



Point-Pattern Synthesis using Gabor and Random Filters

Xingchang Huang¹ , Pooran Memari², Hans-Peter Seidel¹, Gurprit Singh¹ 

¹Max-Planck-Institut für Informatik, Saarbrücken, Germany

²CNRS, LIX, Ecole Polytechnique, Paris, France

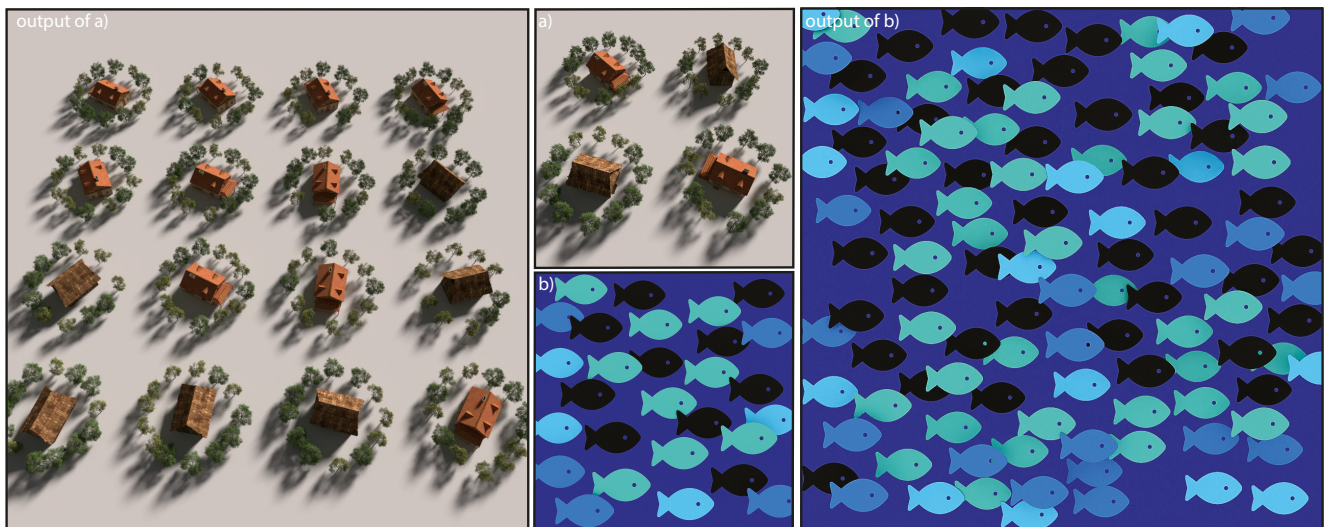


Figure 1: Our method takes simply a point set (with positions, classes, attributes) as input and applies continuous Gabor transform to extract features. We then use these Gabor features to synthesize a larger scale output. We show synthesis results of a 2-class point pattern in (a), and a 4-class point pattern with depth and scale as attributes in (b).

Abstract

Point pattern synthesis requires capturing both local and non-local correlations from a given exemplar. Recent works employ deep hierarchical representations from VGG-19 [SZ15] convolutional network to capture the features for both point-pattern and texture synthesis. In this work, we develop a simplified optimization pipeline that uses more traditional Gabor transform-based features. These features when convolved with simple random filters gives highly expressive feature maps. The resulting framework requires significantly less feature maps compared to VGG-19-based methods [TLH19; RGF*20], better captures both the local and non-local structures, does not require any specific data set training and can easily extend to handle multi-class and multi-attribute point patterns, e.g., disk and other element distributions. To validate our pipeline, we perform qualitative and quantitative analysis on a large variety of point patterns to demonstrate the effectiveness of our approach. Finally, to better understand the impact of random filters, we include a spectral analysis using filters with different frequency bandwidths.

CCS Concepts

• **Computing methodologies** → Point pattern synthesis; Point-based texture synthesis;

1. Introduction

Synthesizing point patterns from small exemplars has various applications from object placement, discrete texture synthesis to creative pattern generation. Synthesizing such patterns has been an active area of research in computer graphics [Lew89; DHL*98; ÖG12;

RÖM*15; TLH19] and involves two major steps. First step requires robustly characterizing the underlying correlations from an exemplar. The second step involves point-pattern expansion while preserving the underlying local and non-local correlations.

Characterizing different point patterns are traditionally performed

using Fourier and spatial (pair correlation function) tools. Among different point-patterns, blue noise point patterns [Uli88] have received special attention in computer graphics due to their minimum distance preservation [Ye183; YGW*15]. Blue-noise patterns are named *blue noise* based on their power spectrum characteristics. But not all patterns are easily characterised by traditional Fourier or spatial (pair correlation function) tools. Many recent works [ZHWW12; LSM*19; HSD13; RÖG17] have devoted significant effort in characterising such patterns by matching the low level statistics.

Synthesizing a large canvas output from an exemplar requires faithful extraction of both local and non-local features. To tackle this, various example-based approaches have been proposed [MWT11; MWLT13; RÖM*15; TLH19; EMGC19a; LGH13] that take as input a small-scale point set, extract both local and non-local features from the exemplars and use these features to synthesize point patterns on a larger scale.

Recently, Tu et al. [TLH19] use a pre-trained VGG-19 network [SZ15] to extract point pattern correlations. VGG-19 features are used as local and non-local features of an exemplar point set. These features are then used as a target during an optimization process to synthesize large canvas point patterns. Their method shows that hierarchical image-space features and losses are powerful for point pattern synthesis. However, the patterns synthesized by this method struggles to preserve accurate local structures and requires multiple steps of optimization or manual intervention for refinement. Moreover, since the VGG-19 network is limited by 2D images as input, it limits the scope of their approach to at most 3-class and limited per-point attribute characterization.

In this paper, we propose a simple Gabor transform-based pipeline that extract local and non-local features. Our method applies the Gabor transform to an exemplar point pattern and extract multi-resolution image-space features with a subsequent convolutional filtering step. The filtering step is performed using random filters which have shown success in reducing the feature space in neural networks [UBGB16; UVL18]. We then optimize a stochastic point set by using these features as target for by-example point pattern synthesis. We conduct qualitative and quantitative experiments on a large variety of patterns to demonstrate the applicability of our method for pattern creation and object placement. We further perform a spectral analysis for our filtering pipeline to interpret why random filters work. Our code is available at https://github.com/xchhuang/pps_gabor_random.

2. Related Work

By-example pattern synthesis are well-studied in the past decade, including image-based [GEB15b; GEB15a; UBGB16], discrete [MWT11; ÖG12; MWLT13; RÖM*15; RÖG17; TLH19] and continuous [TWY*20] texture synthesis. In general, these methods are designed for synthesizing a larger pattern given a small exemplar as input. Our approach is inspired from recent developments in point pattern [TLH19] and texture [UBGB16; GAD*20] synthesis. In this section, we cover the most relevant works and direct interested readers to a recent survey [GAM*21] for an extensive study.

Image-based texture synthesis. Image-based texture synthesis considers synthesizing image texture from an exemplar in the pixel

space. Heeger et al. [HB95] propose to use image pyramid whereas Portilla et al. [PS00] use wavelet-based descriptors for texture synthesis. We instead use Gabor filter bank to extract exemplar features. More recently, Guehl et al. [GAD*20] present Point Process Texture Basis Functions (PPTBF), together with stringed Gabor functions to synthesize a large variety of binary textures. Our work, on the other hand, is designed for by-example point-pattern synthesis.

By-example point pattern synthesis. Point pattern synthesis is usually defined as by-example synthesis as well: given a small point set as input, algorithms are designed to synthesize a larger one preserving both global and local structure. Ma et al. [MWT11] propose a sample- or multi-sample based representation for point pattern synthesis, or in general, element synthesis that allows example-based synthesis. Roveri et al. [RÖM*15] propose meshless representation where structures in the example and output are converted into a functional representation. However, both methods are neighborhood-based, meaning that they can easily fail to synthesize patterns preserving global structure well. A more recent work on point pattern synthesis is from Tu et al. [TLH19], which use the VGG-19 network for point pattern synthesis. Though they achieve improved results in terms of preserving global structure, they struggle with synthesizing local structures faithfully.

