DOI: 10.1111/cgf.13793
Eurographics Symposium on Geometry Processing 2019
D. Bommes and H. Huang
(Guest Editors)

Volume 38 (2019), Number 5

Point Pattern Synthesis via Irregular Convolution

Peihan Tu[1] Dani Lischinski[3] Hui Huang[1,2†]

[1]Shenzhen University, China [2]SIAT, China [3]The Hebrew University of Jerusalem, Israel

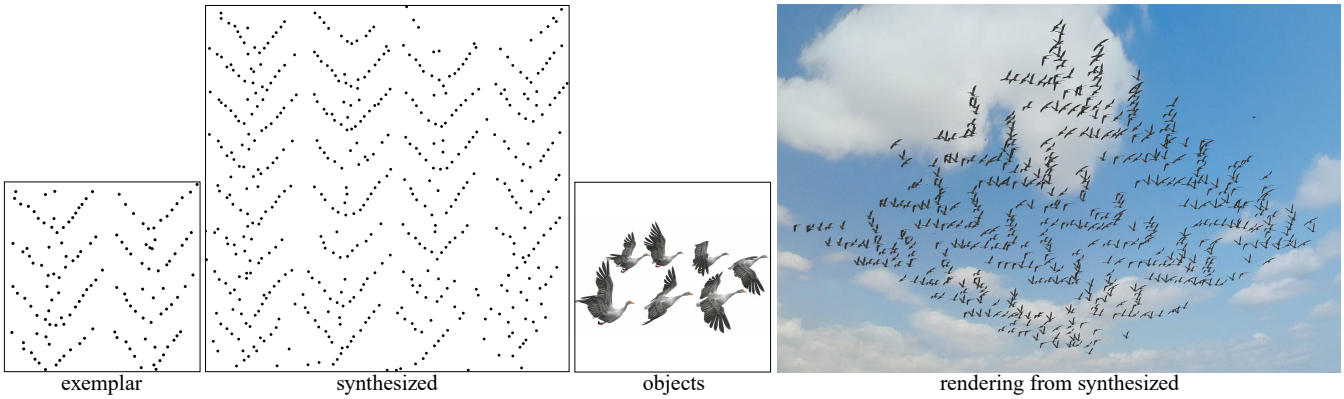


Figure 1: *Wild geese flying to the south. Our point pattern synthesis method takes a small example of a point pattern and synthesizes a larger one. The point pattern may be used to simulate object placement in virtual scenes.*

Abstract

Point pattern synthesis is a fundamental tool with various applications in computer graphics. To synthesize a point pattern, some techniques have taken an example-based approach, where the user provides a small exemplar of the target pattern. However, it remains challenging to synthesize patterns that faithfully capture the structures in the given exemplar. In this paper, we present a new example-based point pattern synthesis method that preserves both local and non-local structures present in the exemplar. Our method leverages recent neural texture synthesis techniques that have proven effective in synthesizing structured textures. The network that we present is end-to-end. It utilizes an irregular convolution layer, which converts a point pattern into a gridded feature map, to directly optimize point coordinates. The synthesis is then performed by matching inter- and intra-correlations of the responses produced by subsequent convolution layers. We demonstrate that our point pattern synthesis qualitatively outperforms state-of-the-art methods on challenging structured patterns, and enables various graphical applications, such as object placement in natural scenes, creative element patterns or realistic urban layouts in a 3D virtual environment.

1. Introduction

Point pattern synthesis is a fundamental tool with a variety of applications in computer graphics, such as rendering [Coo86], noise generation [Lew89], object placement [DHL*98; Wei10] and geometry synthesis [MWT11; MWLT13; EVC*15]. In particular, much effort has been devoted to the generation of blue noise patterns [Coo86; Fat11; KCDL06; Wei08], or patterns with other spectra by matching low level statistics [ZHWW12; ÖG12; HSD13; RÖG17].

Some example-based techniques have been proposed to synthesize point patterns from a small, user-provided, exemplar of the target pattern [ZHWW12; MWT11; ÖG12]. However, it still remains a challenge to synthesize point patterns that capture the structures present in the input exemplar.

In this paper, we present a novel point pattern synthesis method, which is able to preserve both local and non-local structures present in an exemplar pattern. Our method leverages recent neural

† Corresponding author: Hui Huang (hhzhiyan@gmail.com)

example-based texture synthesis techniques that are able to successfully synthesize structured textures [GEB15; SC17; ZZB*18]. A straightforward approach might attempt to synthesize point patterns by applying texture synthesis on a raster image of a point pattern, followed by extracting point locations from the synthesized image. However, as we show and discuss in Section 4.3, such an approach is problematic and error-prone, due to error accumulation. Instead, our approach performs the point pattern synthesis using an end-to-end network. The main idea is to utilize an irregular convolution layer, which converts the pattern into a feature map over a regular grid. The coordinates of points in the pattern may then be optimized directly so as to match the inter- and intra-correlations of responses produced by subsequent convolution layers; see the illustration in Figure 2. Please note that our method does not involve the training of neural network. The VGG-19 used here acts as a feature extractor, and the extracted features are used to define our pattern optimization objective.

Optimization with a neural network is highly non-convex. In texture synthesis, this non-convexity results in noisy synthesis output (examples may be observed in the texture synthesis results of Gatys et al [GEB15] and Sendik and Cohen-Or [SC17]); The prepended irregular convolution layer introduces further non-convexity with respect to the coordinates of the points. The optimization of point coordinates is thus prone to getting stuck at local minima, resulting in significant visual differences between the target exemplar and the converged results. For example, compare the global and local structures present in the exemplar shown in Figure 5(a) to the (poorly) converged results in Figure 5(b)–(d). To overcome this difficulty, our optimization process employs: (i) a *hierarchical strategy* (Section 4.1), where the size of the kernel in the irregular convolution layer is gradually decreased to avoid undesirable local stalling; and (ii) a *point relaxation* scheme (Section 4.2), where each point is associated with an additional attribute indicating the confidence of its existence.

We demonstrate that our point pattern synthesis qualitatively outperforms state-of-the-art methods on challenging structured patterns, and enables various graphical applications, such as object placement in natural scenes, creative element patterns or realistic urban layouts in a 3D virtual environment (Section 5). An analysis of limitations, conclusions, and future work concludes this paper (Section 6).

2. Related Work

2.1. Parametric Point Pattern Synthesis

Stochastic point patterns are sometimes referred to by the color associated with their power spectrum [WW11]. Synthesis algorithms have mostly focused on blue noise generation [Coo86; Fat11; KCDL06; Wei08], because of its good properties for anti-aliasing [Coo86] and visually pleasing uniformity for object placement [LD05]. Blue noise sampling has also been extended to multi-class samples [Wei10] and anisotropic fields [LWSF10].

Methods for generating point patterns with more general spectral properties have also been proposed [ZHWW12; ÖG12; HSD13; RÖG17; IPSS08], based on matching summary statistics of the pattern: the histogram of pairwise sample distances. However, summary statistics fail to capture even small spatial structures within the exemplar. Based on explicit parametric point process models [IPSS08], point patterns can also be generated by first estimating the parameters of a model from a small exemplar and sampling the model to synthesize a larger output. However, such models are domain-specific and cannot provide a universal solution for example-based synthesis. Since existing parametric models are based on low-level statistics, they also cannot deal with large non-local structures in patterns. For example, in discrete element texture synthesis domain, the methods of Landes et al. [LGH13] and Hurtut et al. [HLT*09] utilize complicated, hand-crafted multi-attribute point process models to synthesize textures containing limited categories of elements. Thus, they cannot be generalized to other types of patterns, e.g., biological or urban building patterns.

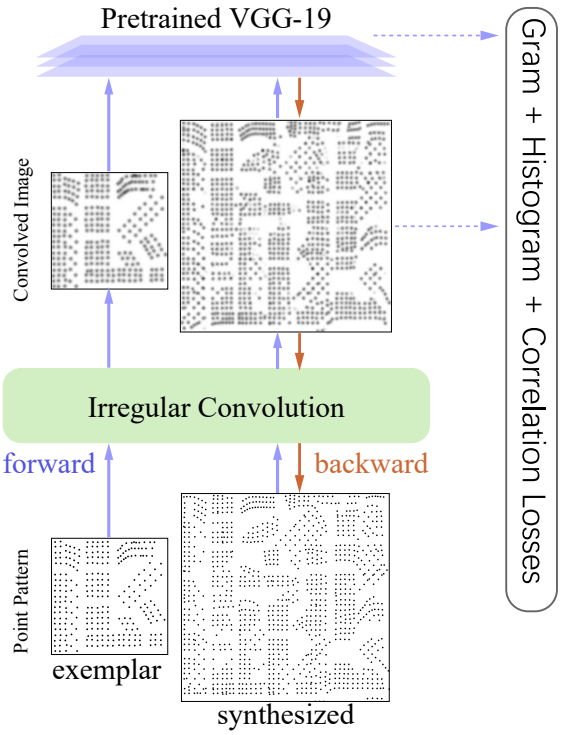


Figure 2: *Our neural net structure. A differentiable irregular convolution is prepended before a pretrained CNN, enabling end-to-end optimization of point coordinates. The irregular convolution produces a gridded feature map. In our design, this map is a density estimation of the point pattern, and the Gram, histogram and correlation losses are employed to encode feature statistics. The synthesis is then performed by penalizing the differences between feature statistics of input and output patterns.*

2.2. Non-parametric Point Pattern Synthesis

Neighborhood matching based methods have been adopted to synthesize both static and dynamic element textures from examples. This task is in essence equivalent to point pattern synthesis,

since elements are represented by single or multiple point samples [MWT11; MWLT13; XCW14; XWSY15; IMIM08]. However, neighborhood matching assumes a local Markovian model and thus cannot capture non-local structures. Also, the synthesis quality depends heavily on the initialization. Ma et al. [MWT11] and follow-up works [RÖM*15] generate the initialization by randomly copying patches from the exemplar, and this initialization determines the synthesis outcome. In contrast, our approach exhibits good convergence from a randomly distributed initial point pattern, regardless of the input exemplar. In addition, our results demonstrate that it exhibits better structure preservation than existing non-parametric methods.

