Structured Shape-Patterns from a Sketch: A Multi-Scale Approach

Submission Number: 40

Abstract

Structured 2D patterns formed by the anisotropic distribution of arbitrary shapes are ubiquitous in nature and man-made environments. They may include both bounded and unbounded (extended fiber-like) shapes. In this work, we address the problem of interactively generating such patterns from a single exemplar, sketched by a user. We build our solution on a new data structure, the *Support Structure Hierarchy*, computed from a multi-resolution analysis of the input exemplar, that encodes the main anisotropy directions at different scales as well as deviations from them. We propose an efficient method based on this structure to synthesize a similar distribution of shapes in an extended 2D domain. The user may also choose to hybridize several input exemplars, by combining structural shapes extracted at different scales. As shown through a user study, our multi-scale solution generates structured shape-patterns that perceptually compete with state-of-the-art methods, were they from learning-based or not. Moreover, our interactive solution, which does not require any precomputation, matches well the needs of an interactive authoring tool, where the user can not only sketch and extend 2D vector textures but also combine them seamlessly.

Index Terms: Computing methodologies—Computer Graphics—Graphics systems and interface—Texturing.

1 Introduction

From fibers and cellular organisms at microscopic scales to seaweeds, schools of fishes, human queues, and alignments of trees or buildings at a larger scale, anisotropic distributions of shapes are ubiquitous in nature and man-made environments. Moreover, such structured shape-patterns have been used extensively in 2D for decorative purposes, from mosaics and wallpapers to the distribution of windows and architectural decorations on building facades. The perceived structure emerges from the anisotropy of these distributions of shapes. In particular, the specific ranges and variances of perceived orientations, both in terms of salient shapes and alignments, convey their unique visual appearance. This work explores the synthesis of such structured 2D shape-patterns from a sketch.

Example-based texture synthesis has already been extensively studied. However, existing methods have mostly focused on point distributions. They have achieved statistical accuracy using noise models, continuous representations of discrete distributions such as pair-correlation or probability-density functions, or neighborhood metrics and energy optimization. The few methods tackling anisotropic distributions of shapes have used multiple point samples or proxy geometries to achieve the analysis and synthesis of structured patterns. To the best of our knowledge, none of them have tackled the case of anisotropic shape-patterns that may include both bounded and unbounded (fiber-like) shapes. While the use of deep learning may be a promising alternative, it requires large training databases and long precomputation (learning) times, which has limited its use so far in interactive design scenarios.

This work tackles the interactive, sketch-based design of anisotropic distributions of shapes in 2D. Given any sketched pattern (a distribution of simple bounded shapes, and/or fiber-like unbounded shapes), our method efficiently synthesizes a perceptually similar, consistent, and non-repetitive distribution of shapes in an extended 2D domain. Note that the input pattern is fully preserved at the synthesis stage. Indeed, contrary to previous methods, the input becomes the central part of the extended texture, while being seamlessly integrated into its larger surrounding. Our solution increases user control and also seems to improve the perceived similarity of the results. One such result generated by our interactive system is shown in Fig. 1 with its interface.

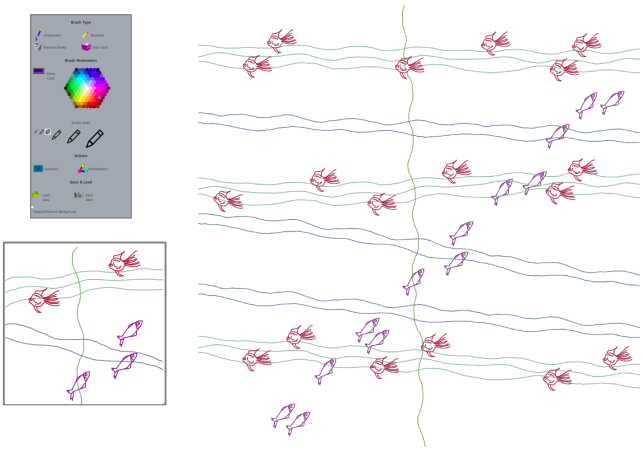

Figure 1: Based on a few perceptual and depiction hypotheses, our method extends an input sketch (bottom left) into a larger vector texture (right). Both bounded (individual fishes) and unbounded shapes (wavy lines) are seamlessly handled. The simple interface (top left) is quick to learn and easy to use.

Real-time analysis and synthesis of distributions require an efficient representation, encoding both local and global correlations between shapes. Our first insight is to introduce a compact encoding for anisotropic distributions, called the *Support Structure Hierarchy*, where individual supporting structures are lead directions of alignments or line skeletons computed from user strokes, all computed at various scales. This representation leads to a particularly simple and efficient multi-scale analysis of the distributions of orientations in the input sketch. It also enables efficient domain extension.

The main challenge at the synthesis stage remains to understand user expectations and the required criteria for perceptual similarity. The (new) case of fiber-like shapes is particularly challenging, because extending fibers that are disjoint in the input exemplar may generate intersections in the extended domain. This could strongly affect our perception of the output as looking different from the input. To support our insights, we formulate a set of perceptual hypotheses to drive our synthesis solution; they were later validated through a user study. In particular, our solution interprets non-intersecting fiber-like strokes as curves that could slightly bend to prevent intersection in the extended domain.

Thanks to its efficiency, we integrated our solution in an interactive authoring tool, where users can progressively test and refine their designs. They can generate a wider variety of vector textures by interactively hybridizing features extracted from several input exemplars, e.g., combining shapes from an exemplar with larger-scale

alignments from another exemplar.

In summary the contributions of our work are threefold, as we introduce:

- a fine-to-coarse analysis method that hierarchically clusters user strokes into a *Support Structure Hierarchy*, based on a new "perceived distance" between line-segments within a domain, depending on both their position and orientation;
- a coarse-to-fine synthesis method that extends the pattern around the input examplar, based on the extracted hierarchy and on a set of perceptual hypotheses validated by a user study;
- an interactive authoring tool, enabling both domain extension and hybridization of structured shape-patterns.

2 Related work

This work addresses 2D sketch-based synthesis of anisotropic discrete distributions. It is related to example-based synthesis that aims at generating an output that minimizes some statistical or perceptual distance with the input, while avoiding artifacts such as salient repetitiveness. We focus below on distributions of 2D shapes, i.e., vector textures formed by arrangements of discrete 2D shapes, and also discuss recent alternatives based on deep learning. We refer the reader to [5, 18] for more general surveys.

