Cyclostationary Gaussian noise: theory and synthesis

Nicolas Lutz, Basile Sauvage, Jean-michel Dischler

EUROGRAPHICS 2021 / N. Mitra and I. Viola
(Guest Editors)

Volume 40 (2021), Number 2

Cyclostationary Gaussian noise: theory and synthesis

Nicolas Lutz, Basile Sauvage and Jean-Michel Dischler

ICube, Université de Strasbourg, CNRS, France

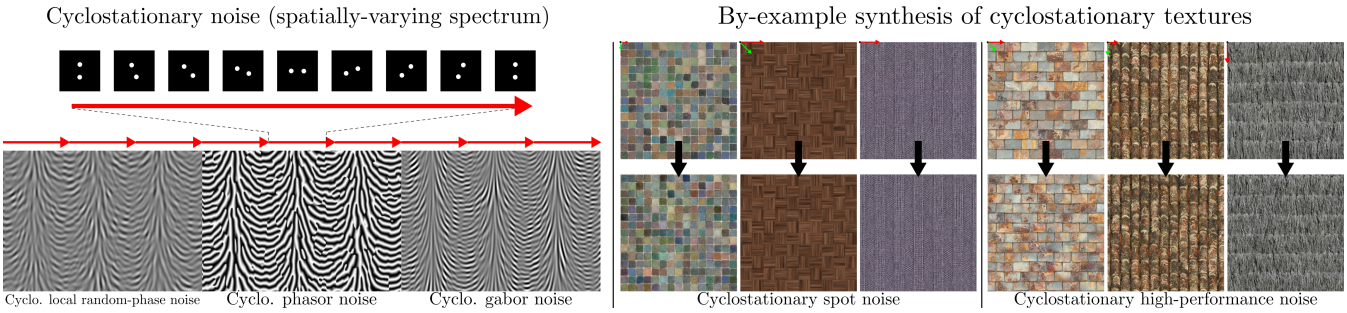

Figure 1: *Cyclostationary noise is a procedural model for cyclostationary textures, which are patterns exhibiting a periodicity in their statistics. We convey existing stationary noises to a cyclostationary context, enabling the synthesis of cyclostationary textures controlled by spectra (left) and by an exemplar (right).*

Abstract

Stationary Gaussian processes have been used for decades in the context of procedural noises to model and synthesize textures with no spatial organization. In this paper we investigate cyclostationary Gaussian processes, whose statistics are repeated periodically. It enables the modeling of noises having periodic spatial variations, which we call "cyclostationary Gaussian noises". We adapt to the cyclostationary context several stationary noises along with their synthesis algorithms: spot noise, Gabor noise, local random-phase noise, high-performance noise, and phasor noise. We exhibit real-time synthesis of a variety of visual patterns having periodic spatial variations.

CCS Concepts

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

1. Introduction

Procedural noise has many applications in computer graphics, ranging from geometric modeling and animation to texture synthesis, as it is able to compactly represent continuous stochastic stationary processes. This explains the many research works that noise generated during the past decades. However, the physical nature of some natural or man-made phenomena often cause the assumption of stationarity to be violated. Modeling non-stationary stochastic processes is more difficult, and such processes have not been well studied yet.

A key application of noise is procedural texture modeling. Purely random patterns can be directly generated with noise; an important class of such patterns are textures that are realizations of stationary Gaussian processes. Recent noise synthesis approaches attempt to enlarge the range of patterns that noise is able to reach directly, that is, with no additional procedural modeling technique, like local random-phase [GSV+14], high-performance [HN18] and Phasor [TEZ+19] noise. Our goal is to follow on from these works and further increase the range of patterns reachable with a stand-alone procedural noise.

Therefore, we study a new class of non-stationary stochastic processes, called cyclostationary stochastic processes, which have not been studied yet in the field of computer graphics and procedural noise synthesis to the best of our knowledge. Such processes, characterized by the periodicity of their statistics, are well known in signal processing. They are used to model periodically-correlated random signals such as signals resulting from rotating machines, radar systems, or weather patterns. We introduce novel noises, called cy-

Figure 2: *Examples of cyclostationary (CS) textures, which exhibit a periodic organization when zooming out and, at the same time, strong stochastic color variations when zooming in. These types of near-regular textures are difficult to model, because colors are neither repetitive nor purely random: color randomness is periodic.*

clostationary Gaussian noises, which we define as models and algorithms implementing cyclostationary Gaussian processes. These noises can compactly and procedurally represent a class of textures we subsequently call *cyclostationary noise textures*, defined by their periodically random pattern. The periodicity and repetition of these textures is not obvious and it can be easier to perceive their random nature. We showcase examples of cyclostationary textures in Figure 2: they exhibit strong stochastic variations, best observable when zooming in, while having a repetitive spatial organization, best observable when zooming out.

Previous techniques introduced a class of textures, called near-regular textures [LLH04], defined as periodic tilings with geometry and color variations. With this respect, our work provides a mathematically grounded, novel and complementary type of color variations, which can be strongly stochastic and detailed. Moreover, our cyclostationary textures can be synthesized in real-time and are unbounded, i.e. they have an infinite expanse.

First, we explore the mathematical properties of cyclostationary Gaussian processes, their statistics, and their link with cyclostationary textures in section 3. We then present various ways of generating patterns using classic stationary processes, notably with a new spatially-varying spectrum model, put in practice by adapting real-time procedural noises known as local random phase noise, Gabor noise and phasor noise in section 4. Then, we show how we achieve a high-quality by-example synthesis of cyclostationary textures by adapting the real-time procedural noises known as spot noise and high-performance noise as well as a cyclostationary version of histogram transfer in section 5. We then show how to estimate the period of cyclostationary textures in section 6; finally, we discuss details, comparisons and limitations in section 7.

2. Related work

Our work is related to noise and Gaussian processes, to the theory of cyclostationary processes and to near-regular textures. We briefly review these topics.

Near-regular textures. Near-regular textures have been introduced to address a specific class of repetitive textures that could not be sufficiently well generated with aperiodic tilings [Sta97, CSHD03, Wei04, LHW+04] or patch-based synthesis [EF01, LLX+01, KSE+03, WY04, LH05]. Liu et al. [LLH04, LTL05] define near-regular textures as a periodic tiling that undergoes spatial, intensity and color transformations. Various works followed: evaluations and comparative studies [WHC+06], analysis methods to detect regularities [DED05, CB11, PBCL09, CB13, LPVV17, LNS+15], and improvements of synthesis quality [NMMK05, HH09, RHE11]. But managing simultaneously regularity and randomness of such textures still remains a challenging issue. When color variations are dominant (e.g. the resulting pattern is close to Gaussian noise), synthesis quality quickly breaks down, PCA [LLH04] not being an appropriate tool and patch-based methods not providing sufficient variety [RHE11]. We remark that none of these methods proposed a procedural noise model for handling color transformations more efficiently and in real-time.

In the last years, texture synthesis has been carried through convolutional neural networks (CNN) [GEB15]. Some of the recent approaches attempt to automatically recover regularities and self-similarities in near-regular exemplars. Liu et al. [LGX16] use spectrum constraints and constraints learned from a CNN, while deep correlations [SCO17] use a structural energy, based on correlations and deep learning. These methods are not adapted to "on-the-fly" synthesis, and are restricted to medium size textures. Faster and unbounded texturing models have been proposed [ULVL16, GRGH20], but remain computationally expensive and are not suited for near-regular texture synthesis.