Other than point patterns, there are works on extending points to shapes. Ecornier et al. [EMGC19a] extend the pair correlation function (PCF) framework to disk distributions. Landes et al. [LGH13] propose a shape-aware model that brings out the elements as polylines which better preserves element distances. Similarly, Barla et al. [BBT*06] develop a specific method for stroke pattern synthesis and Hurtut et al. [HLT*09] consider the appearance of vector elements during synthesis.

Neural networks for pattern synthesis. Resurgence of neural networks through image classification [KSH12] has changed the dynamics of computer graphics research over the past decade. Recent architectures work on both images and unstructured point clouds domains. Point-based neural networks [QSMG17; QYSG17][WSL*19] extend the deep learning approach over irregular point clouds. However, these methods does not take into account point correlations and are not designed for point pattern synthesis. Leimkuhler et al. [LSM*19] propose a convolutional architecture that directly optimizes point samples for prescribed Fourier (power spectra) or spatial (pair correlation function) statistics allowing *blue*-, *green*-, or *pink-noise* sample distributions. However, this mainly captures global correlations. In texture synthesis, neural networks have been recently employed to synthesize large canvas textures from small exemplars using pre-trained network features [GEB15a; SC17] or completely random filters [UBGB16][HWH16]. Similarly, Ulyanov et al. [UVL18] propose deep image prior using a randomly initialized neural network for image restoration. Aberman et al. [ALS*18] use pre-trained CNN features to find image correspondence and Bojanowski et al. [BJS17] present a latent space optimization technique for generative image synthesis. Our method, on the other hand, optimize and synthesize points instead of images.

Compared to these, using neural networks for point pattern synthesis is a relatively unexplored area. Tu et al. [TLH19] propose the first neural network-based idea for point pattern synthesis. They

propose an optimization approach that uses a pre-trained VGG-19 network [SZ15] to guide random point samples to follow structured-patterns from an exemplar. Reddy et al. [RGF*20] develop a differentiable compositing pipeline that allows current deep learning based image methods to effectively handle patterns. Compared to their work, we propose to use Gabor features and a convolutional filtering step that better captures the statistics of point patterns. This enables higher-quality by-example point pattern synthesis and can be extended to multi-class and multi-attribute patterns with minor changes. No training or pre-trained features are required in our framework. It is also worth mentioning that Tu et al. [TLH19] propose an additional soft optimization scheme to heuristically remove outliers from optimized patterns. However, this may still result in missing points in the output patterns. In this work, we do not apply any heuristic approach during optimization.

3. Overview

Given an exemplar pattern with M points $\{\mathbf{p}_i\}_{i=1}^M$ within domain $[0, 1]$, our goal is to perform pattern expansion to a larger point set $\{\mathbf{P}_i\}_{i=1}^N$ in the same domain. First step in this process requires extracting local and non-local correlations from the exemplar. In the second step, these correlations are reproduced on a large-canvas with N points using an optimization. Normally, $N = S \times M$, where S is an up-scaling factor. For simplicity, we focus on a square domain with $S = 4$.

3.1. Preliminaries

In image-based texture synthesis, Gatys et al. [GEB15b] propose to use the pre-trained VGG-19 feature maps to extract local and non-local correlations. These extracted feature maps are then optimized for pattern expansion during synthesis using the following losses:

Gram loss. Gram matrix [GEB15b] is defined as the following:

$$G_{t_1, t_2}^l = \frac{1}{n^l W^l H^l} f_{t_1}^l \cdot f_{t_2}^l \quad (1)$$

where n^l, W^l, H^l are the number, width and height of 2D feature maps at layer l , respectively. G_{t_1, t_2}^l is the gram matrix computed between t_1^{th} and t_2^{th} feature maps at layer l and $f_{t_1}^l, f_{t_2}^l$ are t_1^{th} and t_2^{th} feature maps at layer l . Gram loss \mathcal{L}_{gram} can be defined as the square error between output and input Gram matrices:

$$\mathcal{L}_{gram}^l = \sum_{t_1, t_2} (\tilde{G}_{t_1, t_2}^l - G_{t_1, t_2}^l)^2 \quad (2)$$

where \tilde{G}, G represent gram matrices of output and input patterns, respectively.

Correlation loss. To better capture the correlations, Sendik et al. [SCI17] compute a correlation matrix from the feature maps:

$$C_{a, b}^{t, l} = \frac{f_{ij}^{t, l} f_{i-a, j-b}^{t, l}}{[(W^l - |a|)(H^l - |b|)]} \quad (3)$$

where $a \in [-W^l, W^l], b \in [-H^l, H^l]$. We use zero-padding when $i - a, j - b$ are outside the boundary. t denotes the index of feature maps, meaning that deep correlation matrix is computed at each channel

of feature maps independently. Based on this, deep correlation loss \mathcal{L}_{corr} is defined as:

$$\mathcal{L}_{corr}^l = \sum_{a, b, t} (\tilde{C}_{a, b}^{t, l} - C_{a, b}^{t, l})^2 \quad (4)$$

where \tilde{C}, C represent correlation matrices of output and input patterns, respectively.

Tu et al. [TLH19] use both the Gram and Correlation loss to perform point-pattern expansion. Reddy et al. [RGF*20], on the other hand, use only the Gram matrix to perform pattern expansion. Unlike Tu et al., the input for Reddy et al. is an image with corresponding elements. In our approach, we use Gabor transform to extract local and non-local correlations from an input point pattern.

4. Our Approach

Our pipeline is shown in Fig. 2. First step of our approach is to transform continuous points into a pixelized feature map representation (blue block). Each point can be assigned a class or/and some attributes. Our feature maps encode the associated attributes as well. We apply the following continuous Gabor function to get feature maps from points:

$$h(\mathbf{g}_j, \mathbf{p}, \mathbf{v}, \sigma, \gamma) = \sum_i \mathbf{w}_i \cdot e^{-\frac{\|\mathbf{p}_i - \mathbf{g}_j\|_2^2}{2\sigma^2}} \cos(\mathbf{v} \cdot (\mathbf{p}_i - \mathbf{g}_j))^\gamma \quad (5)$$

where \mathbf{p}_i are continuous point positions of a pattern and \mathbf{g}_j the grid positions of the underlying Gabor feature maps (normalized to $[0, 1]$). \mathbf{w}_i is a vector for each point representing additional attributes such as scale, depth, radii. Points with no attributes and only positions makes $\mathbf{w}_i = 1$ a scalar. When each point has an additional attribute(s) such as radii, \mathbf{w}_i vector represent these attribute values which are normalized to $[0, 1]$. \mathbf{v} represents the frequency. The $\gamma = \{0, 1\}$ parameter can be either 0 or 1 in Eq. 5 and it helps capture both the spatial and spectral features simultaneously. For $\gamma = 0$, the Gabor kernel only captures the spatial statistics of the point pattern. This dials down our model to irregular convolution proposed by Tu et al. [TLH19]. When $\gamma = 1$, it also captures the spectral characteristics of the pattern. By modeling both the spatial and spectral statistics, our method synthesize patterns with better local structure preservation. We create a Gabor filter bank by varying the values of \mathbf{v}, σ and γ . These filters help capture both spatial and spectral features of a point set.

4.1. Multi-resolution Convolutional Filtering

In the second step, we extract multi-scale, hierarchical features from the Gabor feature maps using multi-resolution convolutional filtering (orange blocks in Fig. 2). All convolutional filters are 7×7 pixels wide. For the first layer, we use a stride 1 convolution. For the remaining layers we use stride 2 which down samples the feature maps by $2 \times$. This makes the filtering process multi-resolution. All convolutional filters are drawn from a uniform distribution following Glorot and Bengio [GB10]. More specifically, our model consists of 4 filtering layers. Each layer uses a combination of convolution, instance normalization [UVL16][YTO*18] and ReLU activation