Discrete vector textures (or shape-patterns) have been generated by analyzing distributions of the centroids of individual shapes, and then applying a two-stage synthesis for the new centroids, followed by the creation of the associated shapes. The pioneer work of Barla et al. [1] aims at synthesizing stroke patterns. Their method computes a Delaunay triangulation from the centroids to retrieve the input distribution connectivity. During synthesis, they rely on a Lloyd relaxation and some perturbation to generate a new set of points from which the shapes are recovered. In the same mindset, Ijiri et al. [7] explore local growth processes before the relaxation process. These two methods are however limited respectively to quasi-uniform distributions or to 1-ring neighborhoods. To manage more general shape distributions, Hurtut et al. [6] define the input distribution as a combination of Gibbs point processes from which they generate a new arrangement using Monte-Carlo chains. However, all of these methods are unable to analyze and synthesize structured inputs. In particular, they cannot tackle anisotropic distributions of shapes such as elongated ones, nor analyze correlations between shapes, orientations, and spatial alignments.

Instead of using a single centroid point, Ma et al. [12] characterize each input shape by several sample points. They rely on a neighborhood metric and an energy optimization process to insert individual shapes in a predefined output domain. Their approach has been extended to dynamic textures [11], stroke auto-completion [19], and adapted to other texture workflows [2, 8]. While the use of multiple sample points has also been extended to distribution synthesis of arbitrary shapes, these methods only address bounded shapes—as opposed to unbounded shapes—and require post-processing to avoid inter-penetrations at the synthesis stage, which precludes their use in real time.

Rather than sampling the input shapes, Landes et al. [9] propose to simplify them into proxy geometries. They introduce a spatial relationship measure that takes into account space between pairs of shapes and their relative orientations. By extending the stochastic models to point distributions [13,20], their synthesis method successfully maintains distributions of distances and relative orientations of shapes. Although it handles anisotropic distributions, their method does not meet our goals, as it does not offer real-time performance and is limited to distributions of bounded shapes.

In contrast, Roveri et al. [15] present the first example-based distribution synthesis method applicable to both bounded and unbounded shapes. Regardless of their dimension, shapes are decomposed into point samples that are encoded in a functional representation. A similarity measure is defined in the associated functional space to quantify similarity between input and output. Synthesis is achieved in a few minutes through neighborhood matching and energy optimization. As most other neighborhood-based texture synthesis methods, their method requires input patterns with enough repetitions to avoid bad local minima in the optimization, which would distort the synthesized structures. Moreover, contrary to our method, the use of a fixed neighborhood size prevents their method from capturing repetitive structures at different scales.

Deep learning methods have recently been applied to texture synthesis [4, 10, 14, 16, 17, 21]. They show promising results for capturing, at least partially, local and global correlations present in an input exemplar. In particular, Fish et al. [4] enable sketch stylization via the transfer of geometric textural details from different images; this is related to our secondary goal of pattern hybridization. However, most of these frameworks are image-based and do not extend well to the discrete distributions of vector shapes, which is the scope of our work. The method of Tu et al. [17] is closest to our goal of handling vector shape distributions; it characterizes point patterns via a trained VGG network. Our method, based on a simpler but efficient analysis stage, has the advantage of requiring no time-consuming precomputations (which are inherent to deep-learning techniques), while achieving real-time processing of any newly-created input.

3 Overview

3.1 Hypotheses on Depiction & Perception

Extending a sketched pattern in a perceptually similar way requires making some hypotheses about user depiction and perception of the resulting pattern. Our key hypothesis, common to most sketch-based modeling systems, is that users see their input as a *general view* of the distribution they want to create. Therefore, the input is supposed to include all the necessary information, in a perceptually representative way. This led us to three design hypotheses:

$H1$**: Groupings and alignments are meaningful:** All alignments and groupings are intentional.

$H2$**: Repetitiveness is explicit:** All the shapes that a user wants to see repeated in the output, are repeated in the input.

$H3$**: Non-overlapping shapes should remain disjoint:** Shapes that do not overlap the input should not overlap in the output.

These three hypotheses are used as guidelines for our method at the design stage, and then validated by a user study (see Sect. 6).

3.2 Creation and Preprocessing of an Input Sketch

During a sketching session, the user successively draws strokes of any color in a square representing our 2D Input Space (*IS*). See Fig. 1, left. Two different pens are provided to denote bounded and unbounded strokes. The former are limited to the dimensions of *IS*, while the latter are interpreted as extending beyond the input domain, either in both directions if both extremities reach the border of the *IS*, or in a single direction (in case an unbounded stroke does not reach any border of *IS*, we add a segment to connect it to the closest border). The data stored for each stroke are a list of points, a color, a thickness, a type (bounded or unbounded), and a principal direction computed on the fly from the Principal Component Analysis (PCA) on the coordinates for all points of a stroke.

The user may sketch the input pattern in any order. As several stokes can be used to represent a shape, we provide an automatic clustering mechanism, presented next, to identify shapes at the beginning of the analysis stage.

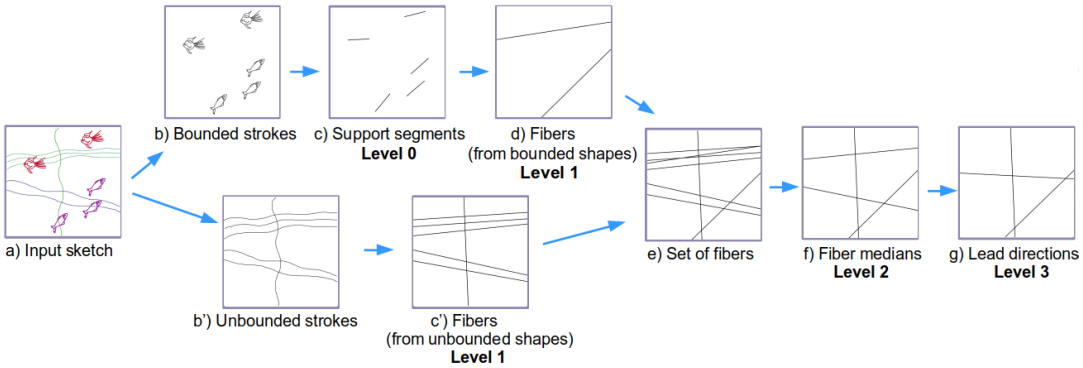

Figure 2: Processing pipeline for the fine-to-coarse analysis of a sketch into a *Support Structure Hierarchy*.

