By-Example Synthesis of Vector Textures

Christopher Palazzolo, Oliver van Kaick and David Mould†

School of Computer Science, Carleton University, Canada

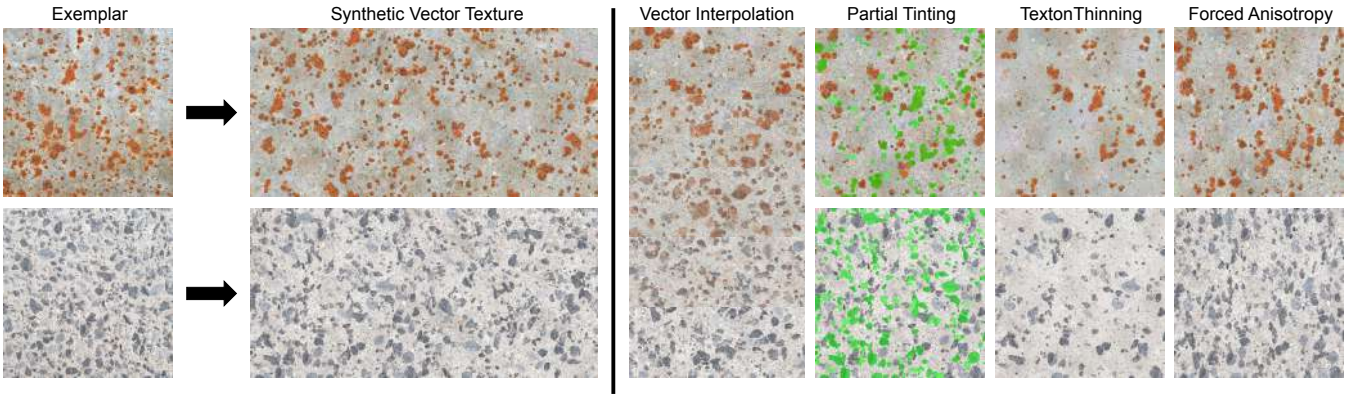

Figure 1: *Left: vector texture synthesis from raster input. Right: post-processing operations facilitated by our vector representation.*

Abstract

We propose a new method for synthesizing an arbitrarily sized novel vector texture given a single raster exemplar. In an analysis phase, our method first segments the exemplar to extract primary textons, secondary textons, and a palette of background colors. Then, it clusters the primary textons into categories based on visual similarity, and computes a descriptor to capture each texton's neighborhood and inter-category relationships. In the synthesis phase, our method first constructs a gradient field with a set of control points containing colors from the background palette. Next, it places primary textons based on the descriptors, in order to replicate a similar texton context as in the exemplar. The method also places secondary textons to complement the background detail. We compare our method to previous work with a wide range of perceptual-based metrics, and show that we are able to synthesize textures directly in vector format with quality similar to methods based on raster image synthesis.

CCS Concepts

• ***Computing methodologies*** → ***Texturing;***

1. Introduction

Texture synthesis is a long-studied problem, with raster images receiving a great deal of attention. Vector textures, however, are much less explored. One advantage of vector textures is editability [OBW*08]: numerous existing operations can be used to manipulate [LTH86] and remove elements from a vector texture. Such operations are much more difficult to apply to raster images. Figure 1 shows two examples of synthesized vector textures along with a number of editing operations that are easily applied to textons, but much harder to apply to a raster image.

Previous work has introduced methods for synthesizing simple polygon distributions through polygon packing [SKA21, XWL*23] or synthesis of vector patterns from exemplars [TWZ22, TWY*20, HLT*09, AKA13]. However, these methods either take simple polygons as input or require the input to be already in vector format with separated vector elements. Other work explores generating vector content that matches a given raster image, but does not permit extrapolating a given exemplar image into a larger texture of a more stochastic nature [LLGRK20]. One might imagine generating vector textures from raster input by first transforming the raster input into vector format with existing tools [SLWS07, LLGRK20], then applying the methods discussed above to generate a new vector texture from the input. However, vectorization tools often create

† Corresponding Author

output with many overlapping polygons exploiting transparency to recreate the input, rather than flatter output that is easier to edit.

We propose a novel algorithm for synthesizing vector textures from a given exemplar in raster format. Our method is aimed at textures composed of distinct, irregular textons with uneven spacing and lacking global structure, e.g., rust and heterogeneous materials (Figure 1). We assume that the textured surfaces are flat, without prominent shadows or self-occlusions. From this input, we generate textures composed of solid-colored polygons on a background gradient field.

To synthesize a texture from an input raster exemplar, we build a texton hierarchy through segmentation and extract a background gradient field on which the textons reside. The texton hierarchy can be used to synthesize a novel vector texture of any size. We synthesize a background gradient over a target output domain, and then place textons over the background to replicate a similar texton context as in the exemplar. Our code can be downloaded at https://github.com/ChrisPGraphics/ByExampleSynthesisOfVectorTextures

Our vector textures closely resemble the input exemplars. Our method is competitive with methods that operate directly on the raster domain. Beyond synthesizing textures, we show how a pure vector result facilitates editing operations that are difficult to accomplish with raster images. Such operations include texture interpolation where the textons of one result (represented as solid-colored polygons) can move and warp into the textons of another. This is in contrast to naïve image interpolation by cross-fading between two images. The supplemental material includes a video of a temporal interpolation; a spatial interpolation is shown in Figure 1.

In summary, this paper makes the following contributions:

- We describe a method to convert a natural raster image to a vector representation containing discrete textons and a background gradient.
- We propose a method to synthesize novel vector textures given our vector representation.
- We demonstrate proof-of-concept editing operations made possible by our discrete texton representation.

2. Related Work

Our method converts a natural image to pure vector format and then synthesizes a novel vector texture from it without user guidance. To our knowledge, no other methods directly synthesize a vector texture from a natural raster image. While our work considers natural textures— i.e., irregular textures with few or no repeated elements—existing work on vector texture synthesis applies to line drawings and textures that are already in vector format, often with distributions of identical elements.

Vector Texture Based Methods. Jagnow et al. [JDR04] propose a parametric method to generate a limited range of solid textures. Barla et al. [BBT*06] describe an algorithm that produces a vector texture, but requires the exemplar to be a distribution of vector elements. Their method also requires the user to perform manual interactive parameter tuning. Following this, Passos et al. [AdPWS10], Landes et al. [LGH13], and Ma et al. [MWLT13] propose fully-automated algorithms with the same objective.

Both Tu et al. [TWZ22,TWY*20] and Hurtut et al. [HLT*09] use polygons to produce novel textures, assuming the example texture is already comprised of polygons (for example, as an SVG file). The approach taken by these methods is based on clustering. Tu et al. group visually similar textons, whereas Hurtut et al. build clusters of samples for each vector element using local neighborhoods. Both methods attempt to identify the underlying local distribution of clusters and use this to synthesize novel textures which are visually similar. AlMeraj et al. [AKA13] propose an equivalent to the raster patch-based texture synthesis for vector images, also assuming input primitives are provided in vector form.

Saputra et al. [SKA21] and Xue et al. [XWL*23] describe vector texture synthesis as a polygon packing problem. These methods try to arrange a set of polygons in a finite area while avoiding overlap and reducing empty space. These methods were designed for simple polygon distributions, and may not perform well on a vectorized natural image. Closer to our problem domain, Qian et al. [QSS*22] generate a vector texture from a raster image, but are limited to exemplars with simple patterns consisting of only a few colors.

