DOI: 10.1111/cgf.14766

EUROGRAPHICS 2023 / K. Myszkowski and M. Nießner
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 2

Preserving the autocovariance of texture tilings using importance sampling

Nicolas Lutz, Basile Sauvage and Jean-Michel Dischler

ICube, Université de Strasbourg, CNRS, France

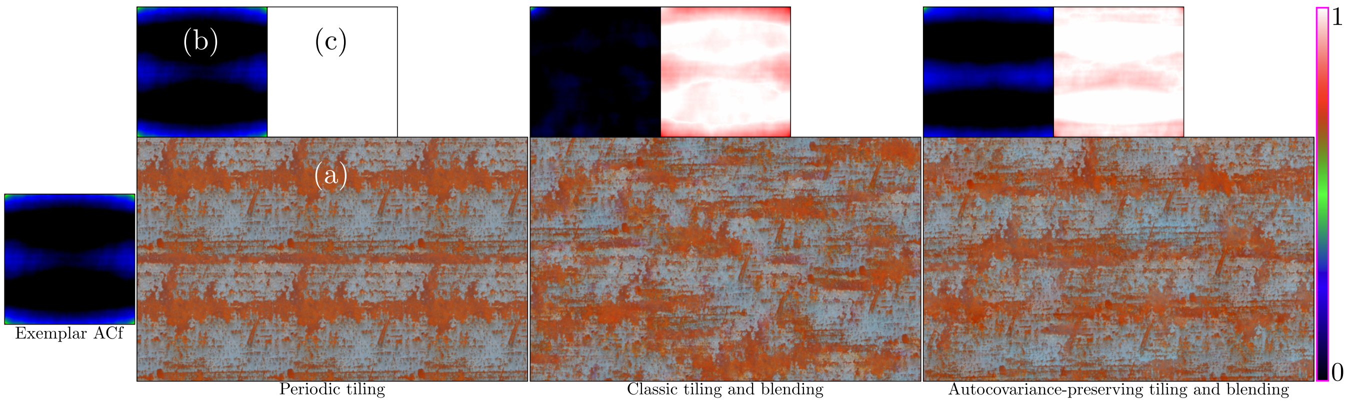

(b)
(c)
(a)
Exemplar ACf
Periodic tiling
Classic tiling and blending
Autocovariance-preserving tiling and blending
1
0

Figure 1: *Left: just repeating an exemplar perfectly preserves the autocovariance function (ACf), but produces an unpleasant repetition effect. Middle: real-time randomized tiling and blending [HN18] avoids this and creates variety by using a random uniform sampler to draw new contents. But by doing so, it does not preserve the ACf. Right: our new approach offers both, variety and preservation of the ACf by applying a well-chosen importance sampler. For each view: (a) synthesized output, (b) ACf of the output, (c) ACf similarity.*

Abstract

By-example aperiodic tilings are popular texture synthesis techniques that allow a fast, on-the-fly generation of unbounded and non-periodic textures with an appearance matching an arbitrary input sample called the "exemplar". But by relying on uniform random sampling, these algorithms fail to preserve the autocovariance function, resulting in correlations that do not match the ones in the exemplar. The output can then be perceived as excessively random.
In this work, we present a new method which can well preserve the autocovariance function of the exemplar. It consists in fetching contents with an importance sampler taking the explicit autocovariance function as the probability density function (pdf) of the sampler. Our method can be controlled for increasing or decreasing the randomness aspect of the texture.
Besides significantly improving synthesis quality for classes of textures characterized by pronounced autocovariance functions, we moreover propose a real-time tiling and blending scheme that permits the generation of high-quality textures faster than former algorithms with minimal downsides by reducing the number of texture fetches.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Rendering; Texturing;***

1. Introduction

Texture synthesis is the process of creating textures from a set of parameters. It can be used to ease the texture creation process or to texture surfaces from a set of parameters in rendering applications.

Depending on the algorithm, the generated textures may or may not have various desirable properties, such as: 1) eliminating repetition and alignment artifacts that both occur when trivially repeating a periodic texture on a surface (cf. the first view of Figure 2); 2) being extendable to a surface of unbounded (possibly infinite) size while

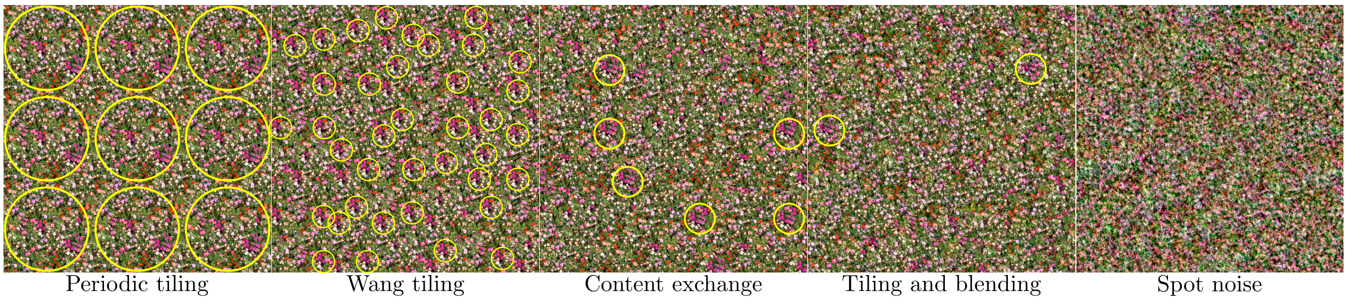


Figure 2: *Comparing most relevant tiling algorithms, with highlighted repetition artifacts (yellow). Periodic tiling, Wang tiling [CSHD03], Content exchange [VSLD13, LSLD19], all exhibit strong repetition artifacts. Tiling and Blending [HN18] generates much less repetitions, but comes along with ghosting artefacts related to the blending of contents. Spot noise [GGM11] does not exhibit any repetition at all and preserves the second-order moment of the exemplar, but the numerous blendings produce Gaussian statistics, thus radically altering the pattern.*

maintaining a high level of detail; 3) being as compact as possible in terms of memory footprint; 4) being computable at very fast framerates, e.g. being intrinsically computable in parallel at texel level to be GPU compliant; 5) being filterable on the fly.

By-example texture synthesis algorithms form a specific family of data-driven texture synthesis algorithms. Their particularity is to infer the parameters from a sample image, called the exemplar. While by-example texture synthesis is now a mature field, the synthesis of textures under the previous five constraints remains highly challenging. Typically, the very large majority of methods, like those based on optimization or neural networks, fail to address these constraints.

Previously, by-example procedural noise was considered to be a powerful tool for satisfying the aforementioned constraints. They succeed at reproducing statistics estimated from the exemplar, notably the first and second-order statistics through the preservation of the power spectral density, which is linked to the autocovariance function [GGM11]; however, these algorithms produce Gaussian statistics that do not preserve the fine structural details like edges and contours of exemplars lacking such statistics. In recent years, tiling texture synthesis has gained growing interest, notably with its recent "randomized tiling and blending" variation, as they conversely well satisfy the previous constraints altogether. Unlike procedural noise, they do not produce purely Gaussian patterns and are able to preserve local structures; but also unlike procedural noise, the preservation of specific statistics is more difficult. The preservation of first-order statistics have seen recent improvements [HN18]. Second-order statistics, on the other hand, have not been paid attention.

Our motivation is to unify these complementary features (local structures, first and second-order moments) to improve aperiodic tiling algorithms. We first generalize known aperiodic tiling models, whether they rely on blendings or not. We then show a simple method to preserve the autocovariance function by replacing the sampler with an importance sampler taking the second-order moment as probability density function. We show how the probability density function can be manipulated for increased control over the second-order moment of the generated output. Then, we use our generalization to present an aperiodic tiling algorithm that requires only two texture accesses for the computation of texel values, thus making it faster than current tiling and blending algorithms. We show that this approach is also well-suited for simulating cyclostationary processes.

2. Related work

This work is related to two families of textures synthesis, namely procedural noises and tilings. Closely related synthesis algorithms are illustrated in Figure 2.