Gaussian processes. Noise synthesis, carried through Gaussian processes, finds its roots in sparse convolution noise [Lew84]. Since then, different variations of this model have been proposed: Gabor noise [LLDD09, LLD11, TNVT19] sums a Gabor kernel with random weights to generate noise with controlled spectrum. Random phase noise [GGM11] uses Fourier series and a random phase to generate noise. Gilet et al. [GSV+14] shows how fixing some phases can help preserve the spatial organization of an exemplar; however, freezing phases makes the output tend towards a perfectly periodic signal, which comes at the expense of variety. Guingo et al. [GSDC17] explores the possibility of using several spectra to provide a higher control over the output spatial organization, but without synthesizing near-regular textures. Phasor noise [TEZ+19] generate patterns with strong variations of intensity, by controlling the instantaneous phase and a wave profile. Spot noise [vW91] is a notable implementation of the ADSN model presented later by Galerne et al. [GGM11], which consists in blending a discrete kernel at random positions. These works were improved with texton noise [GLM17], which computes a summarized version of the kernel, optimized for the synthesis. The work of Pavie et al. [PGDG16] and Cavalier et al. [CGG19] show that controlling the position of these spots helps controlling the visual aspect of the output. High-performance noise [HN18, DH18, Bur19] is a very fast method approximating the ADSN model. It blends hexagonal tiles taken randomly from a discrete exemplar and arranged on a regular lattice. The small number of blended values enables the preservation of non-Gaussian features, but diminishes the variety of the outputs.

These noise models can often be adapted to the by-example case, either by using the spectrum of the exemplar [GGM11, GLLD12,

GSV+14], or, in the case of the ADSN model, by using the exemplar to build a spatial kernel [GGM11, GLM17, HN18].

All these models share issues related to the stationarity of the underlying Gaussian process that they build upon. They are thus unable to synthesize periodic statistics, and subsequently unable to model color transformations in near-regular textures. Works attempting to control the spatial organization of the underling stationary process do so by either incidentally altering the stationarity [GSDC17, PGDG16, CGG19], by varying the sparse convolution using specific point processes; or, by altering their Gaussian nature through the control of the phase [GSV+14, TEZ+19]; or by reducing the convergence towards a Gaussian process [HN18]. Conversely, we propose to alterate stationarity and keep the Gaussian nature by exploiting a generalization to Gaussian cyclostationary processes.

Cyclostationary processes. Cyclostationary processes are periodic processes that can be seen as interleaved stationary processes (and, as such, are a generalization of stationary processes). They were first presented in [Ben58], who noted that the synchronization algorithms used in communication systems exhibited non-stationary signals with a periodic mean and a periodic autocovariance function. Since then, they have been used in various contexts (such as gear systems, mechanical machinery, econometric, or astronomy), but, to our knowledge, have never yet been exploited in texture synthesis in an explicit way. Our presentation of cyclostationary processes and their properties is based on the recent work of Napolitano [Nap19], a book summarizing 60 years of research on cyclostationary processes and their properties. The work of Kipnis et al. [KGE18] was also used, as it establishes the link between our spatially-varying spectrum and the autocovariance function of cyclostationary noises.

It is important to note that even though cyclostationary random fields have been defined, the cyclostationary theory has been mostly elaborated for one-dimensional processes. As such, some of the cyclostationary theory presented in this paper is a generalization of the 1D model.

3. Cyclostationary model

The main idea of this paper is to consider *cyclostationary* (CS) Gaussian processes as generators of procedural noise, characterized by the periodicity of its statistics. This section presents their respective properties and the link between them.

The term "noise" is well defined as a stationary random function characterized by its power spectral density [LLC+10]. In computer graphics, the terms random process, noise and textures are sometimes merged; moreover, despite similarities with classic noise, our model is characterized by additional parameters. To avoid confusions between these three terms, we use them as follows:

- "Process" refers to a general random process, defined through its mathematical properties, as in section 3.1.
- "Noise" refers to a specific process, defined by a computable formula, along with algorithms to compute it. We explore several noises in Sections 4 and 5.
- "Texture" refers to an actual image, one result of an algorithm, and one realization of a process.

3.1. Cyclostationary processes

Processes. A stochastic process Y is a set of random variables $Y(\mathbf{x})$ defined on a probability space for which the sample space is a subset of $\mathbb{R}^d$, and $\mathbf{x}$ lie in Ω. We focus on 2D random fields, thus $\Omega = \mathbb{R}^2$. Their realization can produce grayscale ($d = 1$) or color ($d = 3$) textures.

Gaussian processes. A Gaussian process Y is a stochastic process for which every linear combination of $Y(\mathbf{x_1}), ..., Y(\mathbf{x_N})$ such that $(\mathbf{x_1}, ...\mathbf{x_N}) \in \Omega^N$ is a random variable with a normal distribution. The first order moment and second order moment of these processes are necessary and sufficient to define their N-th order statistics for all N, that is, the probability distribution of the entire process.

Stationarity and cyclostationarity. The invariance of the sets of probabilities P under a translation $\mathbf{t} \in \Omega$ is defined by

$$P(Y(\mathbf{x_1}), .. Y(\mathbf{x_N})) = P(Y(\mathbf{x_1}+\mathbf{t}), .. Y(\mathbf{x_N}+\mathbf{t})). \quad (1)$$

If equation (1) is satisfied for any $\mathbf{t}$, then Y is said stationary. Y is said to be a cyclostationary (CS) or periodically correlated process with a period vector $\mathbf{t_0}$ if equation (1) is satisfied for $\mathbf{t} = k\mathbf{t_0}, \forall k \in \mathbb{Z}$, where $\mathbf{t_0}$ represents the period of the CS process, and there are no vectors $\mathbf{t_0'}$ with any smaller coordinate larger than 0 for which this invariance is true [Nap19]. Most of the time throughout this paper, CS processes are assumed to be periodic with not one, but two vectors $\mathbf{t_0}$ and $\mathbf{t_1}$ which form a basis of Ω.

As shown in Figure 3, $\mathbf{t_0}$ and $\mathbf{t_1}$ define a parallelogram Ω^- and a lattice tiling the plane Ω. We denote as $\mathcal{P}$ the operator $\Omega \to \Omega^-$ that projects any $\mathbf{x}$ modulo $(\mathbf{t_0}, \mathbf{t_1})$:

$$\mathbf{x} = \mathcal{P}(\mathbf{x}) + k\mathbf{t_0} + l\mathbf{t_1}, \quad \text{with } \mathcal{P}(\mathbf{x}) \in \Omega^- \text{ and } (k,l) \in \mathbb{Z}^2 \quad (2)$$

A stationary process can be seen as a CS process where $\mathbf{t_0}$ and $\mathbf{t_1}$ are infinitely small, making CS processes a generalization of the stationary model. Likewise, a CS process with only one period vector $\mathbf{t_0}$ can be seen as one with an orthogonal $\mathbf{t_1}$ infinitely small if it is stationary in this direction, or infinitely large if it has arbitrary statistics.

Ergodicity. A stationary process is said to be ergodic when its statistical properties can be estimated on a single, sufficiently large realization, rather than several realizations of the process. This property typically enables an accurate by-example texture synthesis, because the statistics of the process can be estimated from the example. The extension to CS processes is known as cycloergodicity. We assume all the processes to be cycloergodic, and we will make use of this property to estimate the CS statistics.

3.2. Cyclostationary statistics

As explained previously, the statistics of a cyclostationary process Y are periodic. All statistics presented here have an echo in stationary processes, as they only take an additional argument $\mathbf{x} \in \Omega$, due to the spatially-varying nature of CS processes.

The first order moment $M(\mathbf{x}) \triangleq \mathbb{E}[Y(\mathbf{x})]$ is the expectation of the process Y in $\mathbf{x}$. The cyclostationarity implies $M(\mathbf{x}) = M(\mathcal{P}(\mathbf{x}))$. The

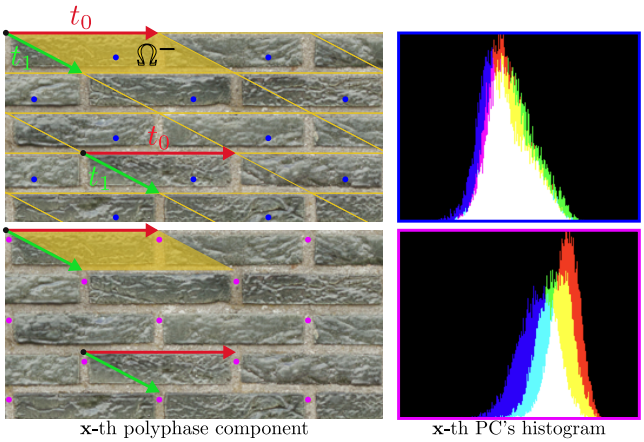

Figure 3: *A cyclostationary signal characterized by two periods* $\mathbf{t_0}$ *and* $\mathbf{t_1}$ *which define a regular lattice (yellow). Left: the signal can be decomposed into polyphase components (collection of blue dots, or magenta dots), which are discrete stationary processes. The* $\mathbf{x}$*-th polyphase component (PC) is made of all values on* $\{\mathbf{x}+k\mathbf{t_0}+l\mathbf{t_1}, (k,l) \in \mathbb{Z}^2\}$ *congruent to* $\mathbf{x}$*. Right: histograms of the PCs are estimators of the statistics of the underlying process (provided it is cycloergodic).*

cycloergodicity implies that it can be estimated from an exemplar E by an empirical mean

$$\widetilde{M}(\mathbf{x}) = \frac{1}{\#samples} \sum_{k,l} E(\mathbf{x}+k\mathbf{t_0}+l\mathbf{t_1}) \tag{3}$$

where $\#samples$ is the number of samples $\mathbf{x}+k\mathbf{t_0}+l\mathbf{t_1}$ in E. This equation is later used for our cyclostationary ADSN model in section 4.3 and can be visualized in the bottom row of figure 6.

