Supplementary Material: Gradient-based Approximation of Nonuniform Low-Discrepancy Samples

ACM Reference Format:
. 2026. Supplementary Material: Gradient-based Approximation of Nonuniform Low-Discrepancy Samples. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3799902.3811140

1 Neural Architecture

Input encoding and parameters. To alleviate the difficulty of learning high-frequency variation with MLPs, we encode both the toroidal shift and the distribution parameter using Fourier features [Tancik et al. 2020]. $e_x = [cos(2\pi Bx), sin(2\pi Bx)]^T$ where each entry in B is sampled from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. We encode all the input including distribution parameters to a 32-dimensional feature vector with $\sigma = 0.1$. The network is initialized as the identity map (i.e., output Sobol points given input Sobol points) by simply setting the last layer's weight and bias to zero.

Sample randomization. In rendering, randomization of a low-discrepancy sequence is often used to produce different sample sets from the same set. Randomization of non-uniform point sets is nontrivial. Our network can generate such different point sets for the same distribution by incorporating randomization as the input to the network. To this end, we apply Cranley-Patterson rotation [Cranley and Patterson 1976] to the input Sobol samples as the base point set. For directions, because the points do not need to be within $[0, 1]^2$ for later square-to-sphere mapping, we do not apply toroidal mapping for shifted Sobol for better smoothness in training. While a better randomization method exists to keep the discrepancy low [Owen 1995], we found that the exact form of randomization has little influence on our method, since it directly optimizes discrepancy. We also found that scrambling is usually specified as a discrete choice, making scrambling parameters unsuitable as inputs for gradient-based network training. Since Sobol is deterministic and Cranley-Patterson rotation is determined by an offset, we only need to input the Cranley-Pattern offset to the network to represent a randomized point set. The network then outputs the differences in positions from the input randomized point set. These differences are then added to the input point set to obtain the final point set. Encoding differences instead of the point positions directly greatly helps us avoiding mode-collapsing during the training due to the randomization of the input point set.

Author's Contact Information:

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811140

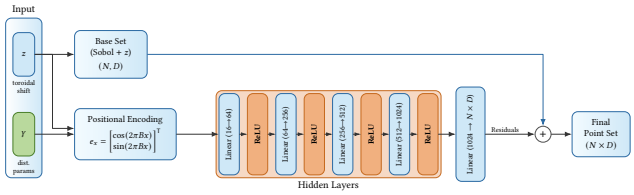

Fig. 1. The illustration for our neural network. The network takes offset z and parameter γ as input, outputs offsets for each point in the base point set to get the target point set. The base point set is generated by applying Cranley-Patterson rotation with offset z to Sobol point set.

ALGORITHM 1: Non-Uniform Low-Discrepancy Set Optimization

Input: *iters*, $\{x_1, \ldots, x_N\}$, M, K, q, α, β
Output: a approximately low-discrepancy set $\{x_1, x_2, ..., x_N\}$

for *iters* number of iterations **do**
  Sample M multiple-corner boxes $A_1, \ldots, A_M$
  Sample K space samples $s_1, \ldots, s_K$ in $[0, 1]^d$ from a density p
  Compute $q(s_i)$ for each space sample s_i
  Compute $\widehat{\mu}(A_m) = \frac{1}{K} \sum_{i=1}^{K} \frac{\mathbb{1}_{A_m}(q(s_i))}{p(s_i)}$ for each box A_m
  Compute distance $d_{(A_m, x_i)}$ between each pair of A_m, x_i
  $\left(\widehat{D}_2^{asd}\right)^2(\mu) = \frac{1}{M} \sum_{m=1}^{M} \left(\frac{1}{N} \sum_{i=1}^{N} \sigma\left(\alpha\, d_{(A,x_i)}\right) - \widehat{\mu}(A_m)\right)^2$
  $x_i = x_i - \beta \frac{\partial \left(\widehat{D}_2^{asd}\right)^2(\mu)}{\partial x_i}$
end for
Return: $\{x_1, x_2, ..., x_N\}$

Initialization. The network is initialized as the identity map (i.e., returns the input Sobol points) by simply setting the last layer's weight and bias to zero, meaning the difference would be initially zero.