2.1. By-example procedural noise

We define procedural noises as algorithms simulating stochastic processes by implementing a random function. Procedural noise modeling has a long history [LLC+10], but by-example procedural noise is more recent. Galerne et al. [GGM11] showed that random phase noise, which consists in computing the inverse Fourier transform of a spectrum with randomized phases [HWM06], and Spot noise [vW91], which consists in computing the sparse convolution of a spot, are both suitable for *by-example* noise synthesis: the former by using the spectrum of an exemplar, and the latter by using the exemplar itself as the spot. Sparse convolution noise was also adapted to the by-example scheme by computing kernels from the spectrum of an exemplar [LLDD09, LLD11, GDG12]. Local random-phase noise [GSV+14] introduced local Fourier Series to compute noises from exemplars with the additional possibility to fix some selected phases. Texton noise [GLM17] consists in optimizing the frequency content of the spot of spot noise to hasten power spectral density convergence of the result obtained by summing up about 30 spots, called textons, per texel on the GPU (it subsequently requires also about 30 texture accesses).

Procedural noise algorithms exploit sparse convolution to create more variety compared to other families of texture synthesis techniques. This allows procedural noise to avoid repetition and alignment artifacts by construction. One of the major advantages is that they preserve the second-order moment estimated from the exemplar, also by construction, since the latter is linked to the power

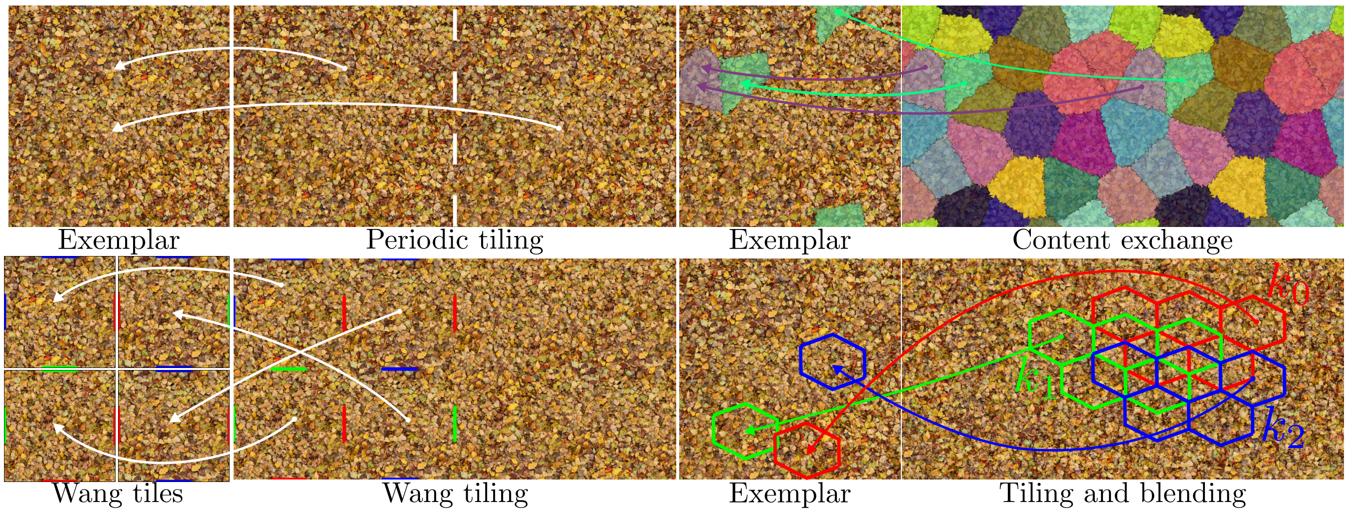


Figure 3: *Illustrations of relevant synthesis algorithms: periodic tiling and Wang tiling [CSHD03] use a single square tiling; content exchange [VSLD13,LSLD19] uses a single irregular tiling; tiling and blending [HN18] blends three regular, hexagonal tilings. These techniques use different samplers to fetch contents from the exemplar.*

spectral density. For exemplars that can be expressed as the realization of a stationary Gaussian process, this means that all statistical relationships are perfectly preserved, since stationary Gaussian processes only require the preservation of the mean and autocovariance function (first and second-order moments). However, when the exemplar is not the realization of a Gaussian process, statistics of first, second-order and beyond are lost in the output, despite the preservation of the first and second-order moments. This loss is related to the sums applied either in spatial or spectral domain, these sums enforcing the creation of Gaussian patterns as a consequence of the central limit theorem.

Heitz and Neyret [HN18] showed that it is possible to preserve first-order statistics of exemplars despite blending. This enables to synthesize stationary procedural noises from exemplars that do not necessarily have a normal histogram; however, it does not act on higher-order statistics. Hence it does not preserves statistical relationships among multiple texels at arbitrary given scales.

Some recent works try to preserve local structures, i.e. correlations in a neighborhood, in the context of procedural noises. Gilet et al. [GSV+14] fix phases in Fourier series, but it makes the synthesis tend towards periodic tiling and introduces both repetition and alignment artifacts. Phasor noise [TEZ+19] produces stochastic non-Gaussian patterns by separating a phase field and a highly contrasted wave profile. Grenier et al. [GSDT22] use 2D transfer functions indexed by a vector noise to produce structured stochastic patterns. However, these last two works are not example-based, thus making their practical usage difficult for non-experts.

Lutz et al. [LSD21] enhanced by-example procedural noise algorithms to produce textures characterized by periodically-correlated patterns, like brick walls. They introduce the mathematical model of cyclostationary Gaussian noise. Corresponding textures are generated by modifying the usual uniform random sampler to blend contents only at positions congruent to period vectors representing the period of the pattern.

Our solution for preserving the second-order moment of exemplars is inspired from cyclostationary noise. Our model fetches contents only at specific positions that optimize the output's autocovariance function.

2.2. Offline tilings

Early works in the field of texture synthesis, which apply tiling with or without blending, are able to preserve complex correlations. But these methods are too slow and cannot be used for on-the-fly infinite texturing. Patch-Based Sampling [LLX+01] constructs textures offline by placing carefully-chosen tile contents side by side with a small overlap. Carefully-chosen means that contents in overlapping regions are optimized so that they best match according to some error measure. Blending is then applied on the overlap. Quilting [EF01] and Graphcut [KSE+03] avoid blending by computing an optimal cut, but at the cost of even more important calculations. Liu et al. [LCT04] use the autocovariance function to detect the periods of a near-regular pattern and exploit a quilting method to reorganize content while keeping the periodic global aspect of the output. Our approach makes use of the autocovariance function to preserve the global aspect as well.

2.3. Aperiodic tilings

Synthesizing high-resolution textures can be trivially done using periodic tiling, which consists in repeating an identical and periodic (also called "seamless") texture sample over the surface (see Figure 2, left). While this guarantees a perfect preservation of higher order statistics, the goal of synthesis is generally to avoid the resulting tartan-like repetition and perfect alignment artifacts. "Aperiodic tiling" is the mathematical term used to designate a tiling that does not contain arbitrarily large periodic regions. Designing an aperiodic tiling requires choosing the shapes and contents of the tiles, how they are placed to tesselate the plane, and how potential discontinuities between connected tiles are hidden. We illustrate some examples of aperiodic tiling algorithms in Figure 3 : each algorithm has specific tile shapes and different ways of fetching contents (illustrated by arrows).

Wang tiling is one of the most famous aperiodic tiling methods. It was first introduced to computer graphics by Stam [Sta97]: square tiles of procedural noise are pre-computed with an identifier on each border and stored in memory; during synthesis, a tile is placed next to another so that the identifier on their common border is the same. Later works propose algorithms to synthesize well-connecting tile collections from an exemplar [CSHD03, ZK08]. Wei [Wei04] determines which tile can be put next to another on the fly e.g. (without pre-computation), and how to pre-filter the tiles to avoid aliasing, making it viable for real-time synthesis. The issue with Wang tiling is that any increase of variety on the output requires the creation of a greater tile collection. The growth is unfortunately exponential, meaning that in practical uses, alignment artifacts are still visible, although toned down. Vanhoey et al. [VSLD13] present content exchange to generate more varied outputs in real time. A periodic exemplar is repeated, and pre-computed alternative contents, chosen at random among a set, are drawn from this exemplar in pre-computed, periodically-repeating regions of the output. The advantage is that generating more variety only requires increasing the number of contents, which boils down to storing more 2D offsets. However, the more contents are used, the less these contents are likely to match each other when placed side by side, so that the number of visible seams might also increase. These artifacts can be reduced by carefully chosen irregular tile shapes [VSLD13], and by local blending on the border [LSLD19].