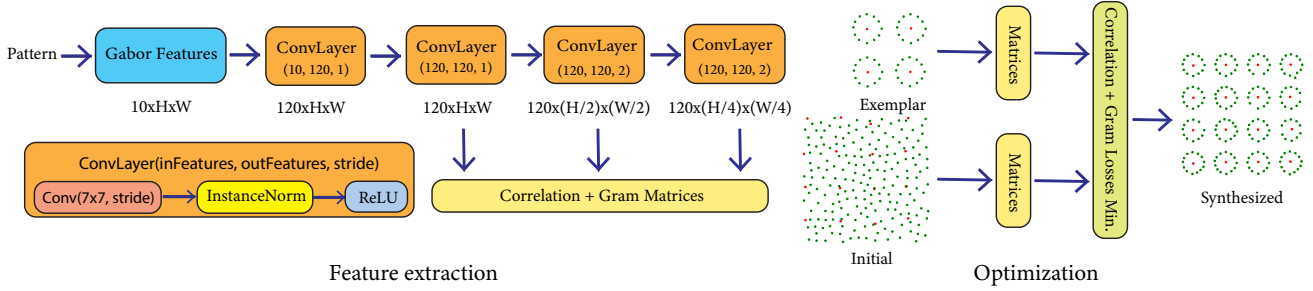


Figure 2: Overview of our pipeline. We propose to use continuous Gabor transform combined with a multi-resolution convolutional filtering step to compute Correlation and Gram statistics. Since all components are differentiable, we perform end-to-end optimization to update output pattern by minimizing the Correlation and Gram losses.

function. For the instance normalization layer, we use the following equation without any trainable parameter:

$$\tilde{q} = \frac{q - E[q]}{\sqrt{\text{Var}[q] + \epsilon}} \quad (6)$$

where q is the feature map values after convolution and \tilde{q} is the output after normalization, $\epsilon = 1e-5$. The mean ($E[q]$) and standard deviation ($\sqrt{\text{Var}[q]}$) is calculated per-dimension of the feature map. Note that our proposed model is data-independent, meaning that our random filters do not need to be trained on a specific data set. The randomly initialized filters are used directly as network weights for feature extraction. In Sec. 5.3, we perform an additional spectral analysis to understand why these convolutional filters work.

4.2. Loss Function

After the convolutional layers, we compute the correlation matrix (Eq. 3) and the Gram matrix (Eq. 1) from the resulting features. The corresponding matrices are used to compute our final objective/loss function that we use for synthesis using optimization. The final loss function is a weighted combination of both losses (Sec. 3.1):

$$\mathcal{L}_{total} = \sum_{l=4} \mathcal{L}_{corr}^l + \alpha \sum_{l=2,3,4} \mathcal{L}_{gram} \quad (7)$$

where $\alpha = 0.08$. We use the 4th layer of the network output for computing \mathcal{L}_{corr} and layers 2 to 4 for \mathcal{L}_{gram} . It is worth mentioning that computing \mathcal{L}_{corr} is more expensive than computing \mathcal{L}_{gram} . The computational cost depends on n^l, H^l, W^l at the same time. We experimentally find that using $n^l = 120$ and the 4-th layer’s output ($4\times$ down-sampling of input resolution) for computing \mathcal{L}_{corr} works best wrt both the optimization time and quality.

4.3. Synthesis via Optimization

The final step of our pipeline is the optimization. With our proposed differentiable pipeline, we use a gradient descent-based optimization to update output point locations by minimizing the loss functions. Output point pattern is initialized as Poisson disk distribution the same as Tu et al. [TLH19] within $[0, 1]$ domain. Note that our model also works for random initialization (see supplemental document

Fig. 3). During optimization, we update point positions by minimizing the proposed loss functions. Since the Gabor transform and convolutional filters are fully differentiable, our optimization process runs end-to-end. Unlike previous methods [RÖM*15; TLH19], we do not apply any heuristics such as removing or re-adding points that might produce distorted local structures. More specifically, we apply the multi-scale optimization procedure [RÖM*15; TLH19] from high σ values to small. Our pipeline naturally extends to handle multi-class and multi-attribute point patterns.

5. Experiments

We implement our full pipeline using PyTorch [PGC*17] and run experiments on an NVIDIA Quadro RTX 8000 graphics card. Additional comparisons and ablation studies can also be found in our supplemental document.

Gabor features. We compute Gabor features in a multi-scale manner by evenly sampling four σ values in the range $[\sigma_1, \sigma_2]$. The value of σ is analogous to a receptive field, higher σ value capture non-local structure while low σ focuses more on local structures. $\sigma_1 = d/c_1$ and $\sigma_2 = d/c_2$, where c_1, c_2 are user-defined and d is the mean minimum distance between any two points within a pattern. We set c_1 to be 0.8 ± 0.2 while c_2 is 2.8 ± 0.2 for all results shown in the paper (more details in the supplemental Table 1).

For each scale σ_s , we create 1 spatial map ($\gamma = 0$) and additional 9 frequency feature maps ($\gamma = 1$) from a pattern. The impact of different σ and γ is shown in Fig. 3. We consider 3×3 grid frequencies \mathbf{v} in the range $\{-1.5, -1.5\} \rightarrow \{1.5, 1.5\}$. For each frequency \mathbf{v} , we compute the Gabor function from Eq. 5. For each pattern, we set the resolution of Gabor feature maps based on underlying mean average distance d . The resolution of each axis is set to be $\text{round}(9/d)$ which is then clamped between $[128, 256]$. The resolution of the output Gabor feature maps is scaled by \sqrt{S} , which is the pattern expansion scaling factor ($S = 2$). In practice, we use k nearest neighbors of each point \mathbf{p} to compute Gabor features for a given grid position, with $k = \min(M, 40)$.

We use the Adam [KB14] optimizer for gradient descent with learning rates $lr = 0.005, 0.002, 0.002, 0.002$ for different 4 scales.

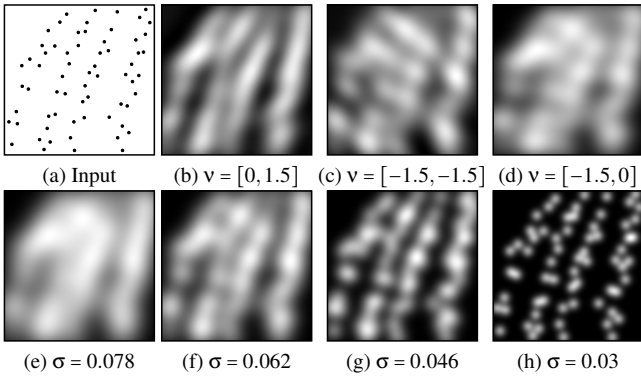


Figure 3: Examples of generated Gabor features from a point pattern (a). (b)-(d) show Gabor features with varying frequencies, the same $\sigma = 0.078$ and $\gamma = 1$. (e)-(h) show Gabor features with varying σ values and $\gamma = 0$ (without considering frequencies).

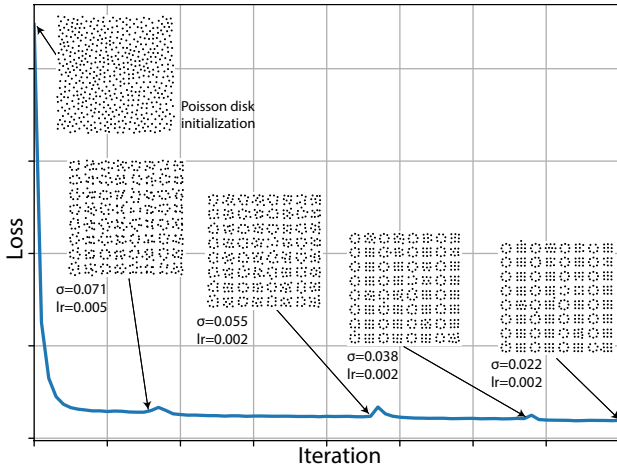


Figure 4: Initial and intermediate results after being optimized at each scale during the multi-scale optimization process.

Note that using other optimizers such as L-BFGS can yield similar results with other components in our pipeline fixed. Fig. 4 shows that the optimization starts from a Poisson disk distribution. At each scale, we optimize point locations within domain $[0, 1]$ until convergence. The convergence criterion is that the decrease of loss during the most recent 100 iterations is smaller than 1%. The whole optimization at all 4 scales takes around 400 ~ 800 iterations.

