Optimization 3".

It is worth noting that our "pressure" metaphor is closely associated with the computation of weights in [24] and the CapVT energy gradient in [29]. Liu et al [53] also used a similar approach, but for a completely different application. We do not claim any performance optimality with respect to these capacity enforcement methods, but our approach is simple, intuitive, and can be applied iteratively to individual points, making it fit within one framework along with conflict and coverage optimization; as discussed below.

Combination: Given that the above mentioned three steps are based on the same data structure, they can easily be mixed by optimizing each target in turn for each visited point. The conflict and coverage constraints do not provide a mechanism for distributing points evenly in a large point set. They work well when starting from an already even distribution (e.g. jittered grid), but tend to leave some energy in low frequencies when starting from a random distribution. Moreover, the capacity optimization enforces global evenness thanks to the capacity constraint, but does not offer control over the aspect ratios of the Voronoi cells. It is therefore constructive to combine the three steps and gain the advantages of all the three: high r_f, low r_c , and even density.

3.3 Discussion

The described algorithm works well and converges reasonably fast in practice; thus, it is reasonable to ask why it works, and how it compares with other methods. At first glance, the conflict optimization step might seem similar to DistMesh [33], since it is based on the edges between adjacent samples. There is, however, a fundamental difference: our algorithm only considers conflicting neighbors, whereas DistMesh pushes on all neighbors. Thus, DistMesh is equilibrium-based, whereas ours is collision-based, and it is this collision nature (broken mesh) that supplements the randomization mechanism in our algorithm and reduces its tendency to create regular structures.

Moreover, our approach seems to be the first work to explicitly optimize coverage radius. It is worth noting that the gap processing framework of Yan and Wonka [16] supports conflict and coverage optimization, and they actually mention coverage. However, our algorithm replaces the substantial overheads for tracking the gaps with a simple trial-and-error logic that seems to be more efficient.

The fact that our conflict and coverage optimization steps are based on collision, not equilibrium, differentiates them from many optimization algorithms. Rather, we find our underlying logic closer to dart throwing: we remove conflicting neighbors and reinsert them in conforming positions from the perspective of the tested sample. They might then conflict with other samples, but these samples will subsequently have their chance to move to a conflict-free region. The algorithm terminates when all samples are in conforming positions to all others. If the target r_f and/or r_c are not very tough, then there are many possible conforming layouts for the samples, and our algorithm iteratively attempts to find one of them.

Within existing blue noise algorithms, there are many which try to enforce equal capacity of Voronoi or power cells, but these algorithms usually follow indirect approaches: discretizing the underlying space [20], imposing weights on samples [24], enforcing equal capacity to Delaunay triangles [22], or reducing an energy function associated with capacity [29]. The only direct (point-by-point) approach

we are aware of is described by Balzer [52], using a downhill algorithm to find optimal shifts to points, but this algorithm is prohibitively slow. In contrast to previous methods, our "move neighbors" approach makes it possible to immediately and explicitly enforce the constrained capacity of each visited sample in a single step.

4 Implementation and Extensions

In this section, we first provide further details on implementation and then analyze the runtime complexity of the proposed algorithm. We subsequently describe two important extensions: adaptive sampling and surface sampling. Finally, we apply the proposed push-pull optimization to surface remeshing, and result showed a surprisingly high remeshing quality. We demonstrate that non-obtuse remeshing can be guaranteed by tuning the ratio between r_c and r_f.

4.1 Implementation Details

The input in our algorithm includes the desired number of samples, n, as well as the three user-specified spatial constraints: a target conflict radius, r_f, a target coverage radius, r_c, and a target maximum deviation of the cell capacities, $\Delta_{\max}$. To render these parameters independent of n, we set r_f and r_c relative to $r_{max} = \sqrt{(2/\sqrt{3}\,n)}$, which is the largest attainable conflict radius, obtained when the samples form a hexagonal lattice [7], [21]. Thus, r_f should be within the $[0,1]$ range. In contrast, r_c can be larger than 1, but for a decent blue-noise quality it should stay below 1. Finally, we scale cell capacities by n so that $\Delta_{\max}$ is relative to 1, independent of n. We will analyze the influence of these parameters in Sec. 5.1.

Starting from any point set, $\mathbf{X}$, which contains n samples (e.g. random distribution or jittered grid), our algorithm iteratively optimizes the positions of the samples by enforcing these three spatial constraints. In each iteration, the algorithm proceeds by repeatedly visiting each point and performing the above three optimization steps, so as to move its conflicting neighbors to suitable positions. Here, a full iteration indicates that each sample in $\mathbf{X}$ is visited once. The optimization performed by one point might undo the optimizations of the points visited earlier, but these points would have another turn in the following iterations. The algorithm converges when no sample is further moved in one iteration. In our experiments, the order for conflict, coverage, and capacity enforcement does not seem to make a significant difference on the performance or the resulting distribution. In the supplementary material we provide a more detailed discussion about the convergence behavior of the algorithm.

To eliminate redundant processing we maintain a list of "stable" points in each iteration, and skip them in the following iteration. A point is stable if it did not move in the last iteration, nor did any of its immediate neighbors. The algorithm starts slowly, where all points are processed, and speeds up dramatically in subsequent iterations.

Parallel Implementation: Instead of applying the push/pull forces immediately, a parallel variant of the algorithm is obtained by aggregating the forces acting on each sample point. Notably, the optimizations are then based on different elements of the Delaunay triangulation: vertices for capacity, edges for conflict, and faces for coverage. Our observation is that the parallel algorithm demonstrates better qualities (less regular point sets), and it is faster per iteration, but it is slower overall (needs more iterations), and fails to converge for some tough targets that are attainable by the serial version. Thus, we will not further consider the parallel algorithm in this paper.

Runtime Complexity: We now consider the runtime complexity of each iteration. Similar to FPO [21], a dynamic global Delaunay triangulation, $DT(\mathbf{X})$, is first constructed in our implementation. Then in each optimization step, the triangulation is updated locally by moving the points one at a time. On average, the local update of $DT(\mathbf{X})$ requires $T_1 = \mathcal{O}(1)$ time. For each visited sample, we have to move no more than all its neighbors, so the runtime for visiting one sample is $T_2 = \mathcal{O}(g)$, where g is the average number of neighbors of each point in the triangulation. As a result, the runtime complexity for a full iteration is $T = n \times T_2 \times T_1 = \mathcal{O}(ng)$. Since $g = \mathcal{O}(1)$ is true for large classes of well-distributed point sets [21], the overall runtime is linear in the number of points: $T = \mathcal{O}(n)$. By contrast, the best time required for one iteration in FPO is $\mathcal{O}(n \log n)$.