Architecture, hyperparameters, and training. Our network has five fully connected layers. The last layer outputs all point set residuals. The hidden layer size can be variable considering the distribution's complexity and point set size to balance time cost and performance. We used (64, 256, 512, 1024) as hidden layer sizes. The input offset z and the parameters γ are concatenated and jointly encoded using Fourier features, producing a 32-dimensional vector. These encodings are also passed to each layer in the network. Fig. 1 illustrates the network architecture. Training uses a batch size of 16, where each entry of the batch samples a different set of input parameters z and γ. We use the Adam optimizer [Kingma and Ba 2014] to update the network weights, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$, and we set the weight decay coefficient to zero. We adaptively decay the learning rate from 5×10^{-4} to 5×10^{-7}.

Pseudocode. Algorithm 1 presents the pseudocode for our optimization procedure.

2 Additional Results

Optimization settings. For 2D uniform optimization, we run the direct optimization for 2000 epochs with a log-shaped learning-rate decay and a linearly increasing sigmoid scale. For point sets up to $N \leq 1024$, we use $M = 8$K sampled regions (rectangles) and decay the learning rate from 10^{-3} to 10^{-7}, while increasing the sigmoid scale from $\alpha = 20$ to $\alpha = 2000$ linearly over the first 1000 epochs. For larger point sets ($N \in \{2048, 4096\}$), we increase the accuracy of the discrepancy estimate by using $M = 32$K regions, reduce the initial learning rate to 10^{-4} (still decaying to 10^{-7}), and increase the maximum sigmoid scale to $\alpha_{\max} = 6000$.

For 2D non-uniform optimization, we run 1000 epochs and linearly increase the sigmoid scale over the first 500 epochs. When the rectangle area $\mu(A_m)$ can be evaluated analytically (e.g., for the bilinear density), we use $M = 8$K regions for $N \leq 1024$ with a learning-rate decay from 10^{-4} to 10^{-7} and α increasing from 20 to 2000. For $N \in \{2048, 4096\}$ we use 32K and increase the cap to $\alpha_{\max} = 6000$. For densities that require Monte Carlo estimation of region mass, we use 4K regions for $N \leq 1024$ and again increase to $M = 32$K for $N \in \{2048, 4096\}$, while keeping the same learning-rate and sigmoid schedules. For Monte Carlo area estimation, we use $K = 2^{17}$ space samples. Since the discrepancy computation scales quadratically in both N and M, these settings can lead to high memory usage. We therefore batch the computation over both sampled regions and points.

Role of Density Mapping. When an inverse CDF transform is available, we apply it during optimization to map points from the unit hypercube to the target density. For unmapped optimization, we found that the results are more sensitive to hyperparameter choices and boundary handling, as points may accumulate near domain edges or remain in zero-density regions and require additional optimization steps. Fig. 3 compares clamping and toroidal wrapping for unmapped points against the mapped case with clamping, showing that toroidal wrapping substantially improves unmapped optimization, although density mapping remains the most effective overall.

Non-uniform discrepancy improvements. To highlight the benefits of directly optimizing the non-uniform discrepancy over uniformly optimized point sets, we also optimized a point set for uniform discrepancy and then mapped it via an inverse CDF transform, as we do for the uniform low-discrepancy baselines. Fig. 4 shows that uniformly optimized point sets behave similarly to uniform low-discrepancy sequences while achieving smaller discrepancy values. For discontinuous target densities, such as the quarter-disc and the image, our non-uniformly optimized samples yield further improvements and exhibit better asymptotic behavior. For a smooth target density such as the bilinear, the two approaches perform similarly. These results also support our claim that mapping can affect discrepancy and it is better to directly generate non-uniform samples than mapping uniform samples.