5.1. Qualitative Comparisons

We compare our point pattern synthesis approach with Ma et al. [MWT11], Roveri et al. [RÖM*15], Tu et al. [TLH19] and Reddy et al. [RGF*20]. We set the input and output domain to be the same for all comparisons. Ma et al. [MWT11] use random patch copy for pattern initialization. They use Hungarian matching algorithm to update the sample locations in a patch-based manner. For a fair comparison, we set the neighborhood size in Ma et al. to ensure the output points are $S = 4$ times the input points. Rover et al. [RÖM*15] also use random-patch copy initialization for output

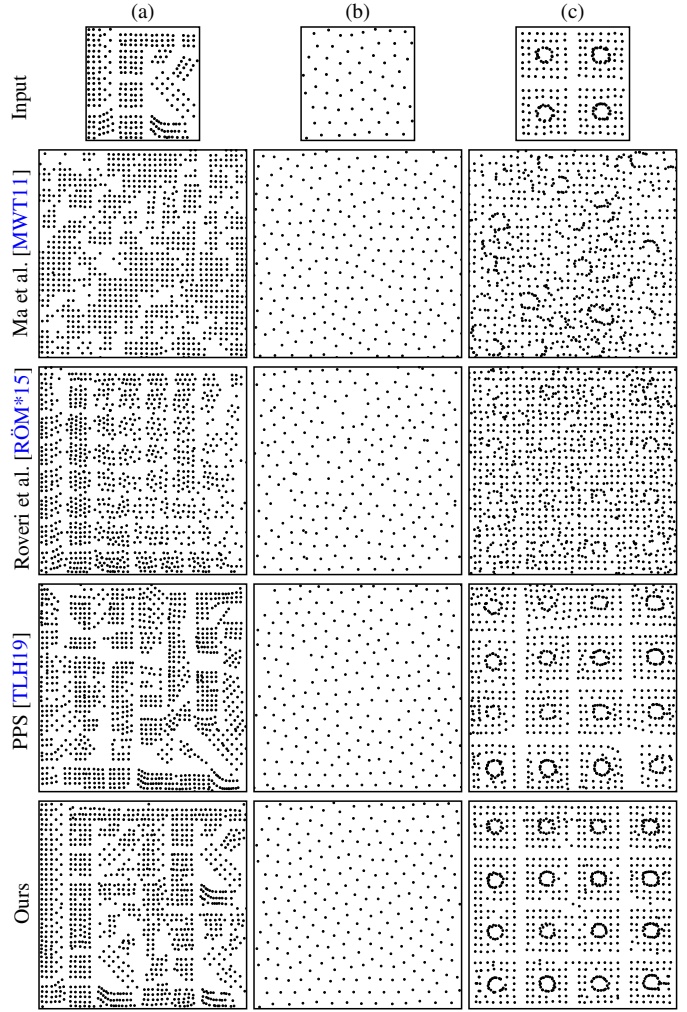


Figure 5: Single-class point-pattern expansion comparison between ours and previous methods. Our approach preserves the local and non-local structures better than previous methods.

pattern. Sample locations are then updated using multi-scale optimization with sampling control by removing and re-adding samples for better convergence. Tu et al. [TLH19] (PPS) use an irregular convolution layer, a pre-trained VGG-19 [SZ14] network and a hierarchical optimization scheme to synthesize point patterns from Poisson disk initialization. We use the same hyper-parameters c_1, c_2 and also initialize the output point patterns with Poisson disk for a fair comparison.

Single-class point patterns. Fig. 5 shows comparisons on regular and irregular structures. Neighborhood-based methods [MWT11; RÖM*15] fail to synthesize the global structure accurately. PPS [TLH19] works well compared to other prior methods [MWT11; RÖM*15] but suffers from visible artifacts like holes or missing points (column b). PPS also does not capture well the regular structures in column (a) and (c). This may happen due to the fact that VGG-based filters are redundantly learned from the images' dataset and is not expressive enough to deal with point patterns. Our



Figure 6: Multi-class point-pattern expansion. Our method performs better in terms of global structure and local distances between point samples. PPS [TLH19] does not handle well point patterns with multiple classes. We enhance PPS with our sequential multi-class synthesis approach but it still shows artifacts. The bottom row shows rendered results with object placement at point locations.

method handles well the regular structures and avoid any artifacts in stochastic distributions (Fig. 5b). Please see the supplemental document for more comparisons with different point patterns.

Multi-class point patterns. Our method naturally extends for multi-class point pattern synthesis. We sequentially optimize for each class similar to Ecornier et al. [EMGC19a]. While synthesizing points of k -th class, we freeze the positions of previous $k - 1$ synthesized classes and compute the Gabor and convolutional feature maps from all these k -class points. Unlike PPS [TLH19], which use one-hot representation for class IDs, our approach does not require additional dimensionality for class IDs. In Fig. 6, we compare our multi-class synthesis approach with PPS. For a fair comparison, we incorporate our sequential multi-class approach within PPS to better handle multi-class patterns. This is shown as PPS++ in Fig. 6.

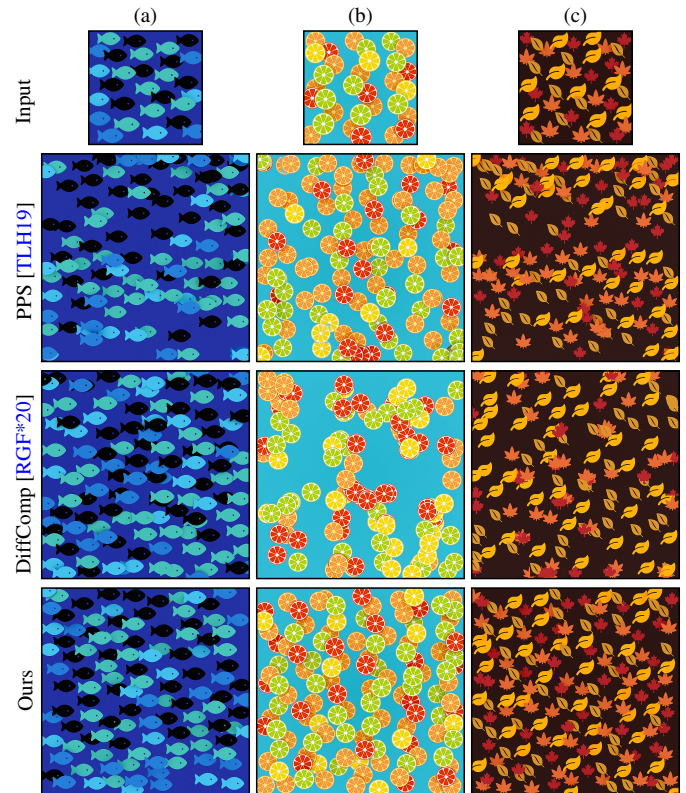


Figure 7: Discrete element-pattern expansion. Our method naturally supports point-based element patterns. We treat each color as a separate class. Each column here is a 4-class pattern with two attributes (scale and depth) per element.

This allows PPS to work with more than 3 classes. The bottom row shows object placement with different objects assigned to different classes. Fig. 6a shows an example that is also presented in Fig. 1.

Multi-attribute point patterns. Our approach can easily incorporate multiple attributes to the pipeline. We assign values (normalized to $[0, 1]$) to attributes w_i in Eq. 5 for each point while computing the Gabor features. As a result, for $|w|$ attributes, we will have $|w|$ Gabor feature maps concatenated together. Similarly, the number of filters in the convolutional layers would increase by a factor of $|w|$, resulting in $|w|$ -channel images going as input to the convolutional filters (see Fig. 2).

In Fig. 7, we show comparison on 4-class patterns with attributes (depth and scale) assigned to each point. DiffComp [RGF*20] work well for patterns with stochastic structures (column a, c), but it fails to capture the vertical structure in column (b). DiffComp also suffers during pattern expansion with more empty spacing as shown in column (c). PPS [TLH19], on the other hand, works slightly better for structured input, like column (b). However, it still shows some holes in all three output patterns. Although we do not explicitly encode element shapes as attributes, our synthesized patterns are visually closer to the exemplars.