4.2 Adaptive Sampling

Our algorithm can be modified to achieve adaptive sampling by applying a density function, $\rho(\mathbf{x})$, to the sample properties. The density function can be derived from the sizing function $I(\mathbf{x})$ (such as the intensity value of a a gray scale image) with $\rho(\mathbf{x}) \propto I(\mathbf{x})^{-\frac{1}{n}}$. In adaptive sampling, the sample density changes spatially according to $\rho(\mathbf{x})$, which indicates the local edge length. To generalize our algorithm to adaptive sampling, we can use the warp method introduced in [54] to map the non-uniform domain to a uniform one. To obtain better results, however, we developed our own exact algorithm for computing the weighted distances (for conflict and coverage) and areas (for capacity). First, we normalize the grayscale values so that the average pixel value is 1, as suggested in [54]. Then, for any line segment, L, the weighted length is computed as

$$|L|_w = \sum_k \sqrt{\rho_k}\,(L \cap A_k)\ , \tag{5}$$

where ρ_k is the (normalized) density of the kth pixel, and A_k is the area covered by that pixel. Please note that the density function applies to areas, and we take its square root to scale distances, as discussed in [55].

Similarly, the weighted capacity of a polygon, Q, is computed as

$$|Q|_w = \sum_k \rho_k \left(Q \cap A_k\right) . \tag{6}$$

We accelerate the computations by pre-computing the accumulated pixel densities along the scan lines [56], so that the processing time is linear (instead of quadratic) in the width of polygons. Note that our density mapping algorithm is different from [57] which produces approximate values, and is more efficient than [24], where individual pixels are assigned to the polygonal power cells. The overall running times, however, remain comparable to those in [24], since our method requires many more iterations to converge. We implemented our density mapping algorithm as a generic density integrator, and we will provide the code in the supplementary material.

Whereas applying the weighted metrics for conflict and capacity is straightforward, the coverage optimization may be problematic for some configurations, because more than two points are involved in computing the distances. We experimented with a few alternatives (e.g. average density in triangles), and we managed to obtain good results (see Fig. 6), but the optimal weighting for coverage optimization is still an open question.

4.3 Surface Sampling and Remeshing

We extend our method to surface sampling and remeshing. Now the sampling domain, $\mathcal{D}$, becomes a curved surface, $\mathcal{M} = \{F, V\}$, where $\mathcal{M}$ is a two-manifold triangular mesh consisting of a set of vertices, V, and triangle facets, F. The goal is to optimize the locations of the samples on the surface and to extract a primal mesh from the optimized samples that satisfies certain properties. For non-uniform remeshing, the sizing function $I(\mathbf{x})$ is typically defined by the local feature size [58], which is a popular choice in the literature.

We use the exact *Restricted Voronoi Diagram* (RVD) and *Restricted Delaunay Triangulation* (RDT) [46] as basic data structures to sample various surfaces, as used in many studies [16], [17], [59]. Similar to the 2D case, we iteratively visit each sample and optimize its neighbors, wherein we constrain the samples to move only on the input surface. When moving one sample, we also update the RVD and RDT locally. It has been demonstrated that the runtime of one movement is $\mathcal{O}(1)$ [59]. Therefore the time required for one iteration is still $\mathcal{O}(n)$. Fig. 3 shows the remeshing results and spectral properties of both uniform and adaptive blue-noise sampling on a mesh surface.

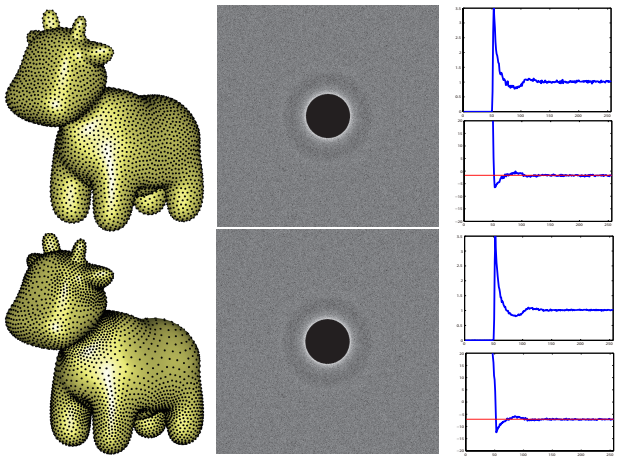


Fig. 3. Surface samplings of a cow model, with uniform (top, 4500 samples) and adaptive (bottom, 6000 samples) point distributions. On the right we show the respective plots of differential domain analysis (DDA) [54].

Once we have obtained a well distributed point set, we are ready to extract the high-quality mesh from the samples by applying the mesh extraction algorithm presented in [46] to compute the remeshing. It is easy to understand that the conflict radius r_{f} bounds the shortest edge length in the remeshing. The coverage radius r_{c} bounds the largest empty Delaunay circumcircle. The works of [31], [60] utilize the Central Angle Theorem to analyze the mathematical relation between the angle and the ratio $\beta = r_{\mathrm{c}}/r_{\mathrm{f}}$. They provide a formal theoretical proof that one can always get a non-obtuse remeshing if $\beta < 1/\sqrt{2} \approx 0.71$. In Sec. 5.3, we provide evidence for this observation using various input surfaces with arbitrary topologies.

5 Experimental Results

This section presents a series of experimental results that demonstrate the effectiveness and validity of the proposed algorithm. First, we compare the main characteristics of our sampler with those of other blue noise sampling patterns. We then present spectral analysis and zone plate plots to demonstrate the performance of our optimized point sets for anti-aliasing. Finally, we study the performance of surface remeshing by comparing to state-of-the-art remeshing techniques with respect to the meshing quality. Our algorithm is implemented in C++. We use the CGAL library [61] for computing the Delaunay triangulation. All the results presented in this paper are obtained on a PC with Intel i7-3770, 3.40 GHz CPU, 16GB memory, and a 64-bit Windows 7 operating system.

5.1 Sampling Evaluation

In the following, we present several characteristics of our resulting point sets, and we discuss the influence

Method	$d_{\min}$	d_{avg}	r_c	β	ν_{eff}	Ω	BOO
Random	0.023	0.465	1.635	71.420	0.0	0.074	0.363
MPS [15]	0.780	0.821	0.781	1.0	0.655	2.168	0.388
BNOT [24]	0.738	0.872	0.744	1.008	0.855	1.994	0.427
CVT [19]	0.805	0.941	0.657	0.817	0.950	4.733	0.842
CCVT [20]	0.711	0.856	0.782	1.10	0.830	1.774	0.403
CapCVT [29]	0.755	0.895	0.772	1.022	0.905	2.957	0.661
KDM [23]	0.645	0.868	0.746	1.157	0.870	2.223	0.451
FPO [21]	0.925	0.933	0.869	0.939	0.900	4.628	0.428
Push-pull-1	0.800	0.847	0.630	0.787	0.860	1.894	0.487
Push-pull-2	0.900	0.905	0.650	0.722	0.895	3.180	0.499
Push-pull-3	0.920	0.922	0.620	0.674	0.920	3.718	0.597

TABLE 1
Comparison of the main characteristics of different blue noise sampling patterns. Here $d_{\min}(=r_f)$ and d_{avg} are the minimal and average nearest-neighbor distance, $\beta = r_c/r_f$ is the aspect ratio of Voronoi cells [31], ν_{eff} is the effective Nyquist frequency [6], Ω is the radial-power oscillation [6], BOO is the bond orientation order [6].

of the target parameters on performance and convergence behavior.

