DOI: 10.1111/cgf.13627
EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)
Volume 38 (2019), Number 2

Accurate Synthesis of Multi-Class Disk Distributions

Pierre Ecormier-Nocca[1], Pooran Memari[2,1], James Gain[3], and Marie-Paule Cani[1]

[1]LIX, École Polytechnique [2]CNRS [3]University of Cape Town

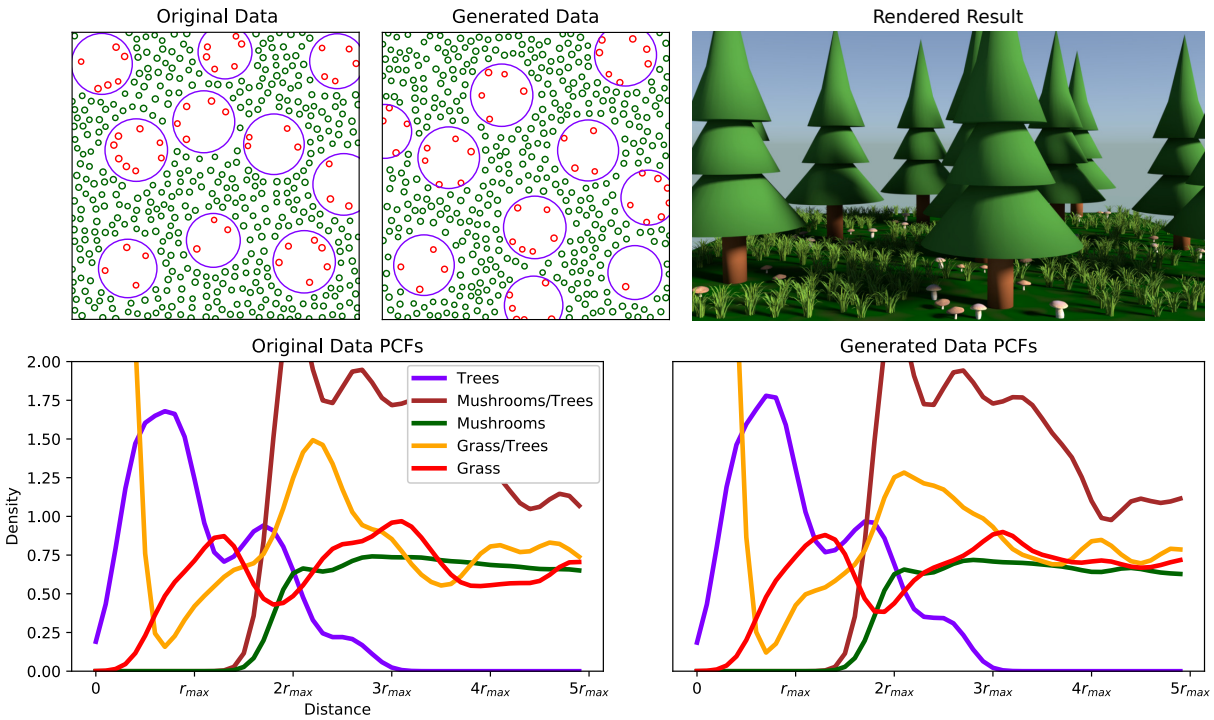


Figure 1: *Disk-based multi-class forest synthesis: mushrooms (in red) grow in the shade of trees (in purple), but grass (in green) is unshaded. Images include: the input exemplar (top left), output synthesis (top center), and corresponding preliminary rendering (top right), with analyzed input (bottom left) and synthesised Pair Correlation Functions (PCFs) (bottom right).*

Abstract

While analysing and synthesising 2D distributions of points has been applied both to the generation of textures with discrete elements and for populating virtual worlds with 3D objects, the results are often inaccurate since the spatial extent of objects cannot be expressed. We introduce three improvements enabling the synthesis of more general distributions of elements. First, we extend continuous pair correlation function (PCF) algorithms to multi-class distributions using a dependency graph, thereby capturing interrelationships between distinct categories of objects. Second, we introduce a new normalised metric for disks, which makes the method applicable to both point and possibly overlapping disk distributions. The metric is specifically designed to distinguish perceptually salient features, such as disjoint, tangent, overlapping, or nested disks. Finally, we pay particular attention to convergence of the mean PCF as well as the validity of individual PCFs, by taking into consideration the variance of the input. Our results demonstrate that this framework can capture and reproduce real-life distributions of elements representing a variety of complex semi-structured patterns, from the interaction between trees and the understorey in a forest to droplets of water. More generally, it applies to any category of 2D object whose shape is better represented by bounding circles than points.

CCS Concepts

• ***Theory of computation*** → *Randomness, geometry and discrete structures;* • ***Computing methodologies*** → *Shape modeling;*

1. Introduction

Creating 2D distributions (also known as arrangements) of discrete shapes is a common requirement in Computer Graphics, be it for vector-based texture synthesis of semi-regular patterns or for populating virtual worlds with varied elements, from rocks to vegetation. Building on recent developments in statistical analysis and synthesis, distributions can be learned from exemplars and seamlessly recreated over larger regions, either automatically or by painting with interactive brushes [HLT*09, EVC*15].

However, the application of point statistics to shape distributions has inherent limitations: the spatial extent of shapes cannot be expressed, so that undesired shape intersection may be inadvertently introduced in dense regions. Conversely, the presence of meaningful overlaps in the input (for instance, the occurrence of under-canopy plants beneath sheltering trees) simply cannot be captured using points distributions. This problem – where arrangements of shapes have different forms of 2D bounding circle intersection – was recently identified, leading to a first model for disk distributions [GLCC17]. Unfortunately, the handling of nested and overlapping cases was limited to a few extra bins in a distribution histogram, making it impossible to accurately describe the full range of overlapping configurations. Furthermore, while continuous representations such as Pair Correlation Functions (PCFs) [ÖG12] have been shown to improve the stability and robustness of point distribution synthesis, these methods have yet to be extended to either multi-class or extent-aware distributions.

In this work, we present an accurate and robust method for the analysis and synthesis of multi-class distributions of potentially overlapping disks, and demonstrate its applicability to populating virtual environments with 2D or 3D shapes.

A Pair Correlation Function (PCF) is a continuous curve that describes the density of neighbouring points as a function of distance from any given reference point. Our method builds on these PCFs to achieve robustness in the multi-class setting. We first present a simple extension of PCF analysis and synthesis to allow for dependency graphs between multiple classes of disks, thereby representing objects with fundamentally different behaviour in an arrangement. Since capturing spatial interactions within and between such classes requires an awareness of both disk location and radius, we also introduce a new metric for disks. This metric meets the differentiability requirements of PCF but is also tailored to distinguish between perceptually different key configurations, such as nested, tangent, or bordering disks. Finally, while retaining the global convergence of the mean PCF, we improve the visual quality of synthesised distributions by constraining each individual PCF to a validity region, based on the variance of input PCFs. As our results show, this framework leads to major improvements in the state of the art for capturing general disk distributions.

Our main technical contributions include:

- An extension of state of the art methods for point distribution analysis and synthesis to multi-class data. This is a crucial consideration when synthesising hierarchical or structured data, as it allows the user to model the relationship between different categories of objects, substantially improving visual fidelity.
- A new metric for evaluating pairs of disks, adapted to disk distribution synthesis and normalised to differentiate those configurations that are key in practice. In particular, this is the first metric that enables precise analysis and synthesis of distributions of possibly overlapping shapes.
- The introduction of variance-aware PCFs and associated error handling routines. This improves synthesis quality, especially in highly-constrained cases with semi-regular patterns. Variance-aware PCFs limit the most prevalent problem in dealing with traditional PCFs, namely the loss of information that arises from solely considering the average curve.

We will discuss each of these contributions in a separate section, after a brief overview of previous work in the field. The final two sections are dedicated to our results and concluding remarks.