Non-parametric Methods. Efros and Leung [EL99] propose a non-parametric sampling approach where pixels are copied from the exemplar directly. Later, Efros and Freeman [EF01] proposed Image Quilting, copying entire patches instead of individual pixels. Such approaches have extremely high quality outputs, but sometimes exhibit visible repetition. Our approach can be considered non-parametric owing to our texton reuse. We provide a quantitative comparison to Image Quilting.

Like our work, Galerne et al. [GGM11] and Heitz and Neyret [HN18] propose effective algorithms for synthesizing a narrow range of textures. Galerne et al.'s random-phase approach is designed for microtextures; Heitz and Neyret's Gaussianization method covers a similar class of textures, although the texture types for which their method is successful are not clearly characterized.

Other methods that work purely in the raster domain include Praun et al. [PFH00], Dischler et al. [DMLG02], and Dischler et al. [DZ06]. These methods require manual segmentation to be performed by the user before synthesis can begin. Liu et al. [LWX*09] and Guehl et al. [GAD*20] require the user to provide a manually created binary mask. Gilet et al. [GDG12] use an intermediate vector representation, which is interpolated to produce a raster result. Their method requires interactive refinement from the user to get the desired results and so it is not fully automatic.

Neural Network Based Methods. The literature on using neural networks for texture synthesis is extensive. Influential works such as Gatys et al. [GEB15], Zhou et al. [ZZB*18], and Jetchev et al. [JBV17] show how networks are able to create high resolution textures from a single smaller exemplar. Subsequent papers enhanced the ability of neural network-based synthesis through novel loss functions and architectures.

Bergmann et al. [BJV17] propose a Periodic Spatial GAN (PSGAN), improving on previous GAN-based texture synthesis approaches by using only convolutional layers in their network; their method can produce periodic textures, with higher quality than

previous GAN-based approaches. We compare with this method as representative of the state of the art of GAN-based methods. Shocher et al. [SBII19] and Shaham et al. [SDM19] train a GAN on small patches of a single exemplar; these methods can generate arbitrary-size and potentially nonstationary output.

Zhou et al. [ZCXH23] propose a new loss function that combines Markov Random Fields with neural networks, the Guided Correspondence Loss (GCD Loss). Their approach is capable of synthesizing high-quality textures of arbitrary size. We use this method as a comparator due to its general effectiveness.

Optimizer Based Methods. Kwatra et al. [KEBK05] propose a method for synthesizing multi-scale textures through optimization, introducing an energy function based on the Markov Random Field and an optimization technique based on Expectation Maximization [MK08]. Kaspar et al. [KNL*15] present an example-based method capable of synthesizing high-quality textures, even those with nonstationary elements due to large-scale structures. We consider Kaspar et al.'s method to be the most effective optimization-based approach and include it as a basis for comparison.

Summary of Methods. Table 1 compares our method to similar algorithms. Ours is the only one capable of producing a pure vector result from a photorealistic raster exemplar. We also require no user guidance apart from selecting the exemplar. The other algorithms that produce vector results either require an exemplar that is already in vector format, have many parameters that need to be manually tuned, or can only take a simple raster exemplar.

3. Vector Texture Synthesis

We take a single stationary texture as input and produce vector output. This is done in two stages: an offline analysis, followed by an online synthesis step.

In the analysis, we decompose the input texture into three layers: a set of *primary textons*, which are large distinct elements; *secondary textons*, which are smaller and less visually distinct than the primary textons; and the remaining background region, comprising all pixels not part of a primary or secondary texton. Then, we create a *descriptor* for each primary texton, consisting of a local map of the other primary textons nearby. We estimate inter-element spacing for the secondary textons, and we sample the background to obtain an approximation of its color distribution.

The analysis and layering process is similar to the approach of Palazzolo et al. [PvM25] for creating abstract expressionist art from arbitrary exemplars. Besides the novel problem domain, the major difference is in texton extraction. Palazzolo et al. naïvely segment the exemplar by color, whereas we extract meaningful objects, even if the objects contain a high amount of color variation. Beyond synthesis, we explore editing, adaptive density, and interpolation.

Once the analysis phase is complete, the online synthesis step begins. Our method uses the descriptors and a scoring system for synthesizing the primary texton distribution, and uses Poisson disk distributions [DW85] for both the secondary elements and control points for the background gradient field.

The overall pipeline is illustrated in Figure 2. The analysis and synthesis are described in detail in the following subsections.

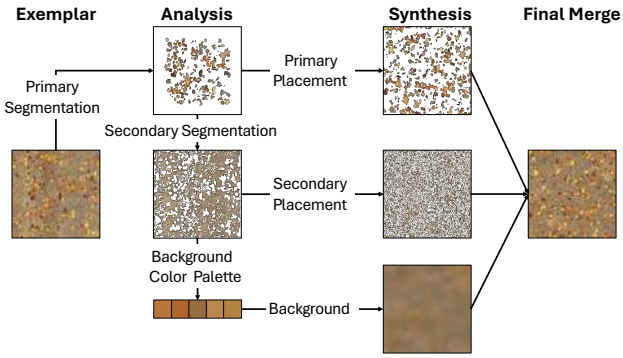

Figure 2: *An overview of our texture synthesis algorithm.*

3.1. Analysis Phase

The analysis phase has three main subcomponents: first, identifying major textons in the input; second, creating descriptors that characterize the local spatial arrangements of textons; and third, estimating secondary texton density and background color gradient. We discuss these three components in turn.

3.1.1. Primary Texton Extraction

We start by identifying primary textons. Our results are only as good as the extracted textons, so the extraction algorithm is critical. A good texton encompasses a texture element that is perceived as a unit and is distinct from the background (e.g., a rock or a leaf), even if it contains a high amount of color variation. Segment Anything (SAM) [KMR*23], executed with a dense grid of query points (10k points on a 500×500 image), provides suitable segments.

The resulting segments are processed to isolate individual connected components and to remove duplicate textons. Primary textons are subsegmented using Felzenszwalb segmentation [FH04] to obtain detail textons (scale parameter 10 used for our results). Detail textons are children of their primary texton. When a primary texton is placed in the result, all of its detail textons are placed on top in the location where they were extracted.

3.1.2. Secondary Texton Extraction

Once the primary textons are identified, we proceed to secondary textons. We extract secondary textons using floodfill segmentation, only checking pixels that are not within a primary texton.

We then estimate the typical spacing, $\bar{s}$, between secondary textons by computing a Delaunay triangulation of the texton centres, then taking a predetermined percentile of the resulting edge lengths. We opted to set $\bar{s}$ to the 40th percentile length; empirically, the edge lengths are fairly stable over roughly the 30th to 60th percentiles, and a slightly lower estimate yields greater texton density in the synthesis and hence greater perceived detail in the output. The value of $\bar{s}$ will be used in the synthesis stage.