Table 1 summarizes the mostly used measures in the blue noise literature, and we thorough compare them with the state-of-the-art sampling patterns. We use Push-pull-$\{1,2,3\}$ as our samplers with different parameters, which will be explained later.

The first two columns, $d_{\min}$ and d_{avg}, show the minimal and average nearest-neighbor distance [21]. They also roughly measure how uniformly distributed the points are: a high $d_{\min}$ indicates that points do not cluster anywhere, and a high d_{avg} means that the points are evenly spaced. $d_{\min}$ also seems to be correlated with regularity, but their relationship is not linear. It was previously speculated that pushing $d_{\min}$ beyond 0.85 would introduce regular configurations [7], but Schlömer et al. [21] subsequently introduced FPO, which reaches $d_{\min} \approx 0.93$ with regularity-free point sets; and the authors suspected the possibility of reaching even higher $d_{\min}$ without introducing regular structures. Our algorithm can reach $d_{\min} > 0.96$, but regular structures are observed for $d_{\min} > 0.93$; agreeing with their assumption. It is worth noting that $d_{\min} \approx 0.93$ is coincident with $d_{\min}$ of a regular grid.

Heck et al. [6] have established two targets for a blue-noise point set used in sampling: high effective Nyquist frequency ν_{eff} (a wide low-energy band) to reduce noise, and low oscillation Ω in the spectrum at higher frequencies to reduce aliasing. An ideal blue noise should have high ν_{eff} with low Ω, but unfortunately high ν_{eff} tends to come with high Ω. It is evident, and understandable, that $d_{\min}$ is positively correlated with ν_{eff}; but once again the relation is not linear. Unfortunately, $d_{\min}$ seems also to correlate with Ω. Thus, even though FPO reaches a much higher $d_{\min}$ than all other blue noise methods, it achieves only a small increase in ν_{eff}, and at the cost of much stronger oscillation. Since increasing $d_{\min}$ was the only known way to increase ν_{eff}, there was no way to increase ν_{eff} without increasing Ω. In contrast, besides directly controlling $d_{\min}$, our method offers another handle to increase ν_{eff} through reducing r_c. Thus, unlike the single combinations for classic methods, or the single curve for [25], our method can achieve a larger set of combinations of ν_{eff} and Ω (see the analysis in the supplementary materials).

Ebeida et al. [31] have considered the contribution of coverage to the quality of point sets, and offered an algorithm to improve coverage; however, their algorithm optimizes the ratio $\beta = r_c/r_f$, rather than handling r_c and r_f individually. While they observed that their algorithm "starts to lose blue noise between $\beta = 0.75$ and $\beta = 0.7$", our method could reach β as low as 0.67 while maintaining blue noise properties, thanks to our dual-parameter configurations.

To quantify irregularity, we compute the bond-orientational order (BOO) [62] in the last column, which measures the similarity of a point distribution to a hexagonal arrangement. A value of 1 means a perfect hexagonal grid. In general, $BOO < 0.6$ indicates a point set to be irregular. From this table, we can see that except the CVT method, other blue noise point sets are irregular in this sense.

Parameter Selection: In summary, three main parameters are used in our approach. A wide range of combinations of parameters are attainable. For capacity optimization, by setting the capacity deviation tolerance $\Delta_{\max} = 0$ the capacity constraint can be enforced up to numerical precision, but we found that this is not needed in practice. Instead, we set $\Delta_{\max} \approx 3.86 \times 10^{-2}$, which is the typical value achieved by CCVT [63]. If not explicitly specified, we use this default setting in our results.

Thus, we only need to tune r_f and r_c. To illustrate the influence of these two parameters, we provide a detailed "datasheet" in the supplementary materials. In Table 1, Push-pull-1 represents the sampler with $\{r_f = 0.80, r_c = 0.63\}$, Push-pull-2 with $\{r_f = 0.90, r_c = 0.65\}$, and Push-pull-3 with $\{r_f = 0.92, r_c = 0.62\}$, respectively.

Performance and Convergence: We now evaluate the performance and convergence behavior of our algorithm. Fig. 4(a) compares the running time of our algorithm to previous methods, using a varying number of sample points. In this example, we use a moderate parameter configuration $\{r_f = 0.85, r_c = 0.67\}$. We see that although our algorithm is slower than MPS [15], we are faster than all the other iterative-based optimization methods. Compared to MPS, we have better sampling quality in many spectral and spatial measures, as shown in Table 1.

Fig. 4(b) analyzes our convergence speed with d-

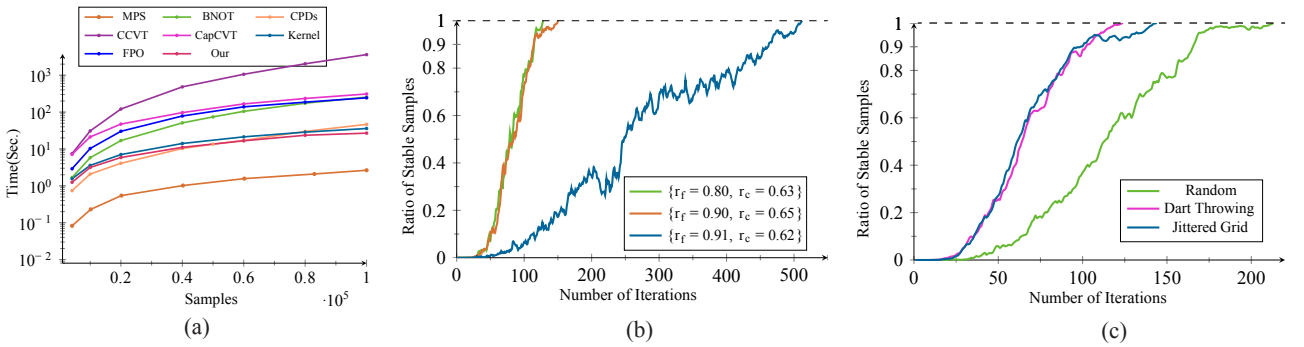


Fig. 4. Performance and convergence analysis. (a) Running time (in log scale) dependency on the number of points. (b) Convergence curves of our method with different parameter configurations. (c) Convergence curves of our method with the same parameters, but using different initialization methods.

ifferent parameter selections. We compute the ratio of stable samples in relation to the number of iterations. Both increasing r_f and reducing r_c needs more iterations to converge. Even though we have no theoretical proof of convergence, we found, according to our extensive testing, that our combined algorithm converges reasonably fast for all combinations of $r_f <= 0.9$ and $r_c >= 0.65$, for any Δ_{max}, leading to superior blue-noise quality. If one of these thresholds is not satisfied, then the convergence will highly depend on the capacity deviation tolerance, Δ_{max}. The further we are away from these thresholds, the higher the Δ_{max} needed for the algorithm to converge. This also considerably increases the number of iterations needed for convergence. Empirically, when $\{r_f > 0.92, r_c < 0.61\}$, the presented algorithm will hardly converge for any Δ_{max}. Please refer to the supplementary materials for a detailed convergence analysis of the algorithm.

