EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

General Point Sampling with Adaptive Density and Correlations Supplementary Material - Algorithms

paper 1192

Abstract

In this supplementary material, we detail an alternative synthesis algorithm that we use to generate results with locally regular structures. We start from an existing discrete texture synthesis method and show how it can be extended to support locally stationary non-ergodic point processes. In the second Section, we explain how strong local anisotropy can be handled in our analysis and synthesis framework.

1. Synthesis of Locally Non-Ergodic Sampling Patterns

Our synthesis algorithm assumes a locally stationary point process and synthesizes a single instance, i.e. distribution, of such a point process. However, for non-ergodic processes, statistics cannot be reliably extracted from a single distribution. In other words, expected values over different distributions generated by the same point process cannot be estimated from a single larger distribution. An example of a non-ergodic stationary process can be obtained by taking a regular grid, and randomly translating or rotating it, which is also called uniform or isotropic jittering [RAMN12, Ö16] in the literature. For these processes, difference vectors between sample points always stay the same as the points always form a regular grid. However, each instance is actually different due to the random global translation of the grid. This cannot be inferred by observing a single distribution.

This is problematic for our synthesis algorihm as the output statistics are extracted and matched for a single distribution, the synthesized distribution. Thus, for locally non-ergodic stationary processes, we need a different synthesis approach. For these cases, we thus extend a recent texture synthesis based approach for synthesizing distributions of discrete entities [ROM*15].

Below we first explain the basics of this algorithm, and then elaborate on how it can be adapted to synthesize distributions from locally stationary ergodic or non-ergodic processes.

1.1. The Synthesis Algorithm

We start from the formulation of a recent patch based texture synthesis algorithm that can handle discrete repeated structures in general domains [ROM*15]. The algorithm takes a single example distribution as the input, and generates a distribution with similar local structures. Due to the point-based representation of the textures, and smooth formulation of the resulting optimization problem, this formulation forms a suitable foundation for our synthesis algorithm.

The original algorithm is based on a local similarity measure between a continuous output function $\mathbf{f}(\mathbf{x})$ and the continuous example function $\mathbf{e}(\mathbf{x})$. The *similarity error density* is defined, for the location $\mathbf{x}$, as

$$S(\mathbf{f}(\mathbf{x}),\mathbf{e}(\mathbf{m}(\mathbf{x}))) = \int_{\mathbb{R}^n} |\mathbf{f}(\mathbf{x}+\mathbf{s}) - \mathbf{e}(\mathbf{m}(\mathbf{x})+\mathbf{s})|^2 w(\mathbf{s})\mathbf{ds} \quad (1)$$

where $w(.)$ is a window function which delimits a local neighborhood $\mathcal{N}$, and $\mathbf{m}(\mathbf{x})$ is a discontinuous mapping that matches each output domain point $\mathbf{x}$ to a point $\mathbf{m}(\mathbf{x})$ within the domain of the example function. Intuitively, S measures how well a local patch in f matches some other local patch in e. The extent of the patches is given by the window function w, and the center locations of the matching patches are $\mathbf{x}$ and $\mathbf{m}(\mathbf{x})$ for f and m, respectively.

The *total similarity error* between the output and input functions is then defined as

$$T = \int_{\mathcal{D}} S(\mathbf{x},\mathbf{e}(\mathbf{m}(\mathbf{x})))\mathbf{dx}, \quad (2)$$

where $\mathcal{D}$ is a subspace in the output domain. The T sums up all matching scores and thus measures the overall matching accuracy of each patch in f to some patch in e. For minimum T, we thus need to have that for each patch in the output f, we have a matching patch in the example e.

As in the common neighborhood matching based texture synthesis methods, the T is minimized alternatively for $\mathbf{m}(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$, with a matching step (where the best matching neighborhoods for the current $\mathbf{f}(\mathbf{x})$ are found, i.e. $\mathbf{m}(\mathbf{x})$ is computed), and a merging step (where the best output function $\mathbf{f}(\mathbf{x})$ is computed for the current $\mathbf{m}(\mathbf{x})$).

The key idea of the synthesis algorithm is forming these functions f and e in terms of the sampling points. Denoting the sample points in the synthesized output as $\{(\mathbf{x}_i), i = 1..n\}$, and those in the given example distribution with $\{(\mathbf{e}_i), i = 1..m\}$, the continuous f and e can be formed by placing Gaussians $g(\mathbf{x},\sigma) = e^{-|\mathbf{x}|^2/\sigma^2}$ at all point locations and summing them together, that is

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \sum_i g(\mathbf{x}-\mathbf{x}_i,\sigma) \\ \mathbf{e}(\mathbf{x}) &= \sum_i g(\mathbf{x}-\mathbf{e}_i,\sigma). \end{aligned} \tag{3}$$

Then, the *similarity error density* S can be analytically evaluated (see the work by Roveri et al. [ROM*15] for details). For the discretization of the integral in Equation 2, the output domain $\mathcal{D}$ is sampled with a regular grid consisting of the background integration points $\mathbf{c}_k$. For each $\mathbf{c}_k$, there is a best matching location $\mathbf{m}_k$ in the example domain. Hence, each $\mathbf{c}_k$ defines a neighborhood $\mathcal{N}_k$ in the output domain, that will match to some neighborhood in the example domain. The *discrete similarity error* is then obtained as

$$T = \sum_k S(\mathbf{q}_k, \mathbf{e}(\mathbf{m}_k)). \tag{4}$$

1.2. Synthesis with Adaptive Density and Correlations

The original algorithm, as described in the last section, is designed for a single example.