Non-uniform baselines. Some point set generation methods such as GBN [Ahmed et al. 2022] and SOT [Paulin et al. 2020] can also directly generate optimized point sets for a prescribed density (albeit not optimizing discrepancy), which we compared against in our paper. For non-uniform SOT, we use the implementation of Salaün et al. [Salaün et al. 2022], since the code released by Paulin et al. [Paulin et al. 2020] does not directly support non-uniform target densities. For completeness, here we compare their direct generation of non-uniform samples and uniform samples mapped to non-uniform samples. Please notice that both SOT and GBN implementations of non-uniform samples focus on cases like image stippling, which could lead to extra errors or bias in our evaluation. Fig. 2 reports numerical integration errors for product integrands using GBN and SOT samples that are optimized either uniformly and then mapped, or directly for the non-uniform target. With the Gaussian as the importance target, both variants achieve similar errors for the product with the step function. In contrast, for the binary quarter-disc target, the uniform-then-mapped variants yield smaller errors than the direct non-uniform variants on the quarter-disc–bilinear product. As such, contrary to what one might believe, we observed that direct non-uniform optimization in SOT and GBN does *not* necessarily outperform uniformly optimized point sets that are subsequently mapped to the target density.

Qualitative point-set visualization. In Fig. 5, we visualize our 2D optimized point sets for both uniform and non-uniform quarter disc, bilinear, and image densities. This figure provides a qualitative view of how the spatial structure of our points evolves as N increases.

Fourier analysis. Based on the visualization of points, one might think our method also produces blue-noise patterns. We additionally analyze the Fourier spectra of the resulting point sets in Fig. 6. Since we initialize the optimization from scrambled Sobol, the optimized point sets exhibit a similar spectral profile. Since we optimize for discrepancy rather than spectral properties, we do not observe the blue-noise characteristic of low energy at small frequencies. Note that we do not claim to generate blue-noise samples from our method, so the results do not invalidate our contribution.

3 Rendering Experiment

BRDF Sampling. In the paper, we compare different sampling methods on the task of BRDF importance sampling for the widely-used Disney BRDF [Burley 2012]. The Disney BRDF consists of several different BRDF components, and we choose three of them: diffuse, subsurface BRDF, and microfacet-based specular lobe. We use an instance whose parameters are $(0.58, 0.2, 0.5, 0.15)$ as roughness, metallic, subsurface, and anisotropy. Other parameters are all zero, and colors are RGB $(1, 1, 1)$. All directions are defined in a local space where $(0, 0, 1)$ is the normal, $(1, 0, 0)$ is the tangent direction, and the view direction ω_o is $(\sqrt{2}/2, 0, \sqrt{2}/2)$.

The Disney model takes two-dimensional random numbers for importance sampling. The deterministic mapping uses the first random number to select a BRDF component uniformly at random, and then rescales it to $[0, 1)$. The rescaled value together with the second number are then used either for cosine-weighted hemisphere sampling for diffuse and subsurface components, or for sampling a visible normal distribution function [Heitz 2018] for the microfacet-based specular lobe.

Environment Map Sampling. We take three environment maps from Debevec's light probe gallery [Debevec 1998]. We use inversion