3.3 Processing Pipeline

Multi-scale analysis: Based on Hypothesis $H1$, the analysis stage consists in iteratively extracting a fine-to-coarse hierarchy of support structures (the *Support Structure Hierarchy*) from the input strokes, according to alignments and multi-scale repetitions in the input (see Fig. 2). We first cluster bounded strokes into *shapes* composed of one to several strokes, while each unbounded stoke is considered as an individual shape (**Level 0**). Note that colors are not used in the clustering, enabling the use of several different colors in a given shape. Bounded shapes are then simplified either into a *central point* or a *support segment* depending on their degree of anisotropy (Fig. 2c). Central points and support segments are clustered according to both orientation and position to find alignments, and then grouped into *fibers* (Fig. 2d), forming the **Level 1** of the *Support Structure Hierarchy*. Other fibers are directly extracted from the unbounded strokes (Fig. 2c'). To capture repetitions at a larger scale, *fibers* of similar orientation are clustered into *fiber medians* (**Level 2**, Fig. 2f), which are finally grouped into *lead directions* (**Level 3**, Fig. 2g). During this hierarchical clustering and simplification process, the input domain IS is progressively partitioned into a hierarchy of *ribbons* that express the variability of position of each substructure around its parent structure. This partitioning will be used to allow an adequate degree of variability while avoiding unwanted overlaps at the synthesis stage. See Section 4 for details.

Synthesis stage: Unlike most existing approaches, our method to synthesize distributions consists in directly replicating local and global correlations between the input shapes, encoded by our *Support Structure Hierarchy*. To avoid exact repetitions, this is done by instantiating each structure from top to bottom of the hierarchy while perturbing their positions within adequate *allowed areas*. These areas are computed to prevent overlaps between strokes belonging to the same *lead direction* and at a low cost since no further overlaps detection will be required.

Structures at the top of the hierarchy are first extended to the user-selected larger 2D domain, defined as a radial extension of ratio $k > 1$ of IS. The support hierarchy is then traversed top-down to the individual strokes. At each level, the repetitive structures are repeated within the larger domain, in order to generate the extended structured pattern. This is done in accordance with our design guidelines: a shape that only appears once in the input (such as the vertical see-weed in Fig. 1) will be extended at its extremities in case of an unbounded stroke, but will not be repeated (consistency with $H2$). Moreover, at each level of the hierarchy, allowed areas within ribbons are used to guide the synthesis of substructures while preventing unwanted overlaps (consistency with $H3$). Note that curving some of the supporting structures is necessary to avoid undesired overlaps in the extended the domain, as illustrated by the three green waves that do not overlap with the two blue waves in Fig. 1, right. This process, an original step of our solution justified by our perceptual guidelines, will be detailed in Section 5.

Interactivity and hybridization Thanks to its real-time performance, our method not only allows users to sketch and extend a given shape-pattern, but also to return to the sketching interface to iteratively improve their input. In our authoring system, all identified shapes are recorded in a shapes database, enabling the user to refine the input by adjusting their position, or to reuse them later for another design. Hierarchical structures extracted from the analysis stages of different inputs can also be combined to create a different design, a process called *hybridization* (see Section 6).

4 Fine-to-Coarse Analysis

4.1 Level 0: From Strokes to Shapes

As illustrated in Fig. 2b,b', the bounded and unbounded strokes in the input are analyzed separately, to extract supporting lines that will then be processed in a combined manner.

We consider the unbounded strokes as individual unbounded shapes. In contrast, we extract bounded shapes by clustering the input bounded strokes as follows: we compute the oriented bounding box of each bounded stroke and group these boxes according to their pair-wise distances. We then associate the resulting bounded shapes to a single *central point* or *support segment*, according to an anisotropy threshold. The resulting set of support segments and central points is the first simplification of the input, efficiently encoding the principal directions and approximate positions of the bounded shapes.

4.2 Level 1: From Shapes to Fibers and their Ribbons

We approximate each unbounded shape with a line, called *fiber*, that best matches its principal direction and position. This support line, augmented with a perpendicular thickness to cover the whole shape, is called a *ribbon*. For bounded shapes, finding such *fibers* and *ribbons* requires analyzing anisotropic information such as alignments. We retrieve the support lines of *support segment* and cluster them using the Mean Shift algorithm. We then compute a central *fiber* within each cluster. *Central points* are first clustered by position, before using Principal Component Analysis to compute their main directions of alignment. Representative *fibers* are defined from the centroid of each cluster and these principal directions. Thicknesses are computed for each of these *fibers*, so that the corresponding *ribbon* fully covers the shapes associated with the clustered points or segments.

4.3 Level 2: From Fibers to Fiber Medians

Fibers with similar orientations and close positions are grouped at this stage. Since we focus on anisotropic distributions, we prioritize orientations of fibers over their positions. We first compute the

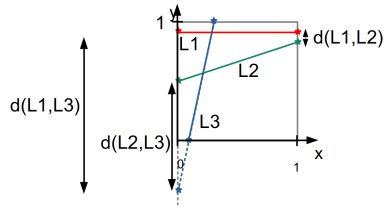

Figure 3: We compute the "perceived distance" between two fibers in a normalized input domain. It is defined as the minimal distance between their intersection points on any of the lines bordering the domain ($X=0, X=1, Y=0, Y=1$), which is extremely fast to compute (for each fiber, only the 4 values $y_{X=0}$, $y_{X=1}$,$x_{Y=0}$, $x_{Y=1}$ are needed). This distance accounts for both position and orientation, and is defined even if the lines intersect in the domain. Here, $d(L1,L2) < d(L2,L3)$, which matches our perception.

histogram of fiber orientations to group those that belong to the same anisotropic distribution. We then refine each cluster using a specific *perceived distance*, which we define as the minimum distance between each fiber intersection points with the domain contour (see Fig. 3). This distance takes into account both position and orientation of the lines: the more parallel and closer two lines are in position, the smaller the distance. Each resulting sub-cluster is stored as a *fiber median* defined as the mean of parameters both in orientation and in position of the clustered fibers (see Fig. 2f). We also store the circular standard deviation associated to each fiber median for later use at the synthesis stage.

Similarly to the previous hierarchy level, a thickness parameter is associated with each newly created fiber median in order to define an associated *ribbon* that fully includes the sub-ribbons of the clustered substructures (see Fig. 4).

4.4 Level 3: From Fiber Medians to Lead Directions

The top level of aggregation in our hierarchical analysis aims at clustering similarly oriented fiber medians. We use the same clustering process as at the previous level of the hierarchy. Each cluster is represented by a *lead direction*, defined using the average of the clustered elements in orientation and position (see Fig. 2g). As at the previous hierarchy level, *ribbons* are defined by associating a thickness parameter to each *lead direction*, as to includes ribbons around clustered median fibers. As a result, the input space *IS* is divided into nested ribbon-like structures (see Fig. 4).

