Example-Based Sampling with Diffusion Models

Bastien Doignies
Univ Lyon, UCBL, CNRS, INSA Lyon, France

Nicolas Bonneel
Univ Lyon, CNRS, UCBL, INSA Lyon, France

David Coeurjolly
Univ Lyon, CNRS, UCBL, INSA Lyon, France

Julie Digne
Univ Lyon, CNRS, UCBL, INSA Lyon, France

Lois Paulin
Univ Lyon, UCBL, CNRS, INSALyon, France

Jean-Claude Iehl
Univ Lyon, UCBL, CNRS, INSA Lyon, France

Victor Ostromoukhov
Univ Lyon, UCBL, CNRS, INSA Lyon, France

ABSTRACT
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.

CCS CONCEPTS
• Computing methodologies → Neural networks; Computer graphics.

KEYWORDS
diffusion models, sampling, point patterns, example-based synthesis

1 INTRODUCTION
A wide range of samplers have been designed in the past, for quasi-Monte Carlo integration, rendering, image stippling, positioning 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]. 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 lacking generically, with methods involving smooth non-convex optimization, integer linear programming, number theory, bruteforce approaches with clever data structures, etc. Finally, it may happen that sample distributions are known only through a set of examples, without a known dedicated sampler, hence the need for an example-based method able to generate point sets from a set of examples, while capturing the fine-grained properties of the point distribution.

Recently, diffusion models have become extremely popular in the context of image generation [Sohl-Dickstein et al. 2015; Ho et al. 2020; Rombach et al. 2022]. By learning how to denoise an image that initially only contains random values, these models have been able to produce impressives 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. We also demonstrate that a mild change in our architecture...
adding distribution class conditioning – allows to train a single network for a whole set of samplers. Hence a single network can provide different types of point sets, and produce points of a desired distribution.

While our network is currently limited to generating 2-d or 3-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 applications. Code is provided in supplementary material.

2 RELATED WORKS
Existing samplers have a wide range of properties. We enumerate important 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 [Subr and Kautz 2013; Pilleboe et al. 2015], digital halftoning [Ulichney 1987] or stippling [Deussen et al. 2000] and well describe arrangements of natural phenomena 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 [Fattal 2011; Ahmed et al. 2022], pair-correlation function [Öztireli and Gross 2012] or optimal transport [De Goes et al. 2012; Qin et al. 2017; Paulin et al. 2020], though fast approximations exist [Nader and Guennebaud 2018]. Tile-based approaches pre-compute tiles for fast synthesis, but are memory demanding [Ostromoukhov et al. 2004; Ostromoukhov 2007; Ahmed et al. 2017; Kopf et al. 2006; Wachtel et al. 2014].

Poisson Disk. As an alternative way to tackle the blue noise point pattern construction and as coined by Ulichney [1987], Poisson disk samples have the property that no point fall within a distance smaller than a threshold from another point [Yuksel 2015; Gamito and Maddock 2009; Wei 2008; Bridson 2007; Dunbar and Humphreys 2006; Dippé and Wold 1985; Cook 1986]. Their spectra resemble those of blue noise distributions, except that they do not decrease towards zero as the frequency decreases [Pilleboe 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 [Niederreiter 1992; Lemieux 2009]. 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], or one can use a profile-based approach to generate LDS samplers with adjustable or with scriptable properties, such as blue-noise properties or stratification on some projections [Paulin et al. 2022; L’Ecuyer and Munger 2016]. 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, Zhou et al. [2012] have proposed an energy formulation and a gradient descent approach to optimize samples targeting a given Fourier spectrum. Leimkühler et al. [2019] have followed a similar approach using a neural network to target specific profiles defined as combinations of radial power spectra. In a texture-synthesis like manner, Huang et al. [2022] proposed to extend point set patterns by using a set of Gabor filter results as input to a CNN and matching resulting feature maps. Tu et al. [2019] follow a similar approach of point pattern upsampling using irregular convolutions by evaluating the convolution operator on a grid. Targeting repetitive patterns, Roveri et al. [2015] use a multiscale local-global optimization that they applied repetitive point set synthesis.