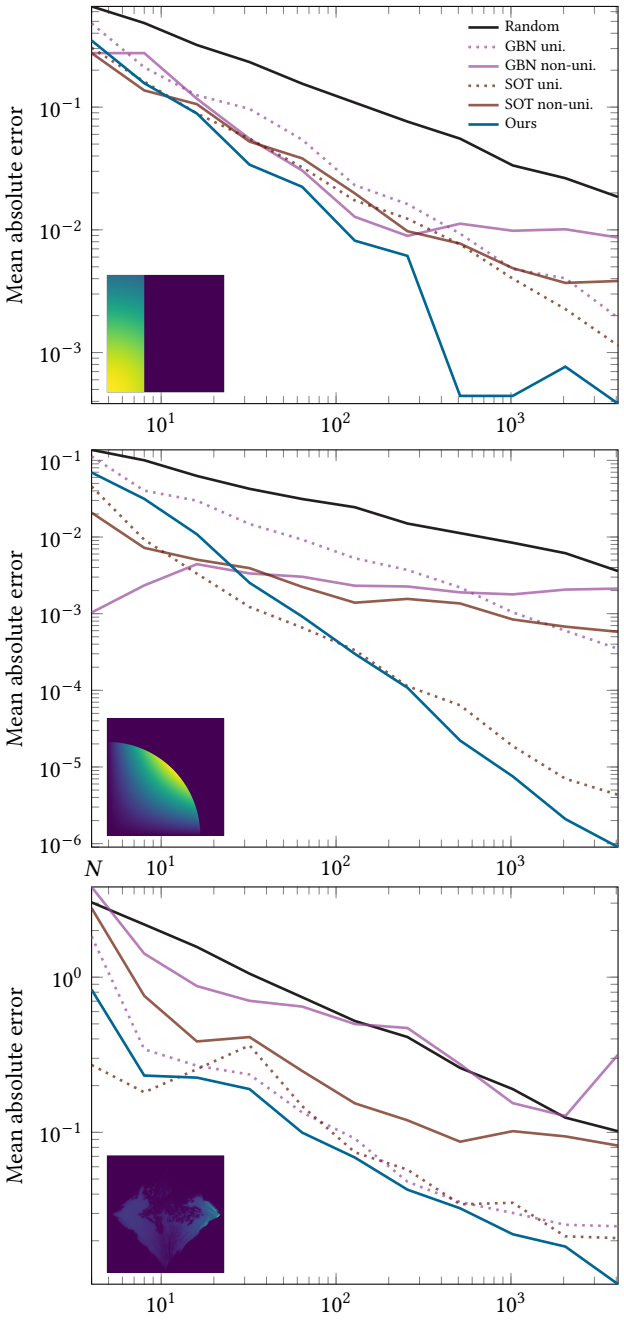

Fig. 2. Numerical integration errors of uniformly and non-uniformly optimized GBN and SOT samples in both 2D (first two rows) and spherical domains (last row). For these methods, non-uniformly optimized samples are not necessarily useful for Monte Carlo integration.

sampling for discrete 2D importance sampling, where the luminance of pixels is used as the unnormalized density.

For both BRDF and environment map, importance sampling for Sobol and Halton is done by deterministic mapping via the inverse CDF and multiple importance sampling [Veach 1998] for Disney BRDF's components. Our neural network is trained to learn the target distribution directly on the spherical domain. For GBN and SOT, we generate non-uniform samples by optimizing with respect to a density map. Since neither method natively supports sampling directly on the spherical domain, we map the spherical target density back to a 2D parameterization and optimize in the square instead. This approximation may introduce distortions and could explain the bias we observe in some experiments.

Our method generates 16 instances starting from an instance of scrambled Sobol samples and learns the same distribution density as other samples. We use 16384 regions, 65536 space samples, and 5000 epochs to train instances. Each instance with 2048 samples takes about 460 seconds to complete the training on an RTX5090 laptop. This time could reduce greatly to about 70 seconds if 4096 regions and 16384 space samples are used, which we found to be sufficient for 2048 samples in most cases. All numerical results are reported as averages over 128 instances of conventional samples and 16 instances of our samples. Uniform blue noise samples and non-uniform SOT averages over 16 instances.

Rendering Application. We use 8192 regions, 32768 space samples, parameter batch size 16 and 15000 epochs to train the model with Adam optimizer. The other parameters are the same as those of the previous experiments. The 3D scenes being rendered are a Light Transport Equation Orb (lte-orb) modeled by Yasutoshi Mori and a dragon model by Christian Schüller. The cup model is modified from the model by Simon Wendsche. The ground truth is rendered with 2^{17} Halton samples for lte-orb, 2^{13} Halton samples for dragon model, and 2^{16} Halton samples for the cup.

