Example-Based Sampling with Diffusion Models

BASTIEN DOIGNIES, Univ. Lyon, France
NICOLAS BONNEEL, CNRS, Univ. Lyon, France
DAVID COEURJOLLY, CNRS, Univ. Lyon, France
JULIE DIGNE, CNRS, Univ. Lyon, France
LOÏS PAULIN, Univ. Lyon, France
JEAN-CLAUDE IEHL, Univ. Lyon, France
VICTOR OSTROMOUKHOV, Univ. Lyon, France

Much effort has been put into developing samplers with specific properties, such as producing blue noise, low-discrepancy, lattice or Poisson disk samples. These samplers can be slow if they rely on optimization processes, may rely on a wide range of numerical methods, are not always differentiable. The success of recent diffusion models for image generation suggests that these models could be appropriate for learning how to generate point sets from examples. However, their convolutional nature makes these methods impractical for dealing with scattered data such as point sets. We propose a generic way to produce 2-d point sets imitating existing samplers from observed point sets using a diffusion model. We address the problem of convolutional layers by leveraging neighborhood information from an optimal transport matching to a uniform grid, that allows us to benefit from fast convolutions on grids, and to support the example-based learning of non-uniform sampling patterns. We demonstrate how the differentiability of our approach can be used to optimize point sets to enforce properties.

1 INTRODUCTION

A wide range of samplers have been designed in the past, for quasi-Monte Carlo integration, rendering, image stippling, positionning objects or generally, to uniformly or non-uniformly cover some space. The generated samples can have various properties, such as being low discrepancy or stratified, having a blue noise spectrum, producing low integration error, with high packing density, satisfying a Poisson disk criterion, or high inter-point distances [Pharr et al. 2016; Singh et al. 2019]. Generating these samples can come at significant cost, especially when points are obtained from complex optimization schemes [Ahmed et al. 2022; De Goes et al. 2012; Fattal 2011; Öztireli and Gross 2012; Paulin et al. 2020; Roveri et al. 2017]. In addition, satisfying multiple properties at the same time is difficult, and is the focus of entire methods – e.g., generating low discrepancy sequences with blue noise properties. Differentiability can also be desirable in contexts involving further optimizations, but may be problematic for specific samplers, for instance when considered in a differential renderer [Jakob et al. 2022b]. The large set of available samplers makes sample generation little generic, with methods involving smooth non-convex optimization, integer linear programming, number theory, bruteforce approaches with clever data structures, etc.

Recently, diffusion models have become extremely popular in the context of image generation [Ho et al. 2020; Rombach et al. 2022; Sohl-Dickstein et al. 2015]. By learning how to denoise an image that initially only contains random values, these models have been able to produce impressive results, i.e., to learn the very fine structure of the manifold of realistic images. It hence seems judicious to take advantage of these models to learn the very fine structure of sample points produced by existing samplers. However, these models heavily rely on convolutions, which makes it impractical to efficiently handle point sets.

In this paper, we propose to learn the distribution of 2-d samples produced by a wide range of samplers using a diffusion model. When point sets are not stratified, we resort to an optimal transport matching to a uniform grid that mostly preserves neighborhood information so as to benefit from efficient convolutional layers. We demonstrate that a single architecture is able to learn sample points produced by different methods, and even allows to reproduce non-uniform point sets. The differentiability of our network allows us to add properties to a given samplers, e.g., allowing to add low discrepancy properties to a given optimal transport-based sampler. While our network is currently limited to generating 2-d samples, it produces samples beyond the range of samples count it has been trained on. We provide trained networks alongside the paper and believe this exciting step will open the door to further conditioning. Code is provided in supplementary material.

2 RELATED WORKS

Existing samplers have a wide range of properties. We enumerate importants classes of samplers below.

Blue Noise. Blue noise samples have a characteristic "ring-like" Fourier power spectrum, with low frequencies converging to zero. They are interesting for Monte Carlo integration purposes [Pilleboue et al. 2015; Subr and Kautz 2013], digital halftoning [Ulichney 1987] or stippling [Deussen et al. 2000] and well describe arrangements of natural phenomenas that have been optimized through evolution such as the retinal distribution of cones [Yellott 1982]. They are often costly obtained through optimization, for instance using kernel approaches [Ahmed et al. 2022; Fattal 2011], pair-correlation function [Öztireli and Gross 2012] or optimal transport [De Goes et al. 2012; Paulin et al. 2020; Qin et al. 2017], though fast approximations exist [Nader and Guennebaud 2018]. Tile-based approaches precompute tiles for fast synthesis, but are memory demanding [Kopf et al. 2006; Ostromoukhov et al. 2004; Wachtel et al. 2014].