The second order central moment is the autocovariance function of Y, defined by $R(\mathbf{x},\mathbf{t}) \triangleq \mathbb{E}[(Y(\mathbf{x})-M(\mathbf{x}))\,(Y(\mathbf{x}+\mathbf{t})-M(\mathbf{x}+\mathbf{t}))]$ for any position $\mathbf{x}$ and any translation $\mathbf{t}$ in Ω. The cyclostationarity implies $R(\mathbf{x},\mathbf{t}) = R(\mathcal{P}(\mathbf{x}),\mathbf{t})$. The cycloergodicity implies that it can be estimated from an exemplar E by

$$\widetilde{R}(\mathbf{x},\mathbf{t}) = \frac{1}{\#samples} \sum_{k,l} \Big(E(\mathbf{y})-\widetilde{M}(\mathbf{x})\Big)\,\Big(E(\mathbf{y}+\mathbf{t})-\widetilde{M}(\mathbf{x}+\mathbf{t})\Big), \tag{4}$$

where $\mathbf{y} = \mathbf{x}+k\mathbf{t_0}+l\mathbf{t_1}$ and $\#samples$ is the number of samples for which both $\mathbf{y}$ and $\mathbf{y}+\mathbf{t}$ fall inside E. It is computed independently for each color channel.

In the stationary case, the autocovariance is often controlled in the spectral domain by its Fourier transform, i.e. the power spectral density (PSD). In the cyclostationary case, the most straightforward formula is the instantaneous power spectral density (IPSD), i.e. the Fourier transform of R with respect to $\mathbf{t}$. However, another, more effective approach is possible using the polyphase components.

Polyphase components. It is possible to decompose a cyclostationary signal into stationary components [Nap19]. One possibility is the translation time series decomposition, or its discrete form, called the polyphase decomposition. For any $\mathbf{x} \in \Omega^-$, the collection of positions $\{\mathbf{x}+k\mathbf{t_0}+l\mathbf{t_1}, (k,l) \in \mathbb{Z}^2\}$ congruent to $\mathbf{x}$ define a discrete sub-process $Y^{\mathbf{x}}$ called the $\mathbf{x}$-th polyphase component of the process Y (also called $\mathbf{x}$-th decimated component), as shown in Figure 3 left. This decomposition is helpful because $Y^{\mathbf{x}}$ are stationary processes: their histogram $H^{\mathbf{x}}$ capture the first order statistics (Figure 3 right), which we may want to control (see section 5.3); their power spectral densities $S^{\mathbf{x}}$ are intuitive local spectra (see sections 3.3 and 4.2), that control the second order moment R [KGE18].

Since our processes are Gaussian, wide-sense and strict-sense cyclostationarity are equivalent [Lap17], which implies that Y is entirely defined by M and R, or by all $S^{\mathbf{x}}$.

3.3. Textures

We model a texture as a function $I : \Omega \to \mathbb{R}^d$, with d the dimension of a random vector returned by noise ($d = 1$ for grayscale, $d = 3$ for color).

Cyclostationary textures. We define cyclostationary textures (CS textures) as realizations of arbitrary cyclostationary processes, implying they have periodic statistics, including but not limited to their mean and autocovariance function; a plausible set of period vectors and lattice by which the CS textures can be generated by a CS process also appear clearly when examining CS textures, as Figure 3 shows. The parameters of the underlying CS process can be estimated as long as the CS texture is sufficiently large, and the CS process is assumed cycloergodic.

Stationary random phase textures. Random phase textures are the result of a random phase noise. They are also commonly called "Gaussian textures", along with the realizations of other types of noises, because they often have statistics similar to those of the realizations of a Gaussian process, although this depends on the spectrum of the noise. They are defined by Galerne et al. [GGM11] as textures having a random phase in the Fourier domain:

$$I(\mathbf{x}) = \sum_{\xi} A(\xi)\cos\left(2\pi\xi\mathbf{x}+\Phi(\xi)\right) \tag{5}$$

for all frequencies ξ in the frequency domain $\hat{\Omega}$, with random phases Φ uniformly distributed in $[0, 2\pi[$. These types of textures are interesting, because they can be compactly modeled in a procedural way using procedural noises.

Cyclostationary random phase textures. Similarly, we define cyclostationary random phase (CSRP) textures as the realizations of CS random phase noise processes, to model cyclostationary textures through a random phase:

$$I(\mathbf{x}) = \sum_{\xi} A(\mathbf{x},\xi)\cos\left(2\pi\xi\mathbf{x}+\Phi(\xi)\right). \tag{6}$$

The novelty is that the amplitude $A : \Omega \times \hat{\Omega} \to \mathbb{R}$ is spatially-varying, i.e. it is a periodic function of $\mathbf{x}$ (periods $\mathbf{t_0}$ and $\mathbf{t_1}$). The phase Φ is random, while A entirely defines the statistics of the CSRP texture. In this scenario, $A(\mathbf{x},\xi)^2$ is equal to the power spectral density $S^{\mathbf{x}}(\xi)$ of the $\mathbf{x}$-th polyphase component.

Although not all CS textures are CSRP textures, this model approximates them well in the same way that random phase textures approximate "stationary" textures (called micro-textures in Galerne et al. [GGM11]), and establishes the possibility of synthesizing textures by defining a spatially-varying amplitude and randomizing their phase. Note that random phase textures and Gaussian textures are very similar but not equivalent [GGM11]. For simplicity, in the CS case, we call them CS noise textures too.

4. Cyclostationary Gaussian noises

In this section, we introduce cyclostationary Gaussian noises. They consist in different formulations of cyclostationary Gaussian processes, giving rise to different synthesis algorithms of cyclostationary noise textures.

4.1. Modulation of a stationary signal

Cyclostationary processes can be expressed as a simple aggregation of stationary components periodically defined on Ω [Nap19]. A simple way to obtain a cyclostationary signal is therefore to modulate a set of jointly stationary signals $\{Y_1, ..., Y_K\}$ as

$$I(\mathbf{x}) = \sum_s w_s(\mathbf{x})\, Y_s(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega, \tag{7}$$

where $w_s : \Omega \to \mathbb{R}$ are $(\mathbf{t_0}, \mathbf{t_1})$-periodic weighting functions.

This approach was used by Guingo et al. [GSDC17] to synthesize heterogeneous textures from an input example, from which they extract a few stationary signals along with weighting functions. It can also be found to a lesser extent in Haindl et al. [HH09], where near-regular textures are considered as a set of cells inside which a single noise is extracted and synthesized. The drawback of this approach is the definition of the weights w_s and the stationary signals Y_s: a manual design is tedious, while an automated extraction is difficult.

4.2. Cyclostationary random phase noise

We have established in equation 6 that cyclostationary random phase textures can be expressed as a sum of cosines with random phases, with statistics being controlled through the spatially-varying amplitude A. This paves the way to adapt existing stationary procedural noises based on the PSD: random phase noise [GGM11, GSV+14], Gabor noise [LLDD09] and phasor noise [TEZ+19]. The main difficulty is the control of the "spatially-varying PSD" $A^2(\mathbf{x}, \bullet) = S^{\mathbf{x}}$.

