Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

CHAVDAR PAPAZOV, HANS-CHRISTIAN HEGE

# Blue-noise Optimized Point Sets Based on Procrustes Analysis

ZIB Report 17-55 (Sept. 2017)

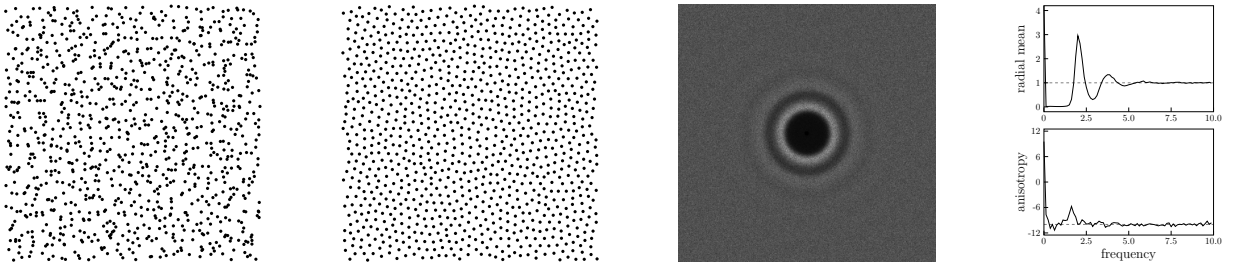
Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

# Blue-noise Optimized Point Sets Based on Procrustes Analysis




input points | optimized result | power spectrum | anisotropy & radial mean

**Figure 1:** A point set optimized with our algorithm exhibits typical blue noise characteristics.

**Abstract**

In this paper, we propose a new method for optimizing the blue noise characteristics of point sets. It is based on Procrustes analysis, a technique for adjusting shapes to each other by applying optimal elements of an appropriate transformation group. We adapt this technique to the problem at hand and introduce a very simple, efficient and provably convergent point set optimizer.

## 1 Introduction

A point set optimizer transforms a given point set to another one, which is, in some sense, better than the input. In this paper, we propose a new algorithm that maximizes the mutual distance between the points and leads to distributions with excellent blue noise characteristics. Methods of this type have important applications in areas like rendering, halftoning, stippling, texture synthesis and remeshing, just to name a few [YGW+15].

Given some input points distributed in a rectangular domain, our method iteratively maximizes the distance between each point and its near neighbors while avoiding regular configurations. This is done by generating the Delaunay triangulation of the input and then optimizing the vertex positions using Procrustes analysis.

Given two finite point sets $\mathbf{Q}, \mathbf{T} \subset \mathbb{R}^k$, the Procrustes problem is to compute the rotation and translation that optimally map $\mathbf{Q}$ to $\mathbf{T}$ by minimizing the sum of squared distances between corresponding points. Applications include shape registration [BM92], deformation modeling [MHTG05], mesh parametrization [LZX+08] and many others. To the best of our knowledge, the proposed algorithm is the first one that utilizes Procrustes analysis in the context of blue-noise optimization.

Our method has several advantages: (i) it is very simple, (ii) it is provably convergent, (iii) it has low time and memory complexity and (iv) it produces high-quality output.

**Related Work.** Perhaps the best-known point set optimizer is Lloyd's relaxation method [Llo82]. It generates point distributions with large inter-point distances but the output exhibits strong regularity artifacts. To account for that, researchers developed methods that generate and optimize capacity-constrained Voronoi tessellations [BSD09, dGBOD12, ZGZ+16]. They enforce that each point's Voronoi region has the same area (volume) throughout the optimization. Other state-of-the-art approaches include farthest point optimization [SHD11], a kernel density model from statistical mechanics [Fat11] and an SPH method from computational fluid dynamics [JZW+15]. Even though all these algorithms produce high-quality blue-noise distributions, they are either computationally expensive or are based on complicated theory.