Poisson Disk. Poisson disk samples have the property that no point fall within a distance smaller than a threshold from another point [Bridson 2007; Dunbar and Humphreys 2006; Gamito and Maddock 2009; Wei 2008; Yuksel 2015]. Their spectra resemble those of blue noise distributions, except that they do not decrease towards zero as the frequency decreases [Pilleboue et al. 2015]. They naturally occur in other natural process such as the placement of trees in a forest. In low dimensions, they are relatively fast to compute.

Low Discrepancy Sequences. Discrepancy is a uniformity measure directly related to Monte Carlo integration error. Low discrepancy sequences (LDS) thus have several advantages. First they are sequences, so that samples can be progressively added. Second, they are low discrepancy, hence guaranteeing good numerical integration error [Lemieux 2009; Niederreiter 1992]. Samplers achieving low discrepancy usually rely on arithmetic and number theory constructions leading to extremely fast generators (*e.g.* in base 2, the i-th sample using [Sobol' 1967] is given by a matrix/vector multiplication in $GF(2)$ on the bitwise representation of i). Alternatively, lattices produce low discrepancy sequences. A rank-1 lattice repeatedly translates an initial point by a given amount in a given direction in a toric domain [Keller 2004]. Rank-n lattices similarly use multiple independent vectors. Good lattices can be similarly hard to optimize for [L'Ecuyer and Munger 2016].

Designing Complex Point Processes. Aside global point set properties such as blue-noise, Poisson disk or low discrepancy, the problem of designing a point process matching some exemplars or satisfying additional constraints has been addressed in several ways. One can design sampler mixing global properties such as low discrepancy and blue-noise [Ahmed et al. 2016; Ahmed and Wonka 2021; Perrier et al. 2018], we can use a profile based approach to generate LDS samplers with adjustable or with scriptable properties (*e.g.* blue-noise properties, stratification on some projections...) [L'Ecuyer and Munger 2016; Paulin et al. 2022]. Mixing point process properties can also be achieved by interpolating their high order statistics such as their pair-correlation functions [Öztireli and Gross 2012]. Focusing on spectral properties, [Leimkühler et al. 2019] have proposed a neural network approach to target specific profiles defined as combinations of radial power spectra.

Point sets through deep learning. Perhaps the closest to our work is that of [Leimkühler et al. 2019]. They learn arbitrary dimensional point sets by matching power spectra. There is a number of important differences with respect to our work. First, they require a power spectrum as input while we require examples from a given sampler. This allows us to capture all characteristics of samplers and not just spectra. Second, our network is able to produce point sets of significantly different sizes without re-training. Third, we propose a way to benefit from efficient convolutions on grids. While this restricts us to low-dimensional settings (we demonstrate our approach in two dimensions), this allows us to use thousands of convolution layers at different scales and to benefit from recent advances in diffusion models. These differences allow us to finely capture the structure of point sets (see Sec. 4.1).

In the context of Monte Carlo integration, deep learning has been used to learn a control variate [Müller et al. 2020], though this does not directly address the location of point samples. Deep learning has also been used for importance sampling [Müller et al. 2019].

Probabilistic Denoising Diffusion. Our method is based on Probabilistic Denoising Diffusion, a concept introduced by [Sohl-Dickstein et al. 2015] in the context of unsupervised learning. The core idea of Denoising diffusion is to gradually remove any structure in the image by progressively adding noise and to train a neural network to invert the degradation process. This allows to capture the data distribution and sample from it. This idea has been extensively used for image synthesis [Ho et al. 2020] with impressive results, either by working directly in pixel space or in the latent space [Rombach et al. 2022]. In this paper, we propose to exploit the capacity of these networks to learn structure from a set of examples to learn point distributions.

3 DENOISING DIFFUSION MODEL

3.1 Architecture

The denoising process involves a sequence of denoising operations which operate at given timesteps. Each denoising is achieved by a forward pass in a single denoising network ε_θ, which takes as input both the noisy image $\tilde{x}_t$ and the embedded timestep t.