3.1.3. Primary Texton Descriptors

The primary textons are clustered based on visual appearance. We take the RGB color channels of the median color from any pixel

Paper	Exemplar Input	Vector Output	User Guidance
Ours	Photorealistic Raster	Yes	Automatic
Dischler et al. [DMLG02]	Photorealistic Raster	No	Manual segmentation
Jagnow et al. [JDR04]	No Exemplar	Yes	NURBS sphere
Dischler et al. [DZ06]	Photorealistic Raster	No	Color quantization
Liu et al. [LWX*09]	Photorealistic Raster	No	Mask of a texton + background marker
Praun et al. [PFH00]	Photorealistic Raster	No	Outlined region
Qian et al. [QSS*22]	Simple Raster	Yes	Automatic
Gilet et al. [GDG12]	Photorealistic Raster	No	Interactive refinement
Guehl et al. [GAD*20]	Photorealistic Raster	No	Binary map of textons and categories
Passos et al. [AdPWS10]	Vector Elements	Yes	Automatic
Landes et al. [LGH13]	Vector Elements	Yes	Automatic
Ma et al. [MWLT13]	Vector Elements	Yes	Automatic
Barla et al. [BBT*06]	Vector Elements	Yes	Interactive parameter tuning

Table 1: *A comparison of our algorithm versus several similar methods. Ours is the only fully automatic algorithm that produces a pure vector output from a photorealistic raster exemplar.*

contained within the texton, in addition to the texton area and Polsby-Popper compactness [PP91]. These five numbers are used in K-means clustering to create some number of clusters (we typically use 15). The three color features are weighted twice as heavily as the others. In stochastic textures, results are not sensitive to the exact number of clusters.

Our texton descriptor is a 2d grid-based map of the surroundings of a given texton, which we call the *central texton*. Each cell of the map contains either a label for the category of the texton found there, if any, or a code for "empty". In addition, where there are textons that lie partially within the map and protrude beyond the initial boundaries, the map is extended to include these textons in full. Figure 3 shows visualizations of some sample descriptors.

By default, we compute descriptors at the pixel resolution of the original raster input. However, we believe that smaller resolution would also work well, especially for future work in which more elaborate texton synthesis would be used, not necessarily matching the high-resolution texton shapes closely.

Descriptors must be fully within the bounds of the image; descriptors extending outside the image are discarded. Thus, a descriptor's centre cannot be close to the image boundary, although the descriptor can include elements up to the edge of the image. We use a descriptor size such that each central texton has on average 2.25 textons in any direction. In order to have adequate variety of content, the method thus requires large exemplars.

3.1.4. Background Gradient Field Summarization

We next proceed to characterizing the background. We remove from the starting image all pixels marked as part of any texton, plus an additional strip of pixels around each texton, ensuring that all remaining pixels strictly belong to the background. We then construct a Voronoi diagram of a Poisson disk distribution of points. We found that a spacing of 25 produces good results. For each Voronoi region, we compute the median color of background pixels. The set of median colors is called the *background color palette* and will be used to synthesize an output gradient field.

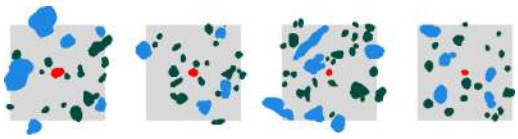

Figure 3: *Illustration of descriptors extracted from a texture. The gray box shows the extent of the descriptor. The red polygon is the central texton; additional polygons are other textons in the descriptor, colored by category. Notice that several textons protrude from the initial descriptor region.*

3.2. Synthesis Phase

To create a novel texture, we synthesize a distribution of primary and secondary textons along with a background gradient field. We then merge the resulting three layers into a single vector image. Details about each layer appear in the following subsections.

3.2.1. Primary Texton Synthesis

We aim to synthesize a spatial distribution of primary textons similar to that in the exemplar. Akin to non-parametric sampling, we attempt to match the surroundings of output textons with arrangements seen in the exemplar.

We track the desired output distribution through an incrementally-developed target map, represented as a grid G. Initially all cells are coded "undefined". Whenever we place a texton, we replace nearby undefined grid cells with target values copied from the descriptor; values can be "empty" or can encode a request for the cell to be covered by a texton of a given category.

The first texton is placed at random. Subsequent placements are made by trialing a fixed number of additional textons (we use 20) and scoring each candidate, with the highest-scoring texton being selected. After finding the best texton, we make an attempt to improve its location, repeatedly shifting it $+/-$ one grid cell in x and y and keeping the highest-scoring location. We use a maximum of five steps. This texton's descriptor is then written onto the map;

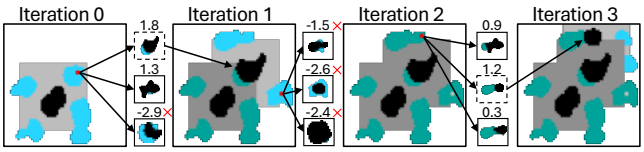

Figure 4: *Example iterations during synthesis. Black regions are placed textons, and blue regions are requested textons. In each iteration, we evaluate how several candidate textons fit in a space and keep the best one, adding its descriptor to the map. If the best score is below 0, placement is not performed.*

Figure 4 illustrates the process. The full process is summarized in Algorithm 1 and Algorithm 2, which respectively describe the process for arranging the primary textons and the subprocess of adding a texton to the map.

Scoring is decided as follows. Call a candidate texton C; it will overlap with a region R of the same category, say α. The score for C is a weighted average $\sum_{i=5}^{5} w_i A_i$ of areas A_i. The areas A_i are computed by counting pixels p with different properties, as follows:

1. Target Area (A_1): $||\{p \in C \cap R\}||$, i.e., overlap between texton and target region.
2. Uncovered Area (A_2): $||\{p \in R - C\}||$, i.e., area of target region left uncovered.
3. Empty Area (A_3): $||\{p \in C \text{ such that } G[p] = \text{empty}\}||$, i.e., texton covers an area that is supposed to be empty.
4. Mismatched Area (A_4): $||\{p \in C \text{ such that } G[p] ! = \{\text{empty}, \alpha\}\}||$, i.e., texton overlaps an area where a different category is desired.
5. Same Overlap Area (A_5) $||\{p \in C - R \text{ such that } G[p] = \alpha\}||$, i.e., texton overlaps a different region of the same category.

By default, the weights $\mathbf{w}$ are $(0.5, -0.4, -0.2, -0.5, -0.5)$, determined empirically. Note that a negative weight implies that the corresponding quantity should be minimized.

3.2.2. Secondary Texton Synthesis

Although fine-scale detail is needed for a convincing texture, such details are individually of lower salience than the larger, higher-contrast primary textons; consequently, such detail can be synthesized using a simpler method. We distribute secondary textons according to a Poisson disk [DW85] distribution with mean inter-element spacing $\bar{s}$ (Section 3.1.2).

3.2.3. Background Gradient Field

In this step we synthesize a gradient field, interpolating between scattered points with associated color values. In this process, we create a Poisson disk point distribution and assign to each point a random selection from the color pool extracted in the background analysis (Section 3.1.4). We found that a Poisson disk spacing of 25 produces good results. From these control points, the full field is computed using Radial Basis Function (RBF) interpolation with a linear radius function $f(r) = r$.