2. Related work

The purpose of point sampling is to dynamically generate a set of points that embodies specific properties. It can be used for stippling [MALI10, DSZ17, MAAI17], but also for rendering and texture synthesis. One common requirement is to match the spectral profile of blue noise, and consequently there have been many improvements to the state of the art in this area, such as improvements in computation [BWWM10, LNW*10, CYC*12], various properties [LWSF10, CGW*13] and analysis [WW11]. Point sampling can also be applied to general distributions by matching a particular spectrum [ZHWW12], or reproducing [ÖG12] or interpolating [ROG17] input exemplars.

More specifically, example-based synthesis aims at deriving from a given representative input, an output that captures key visual aspects of the original, but which differs in certain specifics, such as having larger extent, a constrained boundary, or a different underlying flow field. This has been studied extensively in the context of image-based texture synthesis at the pixel and patch levels [WLKT09]. However, where the basis of similarity is the distribution of identifiable elements, such as motifs, objects, and shapes, a pixel or patch-based approach tends to introduce fragmentation artefacts [MWT11] despite attempts to correct for this [ZZV*03]. This is better solved by discrete pattern synthesis approaches that treat structural elements as the unit of analysis and synthesis.

Dischler et al. [DMLG02] address this by segmenting out pattern elements from an image, based on colour quantisation, and arranging them using a quadrant-based nearest-neighbour scheme. This does not account for higher-order correlation and interactions between multiple classes. Ijiri et al. [IMIM08] improve the matching that is central to effective pattern synthesis by using a Delaunay triangulation over element centers. This is combined with local seeding to grow patterns outwards. The same kind of approach, but with internal seeding, was used to maintain a distribution of surface details during shape deformation [RHC15].

The problem with these approaches is that they do not effectively model interrelationships between classes of elements, are discrete in nature, potentially locking into suboptimal patterns, and also do not account for the shape and orientation of elements.

On the multi-class front, Hurtut et al. [HLT*09] automatically classify vector elements based on histograms of appearance that

consider area, orientation, elongation, extremities and edge crossings. Arrangements among and between classes are then analysed using multi-type point process statistics and synthesised with a variant of Metropolis-Hastings sampling. In a similar vein, multi-class variants of blue noise sampling have been proposed by Wei [Wei10] based on extended dart throwing, and by Qin et al. [QCHC17] based on constrained Wasserstein barycenters. However, these enhancements are particular to blue noise distributions.

Shape issues are addressed by Ma et al. [MWT11] who place multiple point samples per element and couple this with an energy-based iterative solver that supports extra terms to capture user requirements (such as orientation fields). Similarly, Landes et al. [LGH13], uses relative orientation and the distance between the elements' geometry to reproduce shape-aware textures. Both consider correlations between elements across multiple classes.

In terms of continuous matching functions, which overcome sensitivity to initial conditions and improve pattern fidelity, Roveri et al. [RÖM*15] employ a functional sum-of-Gaussians representation and Öztireli and Gross [ÖG12] use point-based pairwise correlation functions. These schemes are amenable to an analytic gradient and thus robust solvers, such as gradient descent. Most work in this area focuses on distributions across a 2D plane or surface. One exception is the synthesis method of Lagae and Dutré [LD06], which supports Poisson sphere distributions in 3D using an efficient tiling algorithm.

Unfortunately, the matching functions, multi-class support and shape adaptations of these methods are ill-suited to the case of intersecting and overlapping disks, since their measures of geometric distance are insufficiently nuanced.

An important application of discrete pattern synthesis lies in the procedural placement of features in natural scenes. For instance, Guérin et al. [GGG*16] model entangled details, such as grass tufts, twigs and fallen leaves, in a custom collision structure, while Worldbrush [EVC*15] encodes the interrelationships within and between categories of scene elements (such as rocks, trees, roads, and buildings) as distributions of points. Artists can then use smart brushes to paint these distributions onto landscapes. In this framework, analysis is conducted using an adaptation of point process statistics to small input exemplars and a user-defined hierarchy of classes, and synthesis is achieved with a modified Metropolis Hastings algorithm. Ecobrush [GLCC17] extends this concept to address the problem of ecoplacement – populating landscapes with plants whose attributes (such as species, position and age) are ecologically sound. Here, input examples are automatically generated using sand-box ecosystem simulations. The focus is on trees and shrubs with potentially overlapping canopies, which represents an instance of the more general problem of analysing and synthesising distributions of overlapping disks. However, Gain et al.'s solution is not general: overlap cases are modelled using three extra bins in the Metropolis Hastings distance histograms, to respectively represent complete inclusion, and more than and less than half inclusion. Moreover, since disk position and radii are not jointly analysed, young and mature trees of the same species are separated into different classes, which prevents any mechanism for continuous optimisation of tree radii at the synthesis stage.

In this work, we revisit these procedural methods and replace their Metropolis Hastings sampling with an extension of the continuous PCF framework of Öztireli and Gross [ÖG12], extended to general dependency graphs between classes and to disk distributions that include overlap configurations, where relative position and radii are considered jointly.

3. Handling multi-class distributions

In this section, we provide background on the Pair Correlation Function (PCF) method [ÖG12] for analysis and synthesis of point distributions, before explaining how it can be extended to multi-class distributions, in such a fashion as to allow a balance between overconstraining and underconstraining interclass dependencies.

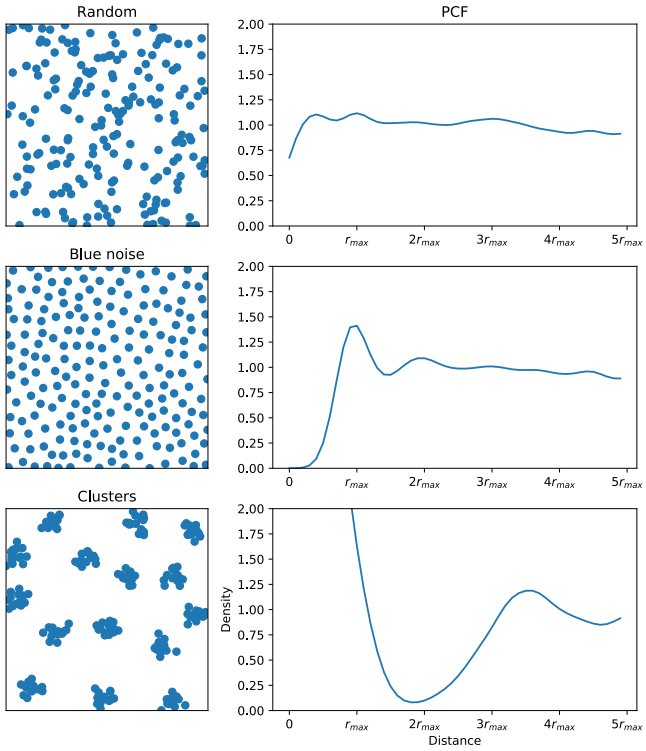


Figure 2: *Three typical point distribution exemplars (left) and their associated PCF representations (right).*

3.1. Analysis and synthesis with Pair Correlation Functions

The PCF framework captures point-based distributions by considering every point in a stationary exemplar in turn and building an average density of surrounding points at different distances. This is encoded as a density function (the *Pair Correlation Function*) normalised according to the number of points in the exemplar, and which intuitively represents the variation in the number of neighbors around each point as a function of increasing distance. The main strength of PCFs is that they represent distributions as normalised continuous functions, which are amenable to derivative-based optimisation, such as by gradient descent, at the synthesis stage.

A PCF provides also an easily interpretable visual signature that

corresponds to the perceptual characteristics of the underlying distribution. Figure 2 illustrates this expressivity: while the PCF of a random distribution of points (top) is mostly flat, the one obtained from a blue noise distribution (middle) shows a very characteristic pattern of ripples of decreasing magnitude, with a global maximum located at the average distance between a point and its closest neighbor. With clustered distributions (bottom), the PCF features two dominant peaks: the first at the mean distance between points within a cluster, and the second, less pronounced, at the mean distance between points in neighbouring clusters.