As DiffComp [RGF*20] only uses Gram loss for pattern expansion, we study two more variants of their method using \mathcal{L}_{corr} and

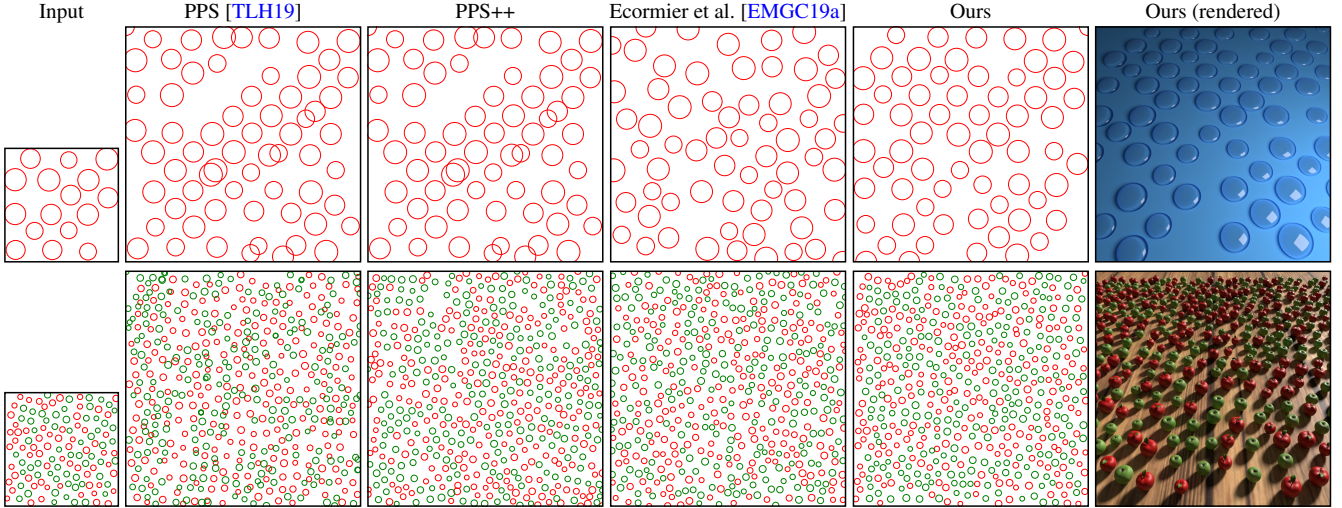


Figure 8: Comparison of our method with PPS [TLH19], PPS++ (our enhanced variant of PPS) and Ecormier et al. [EMGC19a] on disk and multi-class disk distributions. Our method better preserves the overall structure for both regular and irregular structures. Meanwhile, we achieve comparable results with Ecormier et al. [EMGC19a] on the second row, in terms of preserving non-overlapping disks.

Table 1: Quantitative comparison of relative radius [LD08] on a CCVT [BSD09] blue noise pattern as shown in Fig. 5 (b). The closer to the exemplar, the better.

Exemplar	[MWT11]	[RÖM*15]	[TLH19]	Ours
ρ	0.7341	0.6619	0.6507	0.6655

the weighted combination of \mathcal{L}_{gram} and \mathcal{L}_{corr} as shown in supplemental Fig. 7. We also compare the same methods on 2- and 3-class element patterns in supplemental Fig. 6 and show more results for our method on 2- and 6-class, 2- and 5-attribute element patterns in supplemental Fig. 11.

In Fig. 8, we show disk distributions where points are assigned a radius as an attribute. We compare our method against PPS [TLH19] and Ecormier et al. [EMGC19a] (a method based on pair correlation function) on a single- and two-class disk distribution. Our method handles better the regular structures in the first row and achieve comparable results in the second row.

5.2. Quantitative Evaluation

To the best of our knowledge, there is no established metric to quantitatively evaluate structured point patterns. Therefore, we perform several experiments including a user study, providing users side-by-side comparisons between our method and prior methods.

Quantitative metrics. A quantitative measure called relative radius [LD08] is well-defined for blue noise patterns. We choose the CCVT [BSD09] profile as the exemplar and apply 4 methods in Fig. 5 (b) for synthesis. Relative radius is defined as $\rho = r_{max}/r_{min}$. As shown in Table 1, the relative radius with our approach is the closest to the exemplar. This means we better preserve the equidistribution property compared to other methods. Visual inspection consistently supports this behavior as shown in Fig. 5(b).

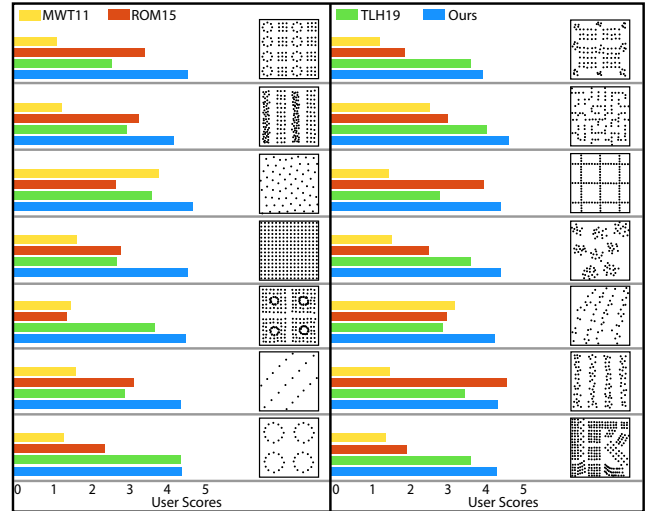
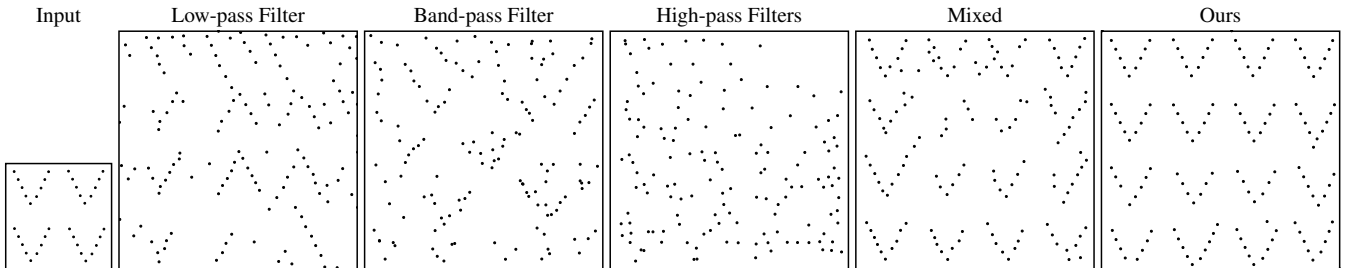


Figure 9: We perform user study and compute an average score across 28 users. We show different point patterns and ask users to score between 1 (worst) and 5 (best). Ours (in blue) gets better score for all but one pattern. All patterns and their comparisons can be found in the supplemental document.

We also include Wasserstein distance, Chamfer distance and the pair correlation function (PCF) as additional metrics for quantitative comparison. Pair Correlation Function (PCF) is also widely used for characterizing stationary point patterns. As our goal is to synthesize output pattern on a larger canvas, direct comparison of these three metrics is not reasonable. We, therefore, split the output pattern into 4 non-overlapping patches. We then compute PCF of each patch and compare it against the PCF of the exemplar. For PCF, we compute the mean-square error (MSE) between the output patch PCF and the exemplar's PCF. The error is averaged across the 4 patches. Similar

Table 2: Quantitative comparisons for single-class point pattern synthesis results of previous methods and ours. More quantitative results can be found in supplemental document Table 3.

Scene	MSE of PCFs (\downarrow)				Wasserstein Distance (\downarrow)				Chamfer Distance (\downarrow)			
	[MWT11]	[RÖM*15]	[TLH19]	Ours	[MWT11]	[RÖM*15]	[TLH19]	Ours	[MWT11]	[RÖM*15]	[TLH19]	Ours
Fig. 5, (a)	0.2678	0.2836	0.2609	0.2459	1.4483	1.1148	1.0222	0.6896	0.0571	0.0440	0.0517	0.0353
Fig. 5, (b)	0.3296	0.3286	0.3103	0.3072	0.4858	0.5166	0.4885	0.3352	0.0962	0.0869	0.0920	0.0429
Fig. 5, (c)	0.6346	0.4913	0.5436	0.3911	2.1869	2.2596	2.2570	0.5287	0.0662	0.0685	0.0719	0.0184

**Figure 10:** Analysis on using different filters in our filtering pipeline. Using Low-pass (Gaussian), band-pass filters (Sobel) and high-pass filters (derivative) fail to preserve complicated structures on the output pattern. While a simple combination of low-, band- and high-pass filters start giving correct orientation and overall structures.

operation is performed for the Wasserstein and Chamfer distance metrics. Table 2 shows the values for all these metrics for point patterns in Fig. 5. Additional quantitative analysis for more patterns can be found in the supplemental Table 3.