Roveri et al. [RÖM*15] extend example-based texture optimization [KEBK05] to example-based point pattern optimization. Similarly to our approach, this is done by converting point patterns to density maps. However, differently from their approach, by using an end-to-end deep architecture, and introducing soft point optimization (Section 4.2), our method is able to better reproduce local and global structures present in the input exemplar. A comparison with the method of Roveri et al. [RÖM*15] is shown in Figure 9.

2.3. Example-based Texture Synthesis

For a general review of traditional example-based texture synthesis, please refer to Wei et al. [WLKT09] and Barnes et al. [BZ17]. Recent synthesis algorithms based on convolutional neural networks are able to produce good synthesis results [GEB15; SC17]. These methods are based on optimizing the input of a pretrained network so as to match the deep inter- or intra-feature correlations of a given exemplar, and provide the inspiration for our work. Wave function collapse [KS17] adopts similar idea from [Mer09] to synthesize raster textures.

Feed-forward networks for texture synthesis have also been proposed [ZZB*18; BJV17; LW16; ULVL16; LFY*17]. In most cases, a network must be trained for each input texture. A diversified synthesis network (i.e., for many textures) in general produces lower quality results [LFY*17], or does not support example-based synthesis (e.g., the type of generated texture cannot be controlled) [BJV17]. Our approach adapts optimization-based neural texture synthesis [GEB15; SC17] to work on discrete point patterns. We will investigate feed-forward synthesis in the near future.

2.4. Point Generation via Deep Learning

A variety of generative models for point clouds have been explored, e.g., [SWL*18; ADMG18; LZZ*18]. These works focus on generation of 3D models represented as point clouds, while we focus on generation of 2D point patterns that are perceptually similar to a provided exemplar.

Deep learning has also been applied to generate samples in rendering systems [LSM*18; MMR*18], where the focus is on producing patterns that improve the resulting rendering quality.

3. Point Pattern Synthesis

Given a two-dimensional n-point pattern $\{\mathbf{p}_i\}_{i=1}^{n}$, our goal is to synthesize a spatially larger point set $\{\widetilde{\mathbf{p}}_i\}_{i=1}^{N}$ $(N \geq n)$, which exhibits the same spatial structures that may be observed in the input pattern. More specifically, the spatial statistics should be roughly preserved. In the remainder of this text, we use $\sim$ to indicate variables associated with the synthesized $\{\widetilde{\mathbf{p}}_i\}_{i=1}^{N}$.

Our work is inspired by the neural texture synthesis methods of Gatys et al. [GEB15], and Sendik and Cohen-Or [SC17]. These approaches leverage a pretrained VGG-19 network [SZ14] and perform example-based texture synthesis by matching activation statistics. Point pattern synthesis is similar to texture synthesis, since point patterns also exhibit repetitive and stochastic characteristics. Thus, one might consider synthesizing point patterns by applying texture synthesis on images of point patterns, followed by extracting points from the synthesized image. However, as we show and discuss in (Section 4.3), such an approach is prone to error accumulation.

In this work, we avoid such error accumulation by formulating the synthesis as an end-to-end optimization process. This approach is enabled by prepending a differentiable convolution layer that converts an irregular point set into a gridded feature map, which is then fed into the pretrained VGG-19 network, as shown in Figure 2. Since the prepended layer is differentiable with respect to the point locations, we can use back propagation and optimize these locations directly, so as to match the statistics of the VGG-19 activations between the feature map of the input exemplar and that of the synthesized pattern.

Our optimization first adjusts the positions of the points $\{\widetilde{\mathbf{p}}_i\}_{i=1}^{N}$, and then improves the local density by decimating the resulting point pattern, where necessary. To this end, each point $\mathbf{p}_i$ is associated with a pair of spatial coordinates $\mathbf{s}_i$, as well as a scalar attribute $a_i \in [0,1]$, which represents the confidence of the point's existence. This attribute is constrained to 1 during the initial *hard point optimization* phase (Section 4.1), and the constraint is subsequently lifted in the *soft point optimization* phase that follows (Section 4.2), to support point removal.

3.1. Irregular Pattern Convolution

The points of a pattern are irregularly distributed across the plane. Recently, several works have addressed convolutions of point clouds, e.g., [LBSC18; AML18; HRV*18]. The goal of these works is to learn convolution operators that extract features from irregular 3D point clouds. In contrast, our goal is to merely transform an irregular 2D point pattern (in a differentiable manner) into a continuous map defined over a regular grid, so that it could be fed into a standard CNN, such as VGG-19. For this purpose, it suffices to use the classical definition of convolution as an integral transform [DM12]. Thus, the convolution of a point pattern is defined as:

$$Conv(\kappa, \{\mathbf{p}_i\}_{i=1}^{n}) = \kappa \circledast \sum_{i=1}^{n} a_i \delta(\mathbf{s} - \mathbf{s}_i) = \sum_{i=1}^{n} a_i \kappa(\mathbf{s} - \mathbf{s}_i), \qquad (1)$$

where $\circledast$ is the standard convolution operator, δ is Dirac's delta function, and $\kappa(\mathbf{s} - \mathbf{s}_i)$ is the convolution kernel centered at $\mathbf{s}_i$. Equation (1) is sampled on a regular grid, and the resulting feature map is replicated to form a three-channel image, which can then be fed into the VGG-19 network.

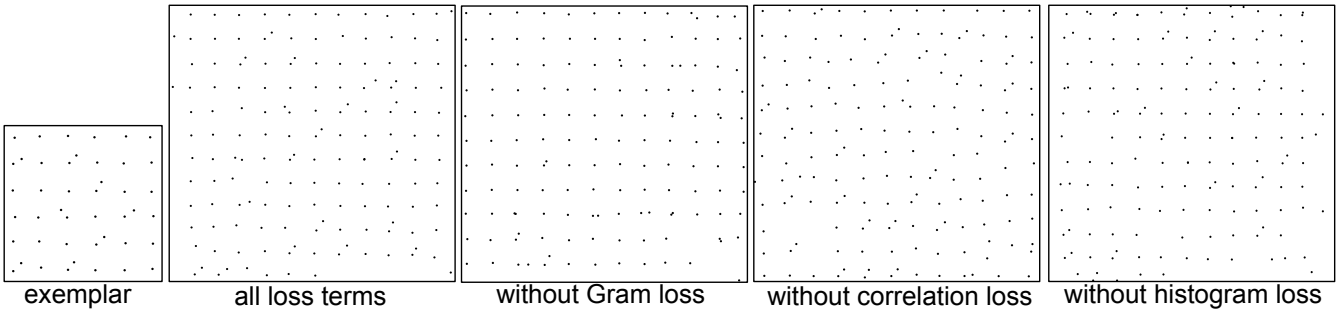


Figure 3: *Ablation study. The results shown above are generated with all defined loss terms (second from left), without Gram loss, without correlation loss, and without histogram loss (right).*

We found that satisfactory results are obtained by using a standard Gaussian kernel, which is defined as:

$$\kappa(\mathbf{s},\mathbf{d}_0,\sigma) = e^{-\frac{\|\mathbf{s}-\mathbf{d}_0\|^2}{2\sigma^2}}, \tag{2}$$

where σ is the standard deviation of the Gaussian filter and $\mathbf{d}_0$ is the filter center. If $a_i = 1$, Equation (1) performs density estimation of the point pattern with the Gaussian kernel κ. If $0 < a_i < 1$, the convolution performs weighted density estimation, where each point might contribute differently.

In principle, more information could be extracted from the pattern using multiple filters, rather than a single one, or by learning the kernel(s), instead of using a Gaussian. However, we did not find that using more than one filter brought forth a perceivable improvement. Furthermore, we opted against learning the kernel, as it has been shown in [UBGB17] that for neural texture synthesis, even random kernels can produce similar results to those obtained with the learned kernels of VGG.

3.2. Deep Texture Model

Gram matrices [GEB15], deep correlation matrices [SC17] and histograms [RWB17] computed from deep features together encode the local and global statistics of a texture. In this work, we adopt these statistics for point pattern synthesis. Computing these statistics involves forwarding the exemplar $\{\mathbf{p}_i\}_{i=1}^n$ and the synthesized pattern $\{\widetilde{\mathbf{p}}_i\}_{i=1}^N$ to the pretrained neural net (Figure 2) and computing the activations. We denote the t-th feature map at layer l by f^{tl}, and its vectorized version by $\vec{f}^{tl}$. The number of feature maps (channels) is denoted by n_l.