References

Abdalla G. M. Ahmed, Jing Ren, and Peter Wonka. 2022. Gaussian Blue Noise. *ACM Trans. Graph.* 41, 6, Article 260 (Nov. 2022), 15 pages. doi:10.1145/3550454.3555519

Brent Burley. 2012. Physically-Based Shading at Disney. In *ACM SIGGRAPH 2012 Courses*. Association for Computing Machinery, New York, NY, USA, 1–7. doi:10.1145/2343483.2343493

Roy Cranley and Thomas N.L. Patterson. 1976. Randomization of number theoretic methods for multiple integration. *SIAM J. Numer. Anal.* 13, 6 (1976), 904–914.

Paul E. Debevec. 1998. Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics With Global Illumination and High Dynamic Range Photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery, New York, NY, USA, 189–198. doi:10.1145/280814.280864

Eric Heitz. 2018. Sampling the GGX distribution of visible normals. *Journal of Computer Graphics Techniques (JCGT)* 7, 4 (2018), 1–13.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Art B. Owen. 1995. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing: Proceedings of a conference at the University of Nevada, Las Vegas, Nevada, USA, June 23–25, 1994*. Springer, 299–317.

Lois Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Antoine Webanck, Mathieu Desbrun, and Victor Ostromoukhov. 2020. Sliced optimal transport sampling. *ACM Trans. Graph.* 39, 4 (2020), 99.

Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022. Scalable Multi-Class Sampling via Filtered Sliced Optimal Transport. *ACM Trans. Graph.* 41, 6, Article 261 (nov 2022), 14 pages. doi:10.1145/3550454.3555484

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS* (2020).

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation.* Stanford University.

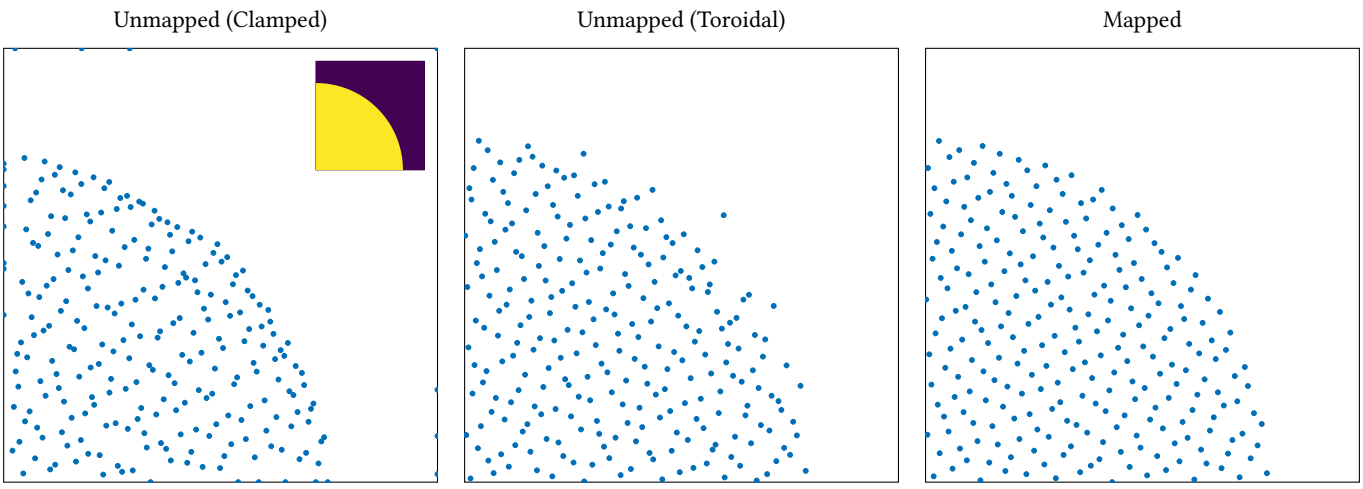

Fig. 3. Comparison of optimization with and without density mapping. Clamping during unmapped optimization can cause points to accumulate at domain edges. Toroidal wrapping greatly reduces this issue, but some points may still remain in zero-density regions. Mapping the points during optimization avoids these problems and better matches the target density.