4.5 Computing the allowed displacement areas

The last step of the analysis stage consists in computing the available space around each clustered shape or ribbon, within their parent structure in the hierarchy. We call this space the *allowed displacement area*, since it will be used at the synthesis stage for adding random displacements to repeated structures, enabling to provide visual diversity while avoiding unwanted overlaps between shapes.

Displacement areas for ribbons Starting at the top of the hierarchy, we recursively decompose each ribbon, using splitting lines that are parallel to its main axis, and evenly split the empty space between neighboring, non-overlapping sub-ribbons (defined as ribbons around one of the clustered sub-structures). The distance between neighboring sub-ribbons (ie. the minimal distance between their contents), used to position these lines, is computed while considering a toroidal topology for *IS*. Based on this distance, two lines are evenly generated between the neighbouring ribbons, to define the limits of extended regions for each of them, as well an empty space between them.

This decomposition results into a displacement region around each sub-ribbon, and an given distance, called *gap* between them. Note that since the parent orientation was used for this decomposition, the sub-ribbons generally have a slightly different orientation. Moreover, they are not necessarily centered in the associated displacement region (see Fig. 4) a) and b).

Finally, the minimum and maximum gaps values are stored in the parent structure, together with the set of displacement areas associated to sub-ribbons.

Displacement areas for bounded shapes These rectangular regions, depicted using dashed lines in Fig. 4 d), represent the areas within the fiber ribbon of a bounded shape in which its bounding box will be allowed to move during instantiation. Their two axes (x,y) respectively correspond to the direction of the associated fiber median and its orthogonal direction. The allowed perturbation along x (tangent to the direction) is set to half of the distance to the next bounding box of a bounded shape, which ensures that overlaps will always be prevented at the synthesis stage. The allowed perturbation along y is set so that the bounding box can cover the whole associated fiber-median ribbon. Again, these computations are done while considering a toroidal topology for the input space *IS*. Therefore, the computed displacement areas can then expand outside *IS* (see the orange areas in Fig. 4 d), which is less restrictive when extending the pattern to a larger input domain.

5 Synthesis of an extended shape-pattern

To enable a seamless exploration of a larger 2D domain by simply zooming out after sketching, our objective is to keep the user-drawn strokes within *IS* while extending and repeating them in a larger output space *OS* (defined as an expansion of *IS* by a ratio $k > 1$). This is done through a coarse-to-fine process in which the elements stored in the *Support Structure Hierarchy* are extended to *OS* and repeated if necessary.

Extension and repetition of lead ribbons: According to $H2$ (see Sect. 3.1), *Lead directions* consisting of only one *fiber median*, (such as the vertical *lead direction* in Fig. 5) should not be repeated. Therefore, we simply extend them as well as their unbounded child structures to span the whole *OS*.

For the remaining *lead directions* (corresponding to repeated substructures in the input), we perform the same extension to the whole *OS*, but also generate new copies of the structure in the remaining space, through an efficient randomized repetition procedure, as follows. For each *lead ribbon*, we start from a displacement area with a single neighbour, randomly generate a new gap using values in the recorded range, and generate the next displacement area as to randomly clone one of the exiting ones (ie. using the same width). We apply this technique to progressively fill *OS*. The randomness in the gap values between displacement areas for sub-ribbons generates different *lead ribbon* configurations, and therefore different outputs from the same input (see Fig. 5).

Fiber medians ribbons will now be generated within the newly extended and repeated lead ribbons, as presented next, at the cost of slightly bending some of them as well as their child structures, if they happen to overlap when extended to *OS*.

Repetition of fiber medians and ribbons For each newly generated *displacement area*, we synthesize its *fiber median* by first copying the parameter values of the original ribbon. We then use the circular standard deviation on the medians' orientations computed during the analysis stage (Sect. 4.3) to perturb its orientation. We also perturb the position of its centroid to place it in the middle of the *displacement area*. While the middle part of generated *ribbons* are guaranteed to remain within their *lead ribbon*, this is not necessarily the case when they extend to *OS*, as illustrated in Fig. 6 (left). When this occurs, we slightly bend a *ribbon* and its *fiber median* (see Fig. 6 (right)) to make it fit entirely inside its allowed displacement area.

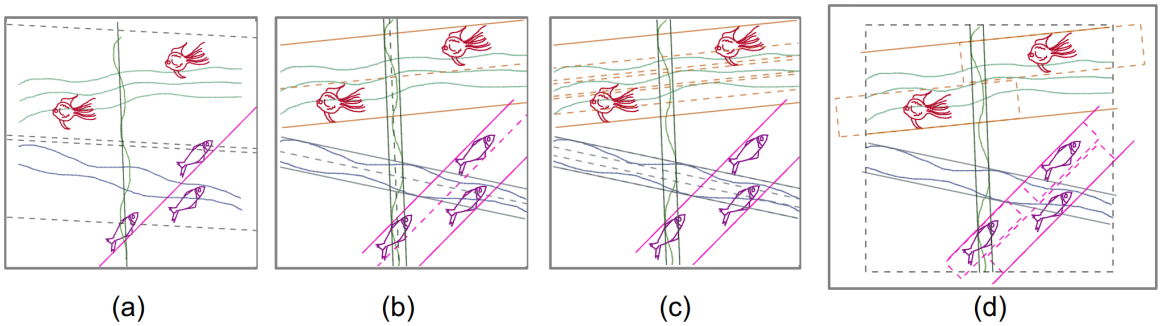

Figure 4: Input domain partitioning: (a) lead ribbons, each between each a pair of dashed lines; (b) ribbons (solid lines) around the fiber medians (dashed lines); (c) the sub-ribbons (dashed lines) inside the ribbons (in plain); (d) the displacement areas, delimited by the dashed areas.

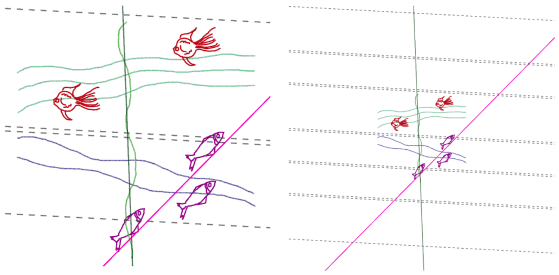

Figure 5: (left) Allowed displacement areas between dashed lines, based on lead directions; (right) Randomized repetition and propagation of lead ribbons.