The inter-feature correlations between feature maps t_1 and t_2 in layer l are encoded by an $n_l \times n_l$ Gram matrix G^l (covariances of feature vectors) [GEB15], defined as:

$$G^l_{t_1t_2} = \frac{1}{n_l Q^l M^l}\langle \vec{f}^{t_1 l}, \vec{f}^{t_2 l}\rangle, \tag{3}$$

where $Q^l \times M^l$ is the spatial size of the feature maps in layer l. The Gram loss in layer l is then defined as:

$$L^l_{gram} = \sum_{t_1,t_2}(G^l_{t_1t_2} - \widetilde{G}^l_{t_1t_2})^2. \tag{4}$$

The deep correlation matrix, which encodes intra-feature correlations, is defined as:

$$C^{tl}_{ab} = \sum_{q,m} w_{ab} f^{tl}_{qm} f^{tl}_{q-a,m-b}, \tag{5}$$

where $a \in [-Q^l, Q^l]$ and $b \in [-M^l, M^l]$. The feature map is padded with zero when $q-a$, $m-b$ fall outside of its border. The normalization factor $w_{ab} = 1/[(Q-|a|)(M-|b|)]$.

The deep correlation loss [SC17] at layer l is defined as:

$$L^l_{corr} = \frac{1}{4}\sum_{a,b,t}(C^{tl}_{ab} - \widetilde{C}^{tl}_{ab})^2. \tag{6}$$

Finally, the histogram loss [RWB17] is defined as:

$$L^l_{hist} = \sum(\widetilde{f}^l - R_{\widetilde{f}^l \to f^l}(\widetilde{f}^l))^2, \tag{7}$$

where $R_{\widetilde{f}^l \to f^l}$ is the histogram matching function from features $\widetilde{f}^l$ to f^l [GW*02]. We have observed that the function R has zero gradient almost everywhere. To compute the gradient of this loss for backpropagation, R can be effectively treated as a constant for the gradient operator [RWB17].

The full optimization objective is defined as a weighted sum:

$$L = w_{gram}\sum_{l\in L_{gram}} L^l_{gram} + w_{corr}\sum_{l\in L_{corr}} L^l_{corr} + w_{hist}\sum_{l\in L_{hist}} L^l_{hist}, \tag{8}$$

where w_{gram}, w_{corr} and w_{hist} are weights of Gram, correlation and histogram losses, respectively. These weights and the layers whose features are used to compute the different loss terms are reported in Table 1. The same weights and set of layers were used in all of our experiments. The ablation study in Figure 3 shows the necessity of these three loss terms. With all loss terms, the synthesis result keeps the global grid structure as well as local distributions within the exemplar. Without Gram loss, the result still keeps the global structure but loses some local distributions; without correlation loss, the synthesized global structure is distorted; without histogram loss, the result also loses some local distributions.

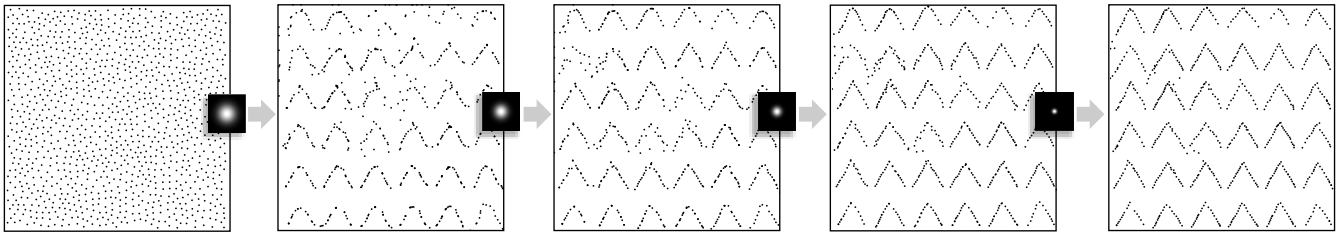
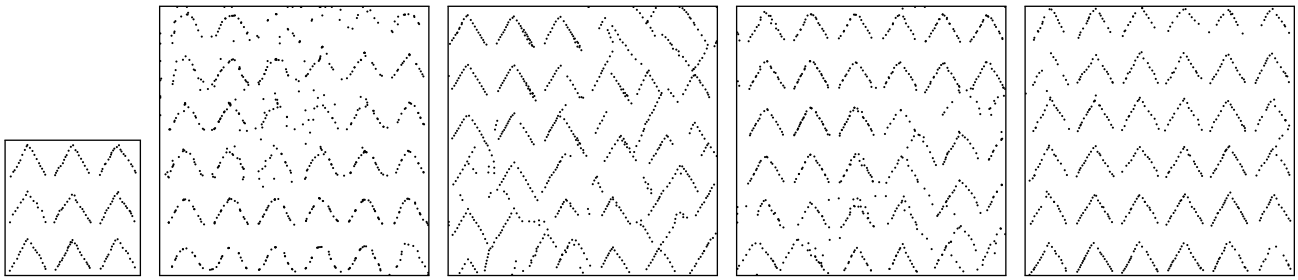


Figure 4: *Hierarchical optimization process. The point pattern used as the initial guess and the filter used in the irregular convolution layer at each of the four stages of the optimization process; the final result is shown on the right.*



(a) *exemplar* **(b)** *with filter size* $\sigma = 2 \cdot \bar{d}$ **(c)** *with filter size* $\sigma = 0.5 \cdot \bar{d}$ **(d)** *with filter size* $\sigma = \bar{d}$ **(e)** *hierarchical optimization*

Figure 5: *Importance of hierarchical optimization. Given an exemplar (a), optimization using a single kernel size fails to deliver satisfactory results (b)–(d). Our hierarchical optimization, which uses a gradually decreasing kernel size, yields a significantly better result (e).* $\bar{d}$ *is the average nearest distances between points within the exemplar.*

Table 1: Parameters in our experiments: *pool* represents pooling layers in VGG net, and the subscript represents its order, please refer to [SZ14] for notations; $conv_{ir}$ represents our prepended irregular convolution layer.

	w_{gram}	w_{corr}	w_{hist}	L_{gram}	L_{corr}	L_{hist}
Hard opt.	1	100	0	$pool_1$ $pool_2$	$pool_2$	$conv_{ir}$
Soft opt.			10	$pool_3$ $pool_4$		$pool_2$

4. Optimization

4.1. Hard point optimization

At the first phase of the optimization, we optimize the spatial positions $\{\widetilde{\mathbf{s}}_i\}_{i=1}^N$ of the points, while keeping their number fixed. The irregular convolution layer produces a single-channel activation map with confidence attributes a_i and $\widetilde{a}_i{}^j$ fixed to 1. The resulting activation amounts to density estimation with a Gaussian kernel. The default weights of different loss terms in Equation (8) are listed in Table 1.

Initialization. Differently from previous algorithms [RÖM*15; MWT11], our algorithm does not require delicate initialization. We use N uniformly distributed points with a blue noise spectrum, as shown in Figure 4. In our experiments, $N = n \cdot c^2$, where n is the number of points within the input exemplar, while c is the upscaling rate of the desired synthesized pattern. For simplicity, we only consider patterns constrained within a square domain in this paper. Other initializations, such as a white noise pattern, are also feasible but, in our experience, are slower to converge.

Hierarchical optimization. A straightforward approach is to optimize the point positions using an irregular convolution layer with a Gaussian filter of a fixed single size, throughout the optimization. However, we have found that it is difficult to achieve satisfactory results using a fixed size filter. As demonstrated in Figure 5, using a large filter results in poor reproduction of local structures, due to loss of spatial information when the pattern is convolved (Figure 5(b)). On the other hand, a small filter (Figure 5(c)) is unable to faithfully capture the global structure of the input pattern. Using a single medium size also fails to provide satisfactory results (Figure 5(d)).

To address the above issue, we adopt a hierarchical, coarse-to-fine optimization strategy for setting the filter size of the irregular convolution layer. Larger filter sizes are used first, allowing each point to affect a larger region. Since the gradients of the loss with respect to point positions are summed from the influenced regions, this reduces the chances of gradients to vanish. In other words, larger filters reduce the number of local optima of the objective function. The optimization is executed four times, while the size of the Gaussian filter is linearly decreased from to σ_1 to σ_2. Section 4.4 explains how these sizes are determined. The filter size is reduced every time after an optimization has converged. The subsequent optimization is initialized with the result taken from the previous one. This hierarchical optimization process is demonstrated in Figure 4. As seen in Figure 5(e) and Figure 4, the results achieved by our hierarchical strategy succeed in preserving both coarse and fine scale structures.