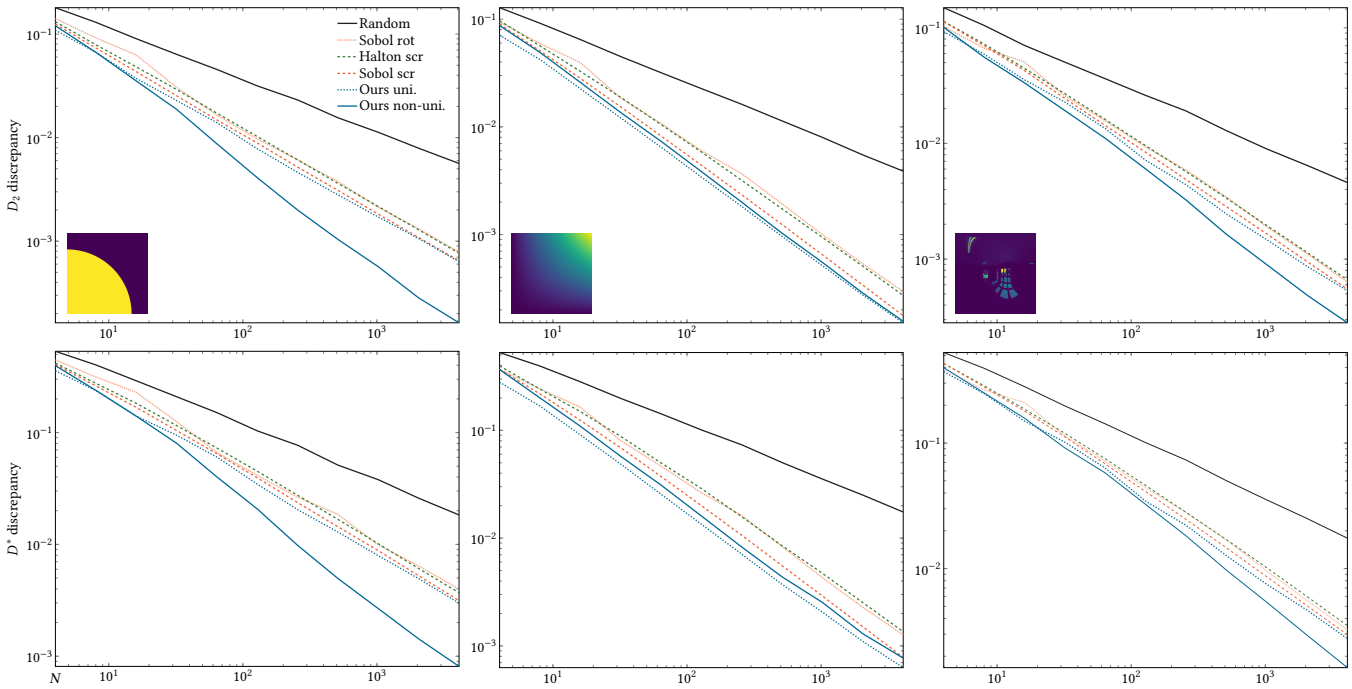

Fig. 4. 2D point samples: L_2 and star discrepancy values (D_2 and D^*) for different numbers N of samples under non-uniform distributions. We additionally plot our uniform optimized samples that are mapped via the inverse transform. We see that directly optimizing for the nonuniform density brings additional improvements.