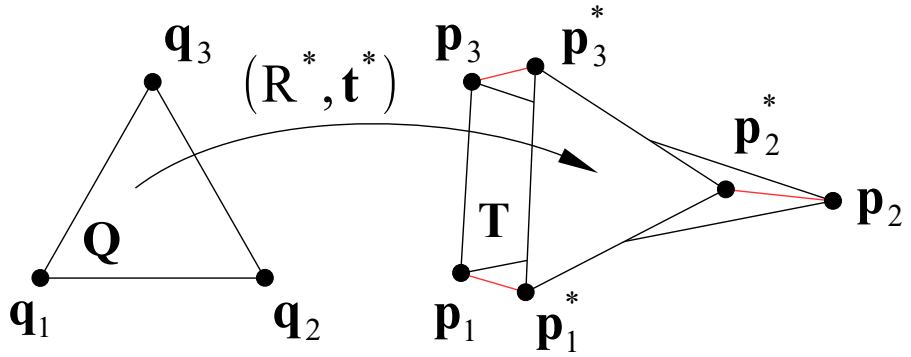



**Figure 2:** The target positions $\mathbf{p}_1^*, \mathbf{p}_2^*, \mathbf{p}_3^*$ computed by rigidly mapping the equilateral triangle $\mathbf{Q}$ to the triangle $\mathbf{T}$. The energy of $\mathbf{T}$ is the sum of squared lengths of the red lines.

As our approach produces a triangulation of the input points, it is somewhat related to mesh generators [ZS00, Che04]. However, such algorithms are not concerned with the blue noise characteristics of the mesh vertices and tend to arrange them in regular hexagonal configurations.

# 2 Algorithm

Let an input point set be distributed in a (periodic or bounded) domain $\mathbf{D} = [x_1, y_1] \times [x_2, y_2] \subset \mathbb{R}^2$. Our method relocates the points such that each one is roughly at a distance $d$ from its near neighbors. How to compute $d$ is explained below. For now, assume that a suitable value is given.

## 2.1 Algorithm Description

**Triangle matching.** We denote by $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$ an equilateral triangle with side length $d$ that has its center of mass at the origin. The exact positions of its vertices do not matter. We call $\mathbf{Q}$ the model triangle. Let $\mathbf{T} = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ be another triangle, for which we compute the target positions $\mathbf{p}_1^*, \mathbf{p}_2^*, \mathbf{p}_3^*$ as

$$\mathbf{p}_i^* = R^* \mathbf{q}_i + \mathbf{t}^*, \quad i = 1, 2, 3, \tag{1}$$

where $R^* \in SO(2)$ is the rotation matrix and $\mathbf{t}^* \in \mathbb{R}^2$ the translation vector, which most closely map $\mathbf{Q}$ to $\mathbf{T}$:

$$(R^*, \mathbf{t}^*) = \underset{R \in SO(2), \mathbf{t} \in \mathbb{R}^2}{\operatorname{argmin}} \sum_{i=1}^{3} \|(R\mathbf{q}_i + \mathbf{t}) - \mathbf{p}_i\|^2. \tag{2}$$

Each target position $\mathbf{p}_i^*$ is the $i$-th vertex of the model triangle $\mathbf{Q}$ optimally mapped to $\mathbf{T}$. Figure 2 provides an illustration. The Procrustes problem (2) has an efficient closed-form solution based on singular value decomposition [AHB87].

**Edge flipping.** We set the energy of a triangle $\mathbf{T} = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ as

$$E(\mathbf{T}) = E(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = \sum_{i=1}^{3} \|\mathbf{p}_i - \mathbf{p}_i^*\|^2. \tag{3}$$

Let $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ and $(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4)$ be two triangles that share the edge $(\mathbf{p}_1, \mathbf{p}_3)$, as shown in Figure 3(c). We flip $(\mathbf{p}_1, \mathbf{p}_3)$ only if this decreases the energy, i.e., if the following holds:

$$E(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4) + E(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) < E(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) + E(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4). \tag{4}$$

**Algorithm steps.** In a preprocessing step, we compute the Delaunay triangulation (periodic or not) of the input points. Let $\mathbf{T}_1, \ldots, \mathbf{T}_m$ be the resulting triangles. In this case, each vertex has as many target positions as adjacent triangles since each triangle contributes with one target position to each of its vertices. The following steps define the rest of the algorithm:

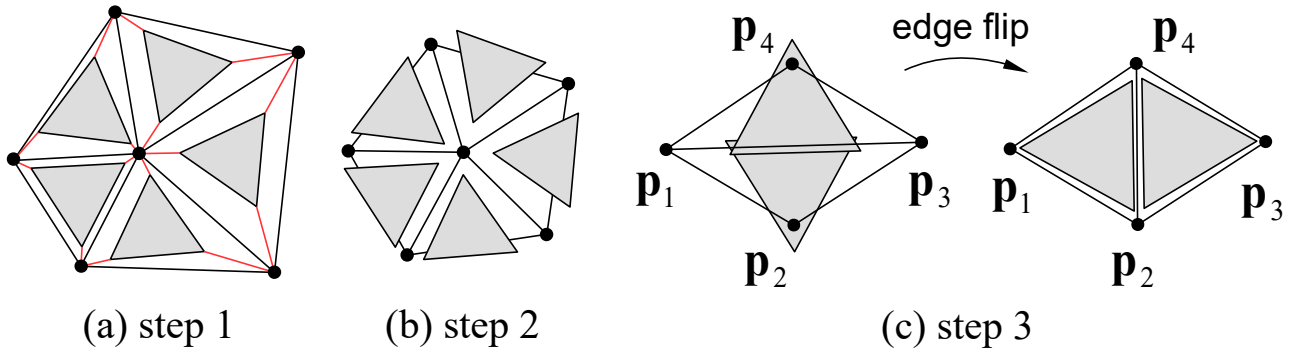



**Figure 3:** (a) The model triangle (gray) is mapped to the triangles in the current triangulation. Each vertex has a corresponding set of target positions (indicated by the red lines). (b) Moving each vertex to the average of its target positions. (c) The edge $(\mathbf{p}_1, \mathbf{p}_3)$ is flipped as the model triangles (gray) better fit the two new triangles and the energy decreases.

1. Match the model triangle $\mathbf{Q}$ to each $\mathbf{T}_k$ (Figure 3(a)).
2. Move each vertex to its average target position (Figure 3(b)).
3. Loop over the edges in random order and flip each one if Equation (4) holds (Figure 3(c)).

One iteration consists of these three steps. The method runs until the energy difference between consecutive iterations falls below a given $\epsilon > 0$. We use $\epsilon = 10^{-5}\big(\text{diagonal}(\mathbf{D})\big)^2$. Note that the Delaunay triangulation is computed only once at the beginning.

**The inter-point distance $d$.** The inter-point distance influences the final point distribution. Assuming $n$ input points, we relate $d$ to the radius $r_{\max} = \big(\text{area}(\mathbf{D})/(2\sqrt{3}n)\big)^{1/2}$ of $n$ equal circles arranged in the densest possible (hexagonal) packing in the rectangular domain $\mathbf{D}$ [SD11].

Circles with a smaller radius can be arranged in very nonuniform ways leading to clusters. This implies that $d$ should be no smaller than $2r_{\max}$. On the other hand, a value too large leads to badly shaped and flipped triangles due to "lack of space". We found experimentally that a value in the range $[2r_{\max}, 3.5r_{\max}]$ leads to similar point distributions with no flipped triangles. This indicates that our method is not particularly sensitive to a specific parameter choice. All results in the paper are obtained with $d = 3r_{\max}$.

## 2.2 Convergence and Complexity

**Convergence.** In the following, we prove the convergence of our method. Let $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the input points and $\mathbf{T}_1, \ldots, \mathbf{T}_m$ the triangles with $\mathbf{T}_k = (\mathbf{p}_{k_1}, \mathbf{p}_{k_2}, \mathbf{p}_{k_3}) \subset \mathcal{P}$. Each $\mathbf{T}_k$ has an instance of the model triangle $\mathbf{Q}$ attached to it by "elastic strings" (shown in red in Figure 3(a)). The sum of squared lengths of all these strings is the energy to minimize. Step 1 reduces it by optimally mapping an instance of $\mathbf{Q}$ to each $\mathbf{T}_k$. Step 2 leads to a further decrease by moving each $\mathbf{p}_i$ to the average of its target positions. (Recall that the average $\mathbf{v}$ of $n$ points minimizes the sum of squared distances from $\mathbf{v}$ to these points.)