We are also aware that different initializations of the samples have different influence on the convergence speed, as illustrated in Fig. 4(c). Intuitively, the more even the initial distribution of samples, the faster the convergence speed will be.

5.2 Spectral and Anti-Aliasing Analysis

To further verify the ability of our algorithm to tune the trade-off between noise and aliasing, we conducted a spectral and anti-aliasing analysis. Fig. 5 shows the power spectra and zone plate plots of point sets obtained with several sets of parameter configurations. The zone plate test reconstructs a function $(x, y) \longmapsto \sin(x^2 + y^2)$ using a single sample per pixel and a Mitchell reconstruction filter [64]. Since it shows the response for a wide range of frequencies, it represents a powerful tool for assessing the aliasing defects of sampling patterns.

First, the power spectra indicate that by varying the parameters, our algorithm could achieve a series of high-quality spectral profiles, which are very similar to the state of the art, ranging from BNOT to FPO. This also shows our potential ability to generate controllable blue noise profiles. Furthermore, we can observe that increasing r_f can effectively reduce the low-frequency noise, but at the cost of introducing structured aliasing. Fortunately, reducing r_c gives us another chance to reduce noise without adding too much aliasing, as shown in the left part of Fig. 5. Hence, we can easily control the trade-off between noise and aliasing by tuning the optimization parameters, yielding images that are of low noise and free of coherent aliasing.

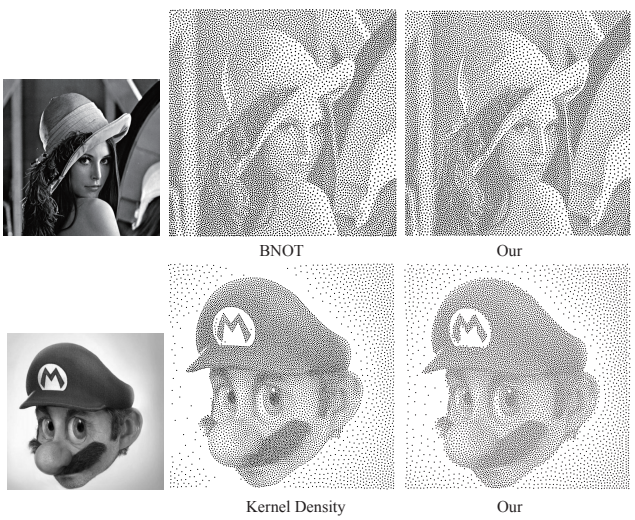


Fig. 6. Comparison of image stippling with the result of state-of-the-art BNOT [24] and KDM [23]. Lena (top) used 10000 points, Mario (bottom) used 8500 points.

Image Stippling: Our adaptive sampling algorithm can be applied to generate non-photorealistic stippling. Fig. 6 shows two examples of stipple drawings from given gray scale images. The density function is defined based on the intensity values of the input images. By comparing with the results of state-of-the-art work, we demonstrate that our method can produce high-quality and visually pleasing blue-noise point distributions.