We call randomized tiling and blending, a class of synthesis algorithms that blend together several tiles, each tile containing a random region of the exemplar. Heitz and Neyret [HN18] present an algorithm which blends 3 hexagonal tiles aligned on a regular grid. Tiling and blending is at half way between aperiodic tiling (fast computation but potential repetition/alignment/discontinuities artifacts) and procedural noise (higher variety and no such artifacts, but heavily modified statistics). Because the synthesis picks tile contents randomly using a uniform random number sampler [HN18], there is no risk of alignment artifacts. Blending hides repetitions as well as discontinuities, at the expense of ghosting or blurring artifacts. Burley [Bur19] proposes to adjust the steepness of blending, in order to balance between these undesired artifacts. Since the contents are picked uniformly, blending mathematically cannot preserve possible correlations that might be present at different scales.

Aperiodic tilings are a powerful tool for real-time synthesis thanks to their speed and their preservation of details; but picking uniform random tiles results in the loss of the second-order moment, taking the form of excessive randomness. In our work, we enhance aperiodic tilings by picking contents with an importance sampler to better preserve the second-order moment.

3. Textures and random fields

The following section presents our notation of a texture, and recalls the mathematical theory of random fields, simplified to keep only the notions we need for our algorithms.

3.1. Textures

We define a 2D texture as a function $I : X \to S$, where X is an index set, and S a value set. X is bi-dimensional and either discrete or continuous. In our case, we consider X to be $\mathbb{R}^2$, because discrete indices can be interpolated. We note an element of X as $\mathbf{x}$, or (x_0, x_1) and refer to elements of X as "positions" for clarity. S is a value set of any dimension d (for instance, $d = 1$ for heightmaps and greyscale textures, and $d = 3$ for RGB) that can be either discrete or continuous. In our case, we consider S to be $\mathbb{R}^d$, because blending works best on a continuous set of values.

3.2. Random fields

A random field is a stochastic process defined by its probability space, a bi-dimensional index set X, and a value set S. We call *statistics* given properties of its probability density function. A texture synthesis algorithm can be expressed as the simulation of a random field, and a texture I is a realization of such a field. In by-example texture synthesis, we are interested in simulating a hypothetical random field that would have generated a given realization E called the *exemplar*. To accomplish this, assumptions must be made about this random process: the invariance of its statistics (for instance, stationary or cyclostationary), and their estimability (i.e. ergodicity). A random process is said to be stationary if its statistics are invariant by any translation [LLC+10]; it is cyclostationary if its statistics are periodic [LSD21].

In this work, all processes are assumed to be both mean-ergodic and autocovariance-ergodic, meaning these statistics can be robustly estimated on a sufficiently large realization. We also focus mainly on first and second-order statistics of the process, which are relevant to the appearance of textures in pre-attentive human vision [Jul81].

The first-order moment of a stationary process is the mean μ. It can be estimated on a realization I of the simulated process as

$$\widetilde{\mu}_I = \frac{1}{|X|} \int_X I(\mathbf{x})\, d\mathbf{x}. \tag{1}$$

The second-order moment of a stationary process is its autocovariance function r (ACf for short). It can be estimated on a realization I as

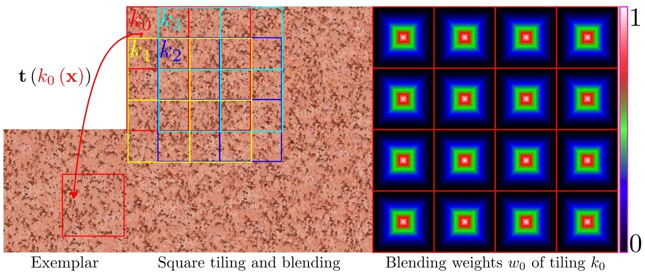
$$\widetilde{r}_I(\tau) = \frac{1}{|X|} \int_X (I(\mathbf{x}) - \widetilde{\mu}_I) \odot (I(\mathbf{x}+\tau) - \widetilde{\mu}_I)\, d\mathbf{x} \tag{2}$$

where $\odot$ represents the pointwise multiplication. The highest peak of covariance is always found at $\tau = \mathbf{0}$, and $r_I(\mathbf{0})$ is equal to the variance of the stationary process. In our figures, we visualize the normalized positive part $\max(0, r_I(\tau)/r_I(\mathbf{0}))$. The goal of this paper is to preserve the autocovariance function r_E of an exemplar E used in a texture tiling; we do this in section 5 with an importance sampler using the ACf r_E as pdf.

4. Algorithmic theory of tilings

In the following subsection, we propose a generic formulation of tiling algorithms. Our objective is to propose a formulation independent of specific implementations, i.e. a formulation that unifies the common aspects of all methods. Let $E : \bar{X} \to S$ be a texture used as an exemplar and $I : X \to S$ a texture synthesized from E. $\bar{X}$ refers to the set of valid positions in E without periodic extension for seamless exemplars. In our context, without loss of generality and to simplify the formulation, we assume E to be a periodic exemplar that can be defined on X through its periodic extension.

4.1. Output formulation



k_0
k_3
k_1
k_2
$\mathbf{t}(k_0(\mathbf{x}))$
1
0
Exemplar
Square tiling and blending
Blending weights w_0 of tiling k_0

Figure 4: *A tiling and blending using 4 square tilings k_0 to k_3 illustrating a synthesis that can be instantiated from our generic model. The blending functions w_0 to w_3 are proportional to the distance to the closest edge, with 1 at the center of each tile. w_0, which corresponds to the blending weights of k_0, is visualized on the right. The sampler **t** takes random tiles in the exemplar.*

In our generic model, the output I is expressed from E as the weighted blending of n mean-centered contents:

$$I(\mathbf{x}) = \sum_{i=0}^{n-1} w_i(\mathbf{x}) \left(E_i(\mathbf{x}) - \mu(\mathbf{x})\right) + \mu(\mathbf{x}), \qquad (3)$$

where E_i represents the content of a tiling, defined by

$$E_i(\mathbf{x}) = E\left(\mathbf{x} + \mathbf{t}\left(k_i(\mathbf{x})\right)\right).$$

k_i are the *tiling functions* as defined in section 4.2, $\mathbf{t}$ is a *sampler* as defined in section 4.3, and w_i are *blending weights* as described in section 4.4. The term $\mu(\mathbf{x})$ is the first-order moment of the process simulated by the synthesis and is used as a “correction factor” to preserve the variance [HN18]. It is constant for stationary processes where $\mu(\mathbf{x}) = \mu_E$, spatially-varying and periodic for cyclostationary processes [LSD21], and spatially-varying for our autocovariance-preserving method as well (see section 5.2).

We illustrate our model in Figure 4 with a tiling and blending model based on the blending of four square tilings, directly instantiated from our generic model.

4.2. Tiling functions

For a given $i < n$, k_i is a piecewise constant function $k_i : X \to \mathbb{N}$ which associates a tile identifier (integer) to any position in the output texture I. We refer to as the *j-th tile* of k_i all indices $\mathbf{x} \in X$ such that $k_i(\mathbf{x}) = j$. This function corresponds to any tiling of the Euclidian plane, where a unique identifier is attributed to each tile. For instance, in Figure 3, the periodic tiling has a unique ($n = 1$) constant tiling function, since the tile is also unique; Wang-tiling uses likewise a unique regular square tiling, but with different identifiers for each square; content exchange uses a unique irregular tiling, where each periodically repeated irregular tile has a unique identifier; the tiling and blending algorithm of Heitz and Neyret overlap three ($n = 3$) regular hexagonal tilings.

4.3. Sampler

A sampler is a function $\mathbf{t} : \mathbb{N} \to \bar{X}$, which associates an offset to a given integer used as a seed. It can be combined with the tiling function as $\mathbf{t}(k_i(\mathbf{x}))$ to associate a unique offset to a given tile (represented by arrows in Figure 3). For instance, in the examples of Figure 3, the sampler of the periodic tiling always yields $\mathbf{0}$; the sampler of Wang tiling draws offsets congruent with the size of a tile in an atlas of pre-computed tiles; the sampler of content exchange draws an offset among a set of pre-selected candidates; the sampler of tiling and blending chooses an offset randomly and uniformly in the exemplar. Lutz et al. [LSD21] reproduce periodic correlations by choosing only offsets congruent to exemplar-specific period vectors. Our method for preserving the autocovariance function is based on a controlled modification of the sampler $\mathbf{t}$.