User study. We perform a user study to analyze the visual differences between our and previous methods. In total, 28 users participated in this experiment. Users were shown the input exemplars and the shuffled outputs from four methods: Ours, Ma et al. [MWT11], Roveri et al. [RÖM*15] and Tu et al. [TLH19]. We use 14 different point patterns for the study. Three of the patterns are shown in Fig. 5. The other patterns can be found in the supplemental Figs. 8, 9 and 10. Users are asked to give a score from 1 (worst) to 5 (best) for each of the output pattern based on their visual preferences over the local and global structure preservation. Fig. 9 shows the final averaged score across all users for each pattern. Our method achieves the highest scores on average for all scenes except one. Detailed results and numbers about the user study can be found in our supplemental document Table 2.

5.3. Analyzing Random Filters

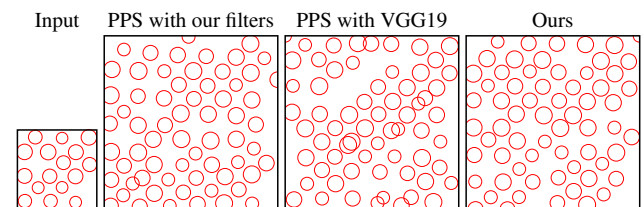
Random filters have shown impressive improvements [UVL18; UBGB16] for feature extraction. We perform spectral analysis (in Fig. 10) to understand the impact of random filters with different frequency spans on the synthesis. For this experiment, we use a simplified filter bank during optimization: 4 gaussian filters (low-pass) with evenly sampled sigma, 4 sobel operators as band-pass and 4 derivative filters as high-pass.

As demonstrated in Fig. 10, when using only low-pass, high-pass or band-pass filters during optimization, we found the resulting synthesis quality to be bad. However, using all filters together starts bringing the quality of synthesis a bit closer to ours (with random filters). This demonstrates that a combined filter bank works better

as it contains filters spanning a larger range of frequencies (from low to high). In theory, random filters span all frequencies based on their Fourier power spectrum. This explains why using random filters can generate even better results, from the perspective of Fourier analysis. VGG-based filters trained on large-scale image data, however, are not guaranteed to span all frequencies. This is consistent with our point pattern synthesis results where random filters outperforms VGG-based filters.

5.4. Ablation Study

We first evaluate the design choices of our pipeline using Gabor and random filters. As shown in Fig. 11, using only spatial features by irregular convolution [TLH19] results in degraded local and global features with overlaps and outliers. Using Gabor features better preserves local and global structures compared with the input. Further, as illustrated in Fig. 12, using only Gabor filters without random filters does not capture the details (column (b)). Using only random filters means we remove the Gabor filters by pixelating the input points. This results in incorrect and broken shapes (column (c)). Our combination of both filters better preserves local and global

**Figure 11:** Ablation study on our proposed Gabor features and filters, compared with the irregular convolutional features and VGG19 network in PPS [TLH19]. Our filters better capture overall structure and Gabor features better preserve local structure.

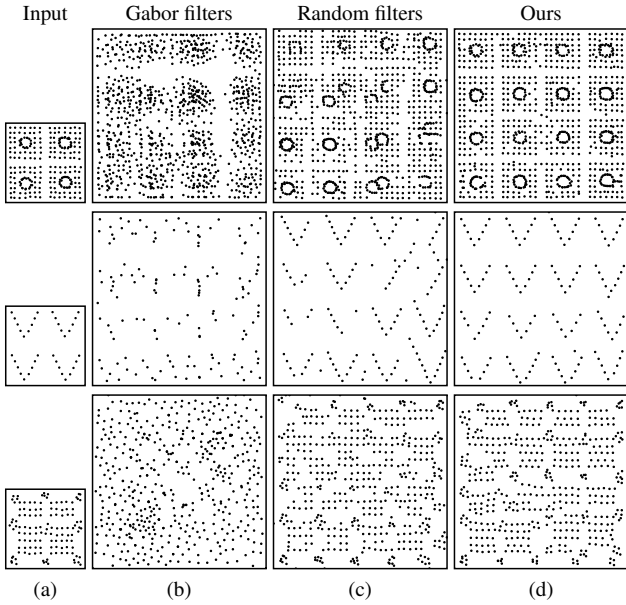


Figure 12: Ablation study on our proposed Gabor and random filters. Our combination of Gabor and random filters better capture overall structure than using only Gabor or random filters.

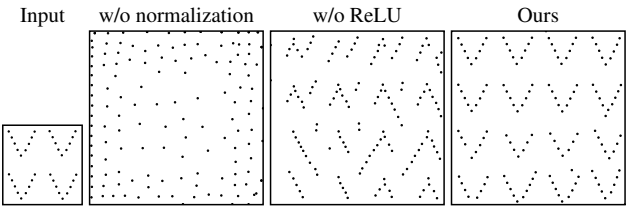


Figure 13: Ablation study on component choices in our convolutional filtering pipeline. We compare our final design with the one without instance normalization or ReLU activation. This shows both normalization and ReLU are important for high-quality synthesis.

structures compared with the input. Also, we study the impact of normalization layers and non-linear activation functions. Fig. 13 demonstrates that our method would completely fail without using normalization. Without non-linear activation function such as ReLU, the synthesized pattern such as V-structure, are much more noisy and the global structure is not preserved well.

Loss functions. We study how different losses affect the synthesis quality. As shown in Fig. 14, deep correlation loss captures better the globally regular structure, while gram loss captures local structures. Without \mathcal{L}_{corr} , the overall regularity is not preserved. Without \mathcal{L}_{gram} , the disks starts to have overlaps. More recently, Heitz et al. [HVCB21] present a Sliced Wasserstein loss for image-based texture synthesis, achieving better results than using Gram loss. We have experimented with Sliced Wasserstein loss noted as \mathcal{L}_{sw} , as a replacement of Gram loss. Fig. 14 illustrates that combining \mathcal{L}_{sw} with \mathcal{L}_{corr} introduces more distortion of the local structure than ours ($\mathcal{L}_{gram} + \mathcal{L}_{corr}$). More ablation studies about the number of random filters, the chosen layer for \mathcal{L}_{corr} computation and loss functions can be found in supplemental Section 3.1.

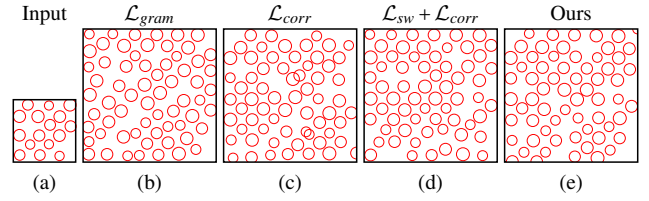


Figure 14: Ablation study on losses. With only \mathcal{L}_{gram} , the output patterns do not preserve the global structure. Though using only \mathcal{L}_{corr} preserves the globally regular structure, the output patterns fail to preserve local details such as the distances between points and disks. \mathcal{L}_{sw} combined with \mathcal{L}_{corr} introduces more distortion than ours (column (d) bottom region).

Table 3: We summarize statistics of different exemplars and runtime of our method and PPS [TLH19]. Note that for the results of multi-class patterns in Fig. 6, Fig. 7, and Fig. 8, we compare against the runtime of PPS++ variant for fair comparisons. R refers row number of the corresponding figure.

Scene	#Classes	#Output Samples	#Attributes	Runtime (m)	
				PPS	Ours
Fig. 5, (a)	1	1124	-	23.9	5.7
Fig. 5, (b)	1	256	-	11.0	3.8
Fig. 5, (c)	1	992	-	22.7	5.6
Fig. 6, (a)	2	208	-	18.1	4.4
Fig. 6, (b)	2	512	-	58.1	7.4
Fig. 6, (c)	4	216	-	44.7	9.7
Fig. 7, (a)	4	112	2	23.8	9.3
Fig. 7, (b)	4	124	2	26.7	11
Fig. 7, (c)	4	160	2	31.7	14.5
Fig. 8, R1	1	64	1	5.6	2.3
Fig. 8, R2	2	512	1	58.1	7.4

5.5. Performance

With our implementation, synthesizing a point pattern from an exemplar takes from 2 to 15 minutes, where the number of output samples varies from 64 to 1124. We show the run-time of PPS [TLH19] and our method using an NVIDIA Quadro RTX 8000 on various exemplars in Table 3. Our method is up to 8 times faster than theirs. Also, the total number of parameters of our filters is about 2.4M, while the number of parameters of VGG-19 feature extractor they used is about 10.6M, about 5 times more than ours.