Computing PCFs: As in standard point processes, the underlying principle of PCF derivation is to count neighbors within rings of increasing radius around each point. However, two modifications are required in order to generate density invariant, continuous curves:

First, distances are normalised according to overall exemplar density. Given an input exemplar with n points that, without loss of generality, occupies a unit square, distances are divided by a value r_{max} based on how far points would be separated if they were maximally spread to occupy the whole input domain. This value, which uniquely depends on the mean density of points in the exemplar, is given by (see [LD06]) :

$$r_{max} = 2\sqrt{\frac{1}{2\sqrt{3}n}} \tag{1}$$

Second, to ensure the function is continuous and robust to noise, the influence of a neighbouring point is spread using a Gaussian Kernel $k_\sigma(x) = \frac{1}{\sqrt{\pi}\sigma} e^{-x^2/\sigma^2}$, centered at the distance r from the reference point. In our work, we use the following simplified version of the original PCF estimator [ÖG12], specialised for the 2D case and expressed over a unit square domain:

$$\mathrm{PCF}(r) = \frac{1}{A_r n^2} \sum_{i \neq j} k_\sigma(r - d_{ij}) \tag{2}$$

where A_r is the area of the ring with inner radius $r - 1/2$ and outer radius $r + 1/2$, and d_{ij} is the distance between reference P_i and neighbour P_j. As in [ÖG12], we compute PCFs in a relatively big neighborhood of P_i, depending on the density. In our experiments, r takes discrete values from zero to kr_{max}, where k and the discretisation step are two prescribed constants.

We often refer to this PCF estimation as "mean PCF" in the remainder of this paper, to distinguish it from the so-called "individual PCF" associated to each P_i, and estimated as follows:

$$\mathrm{PCF}(P_i, r) = \frac{1}{A_r n} \sum_{j} k_\sigma(r - d_{ij}) \tag{3}$$

As we will see in Section 5, considering individual PCFs in the synthesis procedure results in a more accurate synthesised distribution.

Boundary handling: Since the input exemplars may have a limited number of points we cannot afford to simply discard those near the boundary. Instead, following Emilien et al. [EVC*15], we overcome the bias introduced by neighbourhood rings that intersect the boundary, as follows: we weight every point's contribution to the PCF for each ring radius r by the ratio between the length of the circle inside the domain and its full perimeter. This compensates for the fact that points located near the edge of the domain have fewer neighbors than those further in.

Distribution synthesis from PCFs: Following [ÖG12], synthesis is achieved in two steps:

An **initialisation step** computes a first-pass solution using generalised dart throwing to set an initial position for n points in the target domain (with n computed from the exemplar mean density and target area). This step operates by successively creating and accepting or rejecting new random placements until the required number of points is reached.

Each candidate point is only retained if the revised PCF error remains below a threshold ϵ_m (set to increase with the number of accepted points m). The error is computed as $E_{init} = \max_r(PCF_{new}(r) - PCF(r))^+$ where x^+ indicates that negative values of x are truncated to 0. This forces the algorithm to either generate a curve that remains below the targeted PCF if possible, or exceed it by less than the tolerance threshold ϵ_m. In practice, this initialisation step leads to a PCF reasonably close to the target one.

A **refinement step** is then applied, during which point positions are adjusted using gradient descent to better fit the current PCF to the target. We use a least squares cost function $E_{ref} = \sum_r(\mathrm{PCF}_{new}(r) - \mathrm{PCF}(r))^2$, leading to the following normalised gradient at point P_i:

$$\Delta_i = -\frac{\sum_{i \neq j} \mathbf{u}_{ij} w_{ij}}{\sum_{i \neq j} |w_{ij}|}, \tag{4}$$

$$\mathbf{u}_{ij} = \nabla P_i d_{ij}, \tag{5}$$

$$w_{ij} = \sum_r \frac{\mathrm{PCF}_{new}(r) - \mathrm{PCF}(r)}{r} (d_{ij} - r) k_\sigma(d_{ij} - r) \tag{6}$$

In turn each P_i is moved along the associated Δ_i by a random distance in $\{10^{-1}, \ldots, 10^{-5}\}$ and only the single move that most reduces the error is retained. This scheme is repeated until convergence.

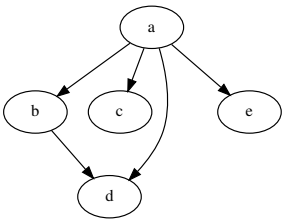
Let us also mention that, by definition, the PCF (and so the defined error) depend on the used metric which indicates the corresponding neighborhoods around each point of distance r. We will specify the metric we use in our method for disk distributions in Section 4.

3.2. Extension to multi-class distributions

An extension of the PCF framework is required for it to cope with more complex cases involving interdependent classes of objects. Although such an extension was mentioned in passing by Öztireli and Gross [ÖG12], they only add control over the overall distribution regardless of the point classes. This approach only works in simple cases and for largely uncorrelated data, since precise interactions between classes are not modelled. We demonstrate these limitations in the third example of Figure 11. A more complete approach was introduced in the context of Metropolis Hastings methods [EVC*15], but this only supports a strict linear hierarchy of classes. Placement of points within a particular class is assumed to depend on all previously instantiated classes. While this approach

works well for simpler cases, our early experiments showed that it struggles to produce adequate results when the number of classes increases, since later classes in the hierarchy are over-constrained and the refinement step fails to converge.

To tackle this issue, we express dependencies in multi-class data using a general *dependency graph*, stored in a Directed Acyclic Graph (DAG) data structure. Rather than insisting that a given subordinate class be fully connected to every previously instantiated class, we instead allow sparse parent-child connections. For example, in Figure 3 class d need not be connected to all of a,b,c,e, but can instead be more parsimoniously reliant on only a and b. In many cases capturing a few strong relationships directly and others transitively is sufficient to model a pattern. Crucially, this reduces the number of constraints, thus facilitating convergence to a significantly better solution.



	a	b	c	d	e
a	✓				
b	✓	✓			
c	✓		✓		
d	✓	✓		✓	
e	✓				✓

Figure 3: *An example of a dependency graph (left) and the corresponding adjacency matrix (right).*

To incorporate such hierarchies, during analysis we derive a PCF for every pair of classes (C_i, C_j) connected by an edge in the dependency graph. This requires $n_i \times n_j$ additional pairs of points and thus terms in Equation (2), where n_i is the number of points in class C_i, in addition to the normal PCF computation within each class C_i (involving $n_i(n_i-1)/2$ terms).

At the synthesis stage, classes are always processed in the topological sorting order of the dependency graph (e.g., a, b, c, e, then d in Figure 3). For each class, synthesis proceeds in two steps as before. We take care to recalculate the initialisation error for each active PCF (the class in question as well as its parent pairings in the dependency graph) when determining if a random point should be retained or discarded. The major change in the refinement step is that points are now moved in a direction computed using the sum of gradient vectors from all active PCFs, rather than a single PCF. Note that in our implementation, disk radii do not vary during refinement, which ensures that we retain the initial distribution of radii for each class.

3.3. Validation

To validate our multi-class synthesis method, we demonstrate its behaviour for a case that exhibits strong interaction between classes. In Figure 4, red points cluster around isolated cyan points, while green points form around isolated purple points. The combined set of cyan and purple points are negatively correlated over the domain. It is impossible to reproduce such clustering behaviour without addressing the inherent multi-class dependencies in evidence. Since clustering only occurs around cyan or purple centers, we construct the DAG in Figure 4 (left) to express this specific dependency: purple and cyan points influence each other, but clustered points only depend on the class they agglomerate around. This reduces the dependency constraints from 10 (since we have 4 classes, $\sum_{i=1}^{4} i = 10$) in the naïve approach to only 7 (10 minus 3 removed relations between the independent centers and clusters) in ours, which leads to a more efficient error minimisation.