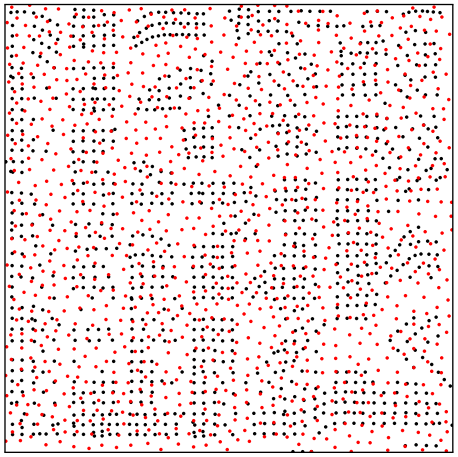


Figure 6: *Initialization for soft point optimization. Points from the hard point optimization phase are plotted in black; Extra uniformly distributed points are added, plotted in red.*

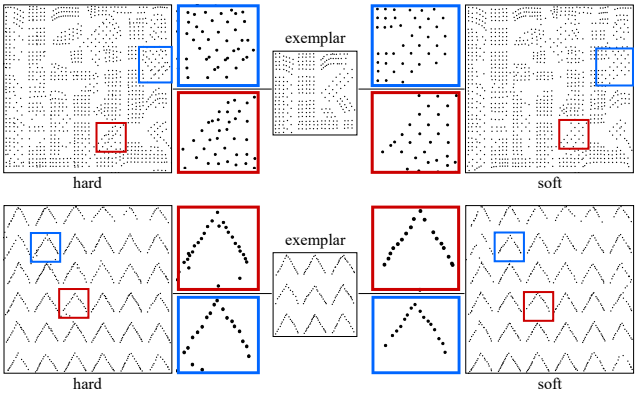


Figure 7: *Two examples that demonstrate the effect of soft point optimization. In each row the input exemplar is shown in the middle, and the patterns produced by hard and soft point optimizations are shown on the left and right, respectively.*

4.2. Soft point optimization

Despite our use of a hierarchical optimization strategy, it may still converge to a pattern where some local regions have too few points, while other regions have too many. However, the points of the pattern might be trapped in local minima, and thus unable to move. To overcome this situation, we add new points to the synthesized pattern, while augmenting all points (original and new) with an existence attribute $\widetilde{a}_i$, which we refer to as *confidence*. We then continue to optimize both the confidence $\widetilde{a}_i$ and the positions $\widetilde{\mathbf{s}}_i$ of the points. Points whose confidence falls below a certain threshold are removed from the pattern.

Initialization. The hard point optimization (Section 4.1) produces a pattern which is globally similar to the exemplar. The idea is to refine the pattern locally by adding and removing points. The optimization is able to remove redundant points by decreasing the confidence attributes of the original points. To add points, because the optimization requires a fixed set of optimized variables, a set of new (extra) points is added, whose number is equal to half that of the original points. The extra points are uniformly distributed across the domain, as shown in Figure 6. The original points confidence is initialized to 1, while that of the extra points to 0.

Optimization. In order to make the joint optimization of locations and confidence attributes stable, the optimization proceeds in two stages. During the first stage the locations of the original points are fixed, and only their confidences are updated. For the new points, both their locations and confidences are optimized. At this stage, the standard deviation of the Gaussian kernel is set to medium ($\frac{\sigma_1+\sigma_2}{2}$). In the second stage, all of the points are allowed to move, and the standard deviation used is small (σ_2). Still, the original points are updated using a smaller step size than the new points. More details about optimization parameters are provided in Section 4.4.

Decimation. For each point, we now have an optimized confidence value $\widetilde{a}_i$. In each iteration, the point that has the lowest confidence is removed. However, we ensure the total confidence sum remains unchanged. Thus, when a point is removed, its confidence is diffused among its k nearest neighbors, with diffusion coefficient proportional to the inverse of the squared distance. The process continues until all the points have confidence that are higher than a threshold th^e. In our experiments, $k = 7$ and $th^e = 0.8$.

Figure 7 demonstrates the effect of soft point optimization. Soft point optimization and the following sparsification step can remove points that "get stuck" in the wrong position due to local minima. It is particularly effective when synthesizing patterns with local regularity, in which such "outliers" are easily noticed.

We have also experimented with skipping the hard optimization phase, and applying soft optimization directly on the initial blue noise distributed pattern. However, we found such optimization process to be less stable, usually failing to converge to a satisfactory result. We attribute this to the soft point optimization being too under-constrained.

4.3. Importance of end-to-end optimization

Our approach synthesizes a point pattern by performing end-to-end optimization, as this avoids the accumulation of error that one might encounter in a heuristic step-by-step approach.

A natural step-by-step approach would be to (i) convert the point distribution into a density map; (ii) apply a general-purpose example-based texture synthesis method to synthesize a larger density map; and (iii) convert the synthesized density map back to a point pattern.

We have experimented with three general-purpose example-based texture synthesis methods. The neural texture optimization method of Sendik and Cohen-Or [SC17], the GAN-based approach of Zhou et al. [ZZB*18] and the graph-cut textures [KSE*03]. In particular, because it is easy for pixel-space method to synthesize high-resolution textures, we synthesize a comparatively larger textures using graph-cut [KSE*03] to reduce information loss during point pattern to image conversion, as will be discussed in the

next paragraph. As may be seen from Figure 8, neural texture optimization generates a noisy density map, while the GAN-based approach [ZZB*18] is able to generate a clean density map, but requires a long training time for each input exemplar. The graph-cut textures cannot preserve the structures within the exemplar. The reason is that matching neighborhoods and computing stitching seams are highly ambiguous for point pattern images, which differ in that respect from natural textures.

We have also experimented with two methods for converting the synthesized density map into a point pattern. The first method is to sample points according to the density, which we did using blue noise sampling [Llo82]. The results of this method are shown in the middle column of Figure 8. Since blue noise sampling assumes globally uniform local pairwise statistics, the results fail to resemble the input pattern. Other studies also show that it's challenging to produce locally regular patterns using existing sampling algorithms, as demonstrated in [ÖG12].

The second method is to extract points from the density map by detecting local density extrema. The resulting patterns are shown in the rightmost column of Figure 8. It is challenging to make such a detection-based algorithm robust, as the parameters must be tuned for different patterns. Even after such tuning, the results suffer from clumping artifacts.

Although it is relatively easy to extract points from high-resolution point images, as shown at the bottom right of Figure 8, the quality is still low due to the poorly generated point image.

In contrast, the results of our end-to-end optimization, shown in the top portion of Figure 8, succeed in capturing and faithfully reproducing the structures resent in the input pattern.

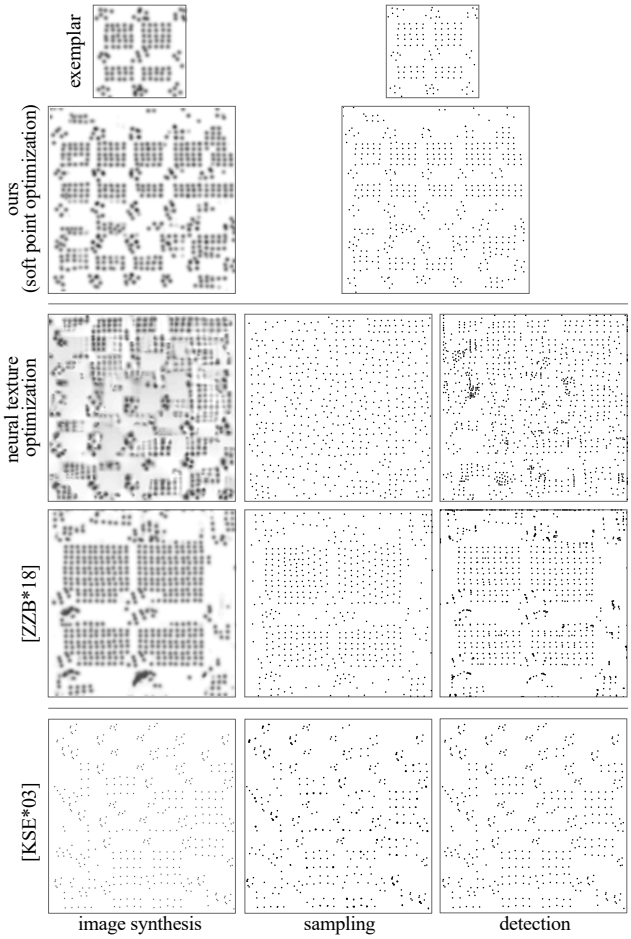


Figure 8: *End-to-end optimization vs. a step-by-step approach. The input exemplar and our results are shown in the above first and second rows, respectively. The two rows in the middle show (in the left column) the comparatively lower resolution density maps generated using general-purpose example-based neural texture synthesis. The last row shows the results by graph-cut textures, which is different from neural algorithms and can generate a high-resolution image as shown at the bottom left. The middle column shows blue-noise sampling of the resulting density map, while the rightmost column shows points extracted by local maxima detection.*

4.4. Implementation Details

The starting size σ_1 and the ending size σ_2 of the Gaussian filter are two hyper-parameters in our algorithm, which are ideally determined by the local point interactions and the global structure size of the exemplar. The larger the global structures, the larger the starting filter size should be. These two parameters play a role similar to that of the patch-size or in traditional texture synthesis algorithms [WLKT09]. We select σ_1 and σ_2 according to the average nearest distances $\bar{d}$ between points within an exemplar: $\sigma_1 = \bar{d}r/c_1$ and $\sigma_2 = \bar{d}r/c_2$, where c_1 and c_2 are user-specified coefficients. The resolution r of sampling grids is selected as $round(2/\bar{d})$, but we ensure r to be within [128, 512] pixels. The corresponding $\widetilde{r}$ equals $round(A \cdot r)$, where A is the upscaling rate. The spatial positions of synthesized points $\widetilde{\mathbf{s}}_i$ are within $[0, A]$.

Hard point optimization. We use the Adam optimization method [KB14], and the learning rates in the four optimization stages are 0.005, 0.002, 0.002 and 0.002, respectively. Each coordinate of $\widetilde{\mathbf{s}}_i$ is constrained to lie within $[0, A]$ by clamping the coordinates after every iteration. The optimization continues until the decrease of the loss during the most recent 100 iterations is smaller than 0.5% of the total loss decrease from the beginning of the optimization.