We validate this idea in Figure 4 by a direct synthesis in the Fourier domain, where A^2 is linearly interpolated from a small set of spectra defining a periodic variation. In Figure 1 left, we adapted the Gabor noise, the local random-phase noise, and the phasor noise. In this example, A^2 is a periodic function, which makes a bi-lobe vary in orientation according to a single period $\mathbf{t_0}$ (red vector). Note that phasor noise is actually not Gaussian; however, it is based on a Gaussian process, so we decided to present it here.

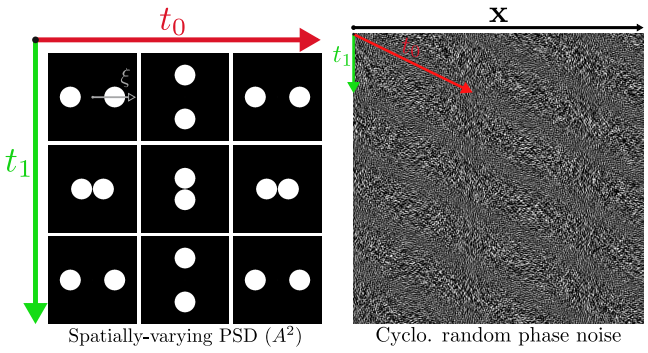

Figure 4: *Cyclostationary random phase noise generated by Fourier series with a spatially-varying amplitude A (computed from an array of PSD) and a random phase. The array was constructed by bilinear interpolation of 9 spectra and defined periodically on the parallelogram described by* $\mathbf{t_0}$ *and* $\mathbf{t_1}$.

4.3. Cyclostationary asymptotic discrete spot noise

The *stationary* asymptotic discrete spot noise (ADSN) [GGM11] is a stationary and ergodic Gaussian process that computes an output I of arbitrary size as the sparse convolution of a discrete spot J:

$$I = \frac{1}{\sqrt{|J|}} (J - \mu \mathbf{1}) * W, \tag{8}$$

where W is the realization of a white noise made of uncorrelated components with distribution $\mathcal{N}(0,1)$ and μ is the estimated mean of the discrete spot J, $\mathbf{1}$ is a constant function equal to 1 with the same support as J, and $|J|$ is the number of texels of the discrete spot.

We introduce a new noise model, the *cyclostationary* asymptotic discrete spot noise (CS-ADSN). It is a cyclostationary and cyclo-ergodic Gaussian process that computes an output texture I from a spot J as

$$I = \frac{1}{\sqrt{|J|}} (J - M) * W_L, \tag{9}$$

where M is the first order moment of the cyclostationary process described in section 3.2, and W_L is a discrete process defined on the lattice whose nodes are congruent to $\mathbf{0}$ with respect to the periods $\mathbf{t_0}$ and $\mathbf{t_1}$. W_L is a sum of diracs (one per node) with uncorrelated random magnitudes distributed according to $\mathcal{N}(0, \sqrt{|\Omega^-|})$. I is a process with first order moment equal to $\mathbf{0}$, requiring the addition of M after the process. The second order moment of I corresponds to the second order moment of the spot, which we prove in appendix A.

This model is close to a specific kind of locally controlled spot noise presented in Pavie et al. [PGDG16] and Cavalier et al. [CGG19], where the results can be made regular by controlling the position where the spots are shot: we conjecture that our CS-ADSN is the underlying cyclostationary model for the perfectly regular version of the controlled spot noise they present without consideration for the cyclostationary mean and the variance of the output, which enables us to go one step further and to achieve a by-example synthesis for cyclostationary textures in the next section.

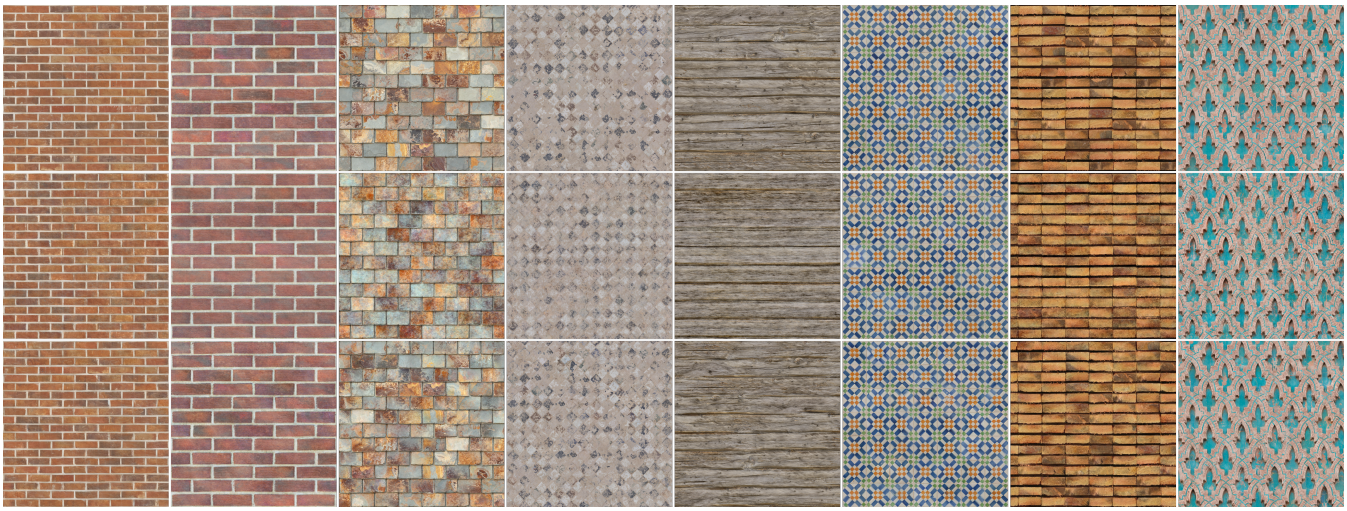

Figure 5: *Exemplars (top row) synthesized with our cyclostationary spot noise (middle) and our cyclostationary high-performance noise (bottom). Our first order statistics transfer (section 5.3) was applied to all results. All textures are unbounded and continuous, and can be sampled in real time at any location of infinite space. Here, we show a sampling matching the resolution of the input.*

5. By-example synthesis

The parameters of the noise models can be difficult to define manually. Instead, noise by-example consists in estimating the parameters from an input example image E. In our context, the idea is to find an underlying process Y which could have been likely to generate the example.

The cyclostationary random phase noise of equation 6 is unfortunately not a convenient model for by-example synthesis, because it requires an accurate estimation of the PSD $S^{\mathbf{x}}$ of polyphase components; however, we dispose of only a few sparse samples in E for each polyphase component. Because of this, we choose to elaborate on the CS-ADSN model presented in section 4.3. We applied it to two procedural noises using an exemplar as a discrete spot: spot noise (section 5.1), and high-performance noise (section 5.2), with results of these algorithms shown in Figure 5.

In this section, we assume that $\mathbf{t_0}$ and $\mathbf{t_1}$ are already estimated from an exemplar; we show how to do this in section 6.

5.1. Cyclostationary spot noise

The by-example *stationary* spot noise algorithm [GGM11] is a direct implementation of the ADSN model of equation (8) that consists in shooting E (the spot $J = E$) on random positions. The random positions are determined by a sparse Poisson process [vW91] instead of a white noise W, which increases the space coverage. As shown in Figure 6 top, it is unable to reproduce the pattern regularity.

We define the *cyclostationary* spot noise from equation 9, by shooting E only on the regular lattice described by $\mathbf{t_0}$ and $\mathbf{t_1}$, as shown in Figure 6 bottom. Since W_L is sparse, it can be faster to compute the cyclostationary spot noise directly instead of simulating it by shooting E several times. For this process, the first order moment M needs to be estimated from the exemplar, which can be done with Equation 3. The asymptotic result of this by-example noise takes the form of a Gaussian cyclostationary process with the statistics of the CS-ADSN of E.