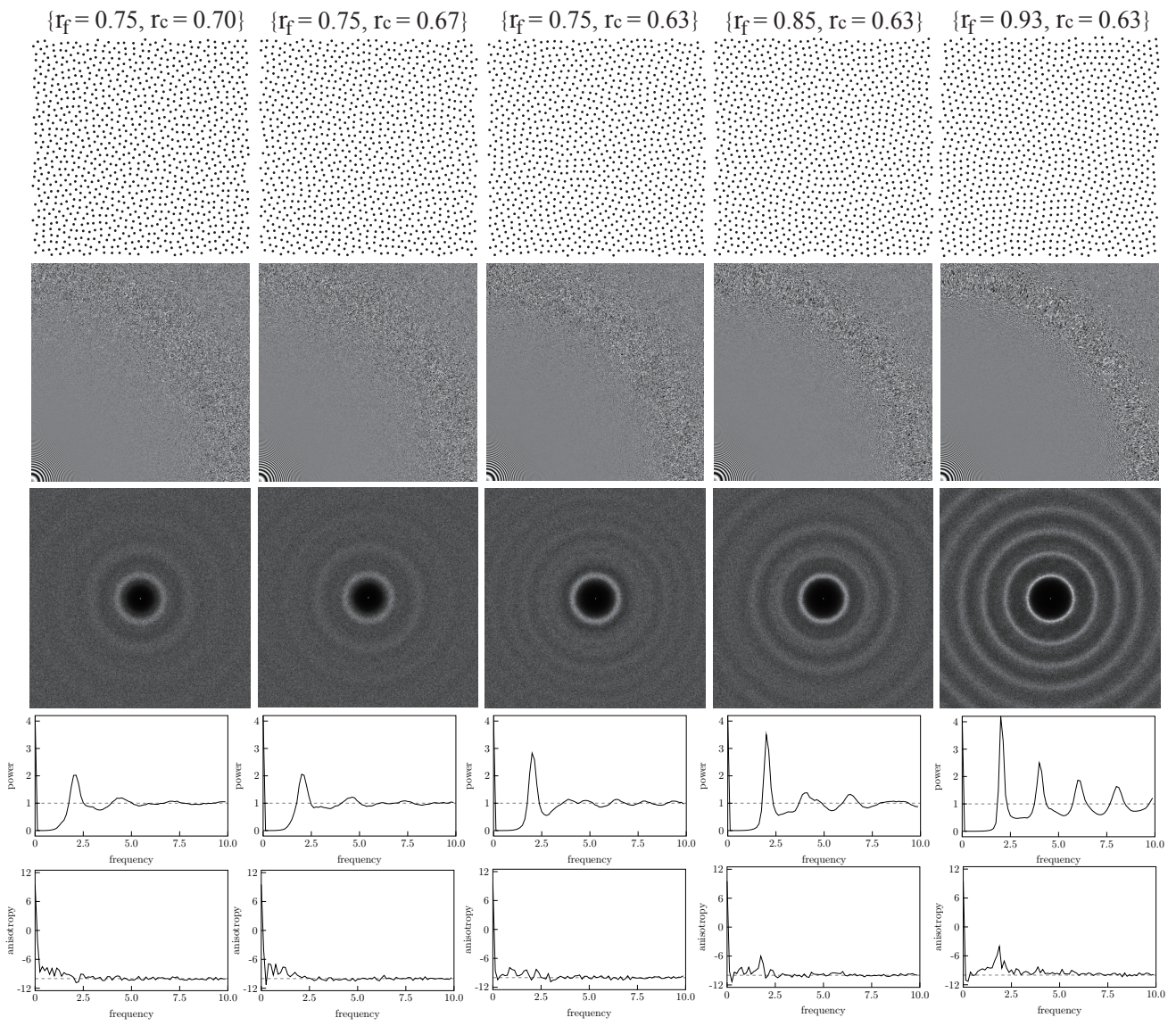


Fig. 5. Spectral and anti-aliasing analysis of the point sets obtained with different parameter configurations. From top to bottom: point sets with 1024 points, zone plate test function sampled by 512^2 points, power spectrum, radial means and anisotropy. The left three columns show the influence of reducing parameter r_c, the right three columns show the influence of increasing parameter r_f.

5.3 Surface Remeshing

Next, we extract new meshes from our surface samples and compare their quality with that obtained using other remeshing techniques, including *Maximal Poisson-disk Sampling* (MPS) [16] [17], *Farthest Point Optimization* (FPO) [59], *Capacity Constrained CVT* (CapCVT) [29]. More recently, there are two excellent works which can remove obtuse angles, either by tuning the spatial density of disks in MPS (DiskTuning) [41], or penalizing short Voronoi edges in CVT (CVT_{nob}) [50]. Table 2 lists the numerical statistics of the remeshing quality compared to previous methods. The comparison contains various standard mesh quality metrics 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 [65]. Here Q_{min} and Q_{avg} is the minimal and average triangle quality; θ_{min} and θ_{max} are the minimal and maximal angle, and $\bar{\theta}_{min}$ is the average of the minimal angles of all triangles; $\theta_{<30^o}\%$ is the percentage of triangles with angles smaller than 30^o; $\theta_{>90^o}\%$ is the percentage of obtuse triangles; V_{567} is the percent of vertices with valences 5, 6 and 7; d_{RMS} is the root mean square distance, and d_H is the Hausdorff distance between input surface and remeshing result, measured with the Metro tool [66]. Fig. 7 shows some selected remeshing results; please refer to the supplementary materials for more comparison results.

We use the parameters $\{r_f = 0.90, r_c = 0.63\}$, where

Model	Method	$\|\mathbf{X}\|$	$\|t\|$	$\|t_{obt}\|$	Q_{min}	Q_{avg}	θ_{min}	$\bar{\theta}_{min}$	θ_{max}	$\theta_{<30^o}\%$	$\theta_{>90^o}\%$	$V_{567}\%$	$d_{RMS}(\times 10^{-3})$	$d_H(\times 10^{-2})$
Fertility	MPS	8.5K	17K	2.6K	0.50	0.81	30.26	45.26	116.0	**0**	15.1	96.4	0.48	0.41
	FPO	8.5K	17K	1.0K	0.52	0.86	32.8	50.8	113.6	**0**	6.16	99.5	0.98	0.33
	CapCVT	8.5K	17K	508	0.52	0.88	30.02	51.3	113.5	**0**	2.98	99.8	0.42	**0.28**
	CVT_{nob}	8.5K	17K	**0**	**0.77**	**0.94**	40.4	**54.8**	**83.4**	**0**	**0**	**100**	0.98	0.34
	DiskTuning	7.7K	15.5K	**0**	0.64	0.82	30.1	45.7	90.0	**0**	**0**	97.8	0.55	0.37
	Our	8.5K	17K	**0**	0.74	0.87	**46.3**	51.1	87.5	**0**	**0**	**100**	**0.29**	0.31
Moai	MPS	12K	24K	3.6K	0.47	0.81	30.4	45.3	118.8	**0**	14.8	96.1	0.91	0.54
	FPO	11K	22K	1.4K	0.54	0.86	32.6	50.9	110.3	**0**	6.08	99.5	0.56	0.51
	CapCVT	11K	2.2K	2.3K	0.48	0.82	21.1	46.4	118.5	0.65	10.4	99.4	0.55	best0.45
	CVT_{nob}	11K	22K	**0**	0.68	**0.95**	32.4	**55.8**	**85.2**	**0**	**0**	**100**	0.88	0.51
	DiskTuning	11K	22K	3	0.64	0.82	30.1	45.6	90.0	**0**	0.01	97.8	0.95	0.60
	Our	11K	22K	**0**	**0.74**	0.87	**45.6**	51.1	87.7	**0**	**0**	99.9	**0.43**	0.46
Homer	MPS	7.5K	15K	1.6K	0.56	0.82	30.1	46.4	105.8	**0**	10.8	99.8	0.61	0.28
	FPO	7.7K	15.6K	1.2K	0.49	0.85	31.0	49.6	117.5	**0**	7.99	99.2	0.45	0.30
	CapCVT	7,5K	15K	1.6K	0.37	0.83	21.3	46.2	131.1	0.98	10.66	95.8	0.42	0.20
	CVT_{nob}	7.5K	15K	**0**	**0.69**	**0.94**	33.7	**54.5**	**85.4**	**0**	**0**	**100**	**0.36**	**0.17**
	DiskTuning	3.8K	7.8K	1	0.51	0.82	21.3	45.0	90	1.64	0.01	**97.3**	1.21	0.57
	Our	7.5K	15K	**0**	0.68	0.90	**34.4**	51.8	89.5	**0**	**0**	**100**	0.40	0.23
Bunny	MPS	7.8K	15.4K	755	0.62	0.83	32.0	47.4	97.9	**0**	4.89	99.8	0.77	0.46
	FPO	8.0K	16K	1.6K	0.50	0.84	30.0	48.6	116.3	**0**	9.92	98.0	0.81	0.42
	CapCVT	8.0K	16K	1.4K	0.39	0.84	15.6	47.3	125.1	1.04	8.46	98.7	0.71	**0.24**
	CVT_{nob}	8.0K	16K	**0**	**0.72**	**0.94**	36.3	**54.4**	**89.4**	**0**	**0**	**100**	0.94	0.35
	DiskTuning	5.8K	12K	**0**	0.50	0.82	19.89	45.2	90.0	0.88	**0**	97.8	1.55	0.52
	Our	8.0K	16K	**0**	0.71	0.87	**37.1**	51.1	89.5	**0**	**0**	99.8	**0.44**	0.52
Kitten	MPS	7.9K	15.8K	2.4K	0.42	0.81	25.3	45.3	125.4	0.29	15.3	96.0	0.78	0.96
	FPO	9.5K	19K	1.3K	0.53	0.85	32.0	50.0	111.6	**0**	6.89	99.4	0.53	0.29
	CapCVT	9.8K	19.6K	1.4K	0.39	0.84	15.6	47.3	125.1	1.04	8.46	98.7	0.29	0.21
	CVT_{nob}	9.8K	19.6K	**0**	**0.71**	**0.95**	33.6	**55.3**	**87.1**	**0**	**0**	**100**	0.27	**0.20**
	Our	9.8K	19.6K	**0**	**0.71**	0.90	**37.3**	52.1	88.9	**0**	**0**	**100**	**0.26**	0.27