Algorithm 1 Primary Texton Layer

G, raster grid used for scoring (see text)
D, set of all descriptor/texton pairs
$\mathbf{w}$, vector of weights for each area type
k, number of tries to find the best placement per iteration
ε, minimum placement score that can be accepted ($\varepsilon = 0$)
S, the set of all textons and their descriptors extracted from the exemplar

Select a random texton and corresponding descriptor from S as T and d respectively, weighted based on area
Add T using Algorithm 2

while at least one cell contains a texton request in G **do**
    $P(x)$ is normalized probability distribution over possible texton placements

    // Try some textons and keep the one with the best score
    $c_c \leftarrow$ Sample $P(x)$ for random centroid pixel
    $b_s \leftarrow -\infty$, best placement score so far

    for $i = 1, 2, \ldots, k$ **do**
        $c_t \leftarrow$ Random texton from D with the category $G[c_c]$

        // Weighted sum to compute the score of the candidate texton's placement
        $c_s \leftarrow \sum_{i=1}^{5} A_i \mathbf{w}_i$
        if $c_s > b_s$ **then**
            // Track best score (b_s) texton (b_t) and centroid (b_c)
            $b_s \leftarrow c_s$, $b_t \leftarrow c_t$, $b_c \leftarrow c_c$
        end if
    end for

    // If the best score is too low, reject the placement, otherwise add it
    if $b_s \leq \varepsilon$ **then**
        For all pixels $p \in b_t$, G[p] $\leftarrow$ empty
    else
        Add b_t at position b_c using Algorithm 2
    end if
end while

Algorithm 2 Texton Addition Procedure

Input: T = the added texton, (a, b) = the location to add it, d = its descriptor, α = its category

// write placed texton into map
For all pixels $p \in T$, $G[p] \leftarrow \alpha$.

// copy descriptor to map wherever map was not yet defined
for (x, y) = all indices within d **do**
    if $G[a+x, b+y]$ is undefined **then**
        $G[a+x, b+y] \leftarrow d[x, y]$
    end if
end for

3.3. Enhancements

Here we describe small enhancements to the base algorithm. All results in Section 4 use these improvements.

Duplication reduction. We do not want to select the same textons repeatedly. Initially, each texton in a category has an equal probability of being selected. Whenever a texton is added to the output, we halve the probability of selecting that texton, and the category probability distribution is renormalized.

Background matching. When computing the median color of a secondary texton, we also compute a color delta against the background. At synthesis time, we determine the texton's color by adding the delta to the background color at the texton centroid. This small change subtly improves visual quality.

Global density correction. Our method tends to place more textons than necessary, so we remove some: we delete a randomly selected texton from any category whose fractional area covered by the category is greater than the exemplar's fractional area. This repeats until the fractional area covered by all textons matches the coverage of the exemplar.

4. Results and Discussion

This section gives sample results and compares our synthetic textures to results generated by previous approaches. We also give examples of texture editing operations enabled by our vector representation, and discuss some limitations and failure cases. Additional results and comparisons appear in the supplemental material.

Our approach is aimed at irregular natural images containing distinct textons, for which a vector representation of individual textons is beneficial. Ideally, the exemplars should be statistically stationary, the common assumption to virtually all example-based texture synthesis methods; we can somewhat cope with nonstationary backgrounds, though large-scale structures will not be preserved. Texton size, shape, and color can be heterogeneous. Many natural textures possess these properties; a few examples appear in the figures of this section.

4.1. Qualitative evaluation

Figure 8 shows textures synthesized by our method. Our method is capable of capturing and reproducing complex relationships between texture elements.

The top-left texture demonstrates the algorithm's ability to produce a convincing result despite somewhat unclear textons. The top-right texture of the matrix demonstrates our method's ability to reproduce uniform textures; however, notice how some of the output textons are slightly perturbed from the grid. The bottom-right shows that our method is capable of handling complex shapes and structures between the flowers and blades of grass. The bottom-left is the ideal exemplar for our algorithm as it has distinct, stochastic textons; this type of texture is particularly successful.

4.2. Quantitative evaluation

We report metrics comparing our results with those of selected previous synthesis methods that output textures in raster format: Image Quilting [EF01], PSGAN [BJV17], GCD Loss [ZCXH23], and Self-Tuning Optimization [KNL*15]. We use the authors' implementation whenever possible and use the default parameters. For Image Quilting, we use a patch size that is half the descriptor size used by our method. Sample outputs from these methods and ours are shown in Figure 10.

Like Rodríguez-Pardo et al. [RPCGLM24], our evaluation uses the metrics SIFID [SDM19], CLIP-IQA [WCL23], DISTS [DMWS20], PieAPP [PCMS18], LPIPS [ZIE*18], SSIM [WBSS04], and BRISQUE [MMB12]. We also estimate the blur in the output images with Kumar and Raj's Laplacian-based metric [KRC16]. We use the implementation of these metrics provided by the PyIQA Python library [CM22]. We also compare the pixel intensity histograms between the exemplar and synthetic texture, reporting the Earth Mover's Distance between the two. Overall, these metrics fall into two categories: metrics that measure similarity and perceptual similarity (LPIPS, PieAPP, SIFID, SSIM, Earth Mover's Distance (EMD)), and metrics that measure image quality (BRISQUE, DISTS, Clip-IQA). Both fidelity to the exemplar and output image quality are important in texture synthesis applications.

We report the quantitative metrics in Table 2. Our method is competitive with state-of-the-art methods, with comparable scores in general and the top SIFID score. We emphasize that our goal here is not to create another conventional raster-based texture synthesis method, but to work with vector textures and to be able to represent and manipulate individual textons while producing results comparable to state-of-the-art methods.

We compare our method to Tu et al. on their sample images, which are composed of a single layer of elements over a simple background. Quantitative results are given in the supplemental material, and sample outputs appear in Figure 5. From the scores, and from visual inspection of results, we see that Tu et al.'s method is better able than our method to capture the target density of textons on their images. Our method scores highly according to PieAPP and SSIM and is comparable according to the other metrics. Figure 5b and c are composed of tightly packed vector elements; our method was not designed for densely packed and overlapping textons and it struggles here. Further, these exemplars contain semi-structured arrangements that our method does not attempt to replicate. Despite this, our method still produces somewhat plausible results. It is not likely that Tu et al.'s method will work well in our target domain of natural, stochastic textures with complex structural relationships, as it assumes a simple distribution of clean textons with duplicates.

4.3. Ablation Study

We conducted an ablation study to confirm the usefulness of the different components of our method. We synthesized textures with various portions of the algorithm disabled: (1) No secondary textons; (2) Simple scoring prohibiting overlaps; (3) No density correction; (4) No reselection decay. Other possible variants include omitting gradient background and secondary texton color adjustment; while these omissions improve the automated scores, they do so by reducing variety and visual interest. Table 3 contains a

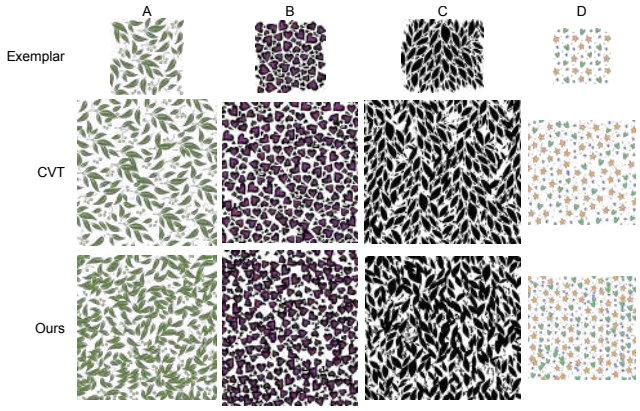

Figure 5: *Our method vs. CVT [TWZ22] on four vector exemplars.*

Metric	Quilting	Tuning	PSGAN	GCD	Ours
SIFID ↓	*0.012*	0.074	0.014	18.032	**0.011**
I. EMD ↓	**0.000**	*0.001*	*0.001*	*0.001*	*0.001*
DISTS ↓	**0.157**	*0.219*	0.253	0.291	0.347
PieAPP ↓	**1.186**	1.466	1.520	1.557	*1.258*
LPIPS ↓	**0.400**	*0.415*	0.428	0.475	0.451
SSIM ↑	*0.201*	**0.219**	0.177	0.141	0.198
BRISQUE ↓	*28.089*	30.765	**20.436**	84.497	76.732
CLIP-IQA ↑	*0.522*	**0.581**	0.484	0.436	0.482
Blur ↑	1750	1617	2391	**7394**	*3060*

Table 2: *Metrics computed for the images in Figure 10; best scores are bold, second best are italicized. Values are the mean of four results from each of 38 exemplars; per-image numbers appear in the supplementary material. The average exemplar BRISQUE score is 27.562 and the average exemplar CLIP-IQA score is 0.521.*

summary of the metrics computed over six textures for each condition. All individual images as well as the full table of metrics can be found in the supplemental material.