Our network architecture is very similar to the one of [Ho et al. 2020]. It corresponds to a U-Net [Ronneberger et al. 2015], where each level is composed of two convolutional residual blocks (ResNet) and the feature maps are downsampled by a factor 2 between each level. While the original architecture only included attention blocks between the two convolutional blocks of the 16×16 level, we add attention to all levels, which we found to work better in practice. Unless specified otherwise, we used 1000 diffusion time steps. The overall architecture design is detailed in supplementary material.

The network learns a time-dependent noise model $\varepsilon_\theta(\tilde{x}_t, t)$ given a noise ε_t added to the input data, $\tilde{x}_t = x_t + \varepsilon_t$ at each time step t. In our setting, x_0 is the offset between strata centers and the input point set as obtained in Sec 3.2. The network thus predicts noise, that can then be progressively removed from a white noise point set to denoise it according to the learned data distribution.

3.2 Convolutions on grids

While computing the required convolutions used in the diffusion model is possible on unstructured point sets [Groh et al. 2019; Hua et al. 2018; Simonovsky and Komodakis 2017], this comes at a prohibitive cost in our context, due to the large number of convolutions involved. Fortunately, our point sets are not arbitrary but may uniformly cover the unit square. In certain cases, they can be stratified, i.e., each stratum of size $\frac{1}{\sqrt{n}} \times \frac{1}{\sqrt{n}}$ contains a single sample. This is notably the case for the large class of $(0, m, s)$-nets samplers [Niederreiter 1992]. In that case, we use a pixel grid of $\sqrt{n} \times \sqrt{n}$ pixels, and store in each pixel the 2-d offset between the stratum center and its corresponding sample location. When this is not the case, we compute a linear assignment using optimal transport between the strata centers and the set of samples (Fig. 1) [Bonneel et al. 2011], and similarly store in each pixel the 2-d offset between the stratum center and its corresponding sample location. Doing so allows to work on 2-d grids and benefit from optimized convolutions. In our settings,

the grid acts as an approximate nearest neighbor acceleration data structure, such that, when a convolution is performed, neighboring samples approximately correspond to neighboring pixels, and are thus appropriately weighted. We evaluate this property with non-uniform sampling in Sec. 4.2. This remapping further allows to remain invariant under re-ordering of samples.

Fig. 1. When input point sets are not stratified, we compute a linear assignment problem between strata centers (red) and sample points (blue) using optimal transport. Each stratum stores its assigned point offset (green arrows). The grid thus serves as an approximate nearest neighbor acceleration data structure and benefits from efficient convolutions.

3.3 Training

The benefit of a convolutional approach is that the same convolution weights can be used for different grid sizes. It thus becomes possible to train *the same* network with point sets of different sizes, and hope that it generalizes. We explore in Sec. 4.1 how it succeeds in generalizing. However, within a single batch, the sample count should remain the same, due to the way batches are processed. For a given batch of size B, we thus build a loss that sums contributions for different input grid sizes $\mathcal{S}$ stored in different batches:

$$\mathcal{L}(\epsilon_\theta, \epsilon_t) = \sum_{j \in \mathcal{S}} \frac{1}{B} \sum_{i=1}^{B} \|\epsilon_\theta(\tilde{x}_{t_i}, t_i) - \epsilon_{t_i}\|^2,$$

for randomly chosen $\{t_i\}$. We typically use $\mathcal{S} = \{8{\times}8, 16{\times}16, 32{\times}32\}$, hence learning from sample sizes $\{64, 256, 1024\}$. We obtain one trained network, of the same architecture but different training weights, per type of sampler, each able to produce point sets of different sample sizes.

We train networks to reproduce Sobol' samples with Owen's scrambling [Owen 1998; Sobol' 1967] as a representative LDS matrix-based sampler, LatNetBuilder samples as a representative LDS lattice-based sampler, a Poisson disk sampler (classical dart throwing approach), SOT [Paulin et al. 2020] as a representative blue noise sampler using optimal transport, GBN [Ahmed et al. 2022] as a representative kernel-based blue noise sampler, LDBN [Ahmed et al. 2016] as a sampler that combines low discrepancy properties and blue noise spectrum, and Rank-1 [Keller 2004] as a representative of lattice based sampler. We train all our models using 64k point sets, except for the SOT sampler trained with only 32 (*not 32k*) point

sets to assess robustness to small training datasets. We train for a constant time of 3 hours, and synthesis time is typically 35 minutes for 1000 point sets of 1024 samples each using 1000 diffusion steps.