6. Conclusions

We present a Gabor transform-based framework for point pattern synthesis. This allows capturing both spatial and spectral features simultaneously. Unlike previous point pattern synthesis approaches, our method naturally extends to multi-class and multi-attribute point pattern synthesis. We simplify the feature space to Gabor feature maps and random filters' features. We also analyze why random filters work better than well-trained VGG-19 based features. Our analysis hints that since random filters span all frequencies, it helps better capture the details previously missed by VGG-19 based features.

Limitations. Feature map representation [RÖM*15; TLH19] allows us to extract positional and structural information from points,

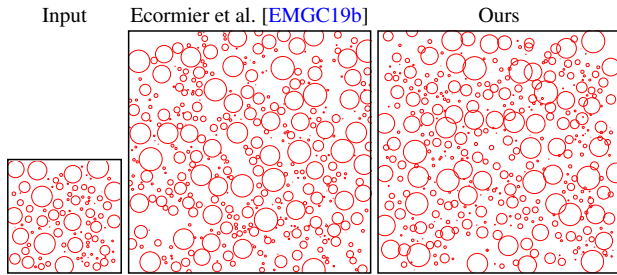


Figure 15: A failure case for highly constrained disk distribution. Since we do not explicitly handle the distance between disks of varying radii, it is hard for our method to synthesize this distribution without overlaps, although the distributions seem still relevant.

which makes it possible for our method to handle patterns with anisotropic, regular and irregular structure. However, similar to previous methods [MWT11; RÖM*15; TLH19], we share the same issue that it requires user to input the window size or kernel size (e.g., σ in Eq. 5) for optimization. These hyper-parameters can affect the quality of the synthesis as shown in [TLH19]. In the multi-class setting, we need to fully run the optimization for each class. Consequently, the time complexity of our optimization grows linearly with the number of classes. This can be problematic for applications where interactive feedback is critical.

Future work. Currently, our feature maps are pixelized. It would be interesting to encode the features in a continuous space. This could potentially avoid artifacts due to the pixelized nature of the feature maps. Though our characterization and synthesis pipeline is end-to-end, with no human intervention, the results are still not perfect according to the user evaluation. Average score for each pattern using our method is from around 4 to 4.5, while 5 is the maximum. One direction for further improvements on local structure can be using an interactive authoring system as shown in Tu et al. [TWY*20]. This would allow users to fix remaining synthesis issues in local regions interactively. Currently, we focus on a unified pipeline for point pattern synthesis. However, as shown in Fig. 15, our method might fail to capture disk distributions where disk can have varying radii and are strictly non-overlapping. Next step should be to consider extending our method to explicitly control the distance between disks or even to handle shape-aware elements. Besides, we are interested in extending our point-based framework to related applications such as continuous curve synthesis [TWY*20], distribution infilling and clustering [NEMC20].

Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments and Pierre Ecormier-Nocca for helping with object placement in Blender [Com18]. Renderings shown in the results use free models from Turbosquid [Tur21] under the "Editorial Use" license.

References

[ALS*18] ABERMAN, KFIR, LIAO, JING, SHI, MINGYI, et al. "Neural best-buddies: Sparse cross-domain correspondence". *ACM Transactions on Graphics (TOG)* 37.4 (2018), 1–14 2.

- [BBT*06] BARLA, PASCAL, BRESLAV, SIMON, THOLLOT, JOËLLE, et al. "Stroke Pattern Analysis and Synthesis". *Computer Graphics Forum* 25.3 (2006), 663–671. DOI: <https://doi.org/10.1111/j.1467-8659.2006.00986.x> 2.
- [BJS17] BOJANOWSKI, PIOTR, JOULIN, ARMAND, LOPEZ-PAZ, DAVID, and SZLAM, ARTHUR. "Optimizing the latent space of generative networks". *arXiv preprint arXiv:1707.05776* (2017) 2.
- [BSD09] BALZER, MICHAEL, SCHLÖMER, THOMAS, and DEUSSEN, OLIVER. "Capacity-constrained point distributions: A variant of Lloyd's method". *ACM Transactions on Graphics (TOG)* 28.3 (2009), 1–8 7.
- [Com18] COMMUNITY, BLENDER ONLINE. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: <http://www.blender.org> 10.
- [DHL*98] DEUSSEN, OLIVER, HANRAHAN, PAT, LINTERMANN, BERND, et al. "Realistic Modeling and Rendering of Plant Ecosystems". *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 1998, 275–286. ISBN: 0897919998. DOI: [10.1145/280814.280898](https://doi.org/10.1145/280814.280898). URL: <https://doi.org/10.1145/280814.280898> 1.
- [EMGC19a] ECORMIER-NOCCA, PIERRE, MEMARI, POORAN, GAIN, JAMES, and CANI, MARIE-PAULE. "Accurate Synthesis of Multi-Class Disk Distributions". *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library, 2019, 157–168 2, 6, 7.
- [EMGC19b] ECORMIER-NOCCA, PIERRE, MEMARI, POORAN, GAIN, JAMES, and CANI, MARIE-PAULE. "Accurate Synthesis of Multi-Class Disk Distributions". *Computer Graphics Forum* 38.2 (2019), 157–168. DOI: <https://doi.org/10.1111/cgf.13627> 10.
- [GAD*20] GUEHL, PASCAL, ALLEGRE, RÉMI, DISCHLER, J-M, et al. "Semi-Procedural Textures Using Point Process Texture Basis Functions". *Computer Graphics Forum*. Vol. 39. 4. Wiley Online Library, 2020, 159–171 2.
- [GAM*21] GIESEKE, LENA, ASENTE, PAUL, MÉCH, RADOMIR, et al. "A Survey of Control Mechanisms for Creative Pattern Generation". *Computer Graphics Forum*. Vol. 40. 2. Wiley Online Library, 2021, 585–609 2.
- [GB10] GLOROT, XAVIER and BENGIO, YOSHUA. "Understanding the difficulty of training deep feedforward neural networks". *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, 249–256 3.
- [GEB15a] GATYS, LEON, ECKER, ALEXANDER S, and BETHGE, MATTHIAS. "Texture Synthesis Using Convolutional Neural Networks". *Advances in Neural Information Processing Systems*. Ed. by CORTES, C., LAWRENCE, N., LEE, D., et al. Vol. 28. Curran Associates, Inc., 2015, 262–270 2.
- [GEB15b] GATYS, LEON A, ECKER, ALEXANDER S, and BETHGE, MATTHIAS. "A neural algorithm of artistic style". *arXiv preprint arXiv:1508.06576* (2015) 2, 3.
- [HB95] HEEGER, DAVID J and BERGEN, JAMES R. "Pyramid-based texture analysis/synthesis". *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 1995, 229–238 2.
- [HLT*09] HURTUT, T., LANDES, P-E., THOLLOT, J., et al. "Appearance-Guided Synthesis of Element Arrangements by Example". *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*. NPAR '09. New Orleans, Louisiana: Association for Computing Machinery, 2009, 51–60. ISBN: 9781605586045. DOI: [10.1145/1572614.1572623](https://doi.org/10.1145/1572614.1572623) 2.
- [HSD13] HECK, DANIEL, SCHLÖMER, THOMAS, and DEUSSEN, OLIVER. "Blue Noise Sampling with Controlled Aliasing". *ACM Trans. Graph.* 32.3 (July 2013). ISSN: 0730-0301. DOI: [10.1145/2487228.2487233](https://doi.org/10.1145/2487228.2487233). URL: <https://doi.org/10.1145/2487228.2487233> 2.