In general, the full algorithm scores better than any reduced variant. This is confirmed by visual inspection of the results, which can be found in the supplemental material. Reselection decay has a more meaningful impact when synthesizing a result larger than the exemplar.

4.4. Timing and statistics

Table 4 reports the average amount of time required to synthesize 4 500 × 500 textures. Our method requires an average of 6 minutes for analysis, and approximately 30 seconds for synthesis. Synthesizing a larger 1000 × 1000 image takes about 2 minutes. Table 5 shows texton counts after analyzing 500 × 500 results. Typical natural textures at this resolution have several hundred primary textons and 5k-10k secondary and detail textons. More artificial textures (like row C in Figure 10) may have far fewer. Note that the counts of detail textons are reasonably consistent owing to the Poisson disk placement, although not all placed textons will be visible in the final image.

Metric	Full	1	2	3	4
SIFID ↓	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
I. EMD ↓	**0.001**	*0.002*	**0.001**	**0.001**	**0.001**
DISTS ↓	**0.348**	0.389	*0.350*	0.351	*0.350*
PieAPP ↓	1.190	4.671	*1.099*	1.131	**1.091**
LPIPS ↓	**0.469**	0.623	*0.470*	0.475	0.475
SSIM ↑	0.110	**0.113**	0.110	0.110	*0.112*

Table 3: *Ablation study metrics. Best scores are bold, second best italics. 1: No secondary textons. 2: Simple scoring prohibiting overlaps. 3: No density correction. 4: No reselection decay.*

Method	Analysis Time	Synthesis Time
Image Quilting†	———	21m
Self-Tuning	———	38.5s
PSGAN	7.5h	0.5s
GCD Loss	———	2.4m
Ours	6m	28s

Table 4: *The average time to synthesize a* 500 × 500 *exemplar. More than 60% of our analysis time is due to Segment Anything (SAM). †Image quilting results are from an unoptimized third-party implementation, not the authors' code.*

4.5. Vector Edits

Results from a few post-processing operations enabled by a vector representation were shown in Figure 1. Figure 9 shows examples of some additional editing operations. Vector interpolation allows textons to seamlessly move and transform into the textons of the destination image. We provide a video of the interpolation in our supplemental material where the full effect is more prominent. Forced anisotropy orients each texton along a desired global direction. Texton thinning has applications in both accessibility and level of detail rendering. Joint synthesis allows a control map to dictate which exemplar to sample from, and supports multiple exemplars. Density map adjustment removes textons with probability proportional to the map intensity at that location. Textures can be made interactive

Exemplar	Primary	Detail	Secondary
A	1376	14368	9328
B	386	9688	5852
C	81	———	———
D	531	6132	9959
E	577	9199	8239
F	445	10036	7627

Table 5: *The number of primary and secondary textons in selected results, plus the total number of detail textons across all primary textons. Exemplar ID refers to the exemplars shown in Figure 10.*

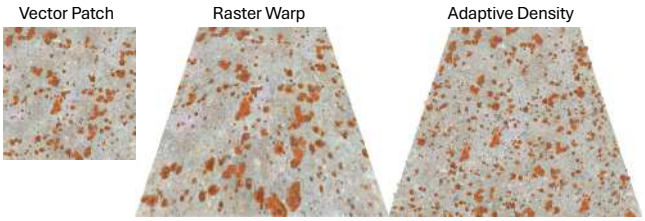

Figure 6: *Effect of domain warps. Left: original texture; middle: warped raster; right: adaptive density.*

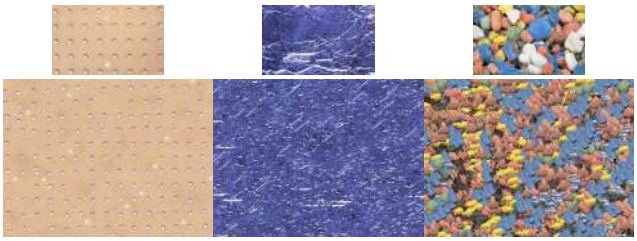

Figure 7: *A few failure cases. Left: Regular texture; center: no discrete textons; right: limited context.*

since the locations, geometry, and color of each texton can be easily modified. For example, a texture of leaves on a sidewalk could have the leaves move and react to a player's footsteps. Texton tinting could be used for dynamic recoloring, especially when simply tinting the full image will not suffice (e.g. certain textons need to be tinted different colors).

Additionally, we can adapt the density of textons if the synthesis domain is modified. We show an example of this capability in Figure 6, where the exterior of the synthesized patch is warped in such a way that increases its area (for example, a balloon inflating). In a raster, this sort of translation would also warp the textons incidentally. In vector format, the textons can be repositioned based on mean value coordinates [Flo03], their descriptors can be placed, and our texture synthesis algorithm can be run from there, creating a plausible distribution. This is different from texture inpainting because the area to fill is not clearly defined.

4.6. Limitations

Our method is intended for a limited class of textures: those with separate, identifiable textons, which most benefit from a vector representation. When individual textons are difficult to distinguish, our method is less effective. Cases where textons overlap or are tightly packed also pose problems: in such cases, texton shape and placement are tightly constrained by nearby textons, whereas we assume more flexible arrangements to be possible. Fine-tuning the scoring weights can improve outcomes on packed textures.

We depend on image segmentation to identify vector elements, but segmentation is not always reliable. Textures without distinct elements will not yield segments and the subsequent synthesis will fail. Shadows may produce separate segments, and subsequently in synthesis can become detached from any possible source of shadow, with unappealing results.

Figure 7 shows the result of our method when used on some of these cases. Regular textures sometimes work, but there is some uncertainty about whether the pattern will be preserved, partly depending on how much unstructured content is also present. Notice how the example regular texture has more textons than just the circles, in contrast to the circles image in Figure 8 (top right) that has no background detail. The problems in the other examples can be attributed to the segmentation: when textons are very small, they may be missed by the segmentation with the subsequent analysis yielding a picture of a much sparser texture. In a texture with no discrete textons, the segmentation still identifies some features; although these can be used to create an appealing texture, the output typically does not resemble the exemplar. Textures with few textons relative to the exemplar size often fail due to limited neighborhood context. Since a large neighborhood is required, only a few segments are accepted as primary textons creating a sparse texture. Sometimes fine-tuning the weights can save these cases.