TABLE 2
Comparison of remeshing quality with previous techniques. We compare the uniform remeshing on the Fertility and Moai models, and adaptive remeshing on some other three models. The best result of each measurement is marked in **bold** font. $|\mathbf{X}|$ is the number of vertices; $|t|$ is the number of triangles; $|t_{obt}|$ is the number of obtuse triangles. The other measurements are explained in Sec 5.3.

$\beta = 0.7$, to achieve excellent surface sampling in all the remeshing results. Table 2 suggests that our approach can effectively remove obtuse angles and exhibits significantly better triangle and angle qualities than the methods with blue-noise profiles. Compared with CVT_{nob}, our algorithm is still competitive and has similar qualities. Further, Fig. 7 demonstrates that the distribution of vertices of our meshing is more irregular than CVT_{nob}, which are preferred for many simulation applications [39], such as fracture simulations [67] and fluid simulation [68].

5.4 Limitations

In the current optimization framework, we haven't considered boundaries or sharp features of the input domains. One possible solution is that we explicitly project nearby samples onto features after each iteration, or adapt other techniques proposed by [69], [70], but the convergence behavior should be studied carefully. We hope to address this issue in future works.

6 Conclusion and Future Work

We presented an iterative blue-noise optimization method by directly enforcing spatial constraints on a given point set. Three optimization steps are proposed, and they are able to generate a variety of blue noise profiles that can be used in different applications, such as rendering, adaptive sampling, surface remeshing, etc. We demonstrate that our method outperforms state-of-the-art blue noise algorithms in terms of speed and many other spectral and spatial measures.

The proposed method was originally conceived for optimizing AA-Patterns [37] and tile-based samplers (e.g. Polyhexes [36]), where indexed points are potentially required to fit well into more than one context. The "move-your-neighbors" approach proved especially useful in these cases. The method was developed thereafter into a simple and intuitive, yet competitive, general-purpose blue noise optimizer. The algorithm also proved effective and efficient for generating *stratified blue noise* point sets, as required in the recent work of Ahmed et al. [71].

Although our algorithm could generate sample patterns similar to various other blue noise methods, our current implementation cannot generate point samples that exactly match a target spectral profile, especially on surfaces.

In the future, we would like to extend our optimiza-

Fig. 7. Quality comparison of remeshing with representative algorithms. From left to right: input meshes, results of FPO [59], DiskTuning [41], CVT$_{nob}$ [50] and our Push-pull method. The obtuse triangles are shown in pink, and triangles with $\theta_{min} < 30$ are shown in blue.

tion algorithm to generate three dimensional blue-noise sets, as well as high-quality tetrahedral meshes.

ACKNOWLEDGEMENTS

We would like to thank Zhonggui Chen, Fernando de Goes, Raanan Fattal, Mohamed S. Ebeida and Daniel Heck for providing the data and executables, Liyi Wei and Rui Wang for sharing the DDA tool. This project was supported by the Deutsche Forschungsgemeinschaft Grant (DE-620/22-1), the National Natural Science Foundation of China (61372168, 61620106003, and 61331018), the National Foreign 1000 Plan (WQ201344000169) and the Leading Talents of Guangdong Program (00201509). Dong-Ming Yan is the corresponding author.

REFERENCES

[1] R. A. Ulichney, "Dithering with blue noise," *Proceedings of the IEEE*, vol. 76, no. 1, pp. 56–79, 1988.

[2] M. A. Z. Dippé and E. H. Wold, "Antialiasing through stochastic sampling," *Proc. ACM SIGGRAPH*, pp. 69–78, 1985.

[3] D. P. Mitchell, "Generating antialiased images at low sampling densities," in *Proc. ACM SIGGRAPH*, 1987, pp. 65–72.

[4] M. Pharr and G. Humphreys, *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., 2010.

[5] A. Pilleboue, G. Singh, D. Coeurjolly, M. Kazhdan, and V. Ostromoukhov, "Variance Analysis for Monte Carlo Integration," *ACM Trans. on Graphics*, vol. 34, no. 4, pp. 124:1–124:14, July 2015.

[6] D. Heck, T. Schlömer, and O. Deussen, "Blue noise sampling with controlled aliasing," *ACM Trans. on Graphics*, vol. 32, no. 3, pp. 25:1–25:12, 2013.

[7] A. Lagae and P. Dutré, "A comparison of methods for generating Poisson disk distributions," *Computer Graphics Forum*, vol. 27, no. 1, pp. 114–129, 2008.

[8] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Trans. on Graphics*, vol. 5, no. 1, pp. 69–78, 1986.

[9] T. R. Jones, "Efficient generation of Poisson-disk sampling patterns," *Journal of Graphics Tools*, vol. 11, no. 2, pp. 27–36, 2006.

[10] D. Dunbar and G. Humphreys, "A spatial data structure for fast Poisson-disk sample generation," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 25, no. 3, pp. 503–508, 2006.

[11] R. Bridson, "Fast Poisson disk sampling in arbitrary dimensions," in *ACM SIGGRAPH 2007 Sketches*, 2007.

[12] L.-Y. Wei, "Parallel Poisson disk sampling," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 27, no. 3, pp. 20:1–20:9, 2008.

[13] M. N. Gamito and S. C. Maddock, "Accurate multidimensional Poisson-disk sampling," *ACM Trans. on Graphics*, vol. 29, no. 1, pp. 8:1–8:19, 2009.

[14] M. S. Ebeida, A. Patney, S. A. Mitchell, P. M. K. Andrew Davidson, and J. D. Owens, "Efficient maximal Poisson-disk sampling," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 30, no. 4, pp. 49:1–49:12, 2011.

[15] M. S. Ebeida, S. A. Mitchell, A. Patney, A. A. Davidson, and J. D. Owens, "A simple algorithm for maximal Poisson-disk sampling in high dimensions," *Computer Graphics Forum (Proc. EUROGRAPHICS)*, vol. 31, no. 2, pp. 785–794, 2012.