Soft point optimization. We constrain $\widetilde{a}_i$ and each coordinate of $\widetilde{\mathbf{s}}_i$ to be within $[0, 1]$ and $[0, A]$ by clamping them after every iteration. The Adam method is also used for optimization [KB14]. The optimization step size of original and extra point energy is always 0.02. In the first optimization stage, the step size of original and extra point position is 0 and 0.01. In the second optimization stage, they are 0.0005 and 0.002, respectively. The convergence criterion is the same as that in the hard point optimization.

5. Results and Applications

Our unoptimized implementation requires from two to ten minutes to synthesize a point pattern, where the number of points varies

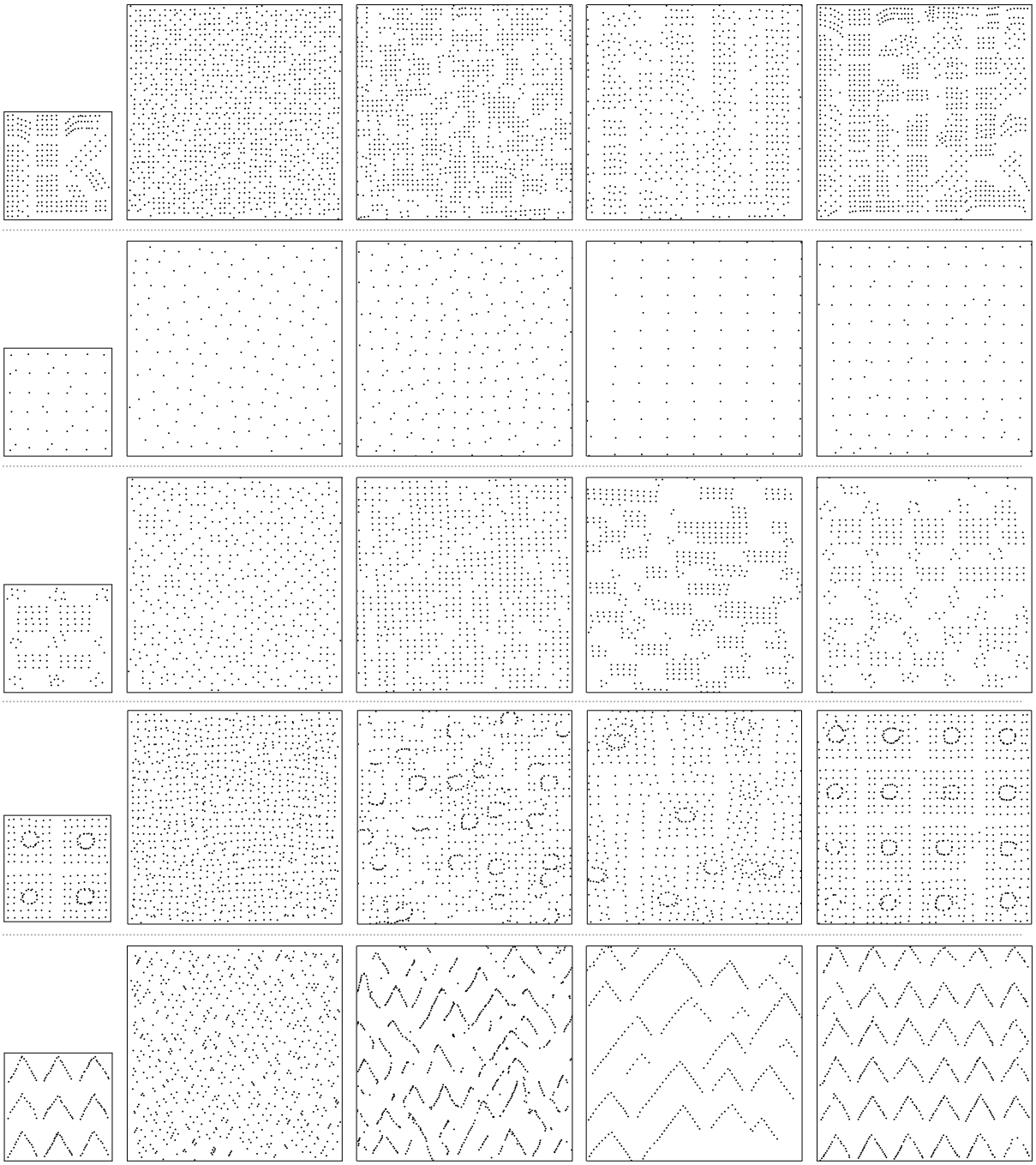


Figure 9: *Point pattern synthesis results. We compare our method with three example-based point patter synthesis methods. From left to right, we show the exemplar, the results by [ZHWW12; MWT11; RÖM*15] and our method. From top to bottom, we use the parameters* $c_1 = 1,1,1,1,0.5$ *and* $c_2 = 4,2,3,2,2$*, respectively.*

from three hundred to two thousand points, with a NVIDIA Quadro P5000. Our current development is unparalleled regarding to the irregular convolution layer, and the computational complexity is about $O(n)$, where n is the number of points. The computational burden mainly lies on the irregular convolution layer, which may be hopefully accelerated in the near future.

Table 2: Quantitative uniformity comparison of example-based methods via synthesizing Poisson-disk distribution.

ours	[ZHWW12]	[MWT11]	[RÖM*15]
0.65	0.68	0.55	0.63

5.1. Qualitative Evaluation

In Figure 9, we compare our algorithm with three existing example-based point synthesis methods [MWT11; ZHWW12; RÖM*15].

Our results preserve both local point interactions and global structures. The method of Zhou et al. [ZHWW12], which is based on matching global pairwise statistics, cannot preserve any structures, as expected. The method of Ma et al. [MWT11] is based on matching neighborhoods. This method can preserve some local structures, but discards global ones. The local point interactions also cannot be perceived. The least squares solver used in their optimization has a blurring effect [KEBK05; MWT11]. While a small degree of blurring of color textures may be acceptable to the human visual system [KEBK05], averaging of point positions may result in a significantly distorted point pattern.

Roveri et al. [RÖM*15] also use a least-square solver. Also, differently from our soft point optimization process that optimizes the existence of points based on our deep texture model, their method controls the local number of points using heuristics that do not always hold. This sometimes also produces distorted local point patterns (e.g., the second row of Figure 9).

5.2. Quantitative Evaluation

To quantitatively compare our method to the existing example-based methods, we apply them to synthesize Poisson-disk distribution and evaluate the synthesis quality by measuring the relative radius $\rho = r_{min}/r_{max}$ [LD08], where r_{min} is the minimum distance between each two points and r_{max} is the inter-sample distance computed from the maximum packing of these samples. We apply dart throwing algorithm to create an exemplar with $\rho \approx 0.7$. Table 2 shows the relative radius values of the synthesized results. Our method performs better than the neighborhood-searching based algorithms, such as [MWT11] that may easily result in broken local patterns. This also conforms to the conclusion draw from [RÖM*15]. Zhou et al. [ZHWW12] preserve a better spatial uniformity than us, since this method is specifically designed to match the local pattern statistics. Our algorithm excels in capturing both the local and non-local structures, as shown in Figure 9.

5.3. Object Placement

The points patterns synthesized by our method may be used to simulate repetitive phenomena in the real world, in order to enhance visual complexity of virtual scenes. Figures 1, 10 and 11 show some examples of 2D element and 3D object placement. These scenes contain both large structures as well as local variations. For example, in Figure 1, the geese form large arrowhead formations, but each formation exhibits small random perturbations. We can observe that our synthesis method preserves both. Similar phenomena may be observed in most of the examples in Figure 11.

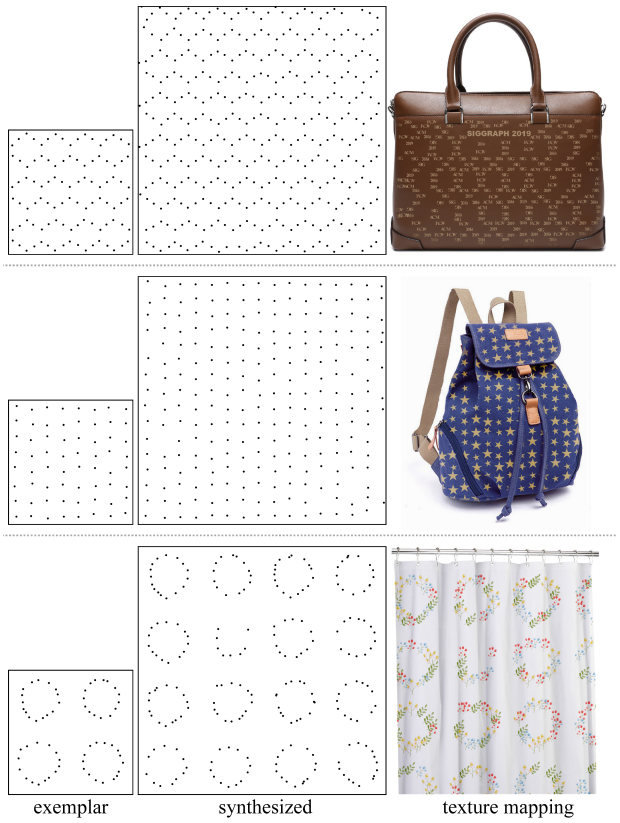


Figure 10: *Texture element placement. Here a synthesized point pattern is used to generate 2D element textures, which can then be mapped onto an object surface. The placed elements are randomly chosen from a set of pattern candidates. From top to bottom, we use the parameters $c_1 = 2, 2, 0.5$ and $c_2 = 4, 4, 2$, respectively.*

Figure 12 demonstrates some point pattern synthesis results by exemplars extracted from real images. We may generate many such realistic 3D scenarios inspired by 2D images, which can steer various applications in game design, virtual or augmented reality.