4.4. Blending weights

Blending weights are relevant when several tilings overlap. A blending weight function is a function $w_i : X \to \mathbb{R}$, which associates a blending weight to a position of the output texture I. The goal of blendings is to introduce variety and avoid discontinuities in I. For instance, Heitz and Neyret [HN18] use blending weights that decrease from 1 at the center of each hexagon to 0 at the edge of each hexagon, and Lutz et al. [LSLD19] blend on the borders of the tiles of content exchange. Heitz and Neyret also noted that the norm of the vector $w(\mathbf{x})$ formed by all $w_i(\mathbf{x})$ should be constrained so that $||w||_2 = 1$ for all $\mathbf{x} \in X$ to preserve the contrast of E in blended areas and the stationary variance of the simulated process.

5. Autocovariance-preserving aperiodic tiling

The second-order moment is not well-preserved by randomized tiling and blendings that use a uniform sampler. We first mathematically explain why this moment is not preserved; then, we introduce our method to replace the sampler of equation 3 with an importance sampler to preserve the second-order moment; finally, we discuss our method, its results and its limits. Our method is illustrated in Figure 5.

5.1. Loss of autocovariance function

We observe that, when $\mathbf{t}$ is a uniform sampler, the autocovariance function $r_I(\tau)$ is biased toward 0 for large values of τ. This can be numerically observed by computing the output autocovariance, such as in Figure 1: in the second view, $r_I(\tau)$ almost vanishes everywhere except for small τ (top left corner).

To understand this, let us look at r_I in Equation 2. In order to

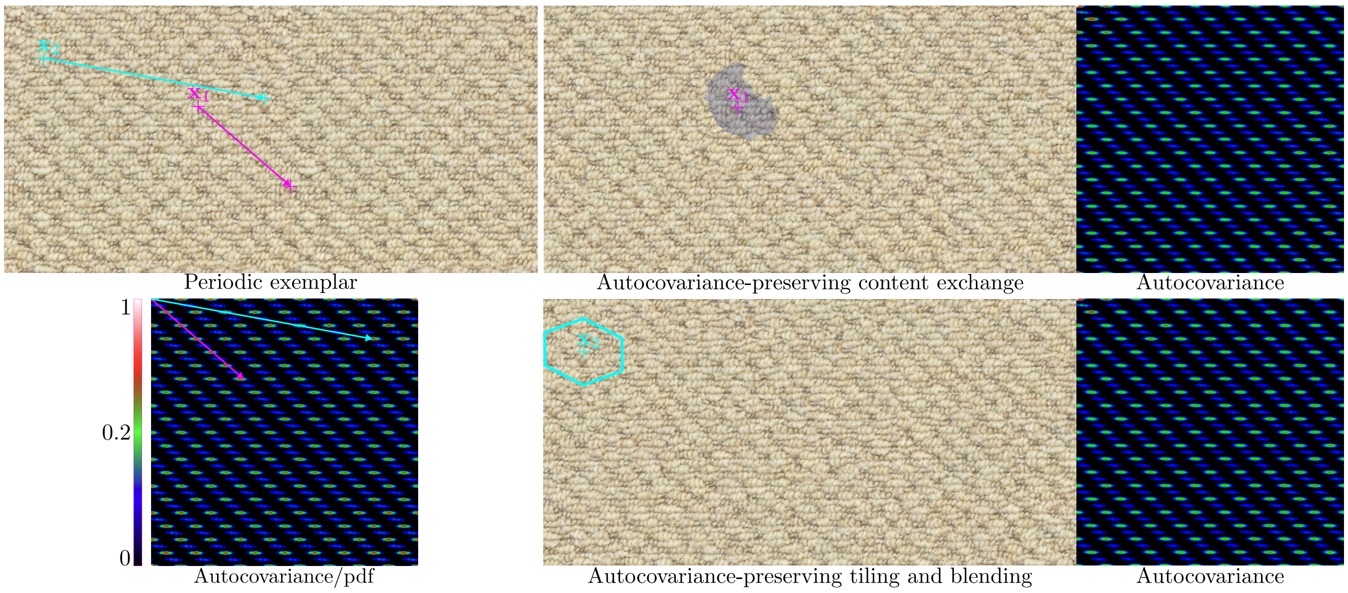


Figure 5: *Our autocovariance-preserving tilings. On a periodic exemplar (top left), we estimate the autocovariance function (ACf) (bottom left, on a log scale for visualization). We use an importance sampler taking the ACf as pdf instead of a random uniform sampler to pick contents whose covariances match better those of the exemplar. This enables a better preservation of the ACf (right).*

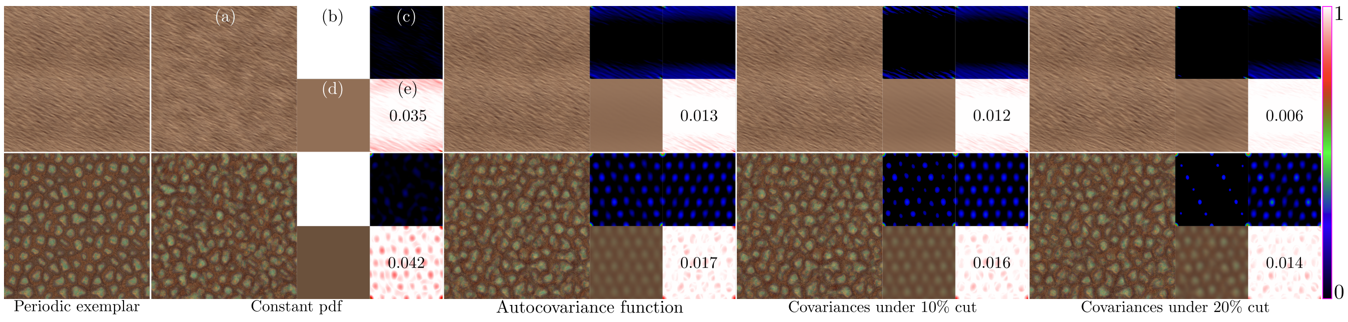


Figure 6: *Cutting low covariances in the pdf improves the match between the autocovariance function (ACf) of the output I and that of the exemplar E, but may increase alignment artifacts in I. (a) is the generated output, (b) is the pdf used, (c) is the ACf of I, (d) is the spatially-varying mean of I estimated using E and the pdf (b), and (e) is the similarity between the ACf of I and that of E (computed as* 1.0 *minus the Euclidian distance), along with its Euclidian norm.*

reproduce $r_I = r_E$, we need the pairs $(\mathbf{x}, \mathbf{x}+\tau)$ to be correlated the same way in I as in E. However, due to Equation 3, the randomness of $\mathbf{t}$ nullifies this correlation for different tilings k_i and k_j $(i \neq j)$ or different tiles $(k_i(\mathbf{x}) \neq k_i(\mathbf{x}+\tau))$. We provide a supplemental material elaborating on these results in more detail, with equations.

Our idea is to modify the sampler $\mathbf{t}$ in order to reestablish similar covariances between any pair of texels.

5.2. Proposed solution

Our goal is to preserve the autocovariance function of a periodic exemplar E in I as best as possible while keeping the benefits of aperiodic tilings. Ideally, r_I should be equal to r_E. As we have established in section 5.1, covariances within one single tile are perfectly preserved. We therefore need to minimize the difference of average products between E and I for all texels that are not on the same tile in I, for each tiling composing the output. The problem could theoretically be expressed as the search for an ideal sampler $\mathbf{t}$ that draws contents which minimize the difference between the autocovariance functions r_I and r_E. There are two issues with solving such a problem: issue 1) the optimum is a periodic tiling, which is not desirable; issue 2) an optimization involving surrounding texels is in contradiction with the parallel computation that we want to obtain, because in a parallel context, content-dependent cross dependencies among multiple texels are too difficult to preserve.

We first show how to partially preserve the autocovariance function (ACf) by using an importance sampler taking the ACf as pdf; then, how to compute the pdf from a multi-channel valued texture;

then, an improvement to control $\mathbf{t}$ to adjust the match of r_I with r_E; finally, how μ is computed differently than usual.