5. Conclusion

In this paper, we introduced an algorithm for synthesizing vector textures composed of solid-colored polygons and a background gradient field. We start with an input raster exemplar, and convert it to a vector representation through image segmentation. This representation is then used to synthesize a novel vector texture. We also showcase some post-processing operations on our vector results that would be difficult to apply to a raster image, illustrating potential applications of vector textures.

We compare the results of our method with a number of other state-of-the-art algorithms in qualitative and quantitative evaluations. Our method is competitive with the other algorithms while our textures are purely composed of vector elements. We also compared our method with a vector texture synthesis algorithm designed for structured textures; despite the different types of textures the methods were designed for, our results are comparable.

There remain some opportunities for future work. Our textons are solid-colored (sub-texton detail is provided by a hierarchy of textons) and using gradients to enhance detail is a clear next step. Exploring alternatives to pure segmentation for texton extraction could yield improvements; manual intervention or texton repair could be investigated. Finally, reconstructing data other than color (e.g., normals) could be beneficial to some applications.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Carleton University. We would also like to thank the members of the Graphics, Imaging, and Games Lab (GIGL) as well as friends outside the lab for their suggestions and comments.

References

[AdPWS10] ALVES DOS PASSOS V., WALTER M., SOUSA M. C.: Sample-based synthesis of illustrative patterns. In *2010 18th Pacific Conference on Computer Graphics and Applications* (2010), pp. 109–116. doi:10.1109/PacificGraphics.2010.22. 2, 4

[AKA13] ALMERAJ Z., KAPLAN C. S., ASENTE P.: Patch-based geometric texture synthesis. In *Proceedings of the Symposium on Computational Aesthetics* (New York, NY, USA, 2013), CAE '13, Association for Computing Machinery, p. 15–19. doi:10.1145/2487276.2487278. 1, 2

[BBT*06] BARLA P., BRESLAV S., THOLLOT J., SILLION F., MARKOSIAN L.: Stroke pattern analysis and synthesis. *Computer Graphics Forum 25*, 3 (2006), 663–671. doi:https://doi.org/10.1111/j.1467-8659.2006.00986.x. 2, 4

[BJV17] BERGMANN U., JETCHEV N., VOLLGRAF R.: Learning texture manifolds with the Periodic Spatial GAN. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017), ICML'17, JMLR.org, p. 469–477. 2, 6, 12

[CM22] CHEN C., MO J.: IQA-PyTorch: Pytorch toolbox for image quality assessment. [Online]. Available: https://github.com/chaofengc/IQA-PyTorch, 2022. 6

[DMLG02] DISCHLER J.-M., MARITAUD K., LÉVY B., GHAZANFARPOUR D.: Texture particles. *Computer Graphics Forum 21*, 3 (2002), 401–410. doi:https://doi.org/10.1111/1467-8659.t01-1-00600. 2, 4

[DMWS20] DING K., MA K., WANG S., SIMONCELLI E. P.: Image quality assessment: Unifying structure and texture similarity. *CoRR abs/2004.07728* (2020). arXiv:2004.07728. 6

[DW85] DIPPÉ M. A. Z., WOLD E. H.: Antialiasing through stochastic sampling. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 69–78. doi:10.1145/325165.325182. 3, 5

[DZ06] DISCHLER J.-M., ZARA F.: Real-time structured texture synthesis and editing using image-mesh analogies. *Vis. Comput. 22*, 9-11 (2006), 926–935. 2, 4

[EF01] EFROS A. A., FREEMAN W. T.: *Image Quilting for Texture Synthesis and Transfer*, 1 ed. Association for Computing Machinery, New York, NY, USA, 2001, pp. 571–576. 2, 6, 12

[EL99] EFROS A., LEUNG T.: Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision* (1999), vol. 2, pp. 1033–1038 vol.2. doi:10.1109/ICCV.1999.790383. 2

[FH04] FELZENSZWALB P., HUTTENLOCHER D.: Efficient graph-based image segmentation. *International Journal of Computer Vision 59* (09 2004), 167–181. doi:10.1023/B:VISI.0000022288.19776.77. 3

[Flo03] FLOATER M. S.: Mean value coordinates. *Computer Aided Geometric Design 20*, 1 (2003), 19–27. doi:https://doi.org/10.1016/S0167-8396(03)00002-5. 8

[GAD*20] GUEHL P., ALLÈGRE R., DISCHLER J.-M., BENES B., GALIN E.: Semi-procedural textures using point process texture basis functions. *Computer Graphics Forum 39*, 4 (2020), 159–171. doi:https://doi.org/10.1111/cgf.14061. 2, 4

[GDG12] GILET G., DISCHLER J.-M., GHAZANFARPOUR D.: Multi-scale assemblage for procedural texturing. *Computer Graphics Forum 31*, 7 (2012), 2117–2126. doi:https://doi.org/10.1111/j.1467-8659.2012.03204.x. 2, 4

[GEB15] GATYS L. A., ECKER A. S., BETHGE M.: Texture synthesis using convolutional neural networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1* (Cambridge, MA, USA, 2015), NIPS'15, MIT Press, p. 262–270. 2

[GGM11] GALERNE B., GOUSSEAU Y., MOREL J.-M.: Random phase textures: Theory and synthesis. *IEEE Transactions on Image Processing 20*, 1 (2011), 257–267. doi:10.1109/TIP.2010.2052822. 2

[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* (New York, NY, USA, 2009), NPAR '09, Association for Computing Machinery, p. 51–60. doi:10.1145/1572614.1572623. 1, 2

[HN18] HEITZ E., NEYRET F.: High-performance by-example noise using a histogram-preserving blending operator. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 2 (aug 2018). doi:10.1145/3233304. 2

[JBV17] JETCHEV N., BERGMANN U., VOLLGRAF R.: Texture synthesis with spatial generative adversarial networks, 2017. arXiv:1611.08207. 2

[JDR04] JAGNOW R., DORSEY J., RUSHMEIER H.: Stereological techniques for solid textures. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 329–335. doi:10.1145/1015706.1015724. 2, 4

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, Association for Computing Machinery, p. 795–802. doi:10.1145/1186822.1073263. 3

[KMR*23] KIRILLOV A., MINTUN E., RAVI N., MAO H., ROLLAND C., GUSTAFSON L., XIAO T., WHITEHEAD S., BERG A. C., LO W.-Y., DOLLÁR P., GIRSHICK R.: Segment anything. *arXiv:2304.02643* (2023). 3

[KNL*15] KASPAR A., NEUBERT B., LISCHINSKI D., PAULY M., KOPF J.: Self tuning texture optimization. *Computer Graphics Forum 34*, 2 (2015), 349–359. doi:https://doi.org/10.1111/cgf.12565. 3, 6, 12

[KRC16] KUMAR P., RAJ G., CHOUDHURY T.: Blur image detection using laplacian operator and open-cv. pp. 63–67. doi:10.1109/SYSMART.2016.7894491. 6

[LGH13] LANDES P.-E., GALERNE B., HURTUT T.: A shape-aware model for discrete texture synthesis. *Computer Graphics Forum 32*, 4 (2013), 67–76. doi:https://doi.org/10.1111/cgf.12152. 2, 4

[LLGRK20] LI T.-M., LUKÁČ M., GHARBI M., RAGAN-KELLEY J.: Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. 39*, 6 (Nov. 2020). doi:10.1145/3414685.3417871. 1