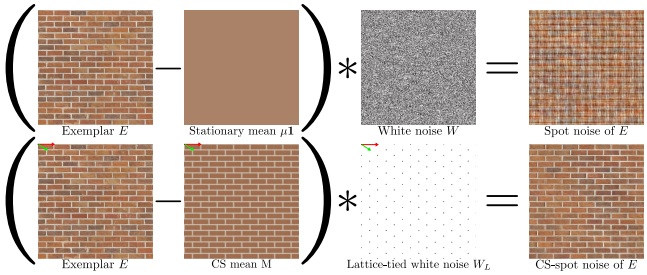

Figure 6: *Comparison between the stationary (top) and our cyclostationary (bottom) spot noise, using an exemplar texture as the spot. The estimation of the lattice and the mean M (Equation 3) are the critical steps for preserving the statistics of the exemplar. The addition of μ and M required after equations (8) and (9) is omitted.*

5.2. Cyclostationary high-performance noise

The *stationary* high-performance noise algorithm [HN18, DH18, Bur19] is a real-time by-example synthesis designed as a compromise between a tiling algorithm and a noise algorithm. It consists in blending hexagon-shaped spots centered on the vertices of a triangular grid. For each vertex $\mathbf{x}$ of the grid, an hexagonal tile is taken from the exemplar E at a random position given by a hash function $\mathbf{h}(\mathbf{x})$. The overlapping tiles are linearly blended within each triangle. As shown in Figure 7 left, it fails to reproduce CS textures. The reason is that $\mathbf{x}$ and $\mathbf{h}(\mathbf{x})$ are not congruent.

We define a *cyclostationary* variant of the high-performance noise by controlling the positions of the tiles. The hash function

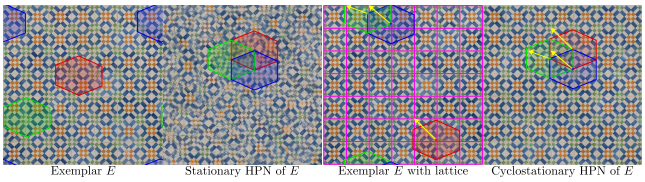

Figure 7: *High-performance noise: stationary (left) and cyclostationary (right) version. For the latter, the positions of the hexagonal tiles in the example ($\mathbf{h}(\mathbf{x})$) and in the result ($\mathbf{x}$) are congruent.*

$\mathbf{h}(\mathbf{x})$ returns a random position congruent to $\mathbf{x}$, i.e. $\mathcal{P}(\mathbf{h}(\mathbf{x})) = \mathcal{P}(\mathbf{x})$ (see yellow vectors in Figure 7). Additionally, it is compatible with the enhancement proposed by Burley [Bur19] that modifies the blending weights.

5.3. First order statistics transfer

The previous algorithms are designed for textures with cyclostationary Gaussian statistics. However, many natural stochastic textures do not have Gaussian statistics. Therefore, many works tried to push the algorithms beyond the pure Gaussian scope of application. Among them, optimal transport has been used successfully to control the first order statistics [GLR17, HN18, DH18], namely to process textures with non Gaussian histograms.

In the following, we briefly recall stationary histogram transfer, show its limitations, propose a CS histogram transfer, discuss robustness issues and filtering.

5.3.1. Stationary histogram transfer

Let E_H be an exemplar with non Gaussian histogram H. The principle is as follows:

- Pre-compute a transfer function T that maps H onto a Gaussian histogram G.
- Pre-compute T^{-1} in a lookup table.
- Pre-compute $E_G = T(E_H)$, a "Gaussianized" example.
- At runtime, compute the result $T^{-1}(Synthesis(E_G))$ by applying the synthesis algorithm to E_G and compositing by T^{-1}.

The results has H as non-Gaussian histogram, just as the example E_H. When applied to a CS texture, this procedure produces artifacts, as shown in Figure 8. These artifacts are due to the global nature of the histogram, which does not capture spatial variations.

5.3.2. Cyclostationary histogram transfer

As explained in section 3, a CS process can be decomposed into a set of polyphase components (PC) which are stationary processes. In this section, the intuition is to preserve the histogram of each PC independently.

Let E be a CS texture we want to synthesize, which we assume to be the result of an underlying CS process. Let $E^{\mathbf{x}}$ be the *polyphase components estimates*: for any $\mathbf{x} \in \Omega^-$, it is an estimation, from the example E, of the $\mathbf{x}$-th PC of the underlying process. In concrete terms, $E^{\mathbf{x}}$ is a subsampling of E with $|E|/|\Omega^-|$ samples. Let $H^{\mathbf{x}}$ be the histogram of $E^{\mathbf{x}}$ (Figure 3), which is an estimation of

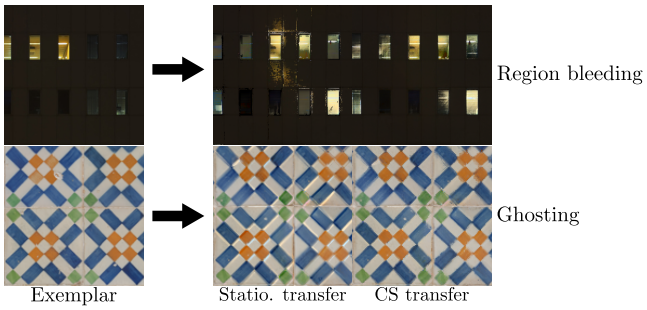

Figure 8: *Histogram transfer: comparison between stationary (left) and cyclostationary (right) transfer with the HPN synthesis algorithm [HN18]. Our CS transfer helps reducing color artifacts.*

the first order statistics of the PC. The collection $\{H^{\mathbf{x}}\}_{\mathbf{x}\in\Omega^-}$ is an estimation of the first order statistics of the CS process.

Our CS histogram transfer operates per PC similarly to the section 5.3.1:

- Pre-compute, for each $H^{\mathbf{x}}$, a transfer function $T_{\mathbf{x}}$ and its inverse $T_{\mathbf{x}}^{-1}$.
- Pre-compute $E_G(\mathbf{x}) = T_{\mathcal{P}(\mathbf{x})}(E(\mathbf{x}))$, because each $\mathbf{x}$ belongs to the $\mathcal{P}(\mathbf{x})$-th PC.
- At runtime, compute the result $T^{-1}_{\mathcal{P}(\mathbf{x})}(Synthesis(E_G))$.

As shown in Figure 8, this cyclostationary transfer helps against false colors. It notably removes region bleeding (when the output color is incorrectly interpreted as that of a different region of the exemplar), and reduces ghosting when attempting to synthesize textures with slight geometric irregularities. The inverse transfer functions $T^{-1}_{\mathcal{P}(\mathbf{x})}$ are stored for all texels $\mathbf{x} \in \Omega^-$. In practice, the memory footprint is at most equivalent to the footprint of the exemplar itself.

5.3.3. Robust histogram estimation

The CS histogram transfer (previous section) relies on the estimations $H^{\mathbf{x}}$. However, for small examples and large periods, $|E^{\mathbf{x}}|$ becomes small, causing issues in the histogram estimation and artifacts in the result (see Figure 9). To make our estimation more robust, we assume a local stationarity: in a small neighborhood of size δ around $\mathbf{x}$ we assume the PC to be similar. Practically, we add into $E^{\mathbf{x}}$ the samples of the neighboring PC estimates: $\delta = 1$ enlarge the sampling by a factor 9, $\delta = 2$ by a factor 25, etc. As

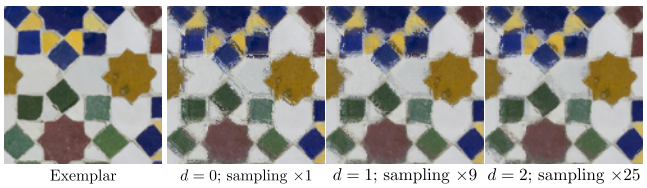

Figure 9: *Robust histogram estimation, by collecting the polyphase component estimates in a small neighborhood of size δ. Extending δ trades noisy for blurry artifacts.*

shown in Figure 9, extending the neighborhood yields a compromise between the stationary (wrong colors, visible blur) and cyclostationary (better colors, visible noise) transfer. In the limit (as the neighborhood covers the example), all estimates are equal and it boils down to a stationary transfer.