4 VALIDATION AND APPLICATIONS

4.1 Properties of generated samples

We study power spectra, optimal transport energy, discrepancy, integration errors and minimum distance statistics of generated point sets, and verify that they match properties they were trained for. We also verify how our network generalizes as we increase the number of samples outside the range it was trained for. For these comparisons, we compare to the approach of [Leimkühler et al. 2019]. For stationary and isotropic point processes or samplers targeting such properties, we have used their publicly available implementation with a 1d radial mean power spectrum loss (same learning parameters as the one provided by the authors for similar experiments). For non-stationary or anisotropic samplers (*e.g.* Sobol'+Owen and Rank1), we had to design our own learning experiment following their examples in 1d, with losses defined as l_1 norm between 2d power spectra (cropped to the central frequency part). We observe that such training turns out to be very difficult in 2d and leads to non-competitive results. In Fig 2, we only show results for Sobol'+Owen in 2d and leave the discussion for Rank-1 in supplementary materials.

While we trained our network on small set of sample sizes ($\{64, 256, 1024\}$), we assess the performance of these metrics for other sample sizes ($\{576, 4096\}$). For most of these properties, we illustrate them with violin plots (Fig. 3, 4, 5, 6), that show the distribution of values in the form of vertical histograms (e.g., similar to a population pyramid). We compute them using 128 point sets.

Power spectra. In Fig. 2, we first show performances of [Leimkühler et al. 2019] and our approach to recover spectral properties of the training sets (either through 1d radial mean power spectra for stationay and isotropic point sets, or 2d spectra for other ones). As discussed above, capturing anisotropic spectra with [Leimkühler et al. 2019] is very challenging using a 2d spectra loss function. Our approach fully captures such characteristics.

Optimal transport energy. Optimal transport (OT) provides a way to characterize the uniformity of a point set by computing the (squared) semi-discrete optimal transport distance between the point set and a uniform distribution [Mérigot 2011]. Fig. 3 illustrates how we match the OT energy.

Discrepancy and integration error. Fig. 4 and 5 show how our network matches integration errors and discrepancy of point sets. For discrepancy, we used the L2 discrepancy [Heinrich 1996; Niederreiter 1992]. For integration error, we compute the average MSE on the integration of wide anisotropic Gaussians (anisotropic ratio between 1:1 and 1:9, and Gaussian sizes ranging from 0.1 to 0.333 for its largest axis) or Heaviside distributions randomly linearly dividing the unit square. We randomly chose 64k integrands among 1 million, whose integral has been estimated with maximum precision as reference. These statistics also often match for sample sizes not seen during training ($\{576, 4096\}$).

Fig. 2. For various input samplers and their spectral content (Fourier power spectrum and radial mean power spectrum), we compare our approach (last three rows) with that of [Leimkühler et al. 2019] (1d radial mean power spectrum loss for Poisson disk, GBN, SOT and LDBN; for Sobol'+Owen and Rank-1, we used the 2d power spetrum cropped to the central part, framed in orange, for the learning to converge).

Fig. 3. We verify that the point sets predicted by our network match the semi-discrete optimal transport distance to a uniform distribution of the original point sets. These plots show these statistics distributions for 128 point sets from the training set and produced by our network, for sample counts of 64, 256, 576, 1024 and 4096 (top to bottom). The network has only been trained with point sets of 64, 256 and 1024 samples, but successfully predicts point sets of 576 and 4096 samples (results highlighted in an orange frame). Labels prefixed by **DC** refer to Deep Point Correlation results [Leimkühler et al. 2019] (on 1d radial power spectral, unless 2d is specified), while **NN** refers to results produced by our Neural Network.

Minimum distance. For distributions such as Poisson Disk, the minimum distance between any pair of samples can be important. We assess this statistics in Fig. 6. This property is highly sensitive as it only depends on the location of 2 points within the entire point set. For this property, the approach of [Leimkühler et al. 2019] performs remarkably well, due to the repulsion of points introduced during learning. In our approach, we tend to produce points with lower minimum distance value.