More formally, using the triangle energy defined in Equation (3), the energy of the whole triangulation is

$$E = \sum_{k=1}^{m} E(\mathbf{T}_k) = \sum_{k=1}^{m} \sum_{l=1}^{3} \|\mathbf{p}_{kl} - \mathbf{p}^*_{kl}\|^2. \tag{5}$$

Note the slight change of notation due to the fact that, in contrast to (3), we now have $m$ triangles instead of a single one. Still, $\mathbf{p}^*_{kl}$ is the target position of $\mathbf{p}_{kl}$ induced by the triangle $\mathbf{T}_k$.

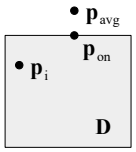
Alternatively, we can sum over all vertices (points) and get

$$E = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \|\mathbf{p}_i - \mathbf{p}^*_{ij}\|^2, \tag{6}$$

where $m_i$ denotes the number of triangles adjacent to $\mathbf{p}_i$ and $\mathbf{p}^*_{ij}$ is the target position of $\mathbf{p}_i$ induced by the $j$-th adjacent triangle to $\mathbf{p}_i$.

Obviously, (5) and (6) are equivalent as in both sums each squared string length appears exactly once, just at a different place. By construction, Step 1 decreases each $E(\mathbf{T}_k)$ in (5) whereas Step 2 decreases each $\sum_{j=1}^{m_i} \|\mathbf{p}_i - \mathbf{p}_{ij}^*\|^2$ in (6). Step 3 also only leads to an energy decrease. Since the energy is bounded from below by zero, the algorithm converges.

**Periodic and bounded domains.** Our algorithm can be implemented both in periodic (toroidal) and bounded domains $\mathbf{D}$. For periodic domains, we compute the periodic Delaunay triangulation [Kru16] and modify the algorithm to take the toroidal nature of $\mathbf{D}$ into account. The changes are obvious and not further discussed. The convergence proof remains unchanged.




In the case of a bounded domain, we initialize the algorithm with the Delaunay triangulation dual to the Voronoi diagram clipped with $\mathbf{D}$ [YWLL13]. In Step 2, each point $\mathbf{p}_i$ is moved to the average $\mathbf{p}_{\text{avg}}$ of its target positions. If $\mathbf{p}_{\text{avg}}$ happens to lie outside $\mathbf{D}$, $\mathbf{p}_i$ is moved to $\mathbf{p}_{\text{on}}$, which is the point in $\mathbf{D}$ closest to $\mathbf{p}_{\text{avg}}$.

Again, the convergence of the algorithm is guaranteed. Note that $\|\mathbf{p}_{\text{on}} - \mathbf{p}_{\text{avg}}\| \leq \|\mathbf{p}_i - \mathbf{p}_{\text{avg}}\|$, i.e, the new position $\mathbf{p}_{\text{on}}$ gets closer (or stays at the same distance) to the optimal $\mathbf{p}_{\text{avg}}$. This implies that $\sum_{j=1}^{m_i} \|\mathbf{p}_{\text{on}} - \mathbf{p}_{ij}^*\|^2 \leq \sum_{j=1}^{m_i} \|\mathbf{p}_i - \mathbf{p}_{ij}^*\|^2$ i.e., moving $\mathbf{p}_i$ to $\mathbf{p}_{\text{on}}$ can only decrease the energy defined in (6).

**Complexity.** Assuming $n$ points in 2D, the Delaunay triangulation can be computed in $O(n \log n)$ time and $O(n)$ space complexity [GM01]. Both the number of edges and triangles in the triangulation depend linearly on $n$ [Lut06]. In each iteration, our method loops once over the triangles, edges and vertices (points), which yields a time complexity of $O(n)$ per iteration. The memory consumption depends linearly on the triangulation size.

# 3 Experimental Results

In this section, we provide an experimental analysis of our algorithm. It is implemented in C++ and all tests were performed on a laptop with 8GB RAM using a single core of an Intel M-5Y70 1.10GHz CPU.

**Blue-noise characteristics.** In Section 2.2, we proved that our method is guaranteed to converge. We now provide a short justification and present experimental evidence that it converges to blue-noise patterns. It is well-known that point distributions generated by optimizing capacity-constrained Voronoi tessellations or capacity-constrained Delaunay triangulations have excellent blue-noise characteristics [BSD09, XLGG11, dGBOD12, JZW+15, ZGZ+16]. Our method also belongs to this class of approaches. It optimizes a triangulation to have all its triangles as similar as possible to one and the same equilateral triangle. In particular, during the optimization the triangles are resized to have roughly the same area (capacity).