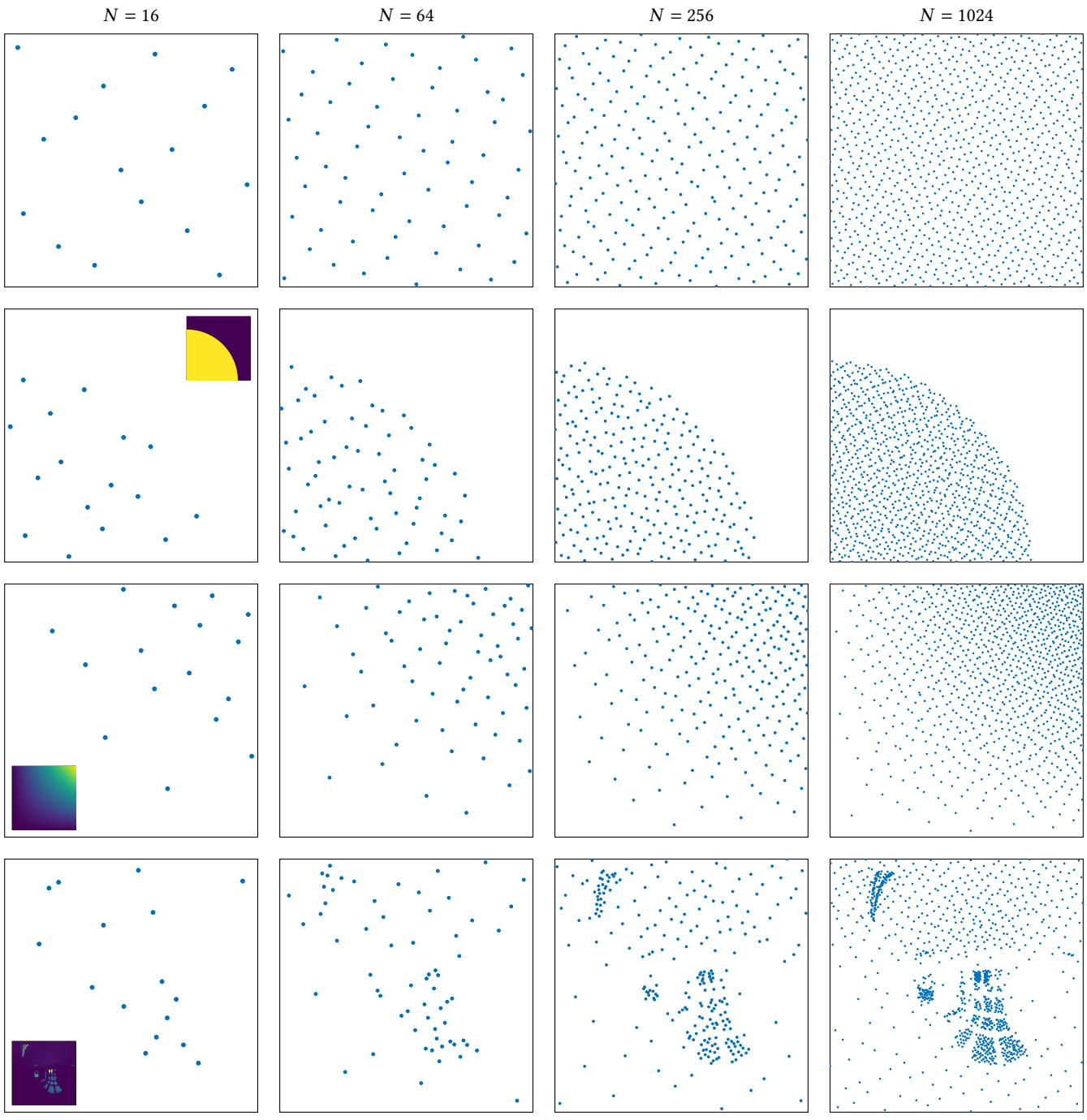

Fig. 5. Visualization of 2D point sets produced by our optimization for uniform and non-uniform targets at increasing sample counts N.