4.2 Non-uniform distributions

The goal of our optimal transport matching to a uniform grid is to infer neighborhood information on the point sets from neighborhood information on the grid, that is, neighboring points on the grid are expected to correspond to neighboring samples. In Fig. 7, using a non-uniform linear ramp sliced optimal transport sampling, we show that, even for non-uniform sampling, our network successfully learns from examples and preserve spectral noise characteristics of the sampler. As a stress test, we also learn to sample a blobby function shown in Fig. 8. In this example, we learn from importance

Fig. 4. Our network matches integration errors on Gaussian integrands (top 4 plots) and Heaviside integrands (bottom 4 plots), even beyond the sample sizes it was trained for ({64, 256, 1024}). Sample counts are 64, 256, 576, 1024 and 4096 (top to bottom for each integrand).

sampled GBN point sets obtained by rejection sampling. Our network reproduces the sampling density well, and mostly preserves

Fig. 5. Our network matches the L2 discrepancy of the original point sets. Sample counts are 64, 256, 576, 1024 and 4096 (top to bottom).

Fig. 6. We evaluate the minimum pairwise distance between samples. This property is highly sensitive as it only depends on the location of 2 samples. Our network tends to produce smaller values, while the sample repulsion of [Leimkühler et al. 2019] better preserve minimum distances. Sample counts are 64, 256, 576, 1024 and 4096 (top to bottom).

important characteristics of the GBN sampler despite inaccuracies in neighborhood information due to the grid embedding. Non-uniform sampling is not possible with the approach of [Leimkühler et al. 2019].

4.3 Applications

Aside from the fast generation of point sets, we also benefit from the differentiability of our network to further optimize point sets within their class.

We illustrate how the differentiability of our network can be used to add properties to generated point sets. Here, we wish to add low discrepancy properties to a sliced optimal transport sampler [Paulin et al. 2020], to benefit from both low discrepancy and low optimal transport energy. We train the network on SOT and then fix the trained weights of the network. Then we optimize the initial white noise samples with an objective function aimed at minimizing the L2 discrepancy measure. As backpropagation requires significant memory overhead, we reduce the number of diffusion steps to 100 (instead of 1000) in the diffusion model. In Fig. 9, we illustrate the result of our optimization in terms of discrepancy and optimal transport energy, and illustrate with an example generated point set.

5 DISCUSSIONS & PERSPECTIVES

We showed that diffusion models provide a powerful tool for learning how to generate point sets directly from examples across a wide range of samplers and they generalize well with sample size. Generalization hints at the fact that the network is correctly learning the general principles that make each point set so particular. An

interesting future work would involve conditioning the network with respect to the particular sampler, sampler type or more general desired properties. This would allow for a single trained network to produce point sets of types. Preliminary experiments showed subpar results, but more complex architectures could alleviate this issue. The capacity of our network to produce possibly non-uniform example-based point sets may open the door to syntheses where sampling data are only available through a small number of measurements (e.g., distribution of trees, cells, etc.) and optimizing only for summarized statistics (power spectrum or PCF) is not desired. This is a promising direction as we have successfully trained our network with 32 examples of the SOT sampler.

While in principle our method would work in arbitrary dimension, the efficiency gained through our convolutions on grids would be lost as storing higher dimensional grids becomes impractical, both in terms of storage (that exponentially grows with dimension) and supported sample size (in the form k^d for some k, similarly to stratified samplers). To date, higher dimensional data would be better supported by the approach of [Leimkühler et al. 2019] that does not rely on grids. To remove this grid-dependency in the Monte Carlo sampling, one could adapt recent diffusion models for 3D point cloud shape synthesis [Luo and Hu 2021; Zeng et al. 2022]. While our network is reasonably efficient, other recent architectures have been proposed to accelerate diffusion models and could be explored as well [Song et al. 2020].

Fig. 7. We sample from a learned sliced OT linear ramp. **Top row, left.** One example point set used for training (among 66,035). **Top row, right.** One synthesized point set. **Bottom row.** Unwarping example and synthesized point sets to recover a uniform distribution shows that their spectra match. The uniformity of the unwarped samples can also be measured: the semi-discrete optimal transport energy averaged for 128 realizations of 256 samples is $7.24.10^{-4}$ for the neural network output, compared with $7.16.10^{-4}$ for the original sliced OT uniform samples.

Fig. 8. As a stress test, we sample from the density $0.2e^{-20(x^2+y^2)} + 0.2\sin(\pi x)^2 \sin(\pi y)^2$ by importance sampling using GBN as a training set (first row). Our sampler reproduces the density well and mostly preserves important characteristics of the sampler (second row).