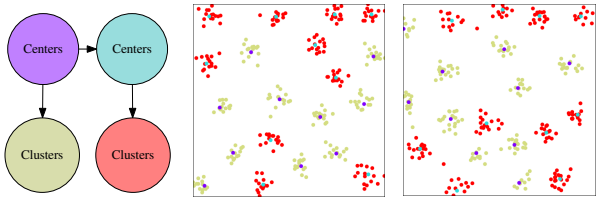


Figure 4: *Synthesis of a 4-class-dataset: dependency graph (left), exemplar input (middle), and synthesised output (right). Note that the chosen dependency direction between purple and cyan classes is arbitrary. Such a distribution is not reproducible without multi-class dependency.*

4. Extension to distributions of disks

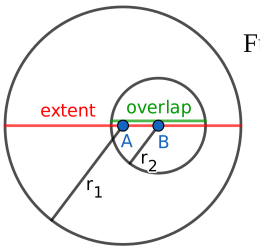
When placing collections of 3D objects on a 2D surface there are cases where considerations of relative size and potential intersection are vital. For instance, in ecosystem simulations (such as Figures 1 and 10) tree canopies often overlap in a botanically meaningful pattern. Such cases can be accommodated by treating them as distributions of potentially overlapping disks instead of points. In this section we consider the range of salient disk configurations, build a distance metric that distinguishes between them, and indicate how this metric can be incorporated into our framework to support disk distributions.

4.1. Distinguishing salient disk configurations

Our goal is to design a disk-aware distance metric suited to PCF processing, which distinguishes between perceptually salient disk interactions, as enumerated in Table 5. These configurations were derived from a study of several application scenarios, including layouts both organic (such as plant ecosystems) and structured (such as table settings and meshing cogwheels).

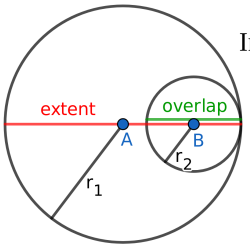
Unfortunately, existing disk metrics fail to disambiguate these salient configurations. The first naïve approach is to add disk radius as a third coordinate and calculate an $\mathbb{R}^3$ Euclidean distance, but this conflates position and radius so that visually distinct configurations can map to the same distance.

A second standard measure, classically used in computational geometry [Aur87], is the sum of the power of each center with respect to the other circle (equal to $2d^2 - r_1^2 - r_2^2$ using the notation of Table 5). Although the sum of powers is strictly increasing and continuous, it becomes negative in certain overlapping cases. Moreover, since these negative values do not admit any general lower



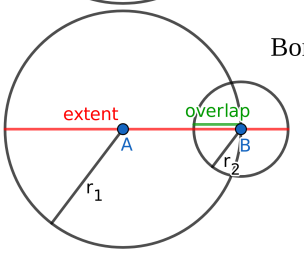
Fully Inside

$$d \le r_1 - r_2$$
$$extent = 2r_1$$
$$overlap = 2r_2$$
$$f \le 4r_1 - 4r_2$$
$$f_{norm} \le 1$$



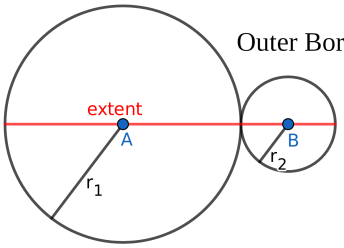
Inner Border

$$d = r_1 - r_2$$
$$extent = 2r_1$$
$$overlap = 2r_2$$
$$f = 4r_1 - 4r_2$$
$$f_{norm} = 1$$



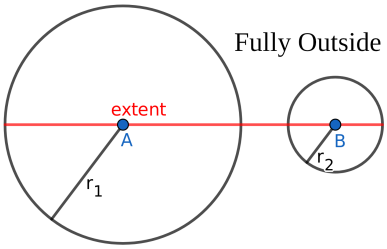
Border

$$d = r_1$$
$$extent = 2r_1 + r_2$$
$$overlap = r_2$$
$$f = 4r_1 - r_2$$
$$f_{norm} = 2$$



Outer Border

$$d = r_1 + r_2$$
$$extent = 2r_1 + 2r_2$$
$$overlap = 0$$
$$f = 4r_1 + 2r_2$$
$$f_{norm} = 3$$



Fully Outside

$$d \ge r_1 + r_2$$
$$extent = d + r_1 + r_2$$
$$overlap = 0$$
$$f \ge 4r_1 + 2r_2$$
$$f_{norm} \ge 3$$

Figure 5: *Principle salient configurations of disk interactions. Relevant notation is as follows: d is the distance between disk centres, extent and overlap are 1D measures linked to the union and intersection of the disks, f is the new disk distance metric that replaces r, and f_{norm} is a normalised version that distinguishes the salient cases.*

bound, a positive metric cannot be guaranteed by adding a constant term.

Lastly, as already mentioned, the EcoBrush measure for histogram binning of plant distributions [GLCC17] is discontinuous, which both prevents fine-grained analysis of overlapping cases and makes it fundamentally unsuited to the gradient-optimisation that underpins the PCF framework.

Consequently, there is a need to develop a continuous metric that distinguishes between salient disks configurations, as identified in Table 5.

4.2. Saliency-based distance between disks

In addition to computational efficiency and meeting the standard requirements for metrics (namely, non-negativity, identity of indiscernibles, symmetry and triangle inequality), our main goal in designing a distance between disks is to ensure detection and distinction between perceptually-different disk configurations, such as nested, bordering or fully disconnected disks.

Let (A, r_1) and (B, r_2) be two disks, where radius $r_1 \ge r_2$, and d be the distance between their centers, A and B.

Our metric is based on what we term the *extent* of the disks, defined as the length of the intersection between the line that goes through A and B, and the convex envelope delimited by the circles (see the red lines in Table 5). This can easily be calculated using Equation (7). We also define the *overlap* (see Equation (8)), which is the length of the intersection between the line through A and B, and the intersection of the two circles (see the green lines in Table 5). Using these two measures, we construct our new metric as the difference between the *extent* and the *overlap* (which can be regarded as a 1D approach to the union minus the intersection, or the length of the *extent* that is not common to both circles), to which we add the distance between centers d and the difference between radii, in order to differentiate all cases (see Equation (9)).

$$extent = \max(d + r_1 + r_2, 2r_1) \tag{7}$$
$$overlap = \text{clip}(r_1 + r_2 - d, 0, 2r_2) \tag{8}$$
$$f = extent - overlap + d + r_1 - r_2 \tag{9}$$

As it stands, our distance is a symmetric, continuous function that differentiates successfully between the three main cases of disk interaction (fully inside but abutting the border, centered on the border, and fully outside but against the border) for fixed r_1 and r_2. Unfortunately, it provides different values for the salient configurations when the radii change, as indicated by the radii-dependent values of f listed in Table 5. This makes it unsuitable for PCF computations, since merging values that are similar but carry a different meaning will destroy that meaning, thus making it impossible to synthesise a perceptually-correct disk distribution.

To circumvent this, we normalise our distance function based on these three special cases, transforming them into three fixed values. More precisely, we choose 1, 2 and 3 as the three normalised values for the specific cases "inner border", "border" and "outer border" (see Table 5), and define the normalised distance f_{norm} as a continuous piecewise linear function of f that maps the specific cases to these values:

$$f_{norm} = \begin{cases} f/(4r_1 - 4r_2), & \text{if } d \le r_1 - r_2 \\ (f - 4r_1 + 7r_2)/(3r_2), & \text{if } r_1 - r_2 < d \le r_1 + r_2 \\ f - 4r_1 + 2r_2 + 3, & \text{otherwise.} \end{cases} \tag{10}$$

4.3. Distribution processing for disks

Analysis: Replacing the standard Euclidean distance with our new distance function for analysing disk distributions is straightforward. The new distance f_{norm} is used as the horizontal axis of the PCF, and therefore both r and d in Equation 2 relate to this metric. However, the Euclidean distance between disk centers is retained for computations not directly related to disk interactions, such as the evaluation of r_{max} (Equation(1)) and compensation for border effects during analysis.

Synthesis: More care is required during synthesis, since both disk positions and consistent radii need to be generated.

During initialisation, we match the original distribution of disk radii by sorting the radius values of the input exemplar within a class in descending order and simply picking them sequentially during dart throwing. When the number of required output disks exceeds the input, we repeat the entire list of input radii as required and use a random non-duplicated sampling to make up any shortfall before sorting as before. This simple scheme has the virtue of being computationally efficient and allowing greater freedom of placement. Smaller disks are easier to fit as they generally have more valid locations, so we prefer to save them for later and focus on the largest and most difficult cases first.