5.4. Real-time rendering and filtering

The synthesis algorithms presented in this paper are tailored for real-time synthesis and rendering, while keeping a low memory load. It requires to filter the resulting texture in real-time. Both the spot noise [vW91] and the high-performance noise [HN18] can be filtered using a pre-computed MIP-map of the example E. This technique is still valid with our cyclostationary model. We directly applied the histogram transfer on the result, as Heitz & Neyret [HN18]. Deliot et al. [DH18] improve the results by computing one pre-filtered lookup table per MIP level: it can be adapted with one lookup table per level per polyphase component, however its practical implementation remains to be investigated.

6. Estimation of period vectors

In this section, we investigate the estimation of the period vectors $(\mathbf{t_0}, \mathbf{t_1})$ from an exemplar E. As stated in the related works section, many methods have been proposed to estimate manually, semi-automatically or completely automatically repetitive structures in patterns. These approaches are useful, but sometimes lack accuracy when combined with our by-example synthesis algorithms, which require a sub-pixel precision. When using them, polyphase components may be incorrectly estimated and cause ghosting artifacts similar to those observed in Figure 8. Furthermore, automatic methods tend to be designed for harder problems and rely on complex software such as neural networks, which are heavy for the task at hand. We propose to use simple techniques such as a manual estimation, which is then refined by exploiting the characteristics of cyclostationary textures.

Our refinement algorithm is based on the observation that $(\mathbf{t_0}, \mathbf{t_1})$ represent the translations to the next element in the repeated pattern, which corresponds to another sample in the same polyphase components. We expect the PC to be strongly coherent, i.e. that any two points $\mathbf{y}$ and $\mathbf{z}$ in a given PC $E^{\mathbf{x}}$ are similar. Thus, we define

$$d_{PC}(\mathbf{t_0}, \mathbf{t_1}, E^{\mathbf{x}}) = \frac{1}{|E^{\mathbf{x}}|^2} \sum_{\mathbf{y} \in E^{\mathbf{x}}} \sum_{\mathbf{z} \in E^{\mathbf{x}}} \sqrt{(E(\mathbf{y}) - E(\mathbf{z}))^2}, \tag{10}$$

which is a measure of the internal coherence of $E^{\mathbf{x}}$. Then we measure the global relevance of the vectors $(\mathbf{t_0}, \mathbf{t_1})$ by

$$d(\mathbf{t_0}, \mathbf{t_1}) = \int_{\mathbf{x} \in \Omega^-} d_{PC}(\mathbf{t_0}, \mathbf{t_1}, E^{\mathbf{x}}) \, \mathrm{d}\mathbf{x}. \tag{11}$$

To refine the initial estimation $(\mathbf{t_0}, \mathbf{t_1})$, we locally minimize d using a stochastic gradient descent. A visualization of d is shown in Figure 10, where an optimal point is found, even on the third exemplar exhibiting geometric irregularity. Note that d is not convex; however, the procedure robustly converges to the local minimum, as long as the the initial estimation is precise enough.

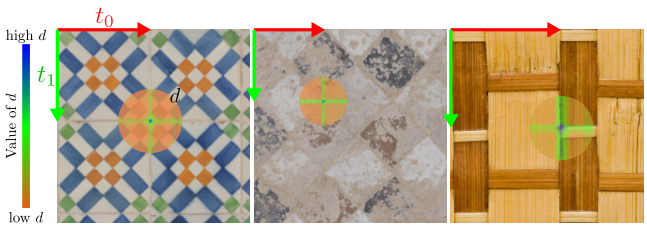

Figure 10: *Distance function d applied around the local minimum of d (corresponding to ground truth period vectors of $\mathbf{t_0}$ and $\mathbf{t_1}$) on various textures. d was only evaluated with orthogonal period vectors to produce this visualization.*

7. Results and discussion

Implementation When a discrete exemplar needs to be sampled at any continuous location (e.g. for the CS-spot noise), it is bilinearly-interpolated between the pixels. The CS-spot noise is implemented with the enhancement of Galerne et al. [GGM11] that consists in smoothing the border of the exemplar to remove linear artifacts. As in the original stationary version [HN18], the CS-HPN is easier to implement with a periodic examplar, otherwise hexagons that cross the border must be avoided. The performances of the stationary and cyclostationary algorithms are similar, so we invite the reader to refer to the original papers to get detailed information about timings. Indeed, our algorithms inherit the pros and cons of the original algorithms. Our codes are publicly available in the ASTex library [ast17] and on shadertoy.

Synthesis of CS random-phase textures We have extended to the CS context several noises from the sate of the art: Figure 4 is a direct implementation of random phase noise ; Gabor noise, local random-phase noise, and phasor noise are shown in Figure 1 (left). Some additional high resolution results are available in our supplemental material.

We designed the spatially-varying PSD using functions or linear interpolation between discrete PSD. Other interpolation operators could be used, such as b-spline interpolation or Wasserstein interpolation [BPC16]. An estimation of this spectrum from an exemplar would be a powerful tool. This is however a difficult task, as samples of polyphase components estimates are sparse. Moreover, Guingo et al. [GSDC17] show that the extraction of spatial variations of the spectrum is difficult.

By-example synthesis We adapted spot noise and high-performance noise. Many results on various patterns are shown in Figure 5. The textures are unbounded and continuous: they can be sampled in real-time at any location of an infinite plane. A histogram transfer extends the application to examples with non Gaussian histograms, and helps reducing artifacts (Figure 8).

We use Figure 11 to further investigate the comparison between these two algorithms: row 1 shows the exemplar, row 2 shows our CS-spot noise with no histogram transfer (to show what unaltered Gaussian cyclostationary results look like), and row 3 shows our CS high-performance noise with histogram transfer. The major strength of the CS-spot noise is to create variety. For instance,

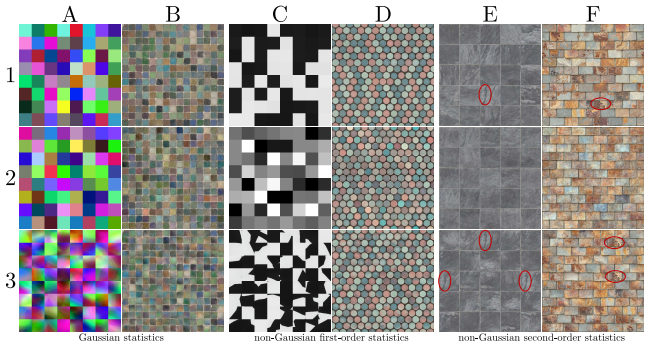

Figure 11: *Exemplars (row 1) synthesized with a CS-spot noise and no histogram transfer (row 2), and with a CS-HPN with cyclostationary histogram transfer (row 3). The Gaussian nature of our CS-spot noise makes it work especially well with exemplars having cyclostationary Gaussian statistics (A,B). Our CS-HPN instead better preserves complex, non-Gaussian structures (E,F), but can produce repetitions (circled in red) and new aberrant tiles (A,B,C).*

columns A and B exhibit colored tiles, for which the colors are close to the result of a normal variable: CS-spot noise (row 2) implicitly estimates the Gaussian CS process that could have generated this tiling and creates new colors that match that process. This result is only possible when the distribution of the colors of each polyphase component corresponds to the result of a Gaussian process, otherwise it creates false colors (columns C and D).

Real-life CS textures corresponding to the realization of a pure Gaussian CS process are pretty rare. The global appearance and periodicity of other CS textures is well kept by the CS-spot noise, but the fine details and complex structures of these textures can be lost by the process. As can be seen in columns E and F, our CS high-performance noise (row 3) better reproduces such features. This is due to the underlying tiling and blending algorithm, which, in counterpart, may create slight visual repetitions (red circles). On some other examples such as colored tiles (columns A, B, C and D), unwanted patterns are created by the linear blending.

To summarize, our CS-spot noise should be used when variety is especially important; our CS-HPN should instead by used when fine details are more important; and neither process is capable of preserving the appearance of CS textures for which the distribution of the colors on each polyphase component is not a Gaussian histogram.