However, in the settings we focus on, in most cases our samples preserve characteristics of major samplers well, including their power spectrum, Monte Carlo integration quality, distance statistics, optimal transport energy and discrepancy. Our diffusion-based sampler allows to generate point sets much faster than some optimization-based samplers by learning from their output. Aside for the fast generation of diverse point sets, we have shown use for our network's differentiability by adding a low discrepancy property to an optimal transport-based sampler. Rendering applications could benefit from our samplers, e.g., through differentiable rendering pipelines [Jakob et al. 2022a] or for generating point sets nicely distributing Monte Carlo error in a blue noise fashion in screen space [Salaün et al. 2022].

Fig. 9. We used a trained SOT sampling network to optimize the discrepancy of the generated point sets among the class of SOT point sets. For 128 SOT (blue) and some Sobol'+Owen (red) point sets as representative of blue noise and LDS samplers, we show their distribution of OT and discrepancy statistics. In orange, we illustrate the SOT and discrepancy value for 10 optimized point sets as well as a representative trajectory during the optimization process. We also show a representative point set before (right) and after (left) optimization.

REFERENCES

Abdalla G. M. Ahmed, Hélène Perrier, David Coeurjolly, Victor Ostromoukhov, Jianwei Guo, Dong-Ming Yan, Hui Huang, and Oliver Deussen. 2016. Low-Discrepancy Blue Noise Sampling. *ACM Trans. on Graphics (SIGGRAPH Asia)* 35, 6 (2016), 247:1–247:13. https://doi.org/f9cpt2

Abdalla G. M. Ahmed, Jing Ren, and Peter Wonka. 2022. Gaussian Blue Noise. *ACM Trans. Graph.* 41, 6, Article 260 (nov 2022), 15 pages. https://doi.org/jtp8

Abdalla G. M. Ahmed and Peter Wonka. 2021. Optimizing Dyadic Nets. *ACM Trans. on Graphics (SIGGRAPH)* 40, 4 (2021), 141:1–141:17. https://doi.org/hn22

Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*. 1–12. https://doi.org/gkcqgt

Robert Bridson. 2007. Fast Poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches* 10, 1 (2007), 1. https://doi.org/gf8tsr

Fernando De Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue noise through optimal transport. *ACM Trans. Graph.* 31, 6 (2012), 171:1–171:10. https://doi.org/gbb6n9

Oliver Deussen, Stefan Hiller, Cornelius Overveld, and Thomas Strothotte. 2000. Floating Points: A Method for Computing Stipple Drawings. *Computer Graphics Forum (EG'00)* 19, 3 (2000), 40–51. https://doi.org/fg9w98

Daniel Dunbar and Greg Humphreys. 2006. A spatial data structure for fast Poisson-disk sample generation. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 503–508.

Raanan Fattal. 2011. Blue-Noise Point Sampling Using Kernel Density Model. *ACM Trans. Graph.* 30 (2011), 48:1–48:12. https://doi.org/cv7pbv

Manuel N Gamito and Steve C Maddock. 2009. Accurate multidimensional Poisson-disk sampling. *ACM Trans. Graph.* 29, 1 (2009), 8:1–8:19. https://doi.org/dr8646

Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. 2019. Flex-Convolution: Million-scale point-cloud learning beyond grid-worlds. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part I 14*. Springer, 105–122.

Stefan Heinrich. 1996. Efficient algorithms for computing the L2-discrepancy. *Math. Comp.* 65, 216 (1996), 1621–1633.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. 2018. Pointwise convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 984–993.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022b. *Mitsuba 3 renderer*. https://mitsuba-renderer.org

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022a. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *ACM Trans. on Graphics (SIGGRAPH)* 41, 4 (2022), 124:1–124:19. https://doi.org/gqjn7p

Alexander Keller. 2004. Stratification by rank-1 lattices. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, Harald Niederreiter (Ed.). Springer, 299–313. https://doi.org/fks8z8

Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. 2006. Recursive Wang Tiles for Real-Time Blue Noise. *ACM Trans. Graph.* 25, 3 (2006), 509–518. https://doi.org/dgvw52

Pierre L'Ecuyer and David Munger. 2016. LatticeBuilder: A General Software Tool for Constructing Rank-1 Lattice Rules. *ACM Transactions on Mathematical Software* 42 (2016), 1–30. https://doi.org/10.1145/2754929

Thomas Leimkühler, Gurprit Singh, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. 2019. Deep point correlation design. *ACM Trans. on Graphics (SIGGRAPH Asia)* 38, 6 (2019), 1–17. https://doi.org/ggfg2x

Christiane Lemieux. 2009. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer. https://doi.org/b8r4z5

Shitong Luo and Wei Hu. 2021. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2837–2845.