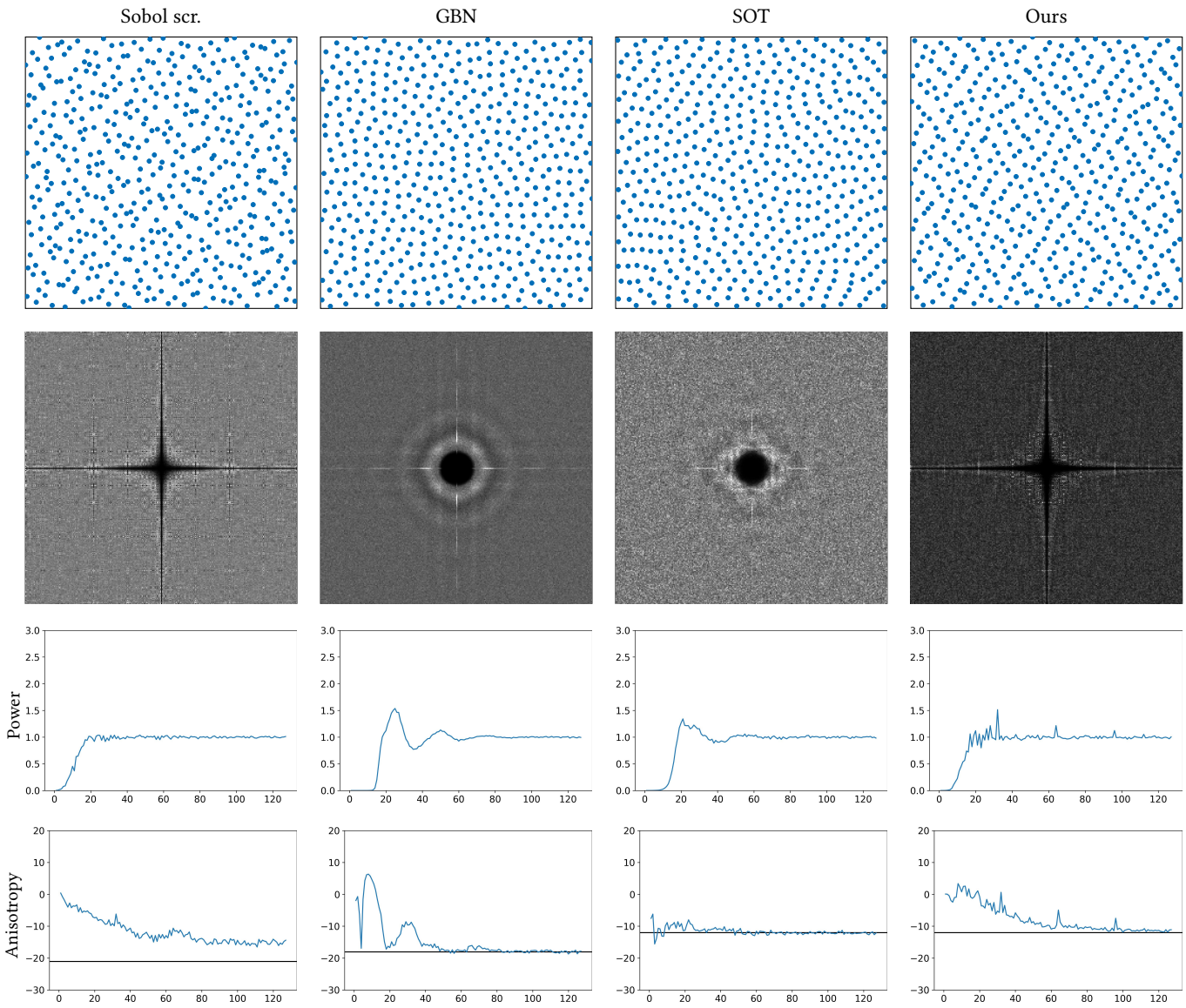

Fig. 6. Power spectrum analysis for several sample patterns. Our samples exhibit a spectrum similar to that of scrambled Sobol, which we use to initialize the optimization.