[16] D.-M. Yan and P. Wonka, "Gap processing for adaptive maximal Poisson-disk sampling," *ACM Trans. on Graphics*, vol. 32, no. 5, pp. 148:1–148:15, 2013.

[17] J. Guo, D.-M. Yan, X. Jia, and X. Zhang, "Efficient maximal Poisson-disk sampling and remeshing on surfaces," *Computers & Graphics (Proc. SMI)*, vol. 46, no. 6-8, pp. 72–79, 2015.

[18] M. McCool and E. Fiume, "Hierarchical Poisson disk sampling distributions," in *Graphics Interface*, 1992, pp. 94–105.

[19] S. A. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[20] M. Balzer, T. Schlömer, and O. Deussen, "Capacity-constrained point distributions: A variant of Lloyd's method," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 6, pp. 86:1–86:8, 2009.

[21] T. Schlömer, D. Heck, and O. Deussen, "Farthest-point optimized point sets with maximized minimum distance," in *High Performance Graphics Proceedings*, 2011, pp. 135–142.

[22] Y. Xu, L. Liu, C. Gotsman, and S. J. Gortler, "Capacity-Constrained Delaunay Triangulation for point distributions," *Computers & Graphics (Proc. SMI)*, vol. 35, no. 3, pp. 510–516, 2011.

[23] R. Fattal, "Blue-noise point sampling using kernel density model," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 3, pp. 48:1–48:10, 2011.

[24] F. de Goes, K. Breeden, V. Ostromoukhov, and M. Desbrun, "Blue Noise through Optimal Transport," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 31, pp. 171:1–171:12, 2012.

[25] M. Jiang, Y. Zhou, R. Wang, R. Southern, and J. J. Zhang, "Blue noise sampling using an SPH-based method," *ACM Trans. on Graphics*, vol. 34, no. 6, p. 211, 2015.

[26] C. Schmaltz, P. Gwosdek, A. Bruhn, and J. Weickert, "Electrostatic Halftoning," *Computer Graphics Forum*, vol. 29, no. 8, pp. 2313–2327, 2010.

[27] D.-M. Yan, J. Guo, B. Wang, X. Zhang, and P. Wonka, "A survey of blue-noise sampling and its applications," *Journal of Computer Science and Technology*, vol. 30, no. 3, pp. 439–452, 2015.

[28] C. Yuksel, "Sample elimination for generating Poisson disk sample sets," *Computer Graphics Forum (Proceedings of EUROGRAPHICS)*, vol. 34, no. 2, pp. 25–32, 2015.

[29] Z. Chen, Z. Yuan, Y.-K. Choi, L. Liu, and W. Wang, "Variational Blue Noise Sampling," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 18, no. 10, pp. 1784–1796, 2012.

[30] S.-Q. Xin, B. Lévy, Z. Chen, L. Chu, Y. Yu, C. Tu, and W. Wang, "Centroidal Power Diagrams with Capacity Constraints: Computation, Applications, and Extension," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 35, no. 6, 2016.

[31] M. S. Ebeida, M. A. Awad, X. Ge, A. H. Mahmoud, S. A. Mitchell, P. M. Knupp, and L.-Y. Wei, "Improving spatial coverage while preserving the blue noise of point sets," *Computer-Aided Design*, vol. 46, pp. 25–36, 2014.

[32] V. Ostromoukhov, "Pseudo-random halftone screening for colour and black&white; printing," in *9th International Congress in Non Impact Printing Technologies*, no. LSP-CONF-1993-002, 1993, pp. 579–582.

[33] P.-O. Persson and G. Strang, "A simple mesh generator in MATLAB," *SIAM review*, vol. 46, no. 2, pp. 329–345, 2004.

[34] Y. Zhou, H. Huang, L.-Y. Wei, and R. Wang, "Point sampling with general noise spectrum," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 31, no. 4, pp. 76:1–76:11, July 2012.

[35] A. C. Öztireli and M. Gross, "Analysis and synthesis of point distributions based on pair correlation," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 31, no. 6, p. 170, 2012.

[36] F. Wachtel, A. Pilleboue, D. Coeurjolly, K. Breeden, G. Singh, G. Cathelin, F. de Goes, M. Desbrun, and V. Ostromoukhov, "Fast tile-based adaptive sampling with user-specified fourier spectra," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 33, no. 4, pp. 56:1–56:11, 2014.

[37] A. G. M. Ahmed, H. Huang, and O. Deussen, "Aa patterns for point sets with controlled spectral properties," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, p. 212, 2015.

[38] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, "Recent Advances in Remeshing of Surfaces," in *Shape Analysis and Structuring*, 2008, pp. 53–82.

[39] M. S. Ebeida, S. A. Mitchell, A. A. Davidson, A. Patney, P. M. Knupp, and J. D. Owens, "Efficient and good Delaunay meshes from random points," *Computer-Aided Design*, vol. 43, no. 11, pp. 1506–1515, 2011.

[40] J. Guo, D.-M. Yan, G. Bao, W. Dong, X. Zhang, and P. Wonka, "Efficient triangulation of Poisson-disk sampled point sets," *The Visual Computer*, vol. 30, no. 6-8, pp. 773–785, 2014.

[41] M. Ebeida, A. A. Rushdi, M. A. Awad, A. H. Mahmoud, D.-M. Yan, S. A. English, J. D. Owens, C. L. Bajaj, and S. A. Mitchell, "Disk density tuning of a maximal random packing," *Computer Graphics Forum (Proc. SGP)*, 2016.

[42] J. Guo, D.-M. Yan, L. Chen, X. Zhang, O. Deussen, and P. Wonka, "Tetrahedral meshing via maximal Poisson-disk sampling," *Comp. Aided Geom. Design*, vol. 43, pp. 186–199, 2016.

[43] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, vol. 41, pp. 637–676, 1999.

[44] P. Alliez, É. C. d. Verdiére, O. Devillers, and M. Isenburg, "Centroidal Voronoi diagrams for isotropic surface remeshing," *Graphical Models*, vol. 67, no. 3, pp. 204–231, 2005.

[45] S. Valette, J.-M. Chassery, and R. Prost, "Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 14, no. 2, pp. 369–381, 2008.

[46] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted Voronoi diagram," *Computer Graphics Forum (Proc. SGP)*, vol. 28, no. 5, pp. 1445–1454, 2009.

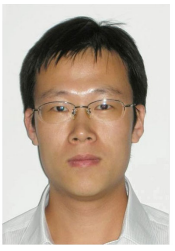
[47] D.-M. Yan, G. Bao, X. Zhang, and P. Wonka, "Low-resolution remeshing using the localized restricted Voronoi diagram," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 20, no. 10, pp. 418–1427, 2014.