Importance sampling driven by the autocovariance function. Our solution consists in using as sampler $\mathbf{t}$ in Equation 3 an importance sampler [GI89] taking a probability density function (pdf) proportional to the ACf of E (r_E). Since a pdf must be positive and integrate to 1, we cut out the negative values in r_E (see related paragraph in section 5.3) and normalize it by its integral. Our approach is illustrated with the colored vectors in Figure 5 for both tiling and blending [HN18] and content exchange [VSLD13].

The intuition for this solution is as follows: we conclude from issue 1) that we have to settle for a sub-optimal solution that partially preserves the autocovariance function; and from issue 2) that we need some kind of guarantee that the content we draw will be well-correlated with surrounding tiles before we draw it. To do this, we use the exemplar E as a "guide" and try to maximize the cross-covariance between I and E, independently for each tile of each tiling:

$$cc_{I,E} = \frac{1}{|X|} \int_X (E(\mathbf{x}) - \mu_E)\, (E(\mathbf{x} + \mathbf{t}(k_i(\mathbf{x}))) - \mu_E)\, d\mathbf{x}. \quad (4)$$

This cross-covariance represents how well the tile $k_i(\mathbf{x})$ is correlated with E, if shifted by $\tau = \mathbf{t}(k_i(\mathbf{x}))$. We remark that $cc_{I,E}$ has the same expression as $r_E(\tau)$ in Equation 2, but the shift $\tau = \mathbf{t}(k_i(\mathbf{x}))$ varies spatially. In conclusion, how high $cc_{I,E}$ is on average for a tile is given by $r_E(\mathbf{t}(k_i(\mathbf{x}))$. Thus, our solution picks tiles with a probability relative to how well it maximizes $cc_{I,E}$.

This is our main compromise between a fully random uniform $\mathbf{t}$ that does not keep the second-order moment at all, and an optimal $\mathbf{t}$ that boils down to a periodic tiling and loses all variety.

Dealing with a multi-valued autocovariance function. The autocovariance estimator of equation 2 yields values in the value set of the simulated process, meaning that the estimation of the ACf on multi-channel exemplars is a multi-channel autocovariance. This is an issue for the importance sampler because explicit pdfs are supposed to be scalar maps. One solution would be to pick a different offset for each dimension according to their respective ACf; however, this would result in lost cross-covariances and cross-autocovariances between channels, an issue that was notably shown in Galerne et al. [GGM11] when dealing with several color channels. In our case, only one offset should be picked for all dimensions to preserve these cross-covariances.

We therefore choose to merge all dimensions of the ACf into one. Since naively blending channels does not enable an optimal preservation of the ACf for each individual dimension, we compute a PCA of the texels of E and only consider the ACf of the principal component. Our results and figures all use this particular ACf.

Dealing with negative correlations. When computed, the ACf usually contains negative values, which represent offsets at which texel values are, on average, negatively correlated. These offsets can be included in the sampler, for instance by re-centering the sampler's pdf by $+r_E(0)$ before normalizing it. Picking these offsets enables a slightly higher variety, but increase the chances that some blended contents do not match at all. We chose to cut them out for this reason, but the user remains free to include them or not.

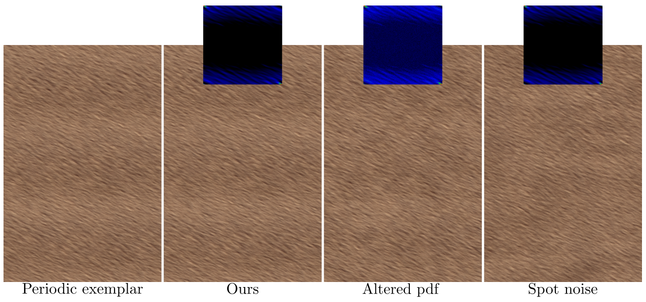


Figure 7: *Periodic exemplar (column 1) compared to the tiling and blending of Heitz and Neyret [HN18] with our autocovariance-preserving method (column 2) and the same algorithm using a pdf perturbated with additive white noise (column 3). Spot noise of the exemplar for comparison (column 4), which preserves the autocovariance function without loss of randomness, but is computed significantly slower and is only compatible with Gaussian inputs.*

Adjusting the match with the ACf of the exemplar. Our ACf-driven importance sampling is a compromise between two undesired extremes: perfect match (periodic tiling) and no match (uniform sampling). Depending on the artistic intention, this compromise may be not conservative enough, or, conversely, too repetitive. We propose to tune the probability density function, so as to increase or decrease the match between the ACf of the exemplar and that of the output.

Increasing the match can be achieved by modifying the provided pdf by increasing the relative weight of the highest covariances in the ACf, for instance by exponentiating all values or cutting out the lowest covariances. This comes at the expense of a reduced output variety due to a lower amount of different offsets being potentially picked and at usually smaller distances, which, in turn, may increase alignment artifacts. We show in Figure 6 the results generated using different input pdfs: none, the ACf, and the ACf with covariances lower than (respectively) 10% and 20% of the maximum cut.

Decreasing the match can be achieved by increasing the relative weight of the lowest covariances in the ACf. We implemented this in Figure 7 by pre-computing a realization of a low-amplitude white noise and adding it to the ACf of E to create the pdf. It enables to slightly increase the match with the autocovariance of E compared to a simple uniform random sampler, but without losing too much randomness.

Impact of the size of the tiles. The preservation of short- and long-range correlations depends not only on the importance sampler, but also on the tile size. Figure 8 shows the result for three tile sizes. Small tiles do not preserve fine details (individual scales are disrupted). Medium tiles preserve fine details but not larger patterns (e.g., regular dark spots). Large tiles improve fidelity, but reduce variety. Repetition artifacts may also appear.

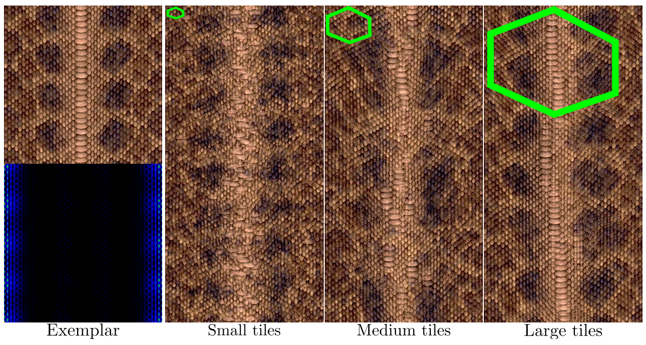


Figure 8: *Impact of the size of the tiles. The same exemplar (left) is used for the synthesis with small ($\times 1/3$), medium (default), and large ($\times 3$) tiles. Larger tiles better preserve fine details, at the expense of some variety.*

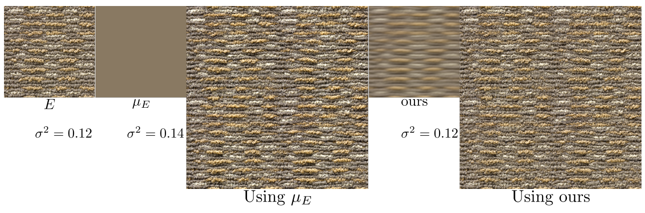


Figure 9: *The modification of the sampler makes the spatial mean μ_E an incorrect estimation of the mean μ of the simulated process, resulting in an incorrect variance (here σ^2 is the variance of the principal component). Our pdf-dependent estimation enables a more accurate estimation of μ that better preserves the variance. The pdf used is the ACf of E with covariances under 10% of the variance cut.*

Computing the first-order moment. In equation 3, $\mu(\mathbf{x})$ is an important term when blending tilings together in order to preserve the variance [HN18], which is required to preserve the autocovariance. The alteration of the sampler means that $\mu(\mathbf{x})$ is not accurately estimated by the spatial mean μ_E of E, but rather by a spatially-varying weighted mean where each texel $E(\mathbf{x}+\tau)$ is weighted by the probability $p(\tau)$ that this texel gets chosen to compute a texel $I(\mathbf{x})$. We therefore accurately estimate $\mu(\mathbf{x})$ as

$$\mu(\mathbf{x}) = \frac{1}{\sum_{\tau \in \bar{X}} p(\tau)} \sum_{\tau \in \bar{X}} p(\tau) E(\mathbf{x}+\tau), \tag{5}$$

where p is the pdf used in the importance sampling.

For real time synthesis, μ must be pre-computed and accessed on the fly, which can be done by pre-computing it on $\bar{X}$ and periodically repeating μ on X, much like the exemplar E. Using the spatial mean μ_E instead would save one texture access, but the contrast can be visibly altered, as shown in Figure 9.