At the refinement stage (Equations (4) – (6)), we replace pointwise distance (d_{ij}) with our new disk-wise distance (f_{norm}) and apply gradient descent to optimise for center positions. The radius distribution is handled in parallel where we take values from the input radius distribution in decreasing order. This way our final radius distribution matches exactly with the given distribution. Indeed, this strategy makes the best use of the given radius distribution, and seems practically more efficient than a gradient descent method optimising on both position and radius simultaneously.

Figure 1, illustrates the use of this method for a distribution with three disk classes, including some overlapping and inner border cases. For comparison, the PCFs of our synthesised example are shown at the bottom right.

5. Towards optimal convergence

Although putatively accurate in their convergence to the mean statistics for each class, PCFs methods can give rise to a clear mismatch between input and output. Furthermore, a few scattered disks may be introduced during initialisation to compensate for inaccuracies in the positioning of other disks, thereby decreasing overall error in the moment. Once present, these outliers cannot be removed, since refinement only locally improves the radius and position of a disk, giving rise to objectionable artifacts.

In this section, we introduce a new strategy for validating inserted disks during synthesis. While retaining global convergence of the mean PCF, these additional constraints, based on the variance of input PCFs, significantly improve the visual quality of the final outcome.

5.1. Variance-aware PCFs

The method presented thus far creates a combined average over individual per-point PCFs. Predictably, this summarisation leads to a loss of information compared to the original, individual PCFs. As Figure 6 shows, such a data reduction can cause visually inaccurate replication of the initial distribution in over-constrained cases, even when convergence to the mean PCFs (one for each class, and one per edge in the dependency graph) is achieved.

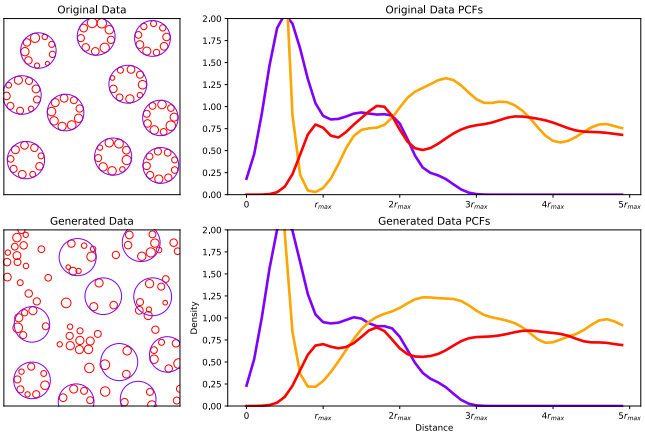


Figure 6: *Correctly matching the average PCF alone is not sufficient for perceptually convincing results in highly constrained cases. (Top) Input disk distribution and the corresponding mean PCFs, where the purple (respectively red) curves represent interrelationships within the purple (respectively red) class of disks, and the yellow curve represents the dependency of red on purple. (Bottom) A perceptually incorrect synthesised distribution, despite convergence of all mean PCFs.*

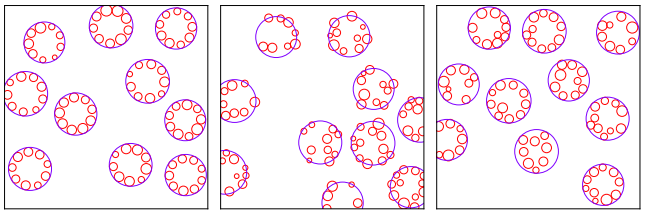


Figure 7: *A challenging over-constrained nested case (left). Synthesis result using Ecobrush [GLCC17] (center). Our variance-aware solution (right).*

To tackle such challenging cases, we introduce a variance-aware extension of the PCF method (for typical outcomes see Figure 7(right)). Our idea is to retain more data from the individual exemplar PCFs. This enables us to ensure during synthesis that the PCF of a candidate disk is never too divergent from individual PCFs in the input distribution. This is done without sacrificing convergence to the mean PCF.

In practice, rather than retaining individual PCFs, which would be both costly in memory and difficult to manipulate, during analysis we extract the lower and upper bound of the set of all the individual PCF curves and employ these bounds as constraints.

During synthesis, a new disk is accepted only if its PCF lies in the so-called *validity region* bounded below and above by the lower and upper envelopes of the union of the original PCFs (as illustrated

in Figure 8, where gray curves represent the original PCFs from reference disks in the exemplar).

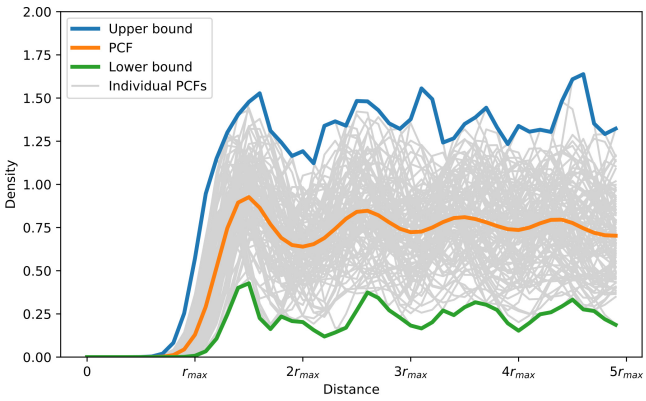


Figure 8: *Validity region: during synthesis the insertion of a new disk is only accepted if its PCF lies in the envelope formed by the union of all original PCFs.*

These bounds, and the difference between them, allow us to differentiate parts of the PCF that must be strictly enforced (an error is likely to be perceptually salient for human observers in regions of low variance) from those with greater leeway. An alternative would be to compute the standard deviation at points along the mean PCF curve, but this is not necessarily representative since the distribution of curves is often asymmetrically skewed about the mean. Moreover, bounds are a harder limit than the standard deviation, allowing us a better fit for the more extreme cases that may have occurred in the source exemplar.

To make effective use of the information provided by our variance-aware PCFs, we add a new term to the error formula used during initialisation. This term represents the largest distance between the new curve and the validity region. We also check that each individual error term remains positive, and divide the differences by the target value. This enables us to associate a higher error with variations where the target PCF value is close to 0. These represent visually salient errors, since they are equivalent to adding an outlying disk where none exists in the input data. Incorporating these changes, we end up with the following error function for synthesis initialisation:

$$E = \left(\max_r \frac{new_{mean} - PCF_{mean}}{PCF_{mean}}\right)^+ + \max\left(\max_r \frac{new - PCF_{upper}}{PCF_{upper}}, \max_r \frac{PCF_{lower} - new}{PCF_{lower}}\right)^+ \quad (11)$$

where x^+ indicates that negative values of x are truncated to 0.

This change in the initialisation proves effective in practice. In particular, we achieve the result in Figure 7 for the case with over-constrained nested disks.

5.2. Control of convergence

Since PCFs are by nature injective and not surjective (any layout of disks can be mapped to a PCF, but some PCFs cannot be realised as a corresponding disk layout), achieving initialisation in reasonable time can be challenging.

Indeed, even with incremental relaxation of the error threshold ϵ, for complex distributions the error can sometimes become so large that it prevents ongoing point placement.

Further relaxing the error threshold could prevent the initialisation from seizing up, but at the cost of lower quality in simpler cases. One solution is to adapt the relaxation factor according to the input configuration, but such automatic tuning in complex multi-class cases seems highly non-trivial.

Instead, we flag a lack of progression when the number of consecutive point rejections exceeds a certain threshold (we generally used a threshold of around 1000 rejections), and switch to a simple grid search. We divide the domain into a regular grid, and compute the expected error for each grid cell. We then sample and accept a random point in the cell with lowest error (or in a random cell among those with minimal error). While this algorithm is slower than dart throwing for easy cases, it runs in constant time regardless of the complexity of the distribution and always returns a solution close to optimal. It is also trivially parallelisable thanks to the independent error computation in each cell. Algorithm 1 sums up the initialisation step of our pipeline with this grid search enhancement.