Quentin Mérigot. 2011. A multiscale approach to optimal transport. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1583–1592. https://doi.org/cjh4q8

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural importance sampling. *ACM Trans. on Graphics* 38, 5 (2019), 1–19. https://doi.org/jtrf

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. 2020. Neural control variates. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–19. https://doi.org/jtrj

Georges Nader and Gael Guennebaud. 2018. Instant transport maps on 2D grids. *ACM Trans. Graph.* 37, 6 (2018), 249:1–249:13. https://doi.org/jtrg

Harald Niederreiter. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA. https://doi.org/fd5fjw

Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. 2004. Fast Hierarchical Importance Sampling with Blue Noise Properties. *ACM Trans. on Graphics (SIGGRAPH)* 23, 3 (2004), 488–495.

Art B. Owen. 1998. Scrambling Sobol' and Niederreiter–Xing Points. *Journal of Complexity* 14, 4 (1998), 466–489.

Cengiz Öztireli and Markus Gross. 2012. Analysis and synthesis of point distributions based on pair correlation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10. https://doi.org/gbb6qr

Loïs Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Alexander Keller, and Victor Ostromoukhov. 2022. MatBuilder: Mastering Sampling Uniformity over Projections. *ACM Trans. on Graphics (SIGGRAPH)* 41, 4 (2022), 84:1–84:13. https://github.com/loispaulin/matbuilder

Loïs Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Antoine Webanck, Mathieu Desbrun, and Victor Ostromoukhov. 2020. Sliced optimal transport sampling. *ACM Trans. on Graphics (SIGGRAPH)* 39, 4 (2020), 99:1–99:17. https://doi.org/gg8xfj

Hélène Perrier, David Coeurjolly, Feng Xie, Matt Pharr, Pat Hanrahan, and Victor Ostromoukhov. 2018. Sequences with Low-Discrepancy Blue-Noise 2-D Projections. 37, 2 (2018), 339–353. https://doi.org/gd2j2d

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3 ed.). Morgan-Kaufmann.

Adrien Pilleboue, Gurprit Singh, David Coeurjolly, Michael Kazhdan, and Victor Ostromoukhov. 2015. Variance Analysis for Monte Carlo Integration. *ACM Trans. Graph.* 34, 4 (2015), 124:1–124:14. https://doi.org/f7m28c

Hongxing Qin, Yi Chen, Jinlong He, and Baoquan Chen. 2017. Wasserstein Blue Noise Sampling. *ACM Trans. Graph.* 36, 4, Article 137a (Oct. 2017). https://doi.org/gcj3d3

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 10684–10695.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241. https://doi.org/gcgk7j

Riccardo Roveri, A Cengiz Öztireli, and Markus Gross. 2017. General point sampling with adaptive density and correlations. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 107–117. https://doi.org/gbm2jp

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. https://doi.org/10.1145/3550454.3555484

Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3693–3702.

Gurprit Singh, Cengiz Öztireli, Abdalla G. M. Ahmed, David Coeurjolly, Kartic Subr, Oliver Deussen, Victor Ostromoukhov, Ravi Ramamoorthi, and Wojciech Jarosz. 2019. Analysis of sample correlations for Monte Carlo rendering. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 473–491.

Ilya M. Sobol'. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7, 4 (1967), 784–802. https://doi.org/crdj6j

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*. PMLR, 2256–2265.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).

Kartic Subr and Jan Kautz. 2013. Fourier analysis of stochastic sampling strategies for assessing bias and variance in integration. *ACM Trans. Graph.* 32, 4, Article 128 (2013), 12 pages. https://doi.org/gbdg7c

Robert Ulichney. 1987. *Digital Halftoning*. MIT Press, Cambridge, MA, USA.