5.4. Extension to Attributed Points

In our work, we append each point with a confidence attribute to support soft point optimization (Section 4.2). Thus, it is a natural extension of our approach to consider other types of attributes, such as the species or the age of each tree in a forest. We distinguish between continuous real-valued attributes, such as age or size, and discrete attributes, such as species. Continuous attributes are represented by a single dimension of a_i. Discrete attributes are encoded using a one-hot vector. As a special case, a binary attribute is represented using a single binary coordinate.

Since point attributes are originally provided in attribute-specific units, we first transform these attributes (using scale and bias) to ensure that the resulting activations lie within an appropriate numerical range ([0, 1] by default in our work). The actual attribute value in the original units can be obtained by an inverse transformation after the synthesis.

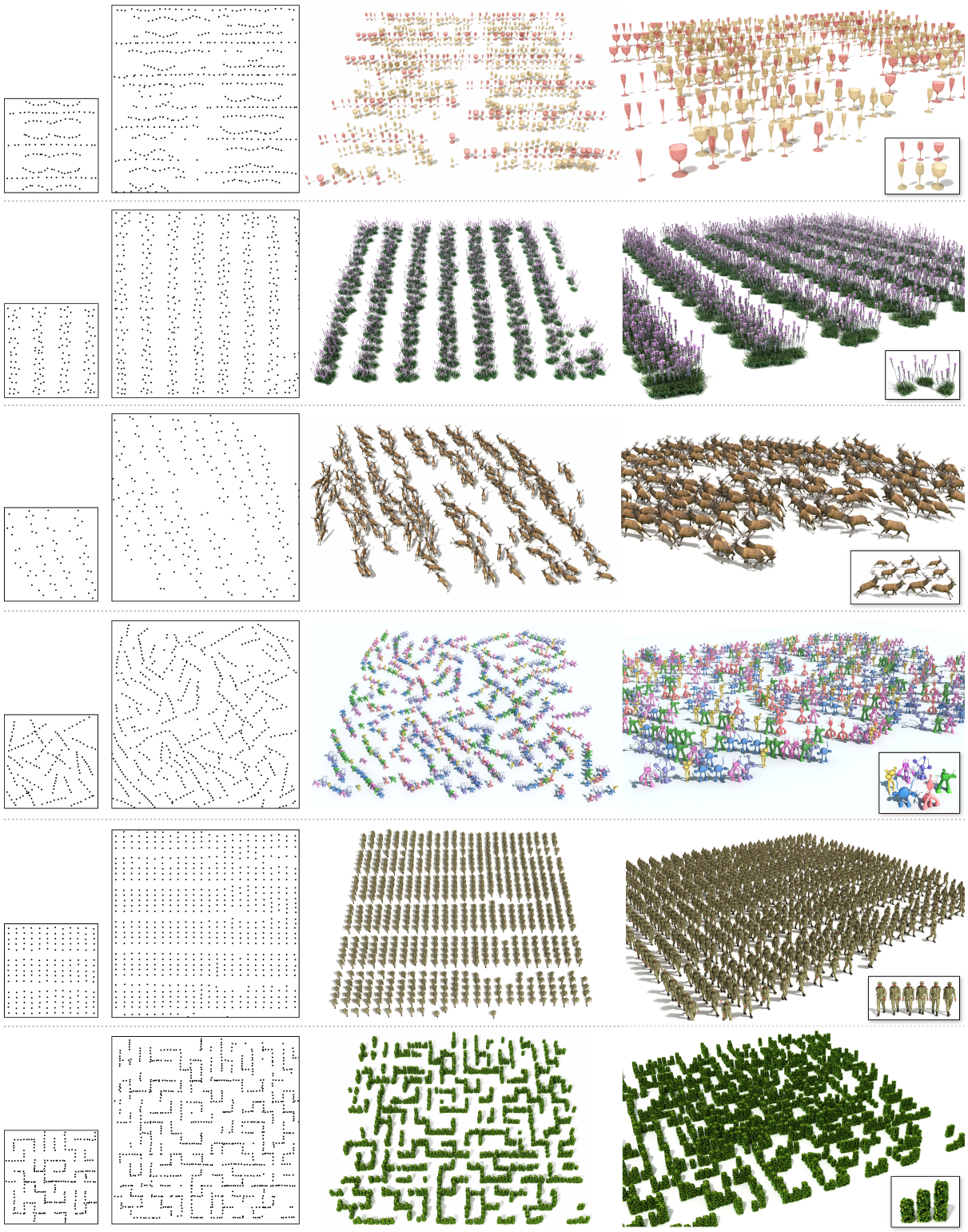


Figure 11: *3D object placement. Our synthesized patterns can be used to generate 3D object placement. Note that the synthesized patterns successfully preserve large structures and small local variations in the exemplars. From top to bottom, the parameters used are* $c_1 = 1,1,2,1,2,1$ *and* $c_2 = 3,4,4,4,4,4$*, respectively.*

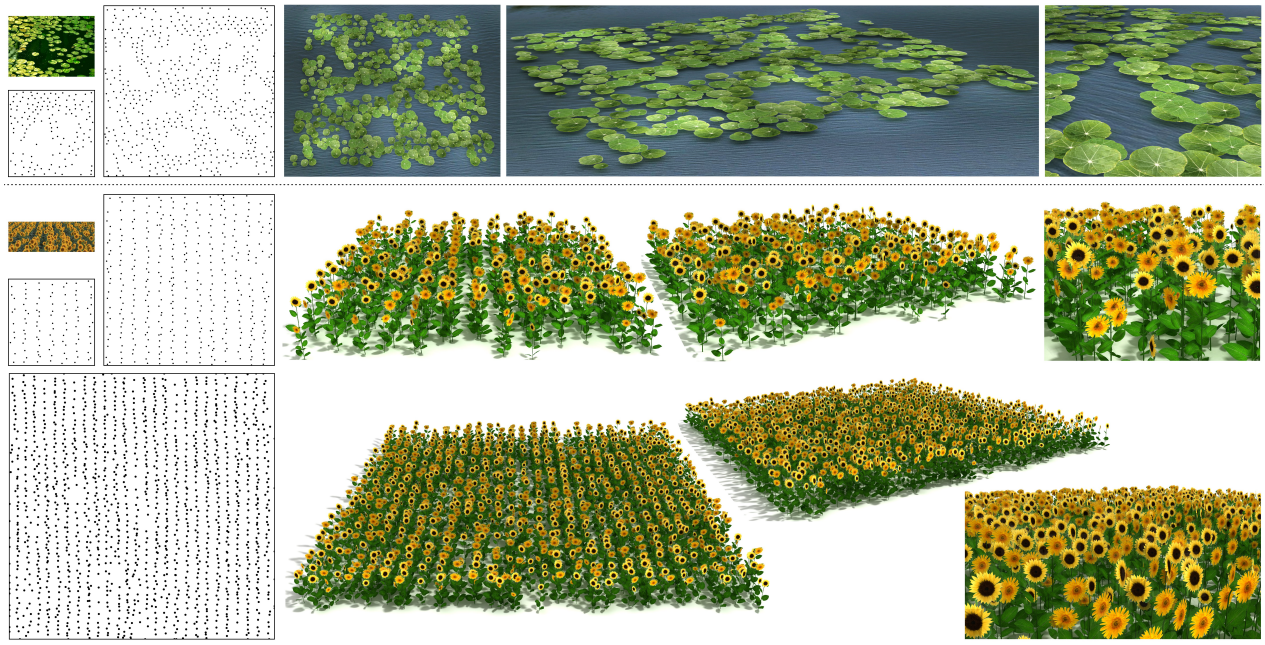


Figure 12: *Natural point pattern synthesis. The input exemplars can be directly extracted from real images, leading to larger realistic 3D virtual scenes. The sunflower pattern has been extended twice while the structure is still well preserved. Parameters used:* $c_1 = 2$ *and* $c_2 = 4$.

Figure 13: *Application: rural layout. This input pattern has two classes of points. Red points indicate regularly distributed houses, while blue points indicate stochastically distributed trees. Parameters used:* $c_1 = 1$ *and* $c_2 = 3$.

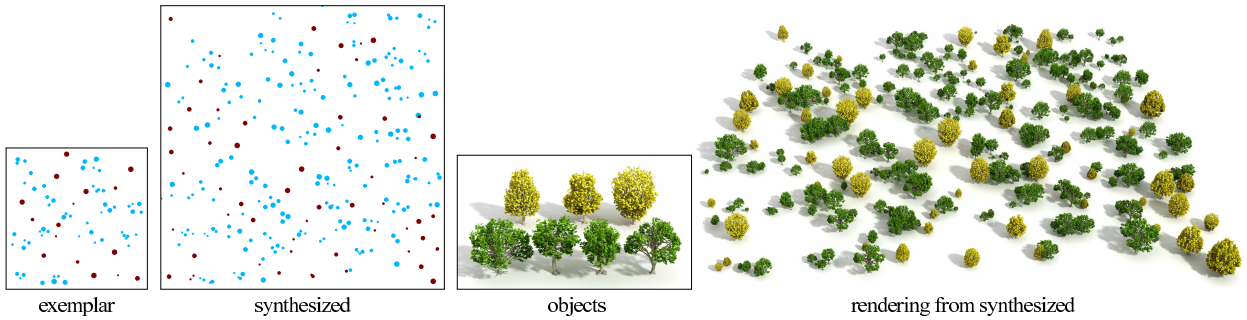


Figure 14: *Application: tree grove. The input point pattern has two attributes, one is a continuous attribute representing the size of a tree, while the other is discrete class attribute indicating the tree type. Parameters used:* $c_1 = 1$ *and* $c_2 = 4$.