5.3. Discussion

We show how faithful the autocovariance function is reproduced in Figure 6, with very little differences when low covariances are cut from the pdf.

Periodic aspect of the output. Our method has the side effect of altering the stationarity of the process and generating periodic statistics in the output, present in the periodically-repeated exemplar, that would otherwise be broken by the stationary algorithm. The periodic aspect of the output comes from the use of the periodic exemplar E as a "guide" for the ACf in section 5.2; if the output I were optimized according to its own autocovariance rather than the cross-covariance between I and E (which is not feasible in parallel), this effect would randomly fade as X increases in size.

We know that it is possible to preserve the autocovariance without creating this periodic effect because spot noise manages it [GGM11], but it does it by blending many tiles together, breaking the major advantage of tiling algorithms to be fast to compute and reduce ghosting artifacts compared to procedural noise. In fact, when Gilet et al. [GSV+14] worked on random phase noise [GGM11] to preserve fine details of the exemplar as well as the power spectrum, they found that perfectly preserving these also yields a periodic tiling. The approaches mirror each other: [GSV+14] get a periodic tiling by optimizing procedural noise, and we get a periodic tiling by optimizing randomized tiling.

Non-periodic exemplars. Our method, as presented, is only compatible with periodic exemplars. Using it on non-periodic exemplars requires to first construct a periodic (also called "seamless") exemplar from a non-periodic one, which usually means losing [MJH+17] or modifying [Moi11] some content of the original exemplar. The main challenge stems from the fact that we use a periodic tiling as a guide. There are other difficulties to overcome, such as possible incompatibilities inherited from tiling algorithms, obtaining a proper estimation of the spatially-varying mean on X that is not periodic, and the estimation of the autocovariance function in E which gets significantly less accurate for larger vectors τ.

6. Dual tiling and blending

In this section, we specialize the general model of section 4 to propose tiling and blending algorithms made of only two dual tiling functions. We show our "dual tiling and blending" algorithm with various tiling choices in Figure 10.

6.1. Construction

Our dual tiling and blending algorithm consists in instantiating the model of equation 3 by using only two tiling functions k_0 and k_1, as well as their respective blending functions w_0 and w_1, where k_1 is the dual of k_0. A dual is created from an arbitrary primal by replacing the center of each tile with a vertex, and joining the centers of adjacent tiles. We construct the blending weights according to the linear distance to the closest edge: 1 for the furthest point, and 0 at the edge (keeping the constraint of section 4.4 that $||w||_2 = 1$). The blending weights can be exponentiated as in Burley [Bur19], yielding different controllable results.

This model reduces the number of texture accesses to two, effectively reducing the theoretical computation time by almost a third if we compare it to the algorithm of Heitz and Neyret [HN18]. Blending only two tilings also reduces ghosting artifacts, which are

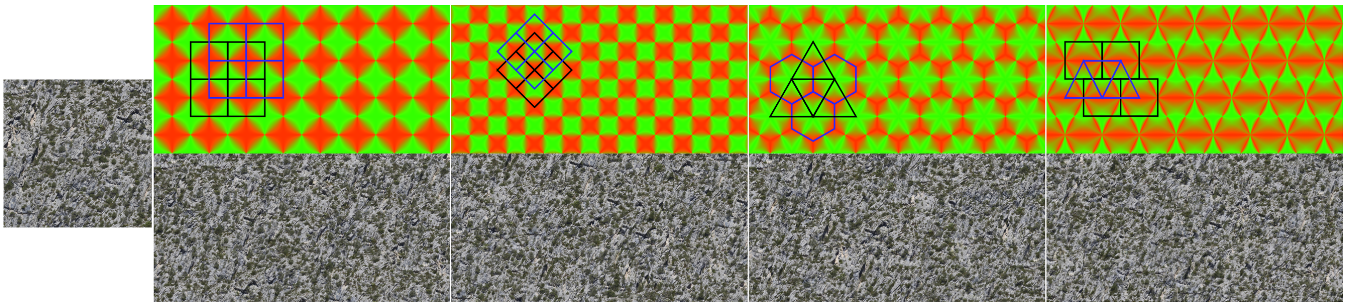


Figure 10: *Various dual tilings of various sizes used for our by-example texture synthesis. Tilings use the distance to the closest edge as blending weights. The leftmost texture is the exemplar.*

known to appear when blending several patterns together due to a Gaussianization of the output [GGM11]. It is directly compatible with the histogram transfer of Heitz and Neyret [HN18], although blendings only two contents makes the histogram transfer have a less significant impact on the first-order statistics. As a downside, the amount of potential repetition artifacts is increased, because reducing the number of blending also reduces the variety.

6.2. Hiding vertex singularities

The disadvantage of having only two tiling functions is that they cannot be superimposed without edge intersections where both blend functions are equal to 0. We refer to these points as "vertex singularities" and our model yields undefined values in these areas.

Instead of modifying our model, we propose to solve this problem by a slight modification of the algorithm. This modification should not add any access to the texture in order not to increase the computation time. We therefore propose two possible approaches: either we slightly amplify the mixing weights so that they never both reach a zero value, or to add a third function w_2 that weakly mixes the (stationary) mean of the exemplar near these singularities. Using either modification requires to adjust $||w||_2$ accordingly for the preservation of the variance (see section 4.4).

6.3. Cyclostationarity

Lutz et al. [LSD21] adapt stationary Gaussian synthesis to cyclostationary Gaussian synthesis. The class of cyclostationary stochastic random fields enable the synthesis of exemplars with strictly periodic statistics, such as brick walls, tilings, or any other periodically-correlated patterns. Firstly, we note that the cyclostationarity model can be encompassed within the tiling model of section 4 for tiling algorithms. It boils down to using a sampler that picks tiles on positions congruent to period vectors and to take μ as the periodic mean of the simulated cyclostationary process [LSD21].

Aligning patterns and tiling. Cyclostationary processes can be used for the representation of man-made structures that can often be expressed as tilings with random variations, much like how near-regular textures were first introduced [LTL05]. A perspective of our generic model is to correlate the tilings with the contents of the tiles

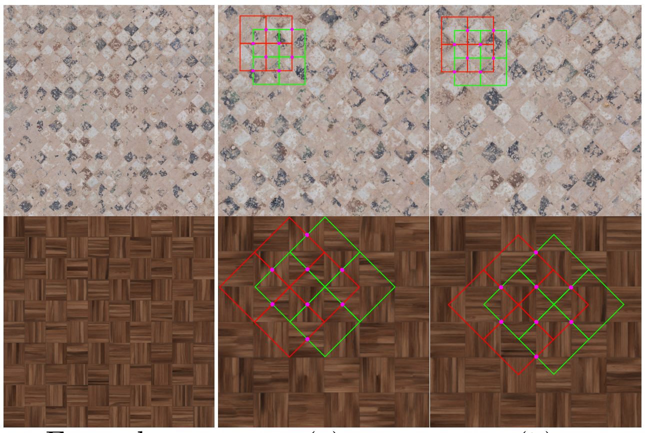


Figure 11: *Exemplars (left) synthesized with our cyclostationary synthesis, using various primal (red) and dual (green) sizes. (a) is synthesized with a misaligned tiling, generating high variety but unnatural-looking tiles (top row: lots of red tiles cut with blue ones; bottom row: shift of line pattern in each element) (b) is synthesized by aligning the singularities of the dual tiling with high frequency corners of the patterns, generating high variety without breaking tile appearance.*

to better preserve the appearance of some elements in the pattern. While this is hardly controllable for stationary patterns that have by nature unpredictable patterns, some cyclostationary patterns have periodic contents that can act as "natural singularities" for our dual tiling and blending synthesis, such as intersections or edges between uncorrelated contents. Since the most abrupt transforms in I happen near singularities, we chose to generate textures by aligning singularities with these points. We validate this enhancement in Figure 11, where we enable high variety while avoiding the creation of otherwise unnatural patterns.