[48] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On centroidal Voronoi tessellation - energy smoothness and fast computation," *ACM Trans. on Graphics*, vol. 28, no. 4, pp. 101:1–101:11, 2009.

[49] J. Li and H. Zhang, "Nonobtuse remeshing and decimation," in *Proc. of Eurographics Symposium on Geometry Processing*, 2006, p. Proc. of Symp. of Geometry Processing.
[50] D. Yan and P. Wonka, "Non-obtuse Remeshing with Centroidal Voronoi Tessellation," *IEEE Trans. on Vis. and Comp. Graphics*, vol. 22, no. 9, pp. 2136–2144, 2016.
[51] S. Zhang, J. Guo, H. Zhang, X. Jia, D.-M. Yan, J.-H. Yong, and P. Wonka, "Capacity constrained blue-noise sampling on surfaces," *Computers & Graphics*, vol. x, no. x, p. x, 2016.
[52] M. Balzer, "Capacity-constrained Voronoi diagrams in continuous spaces," in *Voronoi Diagrams, 2009. ISVD'09. Sixth International Symposium on*. IEEE, 2009, pp. 79–88.
[53] Y. Liu, H. Pan, J. Snyder, W. Wang, and B. Guo, "Computing self-supporting surfaces by regular triangulation," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 32, no. 4, pp. 92:1–92:10, 2013.
[54] L.-Y. Wei and R. Wang, "Differential domain analysis for non-uniform sampling," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 30, no. 4, pp. 50:1–50:8, 2011.
[55] A. G. M. Ahmed, "From stippling to scribbling," in *Proceedings of Bridges 2015: Mathematics, Music, Art, Architecture, Culture*, 2015, pp. 267–274.
[56] A. G. Ahmed, "Modular Line-based Halftoning via Recursive Division," in *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering*, ser. NPAR '14. ACM, 2014, pp. 41–48.
[57] G. Lecot and B. Levy, "Ardeco: automatic region detection and conversion," in *17th Eurographics Symposium on Rendering-EGSR'06*, 2006, pp. 349–360.
[58] N. Amenta, M. Bern, and M. Kamvysselis, "A new Voronoi-based surface reconstruction algorithm," in *Proc. ACM SIGGRAPH*, 1998, pp. 415–421.
[59] D.-M. Yan, J. Guo, X. Jia, X. Zhang, and P. Wonka, "Blue-noise remeshing with farthest point optimization," *Computer Graphics Forum (Proc. SGP)*, vol. 33, no. 5, pp. 167–176, 2014.
[60] S. A. Mitchell, A. Rand, M. S. Ebeida, and C. L. Bajaj, "Variable radii Poisson disk sampling," in *24th Canadian Conference on Computational Geometry (CCCG)*, 2012, pp. 185–190.
[61] "CGAL, Computational Geometry Algorithms Library," http://www.cgal.org.
[62] A. R. Kansal, T. M. Truskett, and S. Torquato, "Nonequilibrium hard-disk packings with controlled orientational order," *The Journal of Chemical Physics*, vol. 113, no. 12, pp. 4844–4851, 2000.
[63] T. Schlömer, "Non-periodic corner tilings in computer graphics," Ph.D. dissertation, Citeseer, 2012.
[64] D. P. Mitchell and A. N. Netravali, "Reconstruction filters in computer-graphics," *ACM Siggraph Computer Graphics*, vol. 22, pp. 221–228, 1988.
[65] P. Frey and H. Borouchaki, "Surface mesh evaluation," in *6th Intl. Meshing Roundtable*, 1997, pp. 363–374.
[66] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998.
[67] J. Bolander and S. Saito, "Fracture analyses using spring networks with random geometry," *Engineering Fracture Mechanics*, vol. 61, no. 5, pp. 569–591, 1998.
[68] F. de Goes, C. Wallez, J. Huang, D. Pavlov, and M. Desbrun, "Power particles: An incompressible fluid solver based on power diagrams," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 34, 2015.
[69] J. Chen, X. Ge, L.-Y. Wei, B. Wang, Y. Wang, H. Wang, Y. Fei, K.-L. Qian, J.-H. Yong, and W. Wang, "Bilateral Blue Noise Sampling," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 32, no. 6, pp. 216:1–216:11, 2013.
[70] K. Hu, D.-M. Yan, D. Bommes, P. Alliez, and B. Benes, "Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement," *IEEE Trans. on Vis. and Comp. Graphics*, 2016, to appear.
[71] A. G. M. Ahmed, H. Perrier, D. Coeurjolly, V. Ostromoukhov, J. Guo, D.-M. Yan, H. Huang, and O. Deussen, "Low-discrepancy blue noise sampling," *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, vol. 35, no. 6, 2016.

Abdalla G. M. Ahmed is a researcher in the Department of Computer and Information Science, University of Konstanz, Germany. He received his BSc in Electrical Engineering from University of Khartoum in 2002, a MBA from University of Khartoum in 2008, and a MSc in Mobile and Personal Communications from Kings College London in 2009. Before joining the University of Konstanz, he worked with many companies in different technical and commercial roles. He also conducted independent research in math, arts, and computer graphics, and published many papers and artworks.

Jianwei Guo is an assistant researcher in National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences(CASIA). He received his Ph.D. degree in computer science from CASIA in 2016, and bachelor degree from Shandong University in 2011. His research interests include computer graphics, geometry processing and 3D shape analysis.



Dong-Ming Yan is an associate professor in National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences(CAS). He received his Ph.D. degree in computer science from Hong Kong University in 2010, and his master and bachelor degrees in computer science and technology from Tsinghua University in 2005 and 2002, respectively. His research interests include computer graphics, geometric processing, and visualization.

Jean-Yves Franceschi is a Master student at the Ecole Normale Supérieure de Lyon (ENS Lyon), France, in Foundations of Computer Science, where he also received his bachelor degree in the same domain. He has various research interests, such as computer graphics, information theory or machine learning.

Xiaopeng Zhang is a Professor in National Laboratory of Pattern Recognition at Institute of Automation, Chinese Academic of Sciences (CAS). He received his Ph.D. degree in Computer Science from Institute of Software, CAS in 1999. He received the National Scientific and Technological Progress Prize (second class) in 2004. His main research interests include computer graphics and image processing.

Oliver Deussen Prof. Deussen graduated at Karlsruhe Institute of Technology and is professor at University of Konstanz (Germany) and visiting professor at the Chinese Academy of Science in Shenzhen. He serves as Co-Editor in Chief of Computer Graphics Forum and is Vice-President of Eurographics Association. His areas of interest are modeling and rendering of complex biological systems, non-photorealistic rendering as well as Information Visualization. He also contributed papers to geometry processing, sampling methods, and image-based modeling.