Although the original algorithm from [ROM*15] can synthesize output point sets with general repeated patterns, these patterns are determined by a single example. Hence, it can be used to synthesize a stationary distribution with fixed pair-wise correlations, while the density can vary in the output space by scaling the example [ROM*15], resulting in the same limitations as the previous sampling papers. Having an example distribution with spatially-adaptive correlations will not generate adaptive correlations in the output since the neighborhoods from the example are assumed to be repeated throughout the output space. Hence, the main challenge of adaptive synthesis is revising the technique to allow for multiple examples and their combinations.

Adaptive Correlations In order to synthesize distributions with adaptive correlations, we thus extend the *similarity error density* to consider multiple input functions and weight their importance depending on the location in the output domain. In this case, instead of having a single example function e, we have multiple functions e_p.

For an arbitrary point $\mathbf{x}$ in the output domain $\mathcal{D}$, there is a weight $\mathbf{w}_p(\mathbf{x})$ that gives how much example e_p is influencing that point. Our *weighted similarity error density* then computes a weighted average of the local similarities between the output function and multiple input functions, as

$$S_w(\mathbf{x}) = \sum_p \mathbf{w}_p(\mathbf{x}) S(\mathbf{f}(\mathbf{x}), \mathbf{e}_p(\mathbf{m}_p(\mathbf{x}))) \tag{5}$$

where $S(\mathbf{f}(\mathbf{x}), \mathbf{e}_p(\mathbf{m}_p(\mathbf{x})))$ is the similarity between the output function f, and the p-th input function e_p, and $\mathbf{m}_p$ is the matching function from the output domain to the domain of e_p.

In the synthesis algorithm in the paper, we define a pair correlation function (PCF) at each point $\mathbf{x}$ as a linear combination of basis PCF-s. The example distributions that define e_p give these basis PCF-s, and the weights w_p correspond to the computed or given weights for linearly combining the basis PCF-s. Hence, instead of having a PCF at each point in the output domain as a linear combination of the basis PCF-s, we have the corresponding basis example distributions and their linear combinations explicitly given in Equation 5.

Adaptive Density and Orientation The original method accounts for adaptive density and orientation by introducing a scale factor $\mathbf{s}(\mathbf{x})$ and a rotation matrix $\mathbf{R}(\mathbf{x})$ defined at every location in the output domain, which scales and rotates the example domain before matching. After computing the sizes of the neighborhoods $\mathcal{N}_k$ according to the given local density λ_k as described in the paper, we adjust the scaling field s such that at each neighborhood, we have the required number of points αn (please see the paper) in the neighborhoods of the same size in the example domain on average. The rotation for each neighborhood is used to rotate the corresponding examples for that neighborhood before synthesis, as done by Roveri et al. [ROM*15].

While simply scaling the input function with $\mathbf{s}(\mathbf{x})$, as proposed in [ROM*15], works for slight changes of scales, we found that, in order to properly handle adaptive densities, it is necessary to accordingly scale some additional parameters as well which are proportional to the input scale. The standard deviation σ of the Gaussians used to define the input and output functions in Equation 3 needs to be adapted according to the size of the neighborhoods and hence the scaling function s. If both σ and the local density of the sample points is small, there is little overlap between the Gaussians, and the minimization routine will not be able to compute a reliable gradient. Conversely, having Gaussians with a large overlap blurs out the distributional details and makes the synthesis not reliable. For a Gaussian placed at a sample position $\mathbf{x}$ in the output domain, we thus scale its initial σ with the scale factor $\mathbf{s}(\mathbf{x})$. We use the same scheme as in the paper for placing the neighborhood center points $\mathbf{c}_k$.

We show an example case where we compare to the original algorithm with non-adaptive neighborhood sizes and regular sampling of neighborhood centers in Figure 1. The example input distribution is shown in (a). Figure 1 (b) shows the output of our synthesis algorithm with our modifications, i.e. setting a proper spacing between the neighborhood centers, while (c) and (d) are obtained with the original algorithm where there are many overlapping large neighborhoods, and neighnorhoods with small overlap, respectively. Unlike the PCF based synthesis case, our adjustments for the neighborhood sizes and the corresponding neighborhood sampling are essential to get correct statistics for point distributions.

Initialization and Control Sampling We start by randomly initializing samples in the whole output domain as explained in the paper. In addition, the optimization procedure described in [ROM*15] includes a control sampling strategy to dynamically adapt the number of samples in the output domain and achieve better convergence. At each iteration, samples at random positions are added and existing samples are deleted if this would improve the similarity error density measure. We refer to the original paper for more details.