Input: Hierarchy of PCFs, number of elements n
Output: Initialised elements

foreach *class* **do**
  $fails \leftarrow 0$;
  repeat
    $\epsilon \leftarrow \epsilon_0 + \epsilon_\Delta fails$;
    Sample a random element from original distribution;
    Compute error E;
    if $E < \epsilon$ **then** Accept this element;
    else $fails \leftarrow fails + 1$;
    if *more than max_fails successive fails* **then**
      while *less than n elements accepted* **do**
        Grid-search the domain for the lowest error;
        Sample a random element in the best cell;
        Accept this element;
      end
    end
  until *n elements are accepted*;
end

Algorithm 1: Initialisation algorithm incorporating grid search.

6. Results and discussion

In most of the examples in this paper, we used $\sigma = 0.25$ (smoothness of the Gaussian kernel), $\delta = 0.1$ (discretisation step for radii analysis), and a neighborhood size around each point set to $5r_{max}$ for PCF computations. If the neighbourhood radius is too small, the PCFs will not incorporate sufficient context area, reducing reproduction accuracy. Conversely, if the radius is too large, the method will overfit the data, damaging its ability to generalise. A similar

trade-off exists for the smoothness parameter σ: reducing it improves the replication of detail and particular configurations, while increasing it speeds up convergence, with the drawback that details may be approximated. The termination test for refinement is when the accumulated adjustments to all disks during the last iteration drops below a distance threshold. Alternatively, since all PCFs are incrementally updated after each iteration, we could check the difference between the target and current PCF curves. We also set a maximal number of refinement iterations to obtain an approximate fallback solution when refinement fails to converge.

6.1. Comparison with previous methods

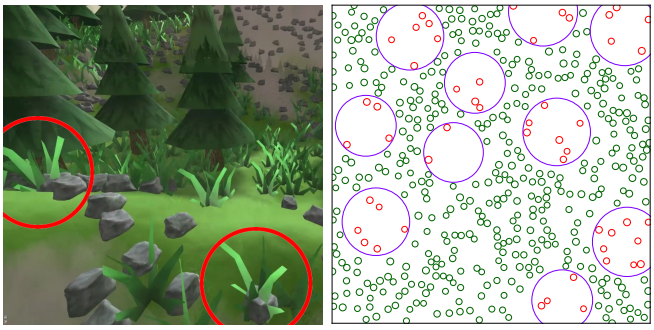


Figure 9: *Comparison with prior work: (Left) Modelling objects as points in Worldbrush makes collision handling impossible (e.g., grass growing through rocks). (Right) For the exemplar from Figure 1 Ecobrush instantiates some mushrooms (in red) too near tree centers (in purple) and fails to achieve an even distribution of grass (in green).*

In WorldBrush [EVC*15], trees, grass and other scene elements are modelled as points for the purpose of placement. Since the varying spatial extents of objects are ignored, fine control over complex interactions, such as collisions, is unattainable. Figure 9 shows a typical failure case in close-up from their video results.

The example in Figure 1 provides a similar context for comparison. In this toy example of a fairytale forest, mushrooms are sun intolerant and must be slightly shaded by trees, while grass favours sunlight and cannot be shaded. In addition to avoiding collisions thanks to the use of disks, our method models such inter-class relationships effectively. We provided the exemplar from Figure 1 as input to an Ecobrush-style synthesis [GLCC17], to evaluate the benefits of our continuous distance function over a binned solution (see Figure 9(right)). While their method respects coarse constraints (mushrooms in the shade, grass in the sun) the discrete distance function fails to reproduce the blue noise distribution of grass and the positioning of mushrooms near the border of tree shade.

Lastly, while the EcoBrush algorithm cannot handle over-constrained cases (as shown in Figure 7), our method proves to be robust to the types of dense ecosystems that is the focus of their work, even with the significant dependencies resulting from a full hierarchy of classes. Figure 10 demonstrates our results on a dense ecosystem with 9 species and more than 2,300 individual plants (taken from Ecobrush, courtesy of the authors).

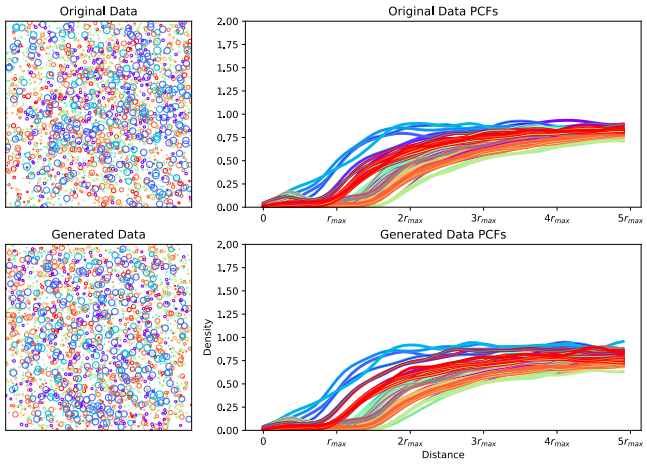


Figure 10: *Results for a dense ecosystem with 9 interdependent classes and more than 2,300 plants. (Top) input exemplar and analysed PCFs. (Bottom) synthesised distribution and the corresponding PCFs.*

Figure 11 demonstrates that our method is able to reproduce point distributions found in previous work. We provide results that are perceptually similar to two examples from Ma et al. [MWT11], with the difference that we cannot capture the orientation of dominant lines, since our method is rotation invariant. We also include a comparison with a multi-class example from Öztireli and Gross [ÖG12]. Here, our results match the input far more closely, as indicated by the analysed curves. For instance, the purple points are never close together in the input nor in our output. More generally, their approach is effective for relatively simple scenarios, but fails in more complex cases, since they only consider interactions within classes and for the dataset as a whole, but not between classes.

Unfortunately, as evident from Figure 13, our method is unable to reproduce Roveri et al.'s [RÖM*15] highly structured patterns, since one of our key goals is the ability to generalise an input pattern rather than perfectly replicate it. Similarly, non-stationary distributions (such as in the work of Rover et al. [ROG17], are beyond the scope of this paper. Importantly, however, our method is able to process arrangements of overlapping shapes, which is not the case for any of the more structured methods.

6.2. Other results

Figure 12 illustrates the use of our disk distribution synthesis in different application scenarios.

The first scenario is derived from the behaviour of scattered water droplets on a smooth surface. Although input and output are rendered in the same style for ease of comparison, the input distribution has a physical analog and was extracted by hand from a photograph. To account for the interaction of size and placement, droplets are divided into 4 classes, with a sequential chain of dependency from largest to smallest.

The second example showcases a more complex distribution of

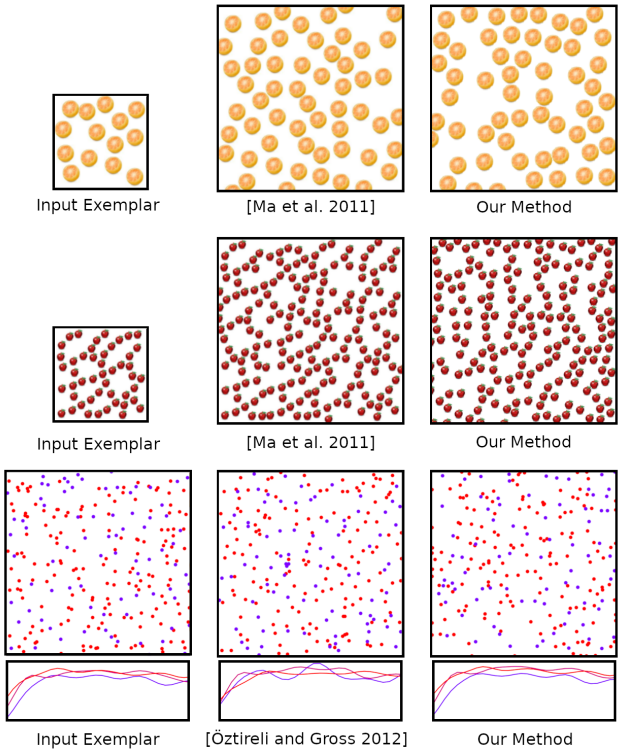


Figure 11: *Comparison with Ma et al. [MWT11] indicates that our method is capable of reproducing previous results, except for the specific orientation of the dominant lines, while a comparison with Öztireli and Gross [ÖG12] shows that we more accurately respect features in multi-class data (refer to the PCFs in the bottom row).*

plates, glasses, bowls and food items in class-specific configurations, with classes that may touch but should not overlap (glasses, bowls and plates), should overlap (food on plates), or may overlap (apples and candies).