Florent Wachtel, Adrien Pilleboue, David Coeurjolly, Katherine Breeden, Gurprit Singh, Gaël Cathelin, Fernando De Goes, Mathieu Desbrun, and Victor Ostromoukhov. 2014. Fast tile-based adaptive sampling with user-specified Fourier spectra. *ACM Trans. on Graphics (SIGGRAPH)* 33, 4 (2014), 1–11. https://doi.org/f6cz6k

Li-Yi Wei. 2008. Parallel Poisson disk sampling. In *ACM Trans. Graph.*, Vol. 27. ACM, 20. https://doi.org/cs3jjv

John I. Yellott. 1982. Spectral analysis of spatial sampling by photoreceptors: Topological disorder prevents aliasing. *Vision Research* 22, 9 (1982), 1205 – 1210. https://doi.org/fsgtr4

Cem Yuksel. 2015. Sample elimination for generating Poisson disk sample sets. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 25–32. https://doi.org/f7k7c7

Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. 2022. LION: Latent Point Diffusion Models for 3D Shape Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Supplementary document

1 DIFFUSION MODEL

Diffusion models date back to the work of Sohl-Dickstein et al. [2015] but were popularized by Ho et al. [2020] for image synthesis. This section recalls the details for completeness.

Probabilistic Denoising Diffusion models involve a *forward* process, where noise is gradually added to the signal (here an image) and a *reverse* process where noise is removed through a learnable network. The forward diffusion process is a Markov Chain, where each transition adds Gaussian Noise to the image, following:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I),\tag{1}$$

where $(\beta_t)_{t=0}^T$ are the noise variances for each time t. The variance schedule is chosen such that nothing distinguishes x_T from a white noise. In our model, we set the variances β_t to be constant $\beta_t = \beta$

One has:

$$q_{x_{1:T}|x_0} = \prod_{t=1\cdots T} q(x_t|x_{t-1}).\tag{2}$$

The reverse (denoising) process is also a Markov Chain, with transitions:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)),\tag{3}$$

μ_θ and Σ_θ are learned by examples. To simplify, following the work of Ho et al. [2020], we consider that $\Sigma_\theta = \sigma_t I$, with $\sigma_t = \beta_t = \beta$. The forward process allows to sample x_t with arbitrary t from x_0, following:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I),\tag{4}$$

with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$.

During training, and image x_0 is drawn from the set of examples, along with a random time $t \in 1\cdots T$, a random noise image ε is drawn following $\mathcal{N}(0, I)$ and the algorithm tries to minimize:

$$\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\alpha_t}\varepsilon, t)\|^2,\tag{5}$$

by gradient descent.

During sampling a random noise image $z \sim \mathcal{N}(0, I)$ is drawn and iteratively denoised by applying:

$$x_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t)) + \sigma_t z,\tag{6}$$

where z is a random noise and in our case, we take $\sigma_t = \beta_t$ The key ingredient of diffusion models is the approximator ε_θ, which is modeled by a neural network.

2 NETWORK

Our network is a slightly modified version of the denoising network of Ho et al. [2020] and is summarized on Figure 10.

3 LEARNING RANK-1 REALIZATIONS WITH [Leimkühler et al. 2019]

Leimkühler et al. [2019] proposed a neural network based point process design using losses defined from spectral or pair-correlation information. In most examples provided by the authors, 1d losses (or combination of 1d losses) are considered using 1d radial mean power spectra or 1d pair correlation functions (allowing complex designs such as a high-dimensional point process with some specific spectral properties for given 1d or 2d projections). When targeting isotropic

Fig. 10. Diffusion network architecture

Fig. 11. Learning Rank-1 realizations using [Leimkühler et al. 2019] using a 2d power spectra loss, and a 1d power spectra loss. We recall the original properties and our results for completeness.

samplers, the authors provided their experimental settings in https://github.com/sinbag/deepsampling. We use the same parameters for Poisson disk, GBN, SOT, LDBN, targeting their respective radial power spectra. For Sobol'+Owen, we keep the same settings but updated the loss function to target a 2d power spectrum. Cropping the spectra to the central part of the domain allowed us to obtain a convergence of the learning step (in our experiments, increasing the cropping domain does not help the convergence). For Rank-1, no satisfactory results have been obtained, for the sake of completeness, we illustrate in Fig.11 the point set and spectra we have obtained. Note that Rank-1 is a very specific anisotropic sampler which is far from the context of [Leimkühler et al. 2019]. Although, additional investigation would be interesting to continue.