[LTH86] LAIDLAW D. H., TRUMBORE W. B., HUGHES J. F.: Constructive solid geometry for polyhedral objects. *SIGGRAPH Comput. Graph. 20*, 4 (aug 1986), 161–170. doi:10.1145/15886.15904. 1

[LWX*09] LIU Y., WANG J., XUE S., TONG X., KANG S. B., GUO B.: Texture splicing. *Computer Graphics Forum 28*, 7 (2009), 1907–1915. doi:https://doi.org/10.1111/j.1467-8659.2009.01569.x. 2, 4

[MK08] MCLACHLAN G. J., KRISHNAN T.: *The EM Algorithm and Extensions, 2E*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA, Feb. 2008. doi:10.1002/9780470191613. 3

[MMB12] MITTAL A., MOORTHY A. K., BOVIK A. C.: No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing 21*, 12 (2012), 4695–4708. doi:10.1109/TIP.2012.2214050. 6

[MWLT13] MA C., WEI L.-Y., LEFEBVRE S., TONG X.: Dynamic element textures. *ACM Trans. Graph. 32*, 4 (July 2013). doi:10.1145/2461912.2461921. 2, 4

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph. 27*, 3 (aug 2008), 1–8. doi:10.1145/1360612.1360691. 1

[PCMS18] PRASHNANI E., CAI H., MOSTOFI Y., SEN P.: Pieapp: Perceptual image-error assessment through pairwise preference, 2018. arXiv:1806.02067. 6

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 465–470. doi:10.1145/344779.344987. 2, 4

[PP91] POLSBY D. D., POPPER R.: The third criterion: Compactness as a procedural safeguard against partisan gerrymandering. *Yale Law & Policy Review. 9 (2): 301–353.* (1991). doi:https://doi.org/10.2139/ssrn.2936284. 4

[PvM25] PALAZZOLO C., VAN KAICK O., MOULD D.: Breaking art: Synthesizing abstract expressionism through image rearrangement. *Computers & Graphics 129* (2025), 104224. doi:https://doi.org/10.1016/j.cag.2025.104224. 3

[QSS*22] QIAN Y., SHI J., SUN H., CHEN Y., WANG Q.: Vector solid texture synthesis using unified rbf-based representation and optimization. *The Visual Computer 39* (07 2022), 1–15. doi:10.1007/s00371-022-02541-y. 2, 4

[RPCGLM24] RODRIGUEZ-PARDO C., CASAS D., GARCES E., LOPEZ-MORENO J.: Textile: A differentiable metric for texture tileability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024). 6

[SBII19] SHOCHER A., BAGON S., ISOLA P., IRANI M.: InGAN: Capturing and Retargeting the "DNA" of a Natural Image. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 4491–4500. doi:10.1109/ICCV.2019.00459. 3

[SDM19] SHAHAM T. R., DEKEL T., MICHAELI T.: SinGAN: Learning a generative model from a single natural image. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 4569–4579. doi:10.1109/ICCV.2019.00467. 3, 6

[SKA21] SAPUTRA R. A., KAPLAN C. S., ASENTE P.: Improved deformation-driven element packing with repulsionpak. *IEEE Transactions on Visualization and Computer Graphics 27*, 4 (2021), 2396–2408. doi:10.1109/TVCG.2019.2950235. 1, 2

[SLWS07] SUN J., LIANG L., WEN F., SHUM H.-Y.: Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics (TOG) 26*, 3 (2007). 1

[TWY*20] TU P., WEI L.-Y., YATANI K., IGARASHI T., ZWICKER M.: Continuous curve textures. *ACM Transactions on Graphics 39*, 6 (Nov. 2020), 1–16. doi:10.1145/3414685.3417780. 1, 2

[TWZ22] TU P., WEI L.-Y., ZWICKER M.: Clustered vector textures. *ACM Trans. Graph. 41*, 4 (jul 2022). doi:10.1145/3528223.3530062. 1, 2, 7

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on 13* (05 2004), 600 – 612. doi:10.1109/TIP.2003.819861. 6

[WCL23] WANG J., CHAN K. C., LOY C. C.: Exploring CLIP for assessing the look and feel of images. *Proceedings of the AAAI Conference on Artificial Intelligence 37*, 2 (Jun. 2023), 2555–2563. doi:10.1609/aaai.v37i2.25353. 6

[XWL*23] XUE T., WU M., LU L., WANG H., DONG H., CHEN B.: Learning gradient fields for scalable and generalizable irregular packing. In *SIGGRAPH Asia 2023 Conference Papers* (New York, NY, USA, 2023), SA '23, Association for Computing Machinery. URL: https://doi.org/10.1145/3610548.3618235, doi:10.1145/3610548.3618235. 1, 2

[ZCXH23] ZHOU Y., CHEN K., XIAO R., HUANG H.: Neural texture synthesis with guided correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2023), pp. 18095–18104. 3, 6, 12

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 586–595. doi:10.1109/CVPR.2018.00068. 6

[ZZB*18] ZHOU Y., ZHU Z., BAI X., LISCHINSKI D., COHEN-OR D., HUANG H.: Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics 37*, 4 (July 2018), 1–13. doi:10.1145/3197517.3201285. 2