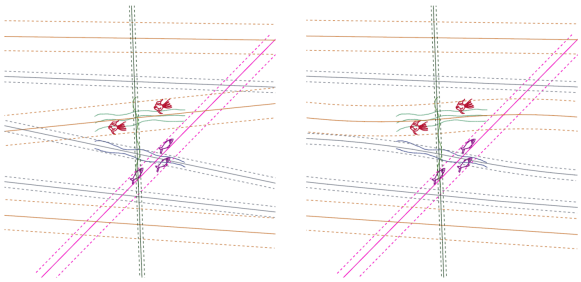

Figure 6: Fiber medians and ribbons repetition in OS: (left) without any bending; (right) with slight bending.

Avoiding overlaps by bending structures Inspired by physical properties of (real) fibers, we consider the following assumption: the thinner the *ribbon*, the more flexible it may be. This can be formalized through the equation $R = \tau w$, relating the curvature radius R to the *ribbon* width w and a stiffness parameter $\tau \in \mathbb{R}^+$.

In case of overlap, each *ribbon* will intersect twice with its *lead ribbon*.For symmetry reasons, we then bend both sides of the *ribbon*, even if one of the intersection regions is out of the output domain. To preserve continuity between the original borderlines of the *ribbon* and their curved version, we consider the midpoints (M_1 and M_2 in green in Fig. 7) between a projected point and the other intersection point as inflection points. For each of these inflexion points (say M), the key idea is to find the arc of circle C that passes through M and remains inside the *lead ribbon* as illustrated in Fig. 7. The details of this computation are provided in the associated Supplementary material.

The same bending process is applied to the children *sub-ribbons*, in order to fit them inside their parent curved *ribbon*.

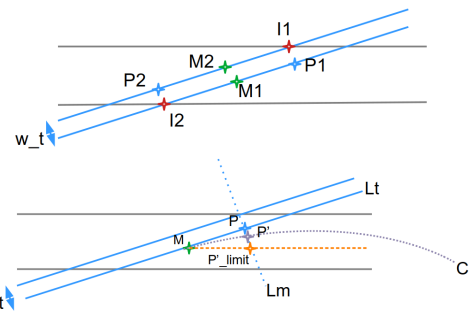

Figure 7: (Top) A ribbon has two intersections ($I1$ and $I2$) with its lead ribbon. (Bottom) The ribbon is bent to remain within its lead ribbon.

Shape distribution synthesis The final step is to synthesize new shapes within each extended or newly created *fiber ribbon*.

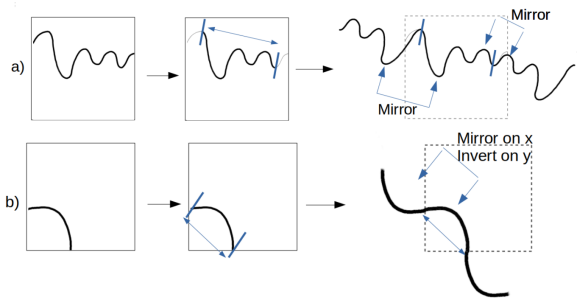

Figure 8: Extension of unbounded strokes: (a) curve; (b) arc.

a) Unbounded shapes We define four unbounded stroke categories (lines, rays, arcs and curves) that respectively stand for perfectly linear unbounded strokes, half-lines, unbounded strokes with a single curvature extremum in IS, and unbounded strokes with more than one curvature extrema. We start by extending these unbounded shapes to OS along their *fiber* direction, which is trivial for lines and rays. Arcs are extended through an alternative mirror duplication that leads to a smooth sinusoidal curve. Curves are first cut at their first and last extrema. Then, we alternatively duplicate the mirror version of the curve-segment to extend it to OS as illustrated in Fig. 8. These extended strokes are stored in the local frame of their corresponding *fiber*. They will therefore be automatically repeated and curved if needed through the repetition process of their parent structures in the hierarchy. The resulting curved structures are shown for different sizes of OS in Fig. 9.

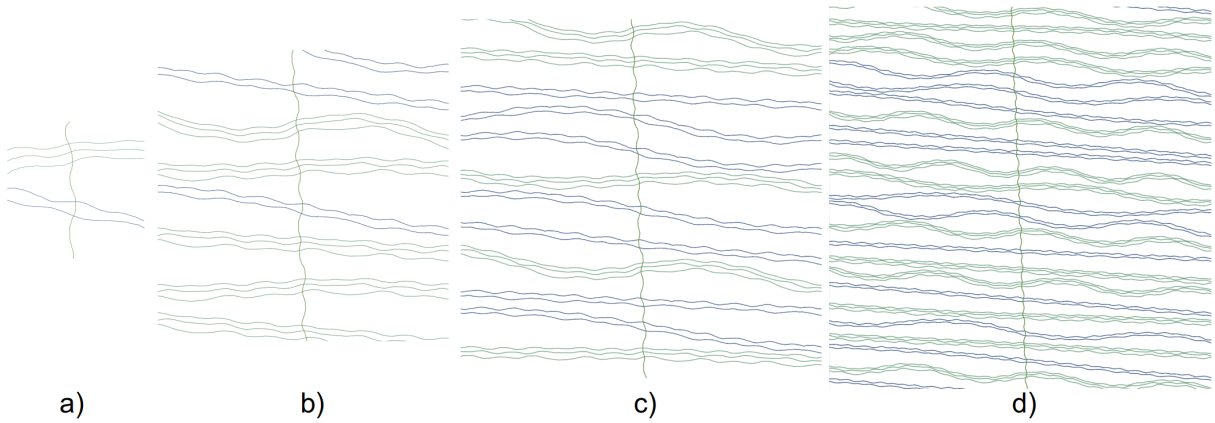

Figure 9: Variation of lateral ratio (k) for unbounded stroke distribution: (a) input; (b) $k = 3$; (c) $k = 5$; (d) $k = 10$. Note that these results have been scaled to fit in the figure.

b) Bounded shapes We process bounded shapes by first, iteratively repeating their representative *support segments* or *central points* along their extended *fiber*, while using the previously computed *displacement areas* to perturb their positions, and drawing the shapes in the resulting local frames. We then reuse their local positions with respect to their *fiber* to repeat them within the parent *fiber medians ribbons*, but with randomly modified positions within in the authorized *displacement areas*. See Figure Fig. 10.