Lastly, we use our method to populate a Mediterranean landscape with plants, based on a distribution of 12 plant species with 78 dependencies, computed by an ecosystem simulator [GLCC17]. The output distribution is synthesised over a larger region, and used to instantiate 3D tree models (Figure 12(bottom)). As demonstrated by this example and the flyover in the accompanying video, our method is not limited to placing shapes with a strictly circular projection. Here, the disks used at the analysis and synthesis stages are only coarse bounds for the footprint of plant canopies.

6.3. Computation times

Table 1 provides the computation times and important parameters for our synthesis examples. Timings were taken on an Intel® Core I5 with 4 cores, clocked at 3.3 GHz with 8GB of RAM, and an NVidia GeForce GTX 960 graphics card. It should be noted that the method was implemented in Python on the CPU without GPU acceleration and its runtime could be improved with further optmisation.

Example	Disks	Classes	PCFs	Time
Toy forest (Fig. 1)	460	3	5	2min
Droplets (Fig. 12)	800+	4	10	7min
Food (Fig. 12)	224	5	11	1min
Mediterranean (Fig. 12)	1500+	12	78	12min
Dense ecosystem (Fig. 10)	2300+	9	45	30min

Table 1: *Runtime and distribution parameters for example scenes.*

6.4. Limitations

There are a few limitations to our approach. First, we have prioritised accuracy over speed. As it stands, the method is not suited to interactive design, where users create and edit distributions with cycles of rapid feedback. However, even in such situations there are options available. For instance, a placeholder distribution, derived from a partial initialisation or by directly cutting and pasting the exemplar, could be displayed while refinement takes place in a background thread.

Another limitation, shared by previous point and disk synthesis methods, is an inability to reconstruct regular or otherwise highly constrained distributions, such as the case in Figure 13(top). One issue is that the pressure of constraints tends to push disks to the same position during refinement. To address this we reuse our PCF validity regions to re-check the individual PCFs of generated points after refinement, and re-initialise outliers as required. This strategy, illustrated by Figure 13(bottom), improves results locally – see for instance the leftmost bumps in Figure 13(middle) that are consequently suppressed – but does not solve the issue more generally.

An efficient handling of such regular configurations would likely require PCFs to be enriched with additional structural information. As a first pass, separating PCFs into directional arcs or quadrants, as in WorldBrush [EVC*15], would improve global alignment.

Since our method is specifically designed for disk distributions, a main limitation is that it does not cater for other shapes. Of course, any 2D shape could be approximated by a union of disks and therefore analysed, but there would be no guarantees as to the quality of the synthesised result. Moreover, in such cases it is necessary to account for relative orientation as well as radial distance, and this is beyond our scope.

Lastly, our method considers the input exemplar as a whole under the assumption that the distribution is homogeneous and isotropic. Although our analysis could be used to test homogeneity using cropped sub-regions we have not investigated spatially-varying distributions further. One possible avenue for catering to non-homogeneous distributions would be to extend Roveri et al.'s [ROG17] method for handling two different distributions and densities.

7. Conclusion and future work

We have presented the first method capable of analysing and synthesising general two-dimensional distributions of disks, including those with nested and partially overlapping configurations. We tackled this from a statistical perspective, by defining a new distance metric adapted to disks and specifically designed to distin-

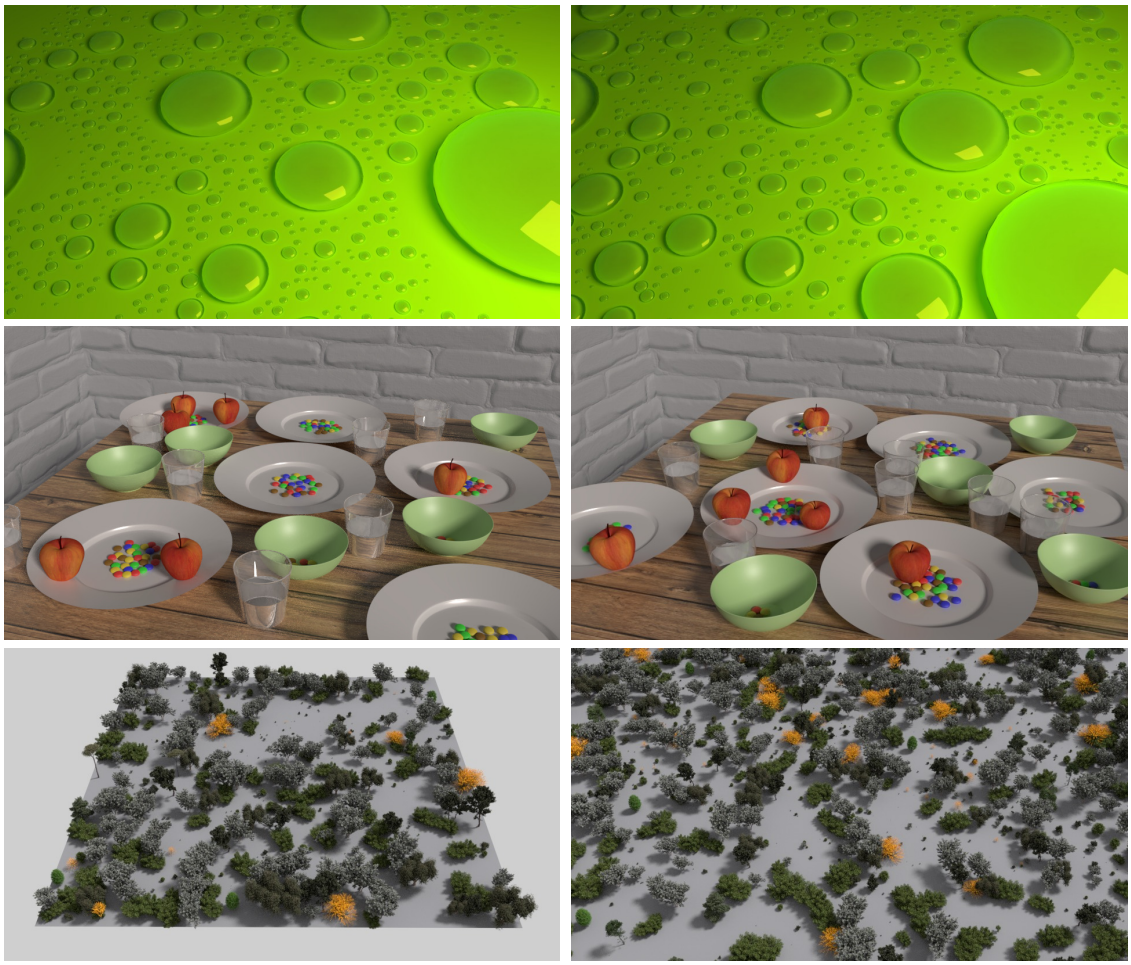


Figure 12: *Application of multi-class disk synthesis to the placement of 3D objects (input exemplars on the left, synthesised results on the right). From top to bottom: Water droplets (4 classes, no overlaps), table set with plates, bowls, glasses and food (5 classes, 6 interdependencies, nested distributions), a Mediterranean biome (12 classes, full dependencies, overlapping distributions).*

guish perceptually salient key configurations, as illustrated by a variety of examples throughout the paper.

Our approach handles multi-class distributions efficiently, and can be further optimised if provided with a user-specified inter-class dependency graph. Moreover, special attention was devoted to the issue of convergence, enabling completion of synthesis even in highly-constrained situations.