Figures 13 and 14 show two examples of attributed point pat-

Figure 15: *Graph synthesis. Our method can also be used to synthesize graphs by adopting our soft point optimization strategy from points to edges. Graphs can be used to model man-made roads, regions or natural rivers. Parameters used:* $c_1 = 1$ *and* $c_2 = 2$.

tern synthesis results. The first pattern (Figure 13) models a rural layout. The pattern has both regular and irregular spatial patterns and represented by points from two classes. One class (red points) represents regularly distributed buildings, while the second (blue points) represents irregularly distributed trees. The synthesized patterns roughly preserve such joint spatial-attribute statistics.

The second pattern (Figure 14) models a tree grove. Blue and red points represent beach and oak trees, respectively. Point size represents the tree age. The oaks that need light keep segregated characteristics but the beeches that can tolerate shade have mingling characteristics. Our synthesized results reproduce these characteristics, present in the exemplar.

However, we found that it's not easy to optimize attributes that have more than three dimensions, as the texture synthesis algorithm we built upon is inherently noisy. High-dimensional attribute synthesis is error-prone, and thus we leave it for future work.

5.5. Extension to Graph Synthesis

Given a graph structure consisting of nodes and edges, our approach may be extended to synthesize a larger graph with similar structures. We first generate a set of nodes by using point pattern synthesis, followed by edge generation. The optimization of the edges is a difficult combinatorial optimization problem. Our solution is to adapt the soft point optimization strategy to generate the graph edges. We connect each node to its 7-nearest nodes, and randomly assign an initial confidence attribute between [0, 1] to each of the resulting edges.

We then optimize the confidence attributes of the edges by extending Equation (1) while keeping it differentiable, and finally remove the edges with small confidence by thresholding. To synthesize practically, we additionally constrain the synthesis by preventing the angle between each pair of intersection edges being smaller than the minimum angle within the given exemplar.

Figure 15 shows some graph synthesis results. The results demonstrate that the synthesized and exemplar graphs have similar local and global structures. Our graph synthesis can thus be used to synthesize layouts of roads, regions, rivers, etc.

Note that these graphs cannot be directly synthesized using traditional texture synthesis algorithms, as they use bitmaps and do not constrain the synthesized results to have a graph structure, i.e., easily resulting in broken graphs.

6. Conclusions and Future Work

In this paper, we propose an end-to-end optimization based approach for example-based point pattern synthesis. By prepending an irregular convolution layer to a pretrained CNN, we are able to use loss functions from neural texture synthesis to directly optimize point positions. The proposed algorithm is simple, which may be considered as a virtue, and is able to achieve state-of-the-art results that preserve large and small-scale structures that might be present in an input exemplar. Our algorithm can also be extended to the synthesis of point patterns associated with simple attributes, as well as to the synthesis of structured graphs.

Methodologically, our method can be further improved by considering spatially varying kernel sizes. Larger kernel sizes can preserve large structure, while small ones are sensitive to local point interactions. Our method is inspired by previous texture synthesis algorithms. However, the current formulation does not scale well to point distributions with high-dimensional attributes or complex attribute statistics. This is an interesting direction for future work.

Our algorithm achieves state-of-the-art point synthesis quality by preserving local and global structures. However, the current approach is relatively slow due to the optimization with a neural network. In future work, we plan to accelerate the synthesis by training feed-forward networks. Also here, we only show results within a square domain for simplicity. It is worth to extend the pattern synthesis to a non-squared domain of a complex shape by defining the calculation of feature statistics on that domain.

Our method deals with stationary point distributions. We are interested in developing algorithms for non-stationary point distribution synthesis, similarly to non-stationary texture synthesis [ZZB*18]. We are also interested in improving the efficiency of our algorithm by training a feed-forward point pattern synthesis model.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was supported in parts by NSFC (61761146002), National 973 Program (2015CB352501), Guangdong Science and Technology Program (2015A030312015), Shenzhen Innovation Program (KQJSCX20170727101233642), LHTD (20170003), ISF (2366/16), and the National Engineering Laboratory for Big Data System Computing Technology.

References

[ADMG18] ACHLIOPTAS, PANOS, DIAMANTI, OLGA, MITLIAGKAS, IOANNIS, and GUIBAS, LEONIDAS. "Learning Representations and Generative Models for 3D Point Clouds". *International Conference on Machine Learning*. 2018, 40–49 3.

[AML18] ATZMON, MATAN, MARON, HAGGAI, and LIPMAN, YARON. "Point Convolutional Neural Networks by Extension Operators". *arXiv preprint arXiv:1803.10091* (2018) 3.

[BJV17] BERGMANN, URS, JETCHEV, NIKOLAY, and VOLLGRAF, ROLAND. "Learning texture manifolds with the periodic spatial GAN". *arXiv preprint arXiv:1705.06566* (2017) 3.

[BZ17] BARNES, CONNELLY and ZHANG, FANG-LUE. "A survey of the state-of-the-art in patch-based synthesis". *Computational Visual Media* 3.1 (2017), 3–20 3.

[Coo86] COOK, ROBERT L. "Stochastic Sampling in Computer Graphics". *ACM Trans. Graph.* 5.1 (Jan. 1986), 51–72. ISSN: 0730-0301. DOI: 10.1145/7529.8927. URL: http://doi.acm.org/10.1145/7529.8927 1, 2.

[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: ACM, 1998, 275–286. ISBN: 0-89791-999-8. DOI: 10.1145/280814.280898. URL: http://doi.acm.org/10.1145/280814.280898 1.

[DM12] DAMELIN, STEVEN B and MILLER JR, WILLARD. *The mathematics of signal processing*. Vol. 48. Cambridge University Press, 2012 3.

[EVC*15] EMILIEN, ARNAUD, VIMONT, ULYSSE, CANI, MARIE-PAULE, et al. "WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds". *ACM Trans. Graph.* 34.4 (July 2015), 106:1–106:11. ISSN: 0730-0301. DOI: 10.1145/2766975. URL: http://doi.acm.org/10.1145/2766975 1.

[Fat11] FATTAL, RAANAN. "Blue-noise Point Sampling Using Kernel Density Model". *ACM Trans. Graph.* 30.4 (July 2011), 48:1–48:12. ISSN: 0730-0301. DOI: 10.1145/2010324.1964943. URL: http://doi.acm.org/10.1145/2010324.1964943 1, 2.

[GEB15] GATYS, LEON, ECKER, ALEXANDER S, and BETHGE, MATTHIAS. "Texture synthesis using convolutional neural networks". *Advances in Neural Information Processing Systems*. 2015, 262–270 2–4.

[GW*02] GONZALEZ, RAFAEL C, WOODS, RICHARD E, et al. *Digital image processing*. 2002 4.

[HLT*09] HURTUT, THOMAS, LANDES, P-E, THOLLOT, JOËLLE, et al. "Appearance-guided synthesis of element arrangements by example". *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering*. ACM. 2009, 51–60 2.

[HRV*18] HERMOSILLA, PEDRO, RITSCHEL, TOBIAS, VÁZQUEZ, PERE-PAU, et al. "Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds". *arXiv preprint arXiv:1806.01759* (2018) 3.

[HSD13] HECK, DANIEL, SCHLÖMER, THOMAS, and DEUSSEN, OLIVER. "Blue Noise Sampling with Controlled Aliasing". *ACM Trans. Graph.* 32.3 (July 2013), 25:1–25:12. ISSN: 0730-0301. DOI: 10.1145/2487228.2487233. URL: http://doi.acm.org/10.1145/2487228.2487233 1, 2.

[IMIM08] IJIRI, TAKASHI, MECH, RADOMIR, IGARASHI, TAKEO, and MILLER, GAVIN. "An Example-based Procedural System for Element Arrangement". *Computer Graphics Forum*. Vol. 27. 2. Wiley Online Library. 2008, 429–436 3.

[IPSS08] ILLIAN, JANINE, PENTTINEN, ANTTI, STOYAN, HELGA, and STOYAN, DIETRICH. *Statistical analysis and modelling of spatial point patterns*. Vol. 70. John Wiley & Sons, 2008 2.

[KB14] KINGMA, DIEDERIK P and BA, JIMMY. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 7.

[KCDL06] KOPF, JOHANNES, COHEN-OR, DANIEL, DEUSSEN, OLIVER, and LISCHINSKI, DANI. "Recursive Wang Tiles for Real-time Blue Noise". *ACM SIGGRAPH 2006 Papers*. SIGGRAPH '06. Boston, Massachusetts: ACM, 2006, 509–518. ISBN: 1-59593-364-6. DOI: 10.1145/1179352.1141916. URL: http://doi.acm.org/10.1145/1179352.1141916 1, 2.