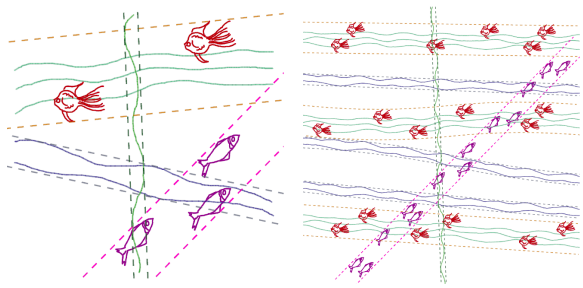

Figure 10: Synthesis outline: (left) input with ribbons between pairs of dashed lines; (right) shape repetition within the extended and synthesized ribbons.

Avoiding residual overlaps: Given that repetitions in different *lead directions* are computed independently, *lead ribbons* with different orientations may naturally intersect. This may lead to perceptual artifacts if these *lead directions* both contain initially non-overlapping bounded shapes. Indeed some undesirable overlaps may occur in the output. We use an *AABB* tree to partition *OS* and efficiently detect overlaps between *displacement areas* of bounded shapes. In such case, we restrict the corresponding *displacement areas*. If this strategy fails (not enough space to insert a shape), we do not instantiate it (see Fig. 11 (b) for such a challenging example).

6 Results and Discussion

6.1 Interactive authoring system

Our prototype system is implemented in WebGL. Creating and extending highly structured vector patterns is made easy by our method, as shown in Fig. 11. In addition to the main sketching and texture expansion interface, the user may store and reuse complex shapes, such as the two categories of fishes in Fig. 1. The use of our system for creating a complex sketch, inspired from biology, is illustrated in Fig. 12.

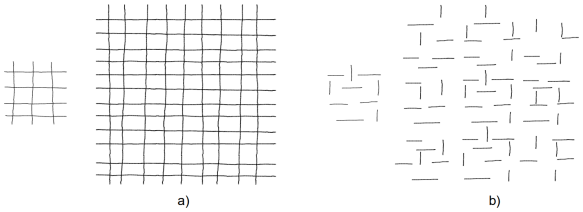

Figure 11: Our synthesis method maintains the perceived regularity of structured distributions (known to be hard to handle) in both cases of unbounded and bounded strokes.

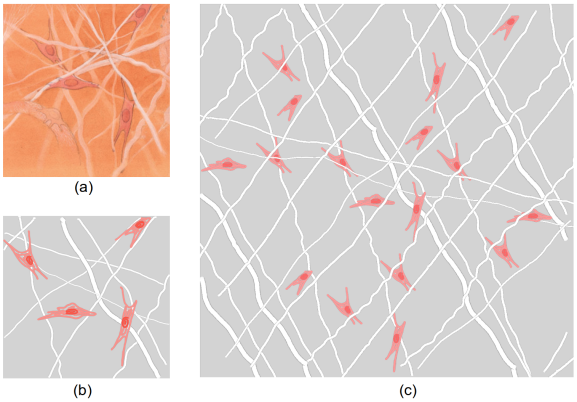

Figure 12: a) Biological illustration depicting cells that navigate in a distribution of fibers; b) Input sketch inspired from (a); (c) Result.

In addition, several input shape-patterns can be interactively combined to create a hybrid one, as follows: Thanks to the *Support Structure Hierarchy*, the user can select the desired level of hierarchy from two different input shape-patterns, and combine them and create an hybrid result. We rely on the fact that our *Support Structure Hierarchy* encodes the input data into structures that are defined in the local frames of their upper structure ribbons, themselves characterized by a main direction and a width. Therefore, consistent patterns can be generated while the input shapes, fibers, fiber-medians or lead directions are changed. Such an hybridization is shown in Fig. 13.

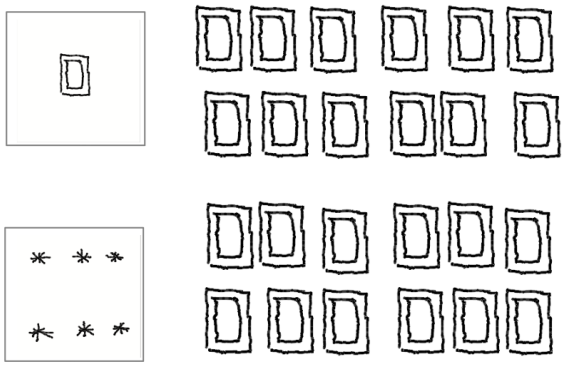

Figure 13: Hybridization example, where two input shape-patterns (left) are combined to create a new result (right)

6.2 Comparison with previous work

We compared our results with both distribution-based and deep-learning-based methods for generating vector textures from examples. Since most classical methods are limited to distributions of bounded shapes, we restricted comparison to this sub-case (see Fig. 14). Since our results seemed close to those of the best classical method, Landes et al. [9], we selected this method for further comparison in our user study (see below).

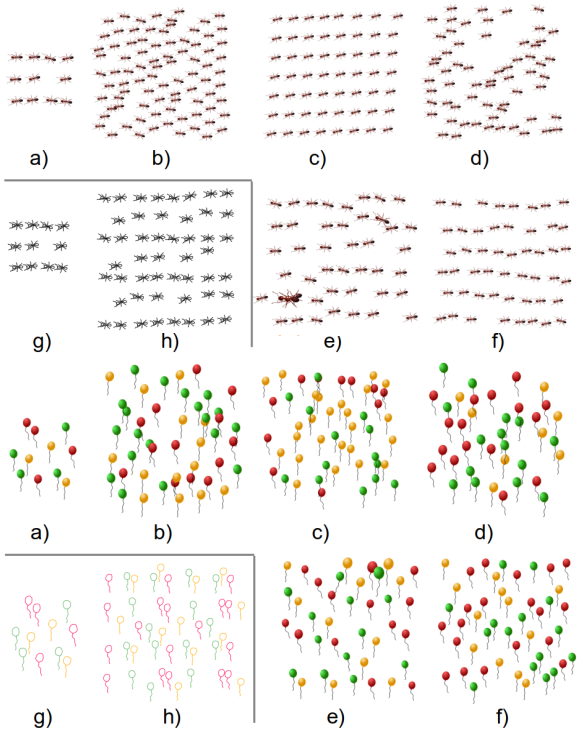

Figure 14: Comparison with distribution-based methods: (a) image input; (b) [1]; (c) [7]; (d) [6]; (e) [12]; (f) [9]; (g) our corresponding sketched input; (h) our result.