- [HVCB21] HEITZ, ERIC, VANHOEY, KENNETH, CHAMBON, THOMAS, and BELCOUR, LAURENT. “A Sliced Wasserstein Loss for Neural Texture Synthesis”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 9412–9420 **9**.
- [HWH16] HE, KUN, WANG, YAN, and HOPCROFT, JOHN. “A powerful generative model using random weights for the deep image representation”. *Advances in Neural Information Processing Systems* 29 (2016) **2**.
- [KB14] KINGMA, DIEDERIK P and BA, JIMMY. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014) **4**.
- [KSH12] KRIZHEVSKY, ALEX, SUTSKEVER, ILYA, and HINTON, GEOFFREY E. “ImageNet Classification with Deep Convolutional Neural Networks”. *Advances in Neural Information Processing Systems*. Ed. by PEREIRA, F., BURGESS, C. J. C., BOTTOU, L., and WEINBERGER, K. Q. Vol. 25. Curran Associates, Inc., 2012, 1097–1105 **2**.
- [LD08] LAGAE, ARES and DUTRÉ, PHILIP. “A comparison of methods for generating Poisson disk distributions”. *Computer Graphics Forum*. Vol. 27. 1. Wiley Online Library, 2008, 114–129 **7**.
- [Lew89] LEWIS, J. P. “Algorithms for Solid Noise Synthesis”. *SIGGRAPH Comput. Graph.* 23.3 (July 1989), 263–270. ISSN: 0097-8930. DOI: [10.1145/74334.74360](https://doi.org/10.1145/74334.74360). URL: <https://doi.org/10.1145/74334.74360>.
- [LGH13] LANDES, PIERRE-EDOUARD, GALERNE, BRUNO, and HURTUT, THOMAS. “A Shape-Aware Model for Discrete Texture Synthesis”. *Proceedings of the Eurographics Symposium on Rendering*. EGSR '13. Zaragoza, Spain: Eurographics Association, 2013, 67–76. DOI: [10.1111/cgf.12152](https://doi.org/10.1111/cgf.12152).
- [LSM*19] LEIMKÜHLER, THOMAS, SINGH, GURPRIT, MYSZKOWSKI, KAROL, et al. “Deep Point Correlation Design”. *ACM Trans. Graph.* 38.6 (Nov. 2019). ISSN: 0730-0301. DOI: [10.1145/3355089.3356562](https://doi.org/10.1145/3355089.3356562).
- [MWLT13] MA, CHONGYANG, WEI, LI-YI, LEFEBVRE, SYLVAIN, and TONG, XIN. “Dynamic element textures”. *ACM Transactions on Graphics (TOG)* 32.4 (2013), 1–10 **2**.
- [MWT11] MA, CHONGYANG, WEI, LI-YI, and TONG, XIN. “Discrete Element Textures”. *ACM Trans. Graph.* 30.4 (July 2011). ISSN: 0730-0301. DOI: [10.1145/2010324.1964957](https://doi.org/10.1145/2010324.1964957) **2, 5, 7, 8, 10**.
- [NEMC20] NICOLET, BAPTISTE, ECORMIER-NOCCA, PIERRE, MEMARI, POORAN, and CANI, MARIE-PAULE. “Pair Correlation Functions with Free-Form Boundaries for Distribution inpainting and Decomposition”. *Eurographics 2020 short paper proceedings*. 2020 **10**.
- [ÖG12] ÖZTIRELI, A CENGIZ and GROSS, MARKUS. “Analysis and synthesis of point distributions based on pair correlation”. *ACM Trans. Graph.* 31.6 (2012) **1, 2**.
- [PGC*17] PASZKE, ADAM, GROSS, SAM, CHINTALA, SOUMITH, et al. “Automatic differentiation in pytorch”. (2017) **4**.
- [PS00] PORTILLA, JAVIER and SIMONCELLI, EERO P. “A parametric texture model based on joint statistics of complex wavelet coefficients”. *International journal of computer vision* 40.1 (2000), 49–70 **2**.
- [QSMG17] QI, CHARLES R, SU, HAO, MO, KAICHUN, and GUIBAS, LEONIDAS J. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 652–660 **2**.
- [QYSG17] QI, CHARLES RUIZHONGTAL, YI, LI, SU, HAO, and GUIBAS, LEONIDAS J. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. *Advances in Neural Information Processing Systems*. Ed. by GUYON, I., LUXBURG, U. V., BENGIO, S., et al. Vol. 30. Curran Associates, Inc., 2017, 5099–5108 **2**.
- [RGF*20] REDDY, PRADYUMNA, GUERRERO, PAUL, FISHER, MATT, et al. “Discovering Pattern Structure Using Differentiable Compositing”. *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: [10.1145/3414685.3417830](https://doi.org/10.1145/3414685.3417830) **1, 3, 5, 6**.
- [RÖG17] ROVERI, RICCARDO, ÖZTIRELI, A. CENGIZ, and GROSS, MARKUS. “General Point Sampling with Adaptive Density and Correlations”. *Computer Graphics Forum* 36.2 (2017), 107–117. DOI: <https://doi.org/10.1111/cgf.13111>.
- [RÖM*15] ROVERI, RICCARDO, ÖZTIRELI, A. CENGIZ, MARTIN, SEBASTIAN, et al. “Example Based Repetitive Structure Synthesis”. *Proceedings of the Eurographics Symposium on Geometry Processing*. SGP '15. Graz, Austria: Eurographics Association, 2015, 39–52. DOI: [10.1111/cgf.12695](https://doi.org/10.1111/cgf.12695) **1, 2, 4, 5, 7–10**.
- [SC17] SENDIK, OMRY and COHEN-OR, DANIEL. “Deep Correlations for Texture Synthesis”. *ACM Trans. Graph.* 36.5 (July 2017). ISSN: 0730-0301. DOI: [10.1145/3015461](https://doi.org/10.1145/3015461) **2, 3**.
- [SZ14] SIMONYAN, KAREN and ZISSERMAN, ANDREW. “Very deep convolutional networks for large-scale image recognition”. *arXiv preprint arXiv:1409.1556* (2014) **5**.
- [SZ15] SIMONYAN, KAREN and ZISSERMAN, ANDREW. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by BENGIO, YOSHUA and LECUN, YANN. 2015. URL: <http://arxiv.org/abs/1409.1556> **1–3**.
- [TLH19] TU, PEIHAN, LISCHINSKI, DANI, and HUANG, HUI. “Point Pattern Synthesis via Irregular Convolution”. *Computer Graphics Forum* 38.5 (2019), 109–122. DOI: <https://doi.org/10.1111/cgf.13793> **1–10**.
- [Tur21] TURBOSQUID. *Turbosquid*. <https://www.turbosquid.com/>. 2021 **10**.
- [TWY*20] TU, PEIHAN, WEI, LI-YI, YATANI, KOJI, et al. “Continuous Curve Textures”. *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: [10.1145/3414685.3417780](https://doi.org/10.1145/3414685.3417780) **2, 10**.
- [UBGB16] USTYUZHANINOV, IVAN, BRENDDEL, WIELAND, GATYS, LEON, and BETHGE, MATTHIAS. “What does it take to generate natural textures?”. (2016) **2, 8**.
- [Uli88] ULICHNEY, ROBERT A. “Dithering with blue noise”. *Proc. IEEE* 76.1 (1988) **2**.
- [UVL16] ULYANOV, DMITRY, VEDALDI, ANDREA, and LEMPITSKY, VICTOR. “Instance normalization: The missing ingredient for fast stylization”. *arXiv preprint arXiv:1607.08022* (2016) **3**.
- [UVL18] ULYANOV, DMITRY, VEDALDI, ANDREA, and LEMPITSKY, VICTOR. “Deep image prior”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 9446–9454 **2, 8**.
- [WSL*19] WANG, YUE, SUN, YONGBIN, LIU, ZIWEI, et al. “Dynamic Graph CNN for Learning on Point Clouds”. *ACM Trans. Graph.* 38.5 (Oct. 2019). ISSN: 0730-0301. DOI: [10.1145/3326362](https://doi.org/10.1145/3326362).
- [Yel83] YELLOTT, JOHN I. “Spectral consequences of photoreceptor sampling in the rhesus retina”. *Science* 221.4608 (1983) **2**.
- [YGW*15] YAN, DONGMING, GUO, JIAN-WEI, WANG, BIN, et al. “A Survey of Blue-Noise Sampling and Its Applications”. English (US). *Journal of Computer Science and Technology* 30.3 (May 2015), 439–452. ISSN: 1000-9000. DOI: [10.1007/s11390-015-1535-0](https://doi.org/10.1007/s11390-015-1535-0) **2**.
- [YTO*18] YI, KWANG MOO, TRULLS, EDUARD, ONO, YUKI, et al. “Learning to find good correspondences”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 2666–2674 **3**.
- [ZHWW12] ZHOU, YAHAN, HUANG, HAIBIN, WEI, LI-YI, and WANG, RUI. “Point Sampling with General Noise Spectrum”. *ACM Trans. Graph.* 31.4 (July 2012). ISSN: 0730-0301. DOI: [10.1145/2185520](https://doi.org/10.1145/2185520). URL: <https://doi.org/10.1145/2185520>. **2185572**.