7. Results

In this section, we discuss the results of our work in terms of computational time, memory consumption, and quality. Our supplemental materials contain the source code of our GPU imple-

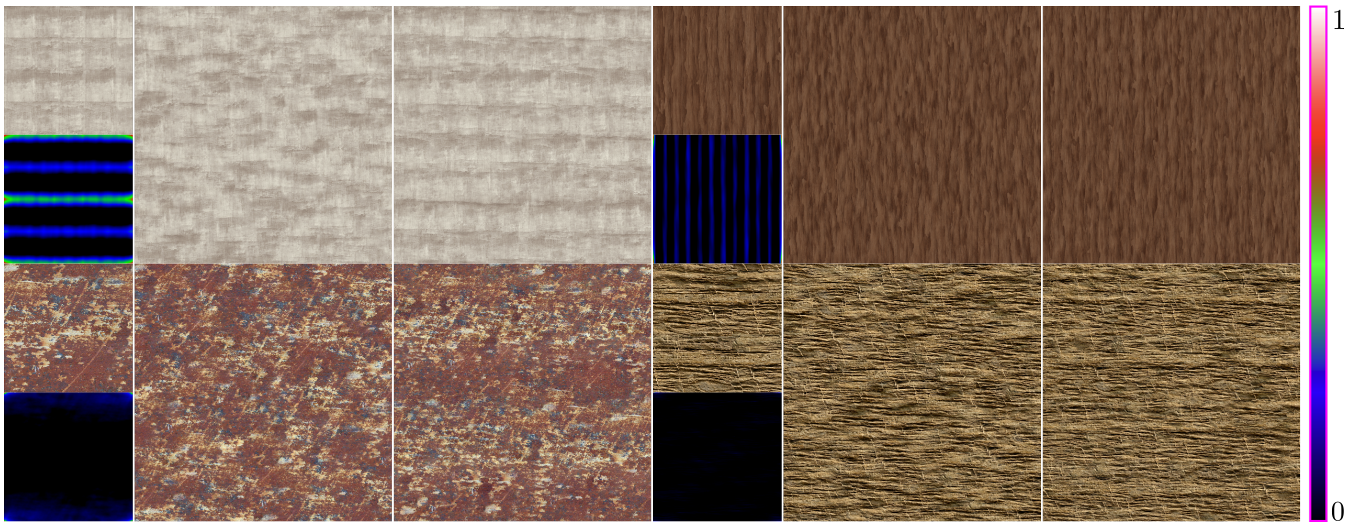


Figure 12: *Exemplars and their autocovariance function used as a pdf for the sampler (left for each view); their synthesis using randomized tiling and blending (middle), and their synthesis using our autocovariance-preserving tiling and blending (right). Our supplemental materials include more results, with comparisons of the ACf.*

mentations with pre-computed material, two shadertoy links of our dual tiling and blending synthesis, and a sheet of results of the autocovariance-preserving tiling and blending with comparisons of the ACf. The code for computing these data along with a CPU version of the synthesis algorithm are also available in the ASTex library [ast].

7.1. Autocovariance preservation

We discuss here the results of the autocovariance-preserving tilings of section 5 in particular, with some relevant instances shown in Figure 12.

Memory and computational time. As shown in section 5.2, the adaptation of any tiling algorithm requires replacing the original sampler with an importance sampler. For tiling and blending algorithms, the proper estimation of the expectation of the output, usually represented by the spatial mean, is spatially-varying on the exemplar. In terms of memory consumption, this requires the storage of the probability density function, and the spatially-varying mean. In terms of computational time, this requires a search in the probability density function for each tiling to pick an offset, and one additional texture access on the spatially-varying mean to add it after blending.

Implementation of the importance sampler. The importance sampler on the 2D discrete pdf can be implemented by pre-computing all conditional densities for one dimension according to the other and then pre-computing the marginal density for the other [PJH16]. At runtime, this requires searching on the marginal density once, then searching on the corresponding conditional density for the final coordinates. Since the importance sampler is an ergodic process, the probability density function can instead be condensed into a single but sufficiently large pre-computed realization of the importance sampler. At runtime, fetching new offsets only requires drawing one random uniform sample from this realization, requiring only one array access.

Quality. We have shown that the contribution of section 5.2 systematically enables a better preservation of the stationary autocovariance function estimated from the exemplar, regardless of the exemplar used. Our method has a greater impact for textures which have highly characterized autocovariance functions. Preserving the autocovariance function or the spectrum of the exemplar in the output has often been considered a “sweet spot” for well reproducing procedural noise patterns [LLC+10]. This makes our method especially efficient for reproducing procedural noise patterns with a standard quality using tiling and blending. We would however like to argue that the preservation of the autocovariance, in some cases, is either not necessary, or not sufficient to achieve a satisfactory appearance.

Preserving the autocovariance function is sometimes not necessary because it comes at the cost of a lowered variety in the output. This loss in variety can be seen in Figure 6, third column group, where the subtle tone change in the sand texture is periodically repeated throughout the entire domain (even if without verbatim copy) since the autocovariance function is periodically preserved. This repetitive effect may not always be pursued by artists. In Section 5.2 we show how to tune this, and Figure 7 illustrates the result with a realization of spot noise for reference.

Preserving the autocovariance function is sometimes not sufficient because some textures have complex and spatially-varying internal statistical relationships. Figure 13 shows three examples: in (a) alignment rules and texture elements are not constant, in (b) the pattern has a complex global coherency and in (c) there are different patterns at different scales. These textures are too difficult to be well-synthesized with our model, regardless of the sampler

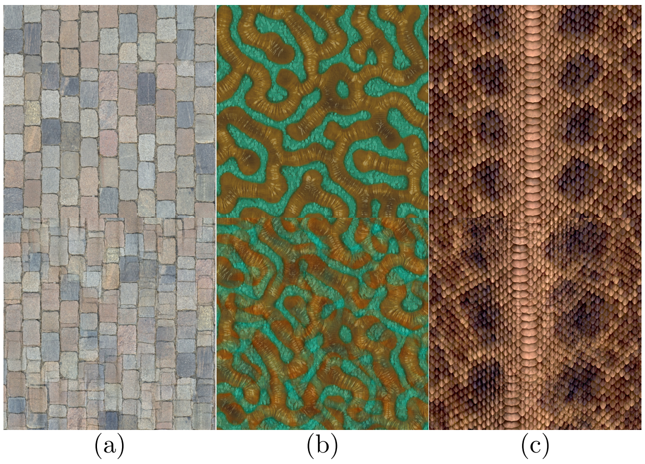


Figure 13: *Typical failure cases: exemplars (top of each view) synthesized with our method (using large tiles). They exhibit internal statistical relations that are too complex to be well-reproduced by simply preserving the ACf.*

used. Such textures would require a higher degree of preservation of statistics rather than only the second-order moment.

7.2. Dual tiling and blending

For discussing this part of our work, we focus on comparing the method of section 6 with the tiling and blending algorithm of Heitz and Neyret [HN18] (that we simply refer to as *T&B* in this section), since it is a direct enhancement of the latter with both pros and cons. We display raw results of our dual tiling and blending with a square primal in Figure 14.

Computational time and memory. Both T&B and our dual tiling and blending take the same amount of memory on a GPU (i.e. the exemplar). Our method spares one texture fetch, diminishing the computational time required. The impact on performances depend on the GPU model, the viewing angle, the size of the exemplar and the scene; in our additional materials, we used the OpenGL T&B implementation of Deliot and Heitz [DH18], and compared it using the same code, with a fragment shader of our method, provided in additional materials.

Quality. The main feature of dual tiling and blending is that the number of blended tiles is limited to two. Compared to T&B, this helps to preserve fine details that get lost in blending, but lowers variety in the output since lesser new content is produced. The quality is also diminished due to the singularities generated which create point discontinuities, although we attenuated this effect in section 6.2. For the simulation of cyclostationary fields, we have shown in section 6.3 that it is possible in some cases to hide singularities in the corners of a pattern, making this synthesis especially suited for these cases.

8. Conclusion

We have presented an importance sampling scheme dedicated to texture tiling algorithms, which allows to control the autocovariance of the result. We based our work on a mathematical model (Equation 3) that encompasses several algorithms from the state of the art. From this model we derived a new algorithm, that we called *dual tiling and blending*, that only requires two texture accesses.

A first perspective is to explore non-stationary signals. A first step into this direction has been made with cyclostationary processes [LSD21]. Such signals are the realization of non-stationary random processes, that are significantly more difficult to control. It would imply to deal with a spatially-varying autocovariance, and thus a spatially-varying sampler.