We also tried our method on examples presented as failure cases in previous papers, such as Fig. 11 (a failure case of [3]) and Fig. 15 (a failure case of [2]). In both cases, our solution was robust and managed to maintain the regularity of the structured input for both bounded and unbounded shapes.

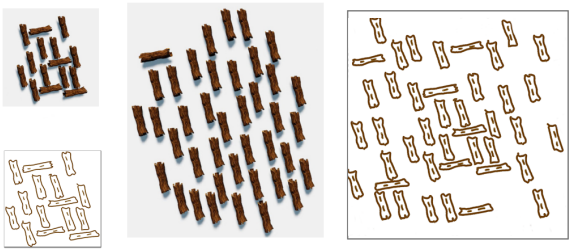

Figure 15: Challenging structured distributions: (a) input; (b) sketched representation of the input ; (c) result of [2]; (d) ours.

Lastly, we compared our results with those of Tu et al. [17], the only deep-learning method tackling point distributions (see Fig. 16). Although our method is interactive and does not require any precomputation stage (in contrast to the hours of training of deep learning methods), the quality of our results looks almost as good (we get more artefacts in the first example, while [17] gets more of them in the second one).

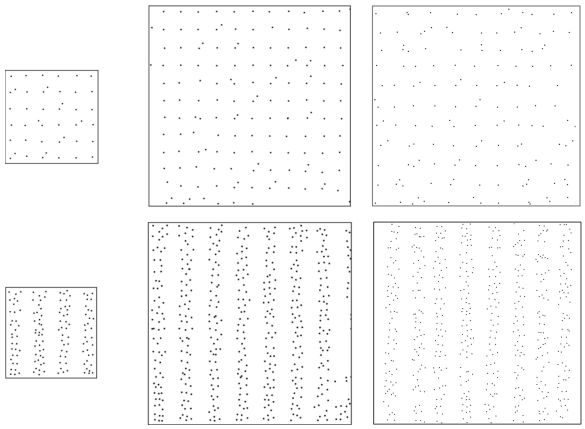

Figure 16: Comparison with the closest deep learning method [17]: (left) input distribution; (middle) results from Tu et al. [17]; (right) our results.

6.3 User study

We carefully designed an online user study to validate the perceptual hypotheses presented in Sect. 3.1, as well as the perceived quality of the extended textures we generate (See our supplemental document for screenshots and detailed results).

The study was conducted by 35 users, from 19 to 61 years old, including 22 males, 9 females and 4 genders unspecified. 14 had an intermediary or expert experience in digital design and 9 as traditional designers. It was composed of two parts: an interactive drawing session, and a comparison session. In the drawing session, users were asked to manually draw an extended texture from a given input pattern. In the comparison session, users were asked to select the closest result from a given a 2D input. Each experiment lasted around ten minutes, most of which in the drawing session.

Among the guidelines to validate, $H1$ (Groupings and alignments are meaningful) was validated by the drawing session, where 97% of the users preserved the grouping of fiber-like shapes and 76% of the users respected the anisotropy directions of bounded shapes in their drawings. $H2$ (repetitiveness is explicit) was validated by most users during the comparison session, and was also observed in the user's drawings as those of Fig. 17. $H3$ (non-overlapping shapes should remain disjoint) was validated as well by the users' drawings,

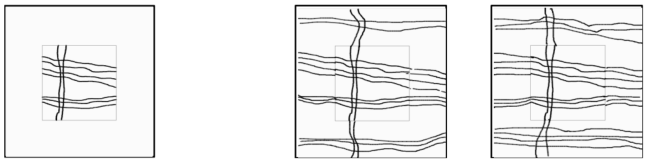

Figure 17: (Left) An example of input for the drawing session; (Right) Example of sketches created by different users

with 73% of overlapping-free drawings when it was the case in the input.

As part of the comparison session, user were asked to choose between our extended textures and the generated ones from Landes et al. [9] (shown in random order, and using the same shape depiction), for the ants and the balloons examples of Fig. 14. Respectively 86% and 77% of users preferred our results. We attribute this unexpectedly good results to the fact we keep the exact input pattern at the center of the generated texture, while seamlessly extending it sideways.

6.4 Performance

The following table was computed using the Google Chrome runtime performance on an Intel(R) Core(TM) i7-7920HQ CPU at 3.10GHz. The second column gives the number of points in the input example, and then the time in milliseconds of respectively the analysis and the synthesis. Note that the synthesis has been performed with a ratio of $k = 3$. As can be observed, the overall computation times take less than a fraction of second.

Example	# Points	Analysis	Synthesis
Fishes (Fig. 1)	7699	73ms	111ms
Biology (Fig. 12)	3094	27ms	86ms
Ants (Fig. 14 top)	9447	134ms	233ms
Balloon (Fig. 14 bottom)	4034	42ms	68ms
Trunks (Fig. 15)	3164	68ms	83ms

6.5 Discussion and limitations

The specificity of our method compared to previous work is that it does not require any neighborhood matching at the synthesis stage, given that our hierarchical representation already captures correlations. This leads to real-time performances, suitable to our application context.

However, the notion of anisotropy being central in our method, it makes it unsuitable to synthesize isotropic distributions. Indeed computing meaningful fiber directions then more difficult, which prevents further extraction a structural hierarchy. While this is the main limitation of our framework. Therefore, in an authoring tool, our solution should be compared with a former method handling isotropic distributions.

A useful extension would be to give the user the possibility of choosing among different perceptual hypotheses, for instance, regarding explicit repetitiveness, which may not be always desired, or to enable overlaps between bounded and unbounded shapes. For instance, in the biology illustration of Fig. 12, the cells depicted in pink/red should remain attached to the underlying fibers, which is not the case in our solution. Indeed, we never cluster bounded strokes with unbounded strokes, even if they overlap. This could easily be added as an option. Generating waving curves rather than having unbounded strokes intersect a design choice, which could also be disabled be the user if necessary.

Fig. 18 presents a failure case for our solutions, as bounding boxes around strokes overlap, while these strokes should not be grouped. To solve this problem, we could allow the choice between bounding-boxes-based distances and centroid-based distances as clustering criteria for bounded strokes. This would facilitate the processing of any dense distribution.

As a last limitation, we do not consider branching curves, which could be another kind of unbounded elements in the input texture pattern. Since it processes the input strokes one by one, our current method would not capture the branching. The input would be split into isolated curves, which would probably intersect when repeated in the output. In addition, even if grouping was forced, our current way of representing unbounded elements using a single linear ribbon of given width would fail. Therefore, seamlessly extending patterns that include such branching structures remains an open problem.