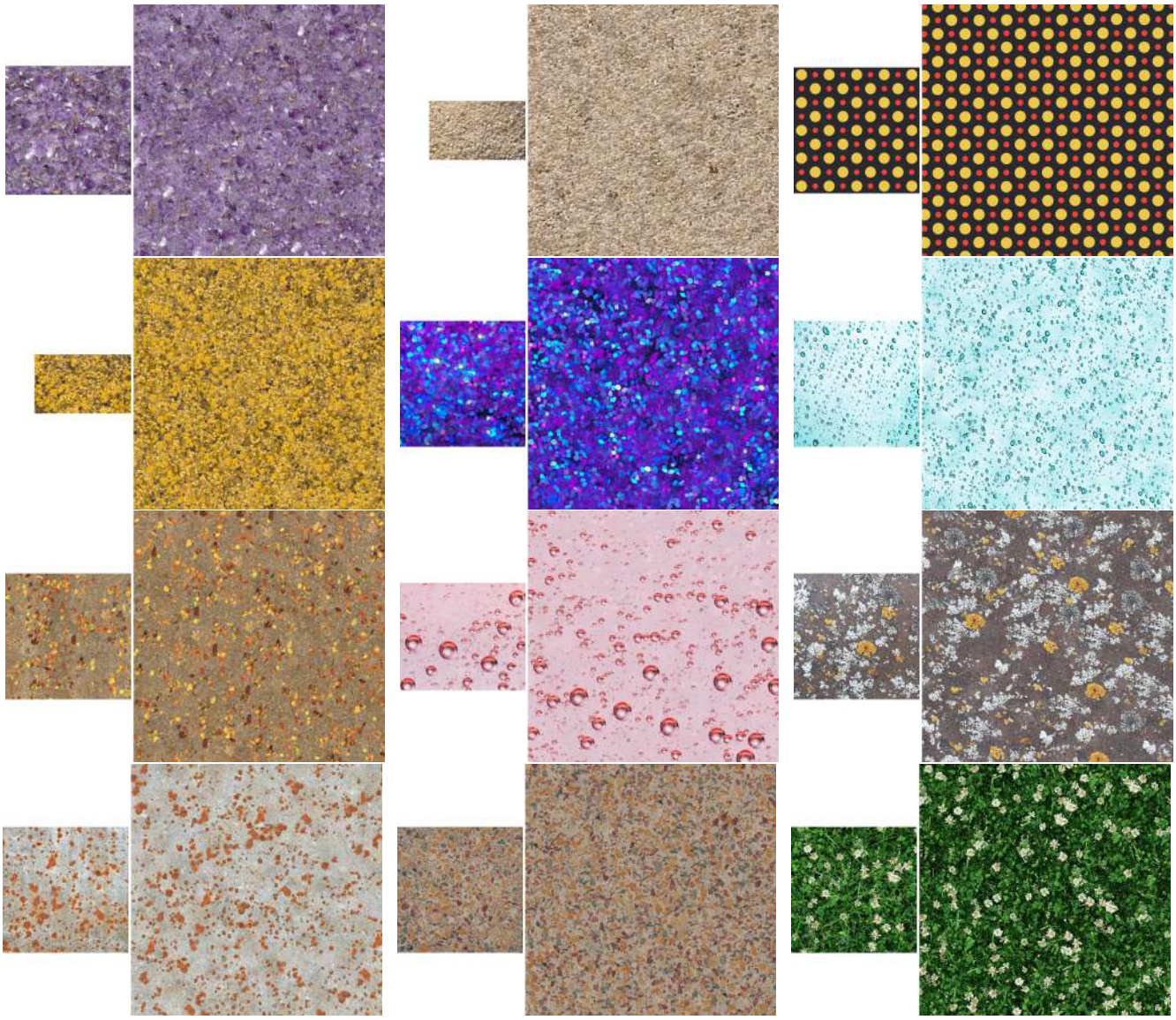

Figure 8: *Textures synthesized using our algorithm. Each texture pair shows a raster exemplar (left) and a synthetic vector image (right, rendered at* 1000×1000*).*

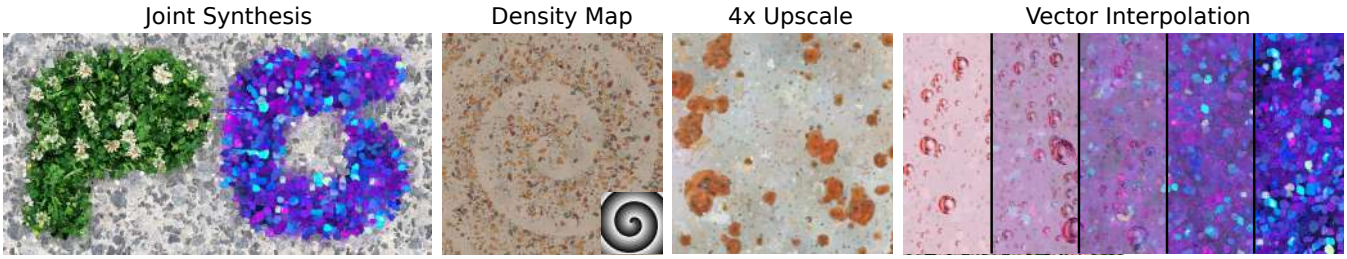

Figure 9: *Some editing operations that are easy to perform given vector textons, but would be more difficult on a raster image. Additional editing operations are shown in Figure 1.*

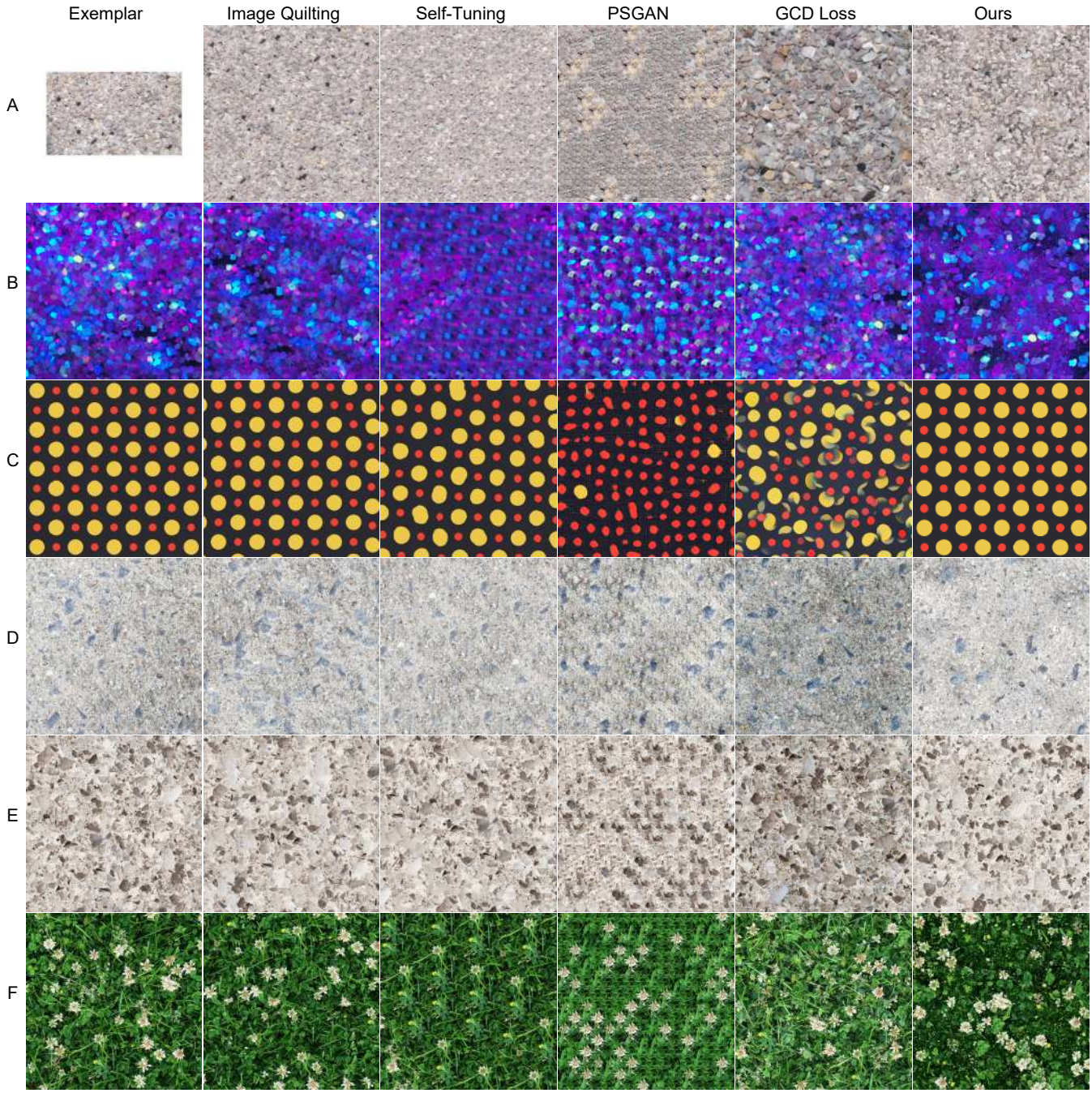

Figure 10: *A comparison of our method to Image Quilting [EF01], Self-Tuning Optimization [KNL*15], PSGAN [BJV17], and GCD Loss [ZCXH23]. Default parameters were used.*