[KEBK05] KWATRA, VIVEK, ESSA, IRFAN, BOBICK, AARON, and KWATRA, NIPUN. "Texture Optimization for Example-based Synthesis". *ACM Transactions on Graphics, SIGGRAPH 2005* (Aug. 2005) 3, 9.

[KS17] KARTH, ISAAC and SMITH, ADAM M. "WaveFunctionCollapse is Constraint Solving in the Wild". *Proceedings of the 12th International Conference on the Foundations of Digital Games*. FDG '17. Hyannis, Massachusetts: ACM, 2017, 68:1–68:10. ISBN: 978-1-4503-5319-9. DOI: 10.1145/3102071.3110566. URL: http://doi.acm.org/10.1145/3102071.3110566 3.

[KSE*03] KWATRA, VIVEK, SCHDL, ARNO, ESSA, IRFAN, et al. "Graphcut Textures: Image and Video Synthesis Using Graph Cuts". *ACM Transactions on Graphics, SIGGRAPH 2003* 22.3 (July 2003), 277–286 6.

[LBSC18] LI, YANGYAN, BU, RUI, SUN, MINGCHAO, and CHEN, BAOQUAN. "PointCNN". *arXiv preprint arXiv:1801.07791* (2018) 3.

[LD05] LAGAE, ARES and DUTRÉ, PHILIP. "A Procedural Object Distribution Function". *ACM Trans. Graph.* 24.4 (Oct. 2005), 1442–1461. ISSN: 0730-0301. DOI: 10.1145/1095878.1095888. URL: http://doi.acm.org/10.1145/1095878.1095888 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 9.

[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. URL: http://doi.acm.org/10.1145/74334.74360 1.

[LFY*17] LI, YIJUN, FANG, CHEN, YANG, JIMEI, et al. "Diversified Texture Synthesis with Feed-forward Networks". *IEEE Conference on Computer Vision and Pattern Recognition*. 2017 3.

[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. URL: http://dx.doi.org/10.1111/cgf.12152 2.

[Llo82] LLOYD, STUART. "Least squares quantization in PCM". *IEEE transactions on information theory* 28.2 (1982), 129–137 7.

[LSM*18] LEIMKÜHLER, THOMAS, SINGH, GURPRIT, MYSZKOWSKI, KAROL, et al. "End-to-end Sampling Patterns". *arXiv preprint arXiv:1806.06710* (2018) 3.

[LW16] LI, CHUAN and WAND, MICHAEL. "Precomputed real-time texture synthesis with markovian generative adversarial networks". *European Conference on Computer Vision*. Springer. 2016, 702–716 3.

[LWSF10] LI, HONGWEI, WEI, LI-YI, SANDER, PEDRO V., and FU, CHI-WING. "Anisotropic Blue Noise Sampling". *ACM SIGGRAPH Asia 2010 Papers*. SIGGRAPH ASIA '10. Seoul, South Korea: ACM, 2010, 167:1–167:12. ISBN: 978-1-4503-0439-9. DOI: 10.1145/1866158.1866189. URL: http://doi.acm.org/10.1145/1866158.1866189 2.

[LZZ*18] LI, CHUN-LIANG, ZAHEER, MANZIL, ZHANG, YANG, et al. "Point cloud gan". *arXiv preprint arXiv:1810.05795* (2018) 3.

[Mer09] MERRELL, PAUL C. "Model synthesis". (2009) 3.

[MMR*18] MÜLLER, THOMAS, MCWILLIAMS, BRIAN, ROUSSELLE, FABRICE, et al. "Neural importance sampling". *arXiv preprint arXiv:1808.03856* (2018) 3.

[MWLT13] MA, CHONGYANG, WEI, LI-YI, LEFEBVRE, SYLVAIN, and TONG, XIN. "Dynamic Element Textures". *ACM Trans. Graph.* 32.4 (July 2013), 90:1–90:10. ISSN: 0730-0301. DOI: 10.1145/2461912.2461921. URL: http://doi.acm.org/10.1145/2461912.2461921 1, 3.

[MWT11] MA, CHONGYANG, WEI, LI-YI, and TONG, XIN. "Discrete Element Textures". *ACM SIGGRAPH 2011 Papers*. SIGGRAPH '11. Vancouver, British Columbia, Canada: ACM, 2011, 62:1–62:10. ISBN: 978-1-4503-0943-1. DOI: 10.1145/1964921.1964957. URL: http://doi.acm.org/10.1145/1964921.1964957 1, 3, 5, 8, 9.

[ÖG12] ÖZTIRELI, A. CENGIZ and GROSS, MARKUS. "Analysis and Synthesis of Point Distributions Based on Pair Correlation". *ACM Trans. Graph.* 31.6 (Nov. 2012), 170:1–170:10. ISSN: 0730-0301. DOI: 10.1145/2366145.2366189. URL: http://doi.acm.org/10.1145/2366145.2366189 1, 2, 7.

[RÖG17] ROVERI, RICCARDO, ÖZTIRELI, A. CENGIZ, and GROSS, MARKUS. "General Point Sampling with Adaptive Density and Correlations". *Comput. Graph. Forum* 36.2 (May 2017), 107–117. ISSN: 0167-7055. DOI: 10.1111/cgf.13111. URL: https://doi.org/10.1111/cgf.13111 1, 2.

[RÖM*15] ROVERI, RICCARDO, ÖZTIRELI, A CENGIZ, MARTIN, SEBASTIAN, et al. "Example based repetitive structure synthesis". *Computer Graphics Forum*. Vol. 34. 5. Wiley Online Library. 2015, 39–52 3, 5, 8, 9.

[RWB17] RISSER, ERIC, WILMOT, PIERRE, and BARNES, CONNELLY. "Stable and controllable neural texture synthesis and style transfer using histogram losses". *arXiv preprint arXiv:1701.08893* (2017) 4.

[SC17] SENDIK, OMRY and COHEN-OR, DANIEL. "Deep correlations for texture synthesis". *ACM Transactions on Graphics (TOG)* 36.5 (2017), 161 2–4, 6.

[SWL*18] SUN, YONGBIN, WANG, YUE, LIU, ZIWEI, et al. "Pointgrow: Autoregressively learned point cloud generation with self-attention". *arXiv preprint arXiv:1810.05591* (2018) 3.

[SZ14] SIMONYAN, KAREN and ZISSERMAN, ANDREW. "Very deep convolutional networks for large-scale image recognition". *arXiv preprint arXiv:1409.1556* (2014) 3, 5.

[UBGB17] USTYUZHANINOV, I., BRENDEL, W., GATYS, L., and BETHGE, M. "What does it take to generate natural textures?": Apr. 2017. URL: https://openreview.net/forum?id=BJhZeLsxx 4.

[ULVL16] ULYANOV, DMITRY, LEBEDEV, VADIM, VEDALDI, ANDREA, and LEMPITSKY, VICTOR S. "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images." *ICML*. 2016, 1349–1357 3.

[Wei08] WEI, LI-YI. "Parallel Poisson Disk Sampling". *ACM Trans. Graph.* 27.3 (Aug. 2008), 20:1–20:9. ISSN: 0730-0301. DOI: 10.1145/1360612.1360619. URL: http://doi.acm.org/10.1145/1360612.1360619 1, 2.

[Wei10] WEI, LI-YI. "Multi-class Blue Noise Sampling". *ACM Trans. Graph.* 29.4 (July 2010), 79:1–79:8. ISSN: 0730-0301. DOI: 10.1145/1778765.1778816. URL: http://doi.acm.org/10.1145/1778765.1778816 1, 2.

[WLKT09] WEI, LI-YI, LEFEBVRE, SYLVAIN, KWATRA, VIVEK, and TURK, GREG. "State of the Art in Example-based Texture Synthesis". Eurographics, Mar. 2009. URL: https://www.microsoft.com/en-us/research/publication/state-of-the-art-in-example-based-texture-synthesis/ 3, 7.

[WW11] WEI, LI-YI and WANG, RUI. "Differential Domain Analysis for Non-uniform Sampling". *ACM Trans. Graph.* 30.4 (July 2011), 50:1–50:10. ISSN: 0730-0301. DOI: 10.1145/2010324.1964945. URL: http://doi.acm.org/10.1145/2010324.1964945 2.

[XCW14] XING, JUN, CHEN, HSIANG-TING, and WEI, LI-YI. "Autocomplete Painting Repetitions". *ACM Trans. Graph.* 33.6 (Nov. 2014), 172:1–172:11. ISSN: 0730-0301. DOI: 10.1145/2661229.2661247. URL: http://doi.acm.org/10.1145/2661229.2661247 3.

[XWSY15] XING, JUN, WEI, LI-YI, SHIRATORI, TAKAAKI, and YATANI, KOJI. "Autocomplete Hand-drawn Animations". *ACM Trans. Graph.* 34.6 (Oct. 2015), 169:1–169:11. ISSN: 0730-0301. DOI: 10.1145/2816795.2818079. URL: http://doi.acm.org/10.1145/2816795.2818079 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), 76:1–76:11. ISSN: 0730-0301. DOI: 10.1145/2185520.2185572. URL: http://doi.acm.org/10.1145/2185520.2185572 1, 2, 8, 9.

[ZZB*18] ZHOU, YANG, ZHU, ZHEN, BAI, XIANG, et al. "Non-stationary Texture Synthesis by Adversarial Expansion". *ACM Transactions on Graphics (Proc. SIGGRAPH)* 37.4 (2018), 49:1–49:13 2, 3, 6, 7, 12.