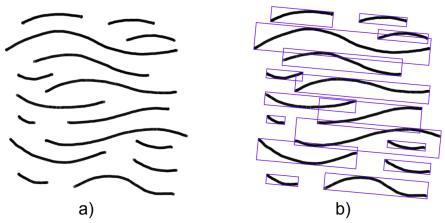

Figure 18: Input example from [12], where our current stroke clustering method fails.

7 Conclusion

Motivated by the interactive design of vector textures, we presented a multi-scale method to efficiently extract anisotropic properties from an input pattern, and seamlessly extend it to a larger 2D domain. While our method runs in real time, visual quality of results compare well with those of state-of-the-art vector texture generation methods, including those requiring either higher computational time and/or training data to learn from. M

The new *Support Structure Hierarchy* we introduced is crucial to our method. Extracted at the analysis stage based on a new *perceived distance* between the salient anisotropic structures within in the input domain, it allows to capture and reproduce the multi-scale structures in an efficient way, while maintaining a good level of visual diversity in the synthesized distribution of shapes. In terms of the interface, our system can be used to quickly design new vector textures by interactively creating new patterns or combining existing ones.

Future work While our solution is well suited to most structured shape-patterns, our use of linear ribbon-like shapes to capture multi-scale anisotropy prevents us from handling more complex, branching structures. Addressing this specific case would be an interesting avenue for future work. In addition, a challenging open problem would be to generate a 3D texture from the 2D exemplar interactively sketched by the user. In cases such as biological illustrations, this would enable users to navigate in a 3D structure created from the sketch, leading to a better understanding the depicted environment.

References

[1] P. Barla, S. Breslav, J. Thollot, F. Sillion, and L. Markosian. Stroke Pattern Analysis and Synthesis. *Computer Graphics Forum*, 25(3):663–671, 2006. doi: 10.1111/j.1467-8659.2006.00986.x

[2] T. Davison, F. Samavati, and C. Jacob. Interactive example-palettes for discrete element texture synthesis. *Computers & Graphics*, 78:23 – 36, 2019. doi: 10.1016/j.cag.2018.10.016

[3] P. Ecormier-Nocca, P. Memari, J. Gain, and M.-P. Cani. Accurate Synthesis of Multi-Class Disk Distributions. *Computer Graphics Forum*, 38(2):157–168, 2019. doi: 10.1111/cgf.13627

[4] N. Fish, L. Perry, A. Bermano, and D. Cohen-Or. Sketchpatch: Sketch stylization via seamless patch-level synthesis. *ACM Trans. Graph.*, 39(6), nov 2020. doi: 10.1145/3414685.3417816

[5] L. Gieseke, P. Asente, R. Měch, B. Benes, and M. Fuchs. A survey of control mechanisms for creative pattern generation. In *Computer Graphics Forum*, vol. 40, pp. 585–609. Wiley Online Library, 2021.

[6] T. Hurtut, P.-E. Landes, J. Thollot, Y. Gousseau, R. Drouillhet, and J.-F. Coeurjolly. Appearance-Guided Synthesis of Element Arrangements by Example. In *Proc. Symposium on Non-Photorealistic Animation and Rendering*, NPAR '09, p. 51–60, 2009. doi: 10.1145/1572614.1572623

[7] T. Ijiri, R. Měch, T. Igarashi, and G. Miller. An Example-based Procedural System for Element Arrangement. *Computer Graphics Forum*, 27(2):429–436, 2008. doi: 10.1111/j.1467-8659.2008.01140.x

[8] R. H. Kazi, T. Igarashi, S. Zhao, and R. Davis. Vignette: Interactive Texture Design and Manipulation with Freeform Gestures for Pen-and-Ink Illustration. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, p. 1727–1736, 2012. doi: 10.1145/2207676.2208302

[9] P.-E. Landes, B. Galerne, and T. Hurtut. A shape-aware model for discrete texture synthesis. *Computer Graphics Forum*, 32(4):67–76, 2013. doi: 10.1111/cgf.12152

[10] T. Leimkühler, G. Singh, K. Myszkowski, H.-P. Seidel, and T. Ritschel. Deep Point Correlation Design. *ACM Trans. Graph.*, 38(6), Nov. 2019. doi: 10.1145/3355089.3356562

[11] C. Ma, L.-Y. Wei, S. Lefebvre, and X. Tong. Dynamic Element Textures. *ACM Trans. Graph.*, 32(4), July 2013. doi: 10.1145/2461912.2461921

[12] C. Ma, L.-Y. Wei, and X. Tong. Discrete Element Textures. *ACM Trans. Graph.*, 30(4), July 2011. doi: 10.1145/2010324.1964957

[13] A. C. Öztireli and M. Gross. Analysis and Synthesis of Point Distributions Based on Pair Correlation. *ACM Trans. Graph.*, 31(6), Nov. 2012. doi: 10.1145/2366145.2366189

[14] P. Reddy, P. Guerrero, M. Fisher, W. Li, and N. J. Mitra. Discovering pattern structure using differentiable compositing. *ACM Trans. Graph.*, 39(6), Nov. 2020. doi: 10.1145/3414685.3417830

[15] R. Roveri, A. C. Öztireli, S. Martin, B. Solenthaler, and M. Gross. Example Based Repetitive Structure Synthesis. *Computer Graphics Forum*, 34(5):39–52, 2015. doi: 10.1111/cgf.12695

[16] O. Sendik and D. Cohen-Or. Deep Correlations for Texture Synthesis. *ACM Trans. Graph.*, 36(5), July 2017. doi: 10.1145/3015461

[17] P. Tu, D. Lischinski, and H. Huang. Point Pattern Synthesis via Irregular Convolution. *Computer Graphics Forum*, 38(5):109–122, 2019. doi: 10.1111/cgf.13793

[18] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. State of the Art in Example-based Texture Synthesis. In *Eurographics 2009 - State of the Art Reports*, 2009. doi: 10.2312/egst.20091063

[19] J. Xing, H.-T. Chen, and L.-Y. Wei. Autocomplete Painting Repetitions. *ACM Trans. Graph.*, 33(6), Nov. 2014. doi: 10.1145/2661229.2661247

[20] Y. Zhou, H. Huang, L.-Y. Wei, and R. Wang. Point Sampling with General Noise Spectrum. *ACM Trans. Graph.*, 31(4), July 2012. doi: 10.1145/2185520.2185572

[21] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang. Non-Stationary Texture Synthesis by Adversarial Expansion. *ACM Trans. Graph.*, 37(4), July 2018. doi: 10.1145/3197517.3201285