Our cyclostationary histogram transfer may be used to correct the colors, as it is able to correct the first order statistics of the exemplar. However, it does not act on higher-order statistics, and neighboring transfers are independent, causing potentially unexpected color differences between neighbors, such as in column D of Figure 11.

Among other existing by-example noise algorithms, it would be worth investigating texton noise [GLM17]. However, a robust estimation of the cyclostationary texton is not trivial.

Comparison with near-regular textures Cyclostationary textures represent a subset of near-regular textures for which there are

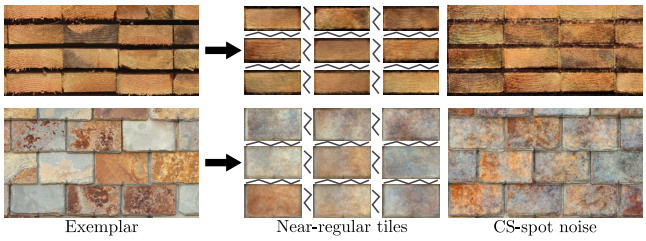

Figure 12: *Comparison with the tile synthesizer of Liu et al. [LLH04]. Tiles produced with their PCA technique (middle) show a loss of contrast and detail, and further require a restitching of tiles to produce output textures. Our real-time CS spot noise (right) avoids these drawbacks.*

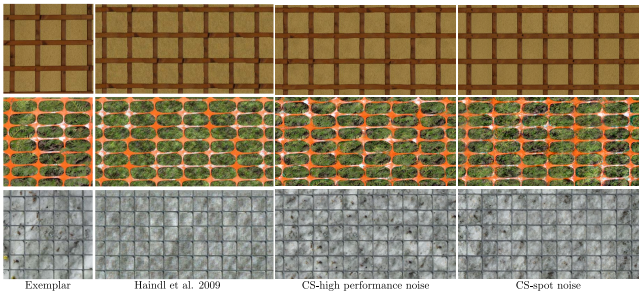

Figure 13: *Comparison with the offline synthesis algorithm of Haindl et al. [HH09] which reproduces a sub-class of CS textures. Our CS spot and CS high-performance noises are real-time, and better preserve finely structured and contrasted random patterns.*

no, or very little, geometric alterations, but important color alterations. Liu et al [LLH04] synthesize color transformations using principal component analysis (PCA). In our experiments, we observed that PCA often induces important losses of contrast and a higher loss of details compared to our CS-spot noise, as shown in Figure 12. Because they separates individual tiles, the output tiles need to be stitched together, making real-time rendering harder to obtain, unless a lower quality blending of tiles is used. Haindl et al. [HH09] propose another offline synthesis algorithm for a subclass of CS textures, which are characterized by a stationary noise within a periodic structure. We show in Figure 13 that CS-spot noise better preserves finely structured and contrasted details. CS-HPN even better preserves complex details.

Comparison with optimization and CNN techniques Until now, optimization and CNN techniques are computationally expensive and rarely suited for the rendering of unbounded textures, whereas speed and unbounded size are two core properties of our methods.

We show in Figure 14 a side by side comparison with "Deep correlations" [SCO17], which attempts to recover automatically repetitive structures in exemplars. Our methods better preserve fine details. More comparisons are given in the supplementary materials. Our model has the advantage of offering some guarantees: exemplars that fall in the class of cyclostationary noise textures are faithfully synthesized, and geometric regularity is perfectly reproduced.

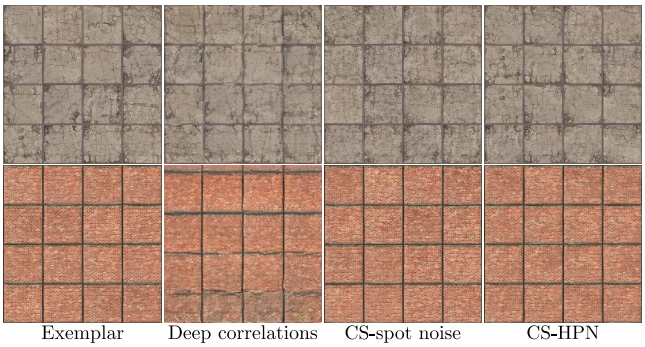

Figure 14: *Comparison with the CNN synthesizer of Sendik et al. [SCO17]. This synthesis technique is not specifically addressing cyclostationary textures, and sometimes fails to capture the periodicity of statistics. Additionally, it is offline and works for small bounded textures: the opposite of our approach.*

We believe CNN techniques could be helpful for detecting optimal period vectors according to more complex criteria, such as visual criteria, and thus replace the method described in section 6, or they could possibly provide a more robust way of estimating cyclostationary histograms.

Geometric variations As of now, geometric variations are not handled with our approach. Liu et al [LLH04] apply both geometric and color transformations, which we could a priori also do. Spatial warping of grids and barycentric coordinates are used to apply geometric alterations. However, we experienced that this works only well if colors have low contrasts and low frequencies. When colors are characterized by high frequencies and strong contrasts, spatial warping generates visual artifacts such as overstretching of details, unnatural distorted patterns and sampling artifacts. Devising an elaborated cyclostationary–like process, which considers both alterations simultaneously, seems like a difficult task that could be an interesting topic for future research.

8. Conclusion

In this paper, we introduced cyclostationary Gaussian noise, a new type of noise able to synthesize cyclostationary noise textures, characterized by the periodicity of their statistics. As a generalization of stationary noise, it broadens the range of patterns that noise is able to address. Thereby, a key application is the more efficient management of stochastic color transformations in near-regular textures. We defined a frame for our models by presenting cyclostationary processes and noises, their statistics and estimators, and cyclostationary textures. We provided a model to synthesize a cyclostationary texture by modulating a stationary process and by designing a spatially-varying power spectral density; we also provided a cyclostationary version of the ADSN model. This allowed us to generalize five procedural noises to the cyclostationary case: local random phase noise, Gabor noise, phasor noise, spot noise and high-performance noise. We showed how to execute a first order statistics correction on outputs generated with by-example synthesis in a cyclostationary context to extend the scope of Gaussian cyclostationary noise to a wider class of cyclostationary textures. We showed how to estimate the periods of an exemplar, required for by-example synthesis. Finally, we compared our synthesis to state of the art methods, and discussed limits and future works.

Acknowledgments

This work has been partially funded by the project HDWorlds from the Agence Nationale de la Recherche (ANR-16-CE33-0001). We thank Xavier Chermain for his help with shadertoy. All textures are provided by https://textures.com, except for those of Figure 13 which are taken from Haindl et al. [HH09].

References

[ast17] ASTex: an open-source library for texture analysis and synthesis. https://github.com/ASTex-ICube/ASTex, 2017. 8

[Ben58] W. R. Bennet. Statistics of regenerative digital transmission. *Bell. Syst. Tech. J.*, 37:1501–1542, 1958. 3

[BPC16] Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: histogram regression using optimal transport. *ACM Trans. Graph.*, 35(4):71–1, 2016. 8

[Bur19] Brent Burley. On histogram-preserving blending for randomized texture tiling. *Journal of Computer Graphics Techniques (JCGT)*, 8(4):31–53, November 2019. 2, 6, 7

[CB11] Y. Cai and G. Baciu. Detection of repetitive patterns in near regular texture images. In *2011 IEEE 10th IVMSP Workshop: Perception and Visual Signal Analysis*, pages 60–65, 2011. 2

[CB13] Y. Cai and G. Baciu. Detecting, grouping, and structure inference for invariant repetitive patterns in images. *IEEE Transactions on Image Processing*, 22(6):2343–2355, 2013. 2

[CGG19] Arthur Cavalier, Guillaume Gilet, and Djamchid Ghazanfarpour. Local spot noise for procedural surface details synthesis. *Computers & Graphics*, 85:92 – 99, 2019. 2, 3, 5

[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

[DED05] Khalid Djado, Richard Egli, and François Deschênes. Extraction of a representative tile from a near-periodic texture. In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '05, page 331–337, New York, NY, USA, 2005. Association for Computing Machinery. 2