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 point statistics such as power spectra or distance statistics. There is a number of important differences with respect to our work. First, they require statistics (e.g., 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 selected statistics). 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 and three 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. 4.2). Point sets generated by our method can be used for Monte Carlo integration purposes. In this context, 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 an input 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]. Denoising diffusion models have been
extended to 3D shape point sets, by conditioning on a shape latent using a PointNet encoder [Luo and Hu 2021], or by relying on a Point-Voxel CNN [Zhou et al. 2021]. A hierarchical approach combining both a global latent representation and a point latent representation using denoising diffusion models for both representation and Point-Voxel CNN for encoding and decoding the shape has also been proposed [Zeng et al. 2022]. While these architectures are able to sample coarse 3d shapes, and are related to our context since they address irregular data, they fail at capturing the fine-grained point distribution properties we are interested in. In this paper, we propose to exploit the capacity of these denoising diffusion models 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 \( \varepsilon_\theta \), which takes as input both the noisy image \( \hat{x}_t \) 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 included attention blocks between the convolutional blocks at some levels, we found that removing this attention led to comparable or better results and reduced the training time. We trained the model using 1000 diffusion time steps but used only 50 time steps at inference time, which allowed for faster synthesis for comparable results (see the supplementary material for an ablation study on our network architecture details).

The network learns a time-dependent noise model \( \varepsilon_\theta(\hat{x}_t, t) \) given a noise \( \varepsilon_t \) added to the input data, \( \hat{x}_t = x_t + \varepsilon_t \) 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 gaussian 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; Simonovsky and Komodakis 2017; Hua et al. 2018], 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 \( \sqrt{M} \times \sqrt{N} \) 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. This assignment is done only once, upon loading a point set at training time, and is not needed at inference time. 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. 4.3. This remapping further allows to remain invariant under re-ordering of samples. This encoding can be extended to point sets of any dimension following the same principle.

Figure 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. 4.2 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 \( S \) stored in different batches:

\[
\mathcal{L}(\varepsilon_\theta, \varepsilon_t) = \frac{1}{B} \sum_{i=1}^{B} \sum_{j \in S} \| \varepsilon_\theta(\hat{x}_{t_i}, t_i) - \varepsilon_t \|^2,
\]

for randomly chosen \( \{t_i\} \). We typically use \( S = \{8 \times 8, 16 \times 16, 32 \times 32\} \), 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.

3.4 Conditioning

Our vanilla architecture allows to learn the characteristics of a set of observations of sample distributions. However, it requires to train a different network for each distribution class. To alleviate this requirement, we propose a simple extension of our method to train
a single network for several sampler classes. In practice, all pixels in the image are concatenated with a vector giving the encoding of the desired sampler class. See the supplementary material for more details and results.

4 VALIDATION AND APPLICATIONS

4.1 Implementation

We train networks to reproduce Sobol’ samples with Owen’s scrambling [Sobol’ 1967; Owen 1998] as a representative LDS matrix-based sampler, LatNetBuilder [L’Ecuyer and Munger 2016] 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 of each sample count in S, 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 0.1s for a point set of 4096 points using 50 diffusion steps (2 seconds for 1000 diffusion steps) on an Nvidia V100. As a comparison, optimization-based samplers usually need several seconds to sample a point set of the same size (multithreaded CPU SOT and GPU-based GBN require 3.0s on AMD Ryzen 7 1700X and 4.5s on Nvidia V100 respectively). The source code is available at https://github.com/BDoignies/ExampleBasedSamplingWithDiffusion.

4.2 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] (DC for short in the graphs). For stationary and isotropic point processes (Poisson disk and GBN), 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). Non-stationary or anisotropic point patterns (SOT, LDBN, Sobol’+Owen and Rank1), fall outside of the scope of the approach of Leimkühler et al. [2019] as it considers anisotropic samplers defined by isotropic properties on axis projections (for instance a 3d point set with blue-noise characteristics on the XY subspace and a step profile on the XZ subspace). We also include in our experiments a comparison to the Point Pattern synthesis approach [Huang et al. 2022] (PPS for short). This method being dedicated to point pattern upsampling, we use it to upscale point sets from 1024 samples to 4096. Additional upscaling results compared with Tu et al. [2019] are available in supplementary materials. On some violin plot figures we omit Sobol’+Owen or Rank1 as either their implementation fails to produce a point set, or the error values are above the others which would hinder the readability of the figure.

Finally, we have also tested point set dedicated denoising diffusion models [Luo and Hu 2021; Zhou et al. 2021] in Figure 2, showing that both methods fail at capturing the point distribution properties. The resulting point sets are very far from both the reference and our results.