A second perspective is to deepen the control of statistics. In this work we investigated the second-order *moment*, i.e. the autocovariance. While it perfectly controls Gaussian patterns, it does not perfectly control non-Gaussian patterns –this is well-known for spot noise [GGM11]. Controlling *full second-order statistics* remains a challenge.

A third perspective is to further exploit the potential of the tiling and blending model. In Equation 3 we controlled the sampler to better reproduce the texture content. The tiling functions k_i could also be optimized to hide artifacts, for instance by adjusting the shape of the tiles to the texture patterns, as we began to do in section 6.3. Another opportunity is to adjust the blending weights w_i depending on the content of the texture, in order to reduce ghosting artifacts.

Acknowledgments

This work has been funded by the project ReProcTex from the Agence Nationale de la Recherche (ANR-19-CE33-0011-01). It was also partially funded by the University of Sherbrooke and Ubisoft Divertissement in the context of a Mitacs Accelerate Fellowship. All textures are provided by https://textures.com.

References

[ast] github.com/ASTex-ICube/ASTex. 10

[Bur19] Brent Burley. On histogram-preserving blending for randomized texture tiling. *Journal of Computer Graphics Techniques (JCGT)*, 8(4):31–53, November 2019. 4, 8

[CSHD03] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *ACM Transactions on Graphics*, 22(3):287–294, 2003. 2, 3, 4

[DH18] Thomas Deliot and Eric Heitz. Procedural stochastic textures by tiling and blending. *GPU Zen 2: Advanced Rendering Techniques*, 2018. 11

[EF01] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 341–346. ACM, 2001. 3

[GDG12] Guillaume Gilet, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Multiple kernels noise for improved procedural texturing. *Vis. Comput.*, 28(6-8):679–689, June 2012. 2

[GGM11] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. Random phase textures: Theory and synthesis. *IEEE Transactions on Image Processing*, 20(1):257 – 267, 2011. 2, 7, 8, 9, 11

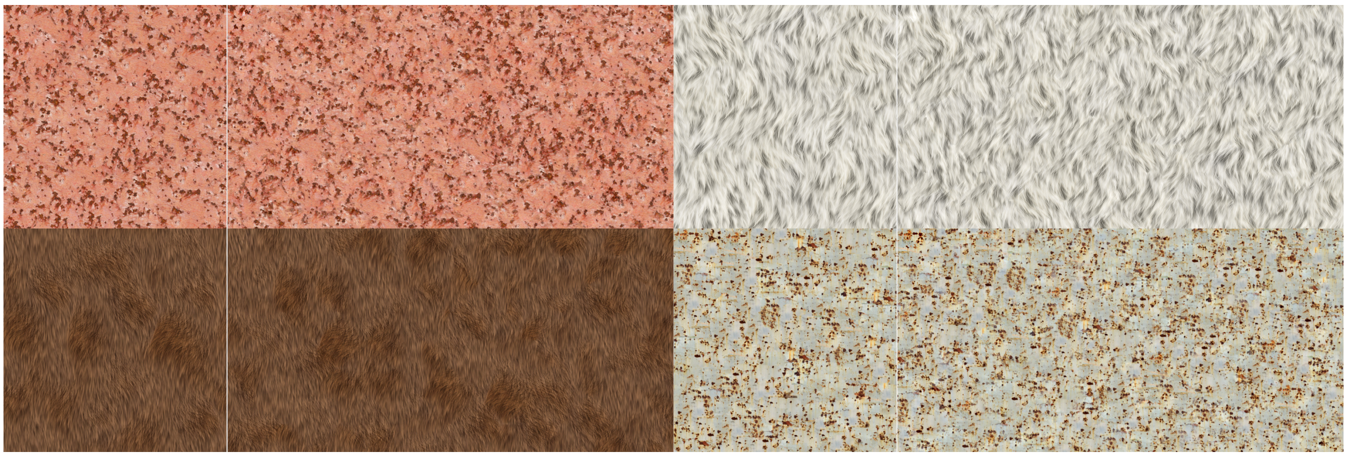


Figure 14: *Exemplars (left of each view) synthesized with our dual tiling and blending. More results can be visualized with our GPU and shadertoy code.*

[GI89] Peter W. Glynn and Donald L. Iglehart. Importance sampling for stochastic simulations. *Management science*, 35(11):1367–1392, 1989. 7

[GLM17] Bruno Galerne, Arthur Leclaire, and Lionel Moisan. Texton noise. *Computer Graphics Forum*, 36(8):205–218, 2017. 2

[GSDT22] Charline Grenier, Basile Sauvage, Jean-Michel Dischler, and Sylvain Thery. Color-mapped noise vector fields for generating procedural micro-patterns. In *Computer Graphics Forum*, volume 40, 2022. 3

[GSV$^+$14] Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Local random-phase noise for procedural texturing. *ACM Trans. Graph.*, 33(6):195:1–195:11, November 2014. 2, 3, 8

[HN18] Eric Heitz and Fabrice Neyret. High-performance by-example noise using a histogram-preserving blending operator. *Eurographics Symposium on High-Performance Graphics 2018*, 2018. 1, 2, 3, 4, 5, 7, 8, 9, 11

[HWM06] Danny Holten, Jarke J Van Wijk, and Jean-Bernard Martens. A perceptually based spectral model for isotropic textures. *ACM Transactions on Applied Perception (TAP)*, 3(4):376–398, 2006. 2

[Jul81] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981. 4

[KSE$^+$03] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003. 3

[LCT04] Yanxi Liu, Robert T. Collins, and Yanghai Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):354–371, 2004. 3

[LLC$^+$10] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, D.S. Ebert, J.P. Lewis, Ken Perlin, and Matthias Zwicker. State of the art in procedural noise functions. In Helwig Hauser and Erik Reinhard, editors, *EG 2010 - State of the Art Reports*. Eurographics, Eurographics Association, May 2010. 2, 4, 10

[LLD11] Ares Lagae, Sylvain Lefebvre, and Philip Dutré. Improving gabor noise. *IEEE Transactions on Visualization and Computer Graphics*, 2011. 2

[LLDD09] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. Procedural noise using sparse gabor convolution. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2009)*, 28(3):54–64, July 2009. 2

[LLX$^+$01] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001. 3

[LSD21] Nicolas Lutz, Basile Sauvage, and Jean-Michel Dischler. Cyclostationary Gaussian noise: theory and synthesis. In *Eurographics 2021*, Vienna, Austria, May 2021. 3, 4, 5, 9, 11

[LSLD19] Nicolas Lutz, Basile Sauvage, Frédéric Larue, and Jean-Michel Dischler. Anisotropic filtering for on-the-fly patch-based texturing. In *Eurographics 2019*, Genoa, Italy, May 2019. 2, 3, 4, 5

[LTL05] Yanxi Liu, Yanghai Tsin, and Wen-Chieh Lin. The promise and perils of near-regular texture. *International Journal of Computer Vision*, 62(1-2):145–159, 2005. 9

[MJH$^+$17] Joep Moritz, Stuart James, Tom S.F. Haines, Tobias Ritschel, and Tim Weyrich. Texture stationarization: Turning photos into tileable textures. *Comput. Graph. Forum*, 36(2):177–188, 2017. 8

[Moi11] Lionel Moisan. Periodic plus smooth image decomposition. *Journal of Mathematical Imaging and Vision*, 39(2):161–179, 2011. 8

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation: Third edition*. 11 2016. 10

[Sta97] Jos Stam. Aperiodic texture mapping. 08 1997. 4

[TEZ$^+$19] Thibault Tricard, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, and Sylvain Lefebvre. Procedural phasor noise. *ACM Transactions on Graphics*, 38(4):Article No. 57:1–13, July 2019. 3

[VSLD13] Kenneth Vanhoey, Basile Sauvage, Frédéric Larue, and Jean-Michel Dischler. On-the-fly multi-scale infinite texturing from example. *Transactions on Graphics*, 32(6):208:1–208:10, 2013. (Proceedings of Siggraph Asia'13). 2, 3, 4, 7

[vW91] Jarke J. van Wijk. Spot noise texture synthesis for data visualization. *SIGGRAPH Comput. Graph.*, 25(4):309–318, 1991. 2

[Wei04] Li-Yi Wei. Tile-based texture mapping on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '04, pages 55–63. ACM, 2004. 4

[ZK08] Xinyu Zhang and Young J Kim. Efficient texture synthesis using strict wang tiles. *Graphical Models*, 70(3):43–56, 2008. 4