We perform a spectral analysis by averaging the periodograms of 10 point distributions generated over the 2D periodic square $[0, 1]^2$ [Uli87]. This provides us with an estimate of the point set's power spectrum, which is used two derive the radially averaged power spectrum and the anisotropy. We compute these quantities with a dedicated open source software tool [SD11].

Figure 1 shows a typical point distribution generated with our method using the convergence criterion presented in Section 2.1. Note that the points are evenly distributed and exhibit no regularity artifacts. Furthermore, the averaged power spectra and the anisotropy plot are typical for blue noise patterns.

**Qualitative Comparison.** Figure 4 provides a qualitative comparison of our method to three other point set optimizers, namely, Lloyd's relaxation [LWL+09], FPO [SHD11] and CCVT [BSD09]. Our approach generates patterns with better blue noise characteristics than Lloyd's method, which is known to produce high-quality triangulations at the cost of regularity artifacts. The artifacts are visible both in the spatial domain as large hexagonal patches as well as in the frequency domain leading to turbulent power spectra. Our algorithm yields results comparable to FPO and CCVT.

**Performance.** We compared the performance of our method to FPO and CCVT. The processing time, the time per iteration and the number of iterations as function of the input size are plotted in Figure 5. The results render our method as the most efficient one and support the linear time complexity per iteration derived in Section 2.2. Furthermore, the figure suggests that the number of iterations does not depend on the number of input points as long as the initial point distribution is uniformly random.

**Triangle shape quality.** Even though not of primary interest, we evaluated the quality of the triangulation produced by our method and compared it to the ones of FPO and CCVT. The middle row in Figure 4 shows typical triangulations generated by the algorithms. To make a quantitative comparison we use the following

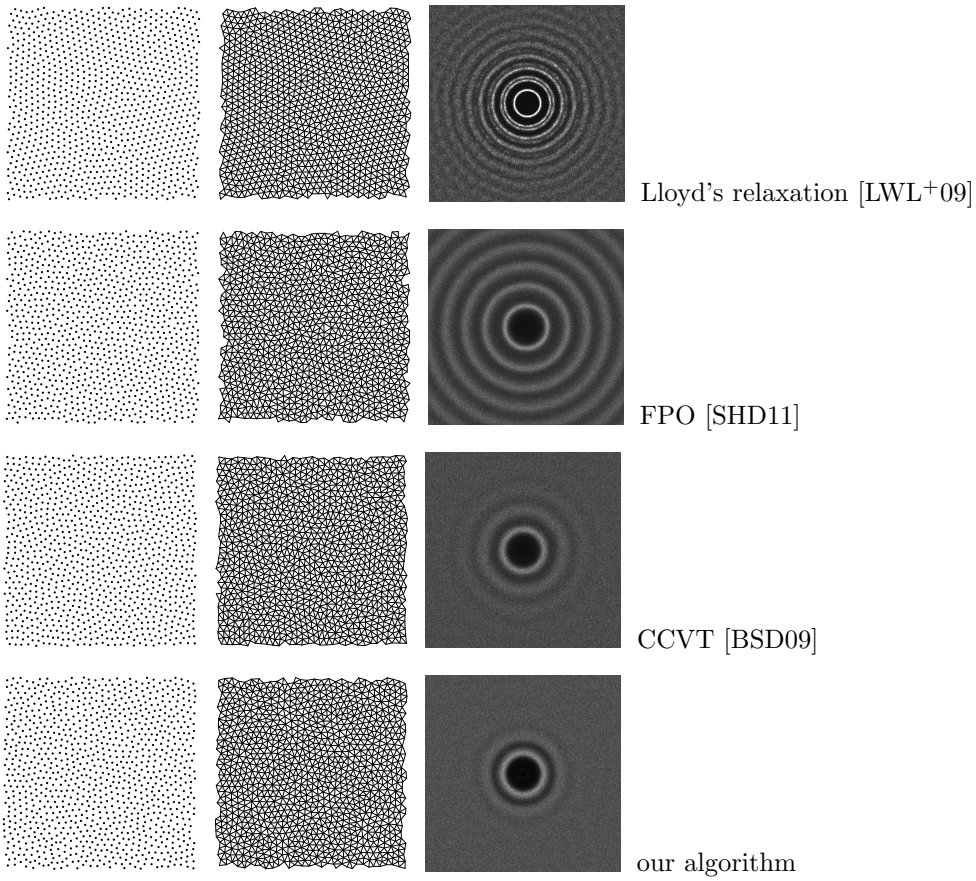



**Figure 4:** Comparison of three point set optimizers with our algorithm. Optimized point sets (1st column), triangulations (2nd column) and power spectra (3rd column).