[DH18] Thomas Deliot and Eric Heitz. Procedural stochastic textures by tiling and blending. *GPU Zen 2: Advanced Rendering Techniques*, 2018. 2, 6, 7, 8

[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. 2

[GEB15] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28:262–270, 2015. 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, 3, 4, 5, 6, 8, 11

[GLLD12] Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. Gabor noise by example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, 31(4):73:1–73:9, July 2012. 2

[GLM17] B. Galerne, A. Leclaire, and L. Moisan. Texton noise. *Computer Graphics Forum*, 36(8):205–218, 2017. 2, 3, 9

[GLR17] Bruno Galerne, Arthur Leclaire, and Julien Rabin. Semi-discrete optimal transport in patch space for enriching gaussian textures. In *Geometric Science of Information*, volume 10589 of *Lecture Notes in Computer Science*, Paris, France, November 2017. 7

[GRGH20] Jorge Gutierrez, Julien Rabin, Bruno Galerne, and Thomas Hurtut. On demand solid texture synthesis using deep 3d networks. In *Computer Graphics Forum*, volume 39, pages 511–530. Wiley Online Library, 2020. 2

[GSDC17] Geoffrey Guingo, Basile Sauvage, Jean-Michel Dischler, and Marie-Paule Cani. Bi-layer textures: a model for synthesis and deformation of composite textures. *Computer Graphics Forum*, 36(4):111–122, 2017. 2, 3, 5, 8

[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. 1, 2, 3, 5

[HH09] Michal Haindl and Martin Hatka. Near-regular texture synthesis. In Xiaoyi Jiang and Nicolai Petkov, editors, *Computer Analysis of Images and Patterns*, pages 1138–1145, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. 2, 5, 9, 10

[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, 6, 7, 8

[KGE18] A. Kipnis, A. J. Goldsmith, and Y. C. Eldar. The distortion rate function of cyclostationary gaussian processes. *IEEE Transactions on Information Theory*, 64(5):3810–3824, 2018. 3, 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. 2

[Lap17] Amos Lapidoth. *A foundation in digital communication*. Cambridge University Press, 2017. 4

[Lew84] John-Peter Lewis. Texture synthesis for digital painting. *SIGGRAPH Comput. Graph.*, 18(3):245–252, January 1984. 2

[LGX16] Gang Liu, Yann Gousseau, and Gui-Song Xia. Texture synthesis through convolutional neural networks and spectrum constraints. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3234–3239. IEEE, 2016. 2

[LH05] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, page 777–786, New York, NY, USA, 2005. Association for Computing Machinery. 2

[LHW+04] Wen-Chieh Lin, James H. Hays, Chenyu Wu, Vivek Kwatra, and Yanxi Liu. A comparison study of four texture synthesis algorithms on regular and near-regular textures. Technical report, Robotics Institute, Carnegie Mellon University, 2004. 2

[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. 3

[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, 5

[LLH04] Y. Liu, W-C. Lin, and J. Hays. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 2004. 2, 9, 10

[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. 2

[LNS+15] Siying Liu, Tian-Tsong Ng, Kalyan Sunkavalli, Minh N. Do, Eli Shechtman, and Nathan Carr. Patchmatch-based automatic lattice detection for near-regular textures. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[LPVV17] L. Lettry, M. Perdoch, K. Vanhoey, and L. Van Gool. Repeated pattern detection using cnn activations. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 47–55, 2017. 2

[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. 2

[Nap19] Antonio Napolitano. *Cyclostationary processes and time series: theory, applications, and generalizations*. Academic Press, 2019. 3, 4, 5

[NMMK05] Andre Nicoll, Jan Meseth, Gero Müller, and Reinhard Klein. Fractional fourier texture masks: Guiding near-regular texture synthesis. *Computer Graphics Forum*, 24:569 – 579, 10 2005. 2

[PBCL09] Minwoo Park, Kyle Brocklehurst, Robert T Collins, and Yanxi Liu. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1804–1816, 2009. 2

[PGDG16] Nicolas Pavie, Guillaume Gilet, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Procedural texture synthesis by locally controlled spot noise. *Computer Science Research Notes*, 2601:71–79, 2016. 2, 3, 5

[RHE11] Diego Lopez Recas, Anna Hilsmann, and Peter Eisert. Near-Regular Texture Synthesis by Random Sampling and Gap Filling. In Peter Eisert, Joachim Hornegger, and Konrad Polthier, editors, *Vision, Modeling, and Visualization (2011)*. The Eurographics Association, 2011. 2

[SCO17] Omry Sendik and Daniel Cohen-Or. Deep correlations for texture synthesis. *ACM Trans. Graph.*, 36(4), July 2017. 2, 9, 10

[Sta97] Jos Stam. Aperiodic texture mapping. 08 1997. 2

[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. 1, 2, 3, 5

[TNVT19] Vincent Tavernier, Fabrice Neyret, Romain Vergne, and Joëlle Thollot. Making gabor noise fast and normalized. In The Eurographics Association, editor, *Eurographics 2019 - 40th Annual Conference of the European Association for Computer Graphics*, Eurographics 2019 - Short Papers, pages 37–40, Genoa, Italy, May 2019. 2

[ULVL16] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016. 2

[vW91] Jarke J. van Wijk. Spot noise texture synthesis for data visualization. *SIGGRAPH Comput. Graph.*, 25(4):309–318, 1991. 2, 6, 8

[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. 2

[WHC+06] Wen-Chieh Lin, J. Hays, Chenyu Wu, Yanxi Liu, and V. Kwatra. Quantitative evaluation of near regular texture synthesis algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 427–434, 2006. 2

[WY04] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Trans. Graph.*, 23(3):364–367, August 2004. 2

Appendix A: Autocovariance of the cyclostationary ADSN

We prove the claim of section 4.3 that the second order moments of the spot J and its CS-ADSN I are identical. We prove it similarly to Galerne et al. [GGM11] for the stationary case. To enable an accurate estimation of the autocovariance on a spot, we consider that J is periodic. Let us denote by L the lattice described by the

vectors $\mathbf{t_0}$ and $\mathbf{t_1}$ as in Figure 3. We call $|L|$ the number of nodes of the lattice on the spot J.

The autocovariance of the spot J (which is the autocovariance of section 3.2 with the mean subtracted to each term) can thus be estimated by

$$\widetilde{R}_J(\mathbf{x},\mathbf{t}) = \frac{1}{|L|}\sum_{\mathbf{u}\in L}\left(J(\mathbf{x}-\mathbf{u})-M(\mathbf{x})\right)\left(J(\mathbf{x}-\mathbf{u}+\mathbf{t})-M(\mathbf{x}+\mathbf{t})\right) \tag{12}$$

since $M(\mathbf{x}) = M(\mathbf{x}-\mathbf{u})$ for all $\mathbf{x}\in\Omega$, $\mathbf{u}\in L$. Let us define $\dot{J} = \frac{1}{\sqrt{|J|}}(J-M)$. The autocovariance function of I is

$$\begin{aligned} R_I(\mathbf{x},\mathbf{t}) &= \mathbb{E}[\sum_{\mathbf{u}\in L}\dot{J}(\mathbf{x}-\mathbf{u})W_L(\mathbf{u})\sum_{\mathbf{v}\in L}\dot{J}(\mathbf{x}+\mathbf{t}-\mathbf{v})W_L(\mathbf{v})] \\ &= |\Omega^-|\sum_{\mathbf{u}\in L}\dot{J}(\mathbf{x}-\mathbf{u})\dot{J}(\mathbf{x}+\mathbf{t}-\mathbf{u}) \\ &= \widetilde{R}_J(\mathbf{x},\mathbf{t}) \end{aligned} \tag{13}$$

since $\frac{|\Omega^-|}{|J|} = \frac{1}{|L|}$, and $\mathbb{E}[W_L(\mathbf{u})W_L(\mathbf{v})]$ is equal to 0 when $\mathbf{u}\neq\mathbf{v}$, $\mathbf{u}\notin L$ or $\mathbf{v}\notin L$, and $\sqrt{|\Omega^-|}$ otherwise. I is therefore a centered cyclostationary Gaussian process with second order moment $\widetilde{R}_J(\mathbf{x},\mathbf{t})$.