Geometry Sampling and Local Anisotropy The original

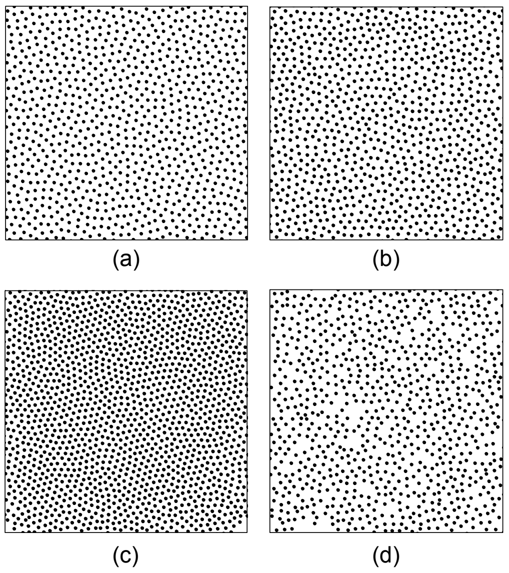

Figure 1: *Different distributions generated for the blue noise input example shown in (a). We obtain distributions with correct statistics when using our adjusted distribution of neighborhood centers (b), while having large (c) or small (d) overlaps leads to skewed characteristics.*

method allows to define guiding fields in 3D to accordingly orientate the input example and synthesize a distribution on a curved suface.

For small anisotropy factors, the original method allows to synthesize anisotropic distribution by simply scaling the input example. For strong anisotropy, the method can be extended to utilize anisotropic Gaussians.

Parameters The parameter σ determines the smoothness of the matched functions. In our implementation, it is set to the average spacing between the samples in the example. We refer to the [ROM*15] for other parameters. The neighborhoods should be large enough to capture the PCF of the distribution within them. For the distributions used in the paper, the neighborhoods sizes were set to include at least 50 samples.

1.3. Limitation

The main limitation of this discrete texture synthesis based algorithm results from using actual example distributions rather than extracted statistics as we do in the paper. A well-known limitation of such neighborhood based texture syntehsis approaches is that they cannot handle repetitions at multiple scales [ROM*15]. For the case of synthesizing point distributions, this means that clustered distributions, where the points in each cluster follow a certain distribution, and the clusters themselves follow another distribution, may not be reliably synthesized. In practice, we have observed that we still get visually accurate results for these cases. However, the statistics deviate from those of the examples slightly.

2. Analysis and Synthesis with Local Anisotropy

2.1. Analysis

The matrix $\mathbf{M}_x$ is a measure of anisotropy. We utilize 2D anisotropy measures from the spatial point processes literature [IPSS08] to compute $\mathbf{M}_x$. This method performs a simple density estimation on the difference vectors $\mathbf{h}_{ij}$. First, a radial histogram of the vectors $\mathbf{h}_{ij}$ is computed. Each bin thus corresponds to a direction on the unit hypersphere and a distance from its center. The dominant anisotropy is then extracted by picking the directions with the smallest and largest bin sums. We then form a covariance matrix $\mathbf{C}_x$ with two eigenvectors set as these directions, and the eigenvalues as the corresponding bin values. Finally, the matrix $\mathbf{M}_x$ is set as the whitening transform $\mathbf{M}_x = \mathbf{C}_x^{-1/2}$ such that the resulting difference vectors $\mathbf{h}_{ij}$ are distributed isotropically. Note that when we apply the transform $\mathbf{M}_x$, we assume that the volume $|\mathcal{N}_x|$ also scales so as to keep the intensity at λ_x. This step is designed for point distributions on two dimensional domains such as the 2D plane or two-manifold surfaces in 3D, or with limited number of anisotropy directions in higher dimensional domains.

2.2. Synthesis

Before feeding into the estimator, the difference vectors $\mathbf{h}_{ij}$ are first whitened with $\mathbf{M}_k \mathbf{h}_{ij}$. The gradient $\frac{\partial}{\partial \mathbf{x}_i} E_k$ is then computed, multiplied with $\mathbf{M}_k^{-1}$, and summed over all neighborhoods to get the final gradient $\Delta_i = \sum_k \mathbf{M}_k \frac{\partial}{\partial \mathbf{x}_i} E_k$ for point $\mathbf{x}_i$.

References

[IPSS08] Illian J., Penttinen A., Stoyan H., Stoyan D. (Eds.): *Statistical Analysis and Modelling of Spatial Point Patterns.* John Wiley and Sons, Ltd., 2008. 3

[Ö16] Öztireli A. C.: Integration with stochastic point processes. *ACM Trans. Graph. 35*, 5 (Aug. 2016), 160:1–160:16. 1

[RAMN12] Ramamoorthi R., Anderson J., Meyer M., Nowrouzezahrai D.: A theory of monte carlo visibility sampling. *ACM Trans. Graph. 31*, 5 (Sept. 2012), 121:1–121:16. 1

[ROM*15] Roveri R., Öztireli A. C., Martin S., Solenthaler B., Gross M.: Example Based Repetitive Structure Synthesis. *Computer Graphics Forum* (2015). 1, 2, 3