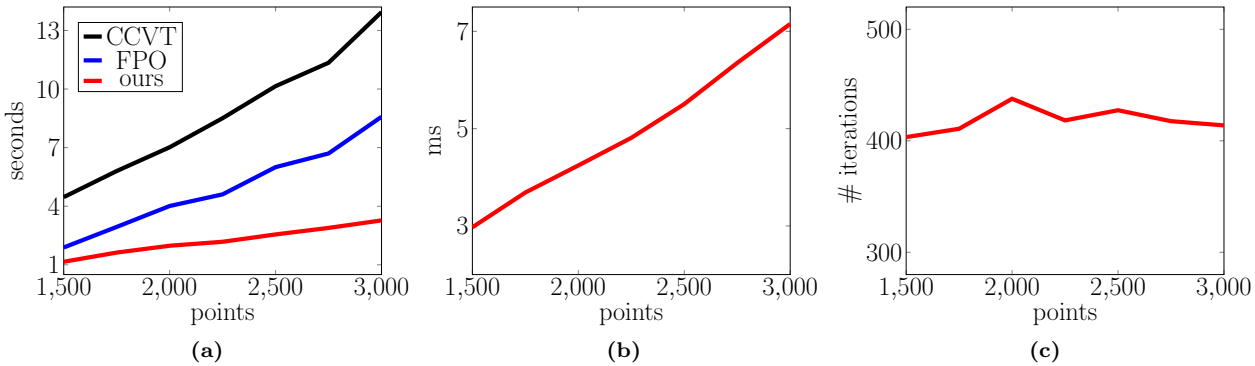



**Figure 5:** (a) Computational time of various optimizers showing that ours is most efficient. (b) Computational time per iteration of our algorithm. The curve indicates the linear time complexity per iteration. (c) Number of iterations. (Best viewed in color.)

measures. The first one is the angle RMS error

$$\mathrm{E}_\alpha = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\alpha_i - 60^\circ)^2}, \tag{7}$$

where $M$ is the number of angles in the triangulation (three times the number of triangles) and $\alpha_i$ is the $i$-th angle. The other measures are the triangle quality $\rho_i$ and its average $\bar{\rho}$ over all triangles:

$$\rho_i = \frac{2r_i}{R_i}, \qquad \bar{\rho} = \frac{1}{m}\sum_{i=1}^{m}\rho_i, \tag{8}$$

where $r_i$ (resp. $R_i$) is the radius of the incircle (resp. circumcircle) of the $i$-th triangle and $m$ is the number of triangles. The factor 2 is included to normalize $\rho_i$ to the range $[0, 1]$, with $\rho_i = 1$ indicating an equilateral triangle [PB03]. The angle RMS errors and the average triangle quality of the three methods are listed in Table 1.

| algorithm | angle RMS error | average triangle quality |
|---|---|---|
| FPO | 13.0 | 0.93 |
| CCVT | 14.5 | 0.91 |
| ours | 11.7 | 0.94 |

**Table 1:** Triangulation quality measures. Our method has the lowest angle RMS error $\mathrm{E}_\alpha$ (see (7)) and the highest average triangle quality $\bar{\rho}$ (see (8)).

# 4 Conclusions and Future Work

We presented a novel approach for optimizing the blue noise profile of point sets in 2D periodic or bounded domains. It is based on Procrustes transformations that optimally match an equilateral triangle of suitable size to the elements of a triangulation. This procedure is used for both vertex displacement and connectivity adjustment, leading to a very simple, efficient and provably convergent optimization algorithm. We experimentally validated that the generated output has very good blue noise characteristics.