Figure 2: Comparisons with alternative deep models dedicated to point cloud processing ([Luo and Hu 2021] and [Zhou et al. 2021]). For this experiment, we have considered an LDBN point set target with 256 samples.

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). Its ability to generalize to unseen sample counts shows that our network did not merely memorize point sets from the training set. 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 (similar to a population pyramid). We compute them using 128 point sets.

Power spectra. In Fig. 13, we first show performances of the method of Leimkühler et al. [2019] and our approach to recover spectral properties of the training sets (either through 1d radial mean power spectra for stationary and isotropic point sets, or 2d spectra for other ones). As discussed above, capturing anisotropic spectra with the method of 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 generalized L2 discrepancy [Niederreiter 1992; Heinrich 1996]. For integration error, we compute the average...
Figure 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 [Leimkühler et al. 2019] (on 1d radial power spectral, unless 2d is specified), while NN refers to results produced by our Neural Network. We upscaled 1024 sample point sets to 4096 using the approach of Huang et al. [Huang et al. 2022] (PPS). For Owen, this resulted in values above our graph ranges. For R1, this produced NaN values.

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)).

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

Figure 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). For PPS, we upscaled 1024 sample point sets to 4096: for Owen, this resulted in values above our graph ranges; for R1 we obtained NaN values.
The goal of our optimal transport matching to a uniform grid is learning. In our approach, we tend to produce points with lower minimum distance value.

Conditioning. Finally, Fig. 7 evaluates the performances of the per class conditioning as discussed in Sect. 3.4. When compared to our vanilla model learned on each class, our conditioned model learned for all classes performs slightly worse than the vanilla models trained separately for each class, but it still provides a reasonable approximation. In the supplementary material, we provide further comparisons for all metrics and sample counts.

Diffusion model in 3d. Using optimal transport in 3d and convolutions on 3d grids, our model can be extended to process 3d point sets. In Fig. 8, a preliminary performance evaluation is given for the heaviside integration test on 4096 samples. Additional results are also given in supplementary material.

4.3 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. 9, 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. 12. In this example, we learn from importance sampled GBN point sets obtained by rejection sampling. Our network reproduces the sampling density well, and mostly preserves important characteristics of the GBN sampler.
despite inaccuracies in neighborhood information due to the grid embedding. As a failure case, for highly non-uniform point sets with many voids and clusters, e.g., pink noise of Fig. 10, the optimal transport matching may lead to suboptimal results. Clusters and voids in the pink noise distribution are located at different random locations in each training point set, which can render distribution learning more difficult. Our approach thus fails when the local density of samples varies among examples of the training set.

4.4 Application to constrained point set optimization

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. This is made feasible since our network is differentiable and produces point sets of a single class it is trained from. First, we train the network on SOT point sets and then freeze the weights of the network. Next, we optimize the input gaussian noise offset grid such that the synthesized point set minimizes the L2 discrepancy while the network ensures that it remains SOT-like. As backpropagation requires significant memory overhead, we reduce the number of diffusion steps to 100 (instead of 1000) in the diffusion model. In Fig. 11, 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. 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 each class (see supplementary material). Another promising direction lies in other types of network conditioning, such as property conditioning. Such a model would allow the user to specify numerical values for all or a subset of well-chosen properties (discrepancy, minimum distance, spectra...) and the network would be trained to fit these properties. However, this would require heavy architecture changes and we leave it as a possible future work.

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. Still, we believe our use of optimal transport matching for adapting a widespread convolutional network to the unstructured setting could benefit other low-dimensional applications.

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. However, sampling speed remains an issue for time-critical applications, notably compared to fast samplers such as Sobol’ or LDBN. 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 [Jakov 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].

ACKNOWLEDGMENTS

This work was partially funded by ANR-20-CE45-0025 (MoCaMed) and gifts from Adobe Inc. We gratefully acknowledge support from the CNRS/IN2P3 Computing Center (Lyon - France) for providing酝酿 2022-AD011012547R1 made by GENCI.

REFERENCES


Figure 12: As a stress test, we sample 1024 points from the density $0.2e^{-20(x^2+y^2)} + 0.2\sin(\pi x)^2 \sin(\pi y)^2$ [Balzer et al. 2009] by importance sampling using 256 GBN stippling pointset as a training set (first row). Our sampler reproduces the density well and mostly preserves important characteristics of the sampler (second row). We also illustrate an image stippling experiment (1024 samples, trained using GBN stippling, image from [Secord 2002]).
Figure 13: 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 and GBN).