There are a number of viable avenues for future work. First, although our method has proven robust in converging to a visually convincing result for all our test cases, providing a general proof of convergence for every realisable PCF (i.e., those for which a disk distribution exists) remains as future work.

Second, in practice, a user may not know, in advance, how best to partition a real-world distribution into classes, nor, having done so, what dependencies to specify. We recently faced just such a challenge when attempting to analyse the distribution of a variety of archaeological remains deposited in cave sediments. Being able to automatically cluster objects into classes and derive a dependency graph on the fly during analysis would be a significant boon to users.

It would be worthwhile extending our framework to shapes other than disks. This could be achieved in several ways. We could use distance functions adapted to other object types, while keeping in mind that additional work might be necessary depending on the geometry, such as keeping track of orientation or length. Another approach would be to represent arbitrary objects as sets of points (similar to [RÖM*15]) and adapt our method to this new structure.

Finally, being able to distribute shapes in 3D as well as in 2D would open up many other applications, ranging from the placement of leaves, flowers and fruit in plants and trees to the instantiation of granular materials. From a theoretical point of view, our metric and the attendant framework are well suited to such a generalisation.

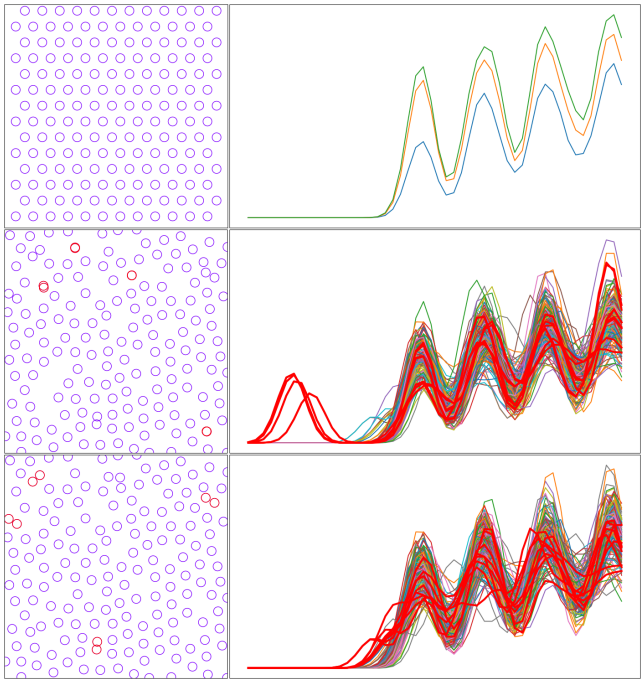


Figure 13: *Failure case on an hexagonal grid: (top) exemplar input, (center) standard disk synthesis, and (bottom) synthesis with outlier removal. The pattern and distance between points are preserved locally but the global layout does not converge to a visually acceptable solution. Disks with the most outlying PCFs are colored in red at each stage. In the center left pattern, these correspond to several superimposed red disks, inducing the initial PCF bumps (center right).*

References

[Aur87] Aurenhammer F.: Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing 16*, 1 (1987), 78–96. 5

[BWWM10] Bowers J., Wang R., Wei L.-Y., Maletz D.: Parallel Poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 166:1–166:10. 2

[CGW*13] Chen J., Ge X., Wei L.-Y., Wang B., Wang Y., Wang H., Fei Y., Qian K.-L., Yong J.-H., Wang W.: Bilateral blue noise sampling. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 216:1–216:11. 2

[CYC*12] Chen Z., Yuan Z., Choi Y.-K., Liu L., Wang W.: Variational blue noise sampling. *IEEE Transactions on Visualization and Computer Graphics 18*, 10 (Oct. 2012), 1784–1796. 2

[DMLG02] Dischler J.-M., Maritaud K., Lévy B., Ghazanfarpour D.: Texture particles. *Computer Graphics Forum 21*, 3 (2002), 401–410. 2

[DSZ17] Deussen O., Spicker M., Zheng Q.: Weighted Linde-Buzo-Gray stippling. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 233:1–233:12. 2

[EVC*15] Emilien A., Vimont U., Cani M.-P., Poulin P., Benes B.: WorldBrush: interactive example-based synthesis of procedural virtual worlds. *ACM Transactions on Graphics 34*, 4 (July 2015), 106:1–106:11. 2, 3, 4, 9, 10

[GGG*16] Guérin E., Galin E., Grosbellet F., Peytavie A., Génevaux J.-D.: Efficient modeling of entangled details for natural scenes. *Computer Graphics Forum* (2016). 3

[GLCC17] Gain J., Long H., Cordonnier G., Cani M.-P.: EcoBrush: Interactive control of visually consistent large-scale ecosystems. *Eurographics 36*, 2 (2017). 2, 3, 6, 7, 9, 10

[HLT*09] Hurtut T., Landes P.-E., Thollot J., Gousseau Y., Drouillhet R., Coeurjolly J.-F.: Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering* (2009), ACM, pp. 51–60. 2

[IMIM08] Ijiri T., Mêch R., Igarashi T., Miller G.: An example-based procedural system for element arrangement. *Computer Graphics Forum 27*, 2 (2008), 429–436. 2

[LD06] Lagae A., Dutré P.: Poisson sphere distributions. In *Vision, Modeling, and Visualization* (2006), pp. 373–379. 3, 4

[LGH13] Landes P.-E., Galerne B., Hurtut T.: A shape-aware model for discrete texture synthesis. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 67–76. 3

[LNW*10] Li H., Nehab D., Wei L.-Y., Sander P. V., Fu C.-W.: Fast capacity constrained Voronoi tessellation. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, ACM, pp. 13:1–13:1. 2

[LWSF10] Li H., Wei L.-Y., Sander P. V., Fu C.-W.: Anisotropic blue noise sampling. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 167:1–167:12. 2

[MAAI17] Martín D., Arroyo G., Alejandro R., Isenberg T.: A survey of digital stippling. *Comput. Graph. 67*, C (Oct. 2017), 24–44. 2

[MALI10] Martín D., Arroyo G., Luzón M. V., Isenberg T.: Example-based stippling using a scale-dependent grayscale process. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2010), NPAR '10, ACM, pp. 51–61. 2

[MWT11] Ma C., Wei L.-Y., Tong X.: Discrete element textures. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 62:1–62:10. 2, 3, 9, 10

[ÖG12] Öztireli C., Gross M.: Analysis and synthesis of point distributions based on pair correlation. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 170. 2, 3, 4, 9, 10

[QCHC17] Qin H., Chen Y., He J., Chen B.: Wasserstein blue noise sampling. *ACM Trans. Graph. 36*, 5 (Oct. 2017). 3

[RHC15] Rohmer D., Hahmann S., Cani M.-P.: Real-time continuous self replicating details for shape deformation. *Shape Modeling International (SMI), Computers & Graphics 51* (2015), 67–73. 2

[ROG17] Roveri R., Öztireli A. C., Gross M.: General point sampling with adaptive density and correlations. *Computer Graphics Forum (Proc. Eurographics) 36*, 2 (2017). 2, 9, 10

[RÖM*15] Roveri R., Öztireli A. C., Martin S., Solenthaler B., Gross M.: Example based repetitive structure synthesis. *Computer Graphics Forum 34*, 5 (2015), 39–52. 3, 9, 11

[Wei10] Wei L.-Y.: Multi-class blue noise sampling. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 79. 3

[WLKT09] Wei L.-Y., Lefebvre S., Kwatra V., Turk G.: State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR* (Munich, Germany, Mar. 2009), Eurographics Association, pp. 93–117. 2

[WW11] Wei L.-Y., Wang R.: Differential domain analysis for non-uniform sampling. *ACM Trans. Graph. 30*, 4 (July 2011), 50:1–50:10. 2

[ZHWW12] Zhou Y., Huang H., Wei L.-Y., Wang R.: Point sampling with general noise spectrum. *ACM Trans. Graph. 31*, 4 (July 2012), 76:1–76:11. 2

[ZZV*03] Zhang J., Zhou K., Velho L., Guo B., Shum H.-Y.: Synthesis of progressively-variant textures on arbitrary surfaces. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 295–302. 2