Our algorithm can be extended to 3D. Instead of computing the Delaunay triangulation we would compute the Delaunay tetrahedralization of the input points. The equilateral model triangle becomes a regular tetrahedron, which would be mapped to the elements of the tetrahedralization. Instead of flipping edges we would flip edges and/or faces.

Some applications require parts of the sampling domain to be resolved in higher accuracy than others. Such local adaptivity could be achieved by varying the side length of the equilateral triangle (or regular tetrahedron) according to a given density function.

# References


[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Trans. PAMI*, 9(5):698–700, 1987.

[BM92] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Trans. PAMI*, 14(2):239–256, 1992.

[BSD09] M. Balzer, T. Schlömer, and O. Deussen. Capacity-Constrained Point Distributions: A Variant of Lloyd's Method. *ACM Trans. Graph.*, 28(3):1–8, 2009.

[Che04] L. Chen. Mesh smoothing schemes based on optimal Delaunay triangulations. In *Proceedings of the International Meshing Roundtable*, pages 109–120, 2004.

[dGBOD12] F. de Goes, K. Breeden, V. Ostromoukhov, and M. Desbrun. Blue Noise Through Optimal Transport. *ACM Trans. Graph.*, 31(6):171:1–171:11, 2012.

[Fat11] R. Fattal. Blue-noise Point Sampling Using Kernel Density Model. In *Proceedings of ACM SIGGRAPH*, pages 48:1–48:12, 2011.

[GM01] C. I. Grima and A. Marquez. *Computational Geometry on Surfaces*. Springer Netherlands, 2001.

[JZW+15] M. Jiang, Y. Zhou, R. Wang, R. Southern, and J. J. Zhang. Blue Noise Sampling Using an SPH-based Method. *ACM Trans. Graph.*, 34(6):211:1–211:11, 2015.

[Kru16] N. Kruithof. 2D Periodic Triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.9 edition, 2016.

[Llo82] S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. Inf. Theor.*, 28(2):129–137, 1982.

[Lut06] F. H. Lutz. Triangulated Manifolds with Few Vertices: Combinatorial Manifolds. Technical Report 06/08, ZIB, 2006.

[LWL+09] 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. Graph.*, 28(4):101:1–101:17, 2009.

[LZX+08] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A Local/Global Approach to Mesh Parameterization. In *Proceedings of the Symposium on Geometry Processing*, pages 1495–1504, 2008.

[MHTG05] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, 2005.

[PB03] P. P. Pébay and T. J. Baker. Analysis of Triangle Quality Measures. *Math. Comput.*, 72(244):1817–1839, October 2003.

[SD11] T. Schlömer and O. Deussen. Accurate Spectral Analysis of Two-Dimensional Point Sets. *Journal of Graphics, GPU, and Game Tools*, 15(3):152–160, 2011.

[SHD11] T. Schlömer, D. Heck, and O. Deussen. Farthest-point Optimized Point Sets with Maximized Minimum Distance. In *Proceedings of the ACM Symposium on High Performance Graphics*, pages 135–142, 2011.

[Uli87] R. Ulichney. *Digital Halftoning*. MIT Press, 1987.

[XLGG11] Y. Xu, L. Liu, C. Gotsman, and S. J. Gortler. Capacity-Constrained Delaunay Triangulation for Point Distributions. *Comput. Graph.*, 35(3):510–516, 2011.

[YGW+15] D.-M. Yan, J.-W. Guo, B. Wang, X.-P. Zhang, and P. Wonka. A Survey of Blue-Noise Sampling and its Applications. *J. Comput. Sci. Technol.*, 30(3):439–452, 2015.

[YWLL13] D.-M. Yan, W. W., B. Lévy, and Y. Liu. Efficient Computation of Clipped Voronoi Diagram for Mesh Generation. *Computer-Aided Design*, 45(4):843–852, 2013.

[ZGZ+16] S. Zhang, J. Guo, H. Zhang, X. Jia, D.-M. Yan, J. Yong, and P. Wonka. Capacity Constrained Blue-Noise Sampling on Surfaces. *Comput. Graph.*, 55:44–54, 2016.

[ZS00] T. Zhou and K. Shimada. An Angle-Based Approach to Two-Dimensional Mesh Smoothing. In *Proceedings of the International Meshing Roundtable*, pages 373–384, 2000.