Local Random-Phase Noise for Procedural Texturing

Guillaume Gilet*, Basile Sauvage†, Kenneth Vanhoey†, Jean-Michel Dischler†, Djamchid Ghazanfarpour*
* XLIM, Université de Limoges, CNRS, France
† ICube, Université de Strasbourg, CNRS, France

(a) Noise based on a continuous target spectrum.

(b) Structure-preserving procedural texture deduced from an example.

Figure 1: *Local random-phase noise can approximate an arbitrary power spectral density. It provides high-speed procedural reproduction of Gaussian patterns defined by a continuous spectrum (a) or by an input example. A broader range of procedural textures by example can be generated by preserving input structures such as skin wrinkles in (b).*

Abstract

Local random-phase noise is a noise model for procedural texturing. It is defined on a regular spatial grid by local noises, which are sums of cosines with random phase. Our model is versatile thanks to separate sampling in the spatial and spectral domains. Therefore, it encompasses Gabor noise and noise by Fourier series. A stratified spectral sampling allows for a faithful yet compact and efficient reproduction of an arbitrary power spectrum. Noise by example is therefore obtained faster than state-of-the-art techniques. As a second contribution we address texture by example and generate not only Gaussian patterns but also structured features present in the input. This is achieved by fixing the phase on some part of the spectrum. Generated textures are continuous and non-repetitive. Results show unprecedented framerates and a flexible visual result: users can control with one parameter the blending between noise by example and structured texture synthesis.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: procedural texturing, noise synthesis, by-example texturing

Links: DL PDF WEB VIDEO

*e-mail: {Guillaume.Gilet|Ghazanfarpour}@unilim.fr
†e-mail: {Sauvage|Kenneth.Vanhoey|Dischler}@unistra.fr

1 Introduction

Procedural textures based on noise, like Perlin's famous marble [1985], have been introduced in the mid-eighties [Ebert et al. 2002]. They unify a number of properties that still no other texturing technique is able to unify, four of the most compelling ones being:

1. not any kind of repetition, due to the random nature of noise;
2. continuity (i.e. resolution independence);
3. computation on a per-pixel basis, thus allowing for straightforward parallel implementations;
4. genericity (*i.e.* a single texture model allows users to generate visual variants by tuning parameters).

These properties have probably contributed to the widespread use of this kind of textures for a broad range of computer graphics applications like simulators and computer generated images in animated movies. Noise is however hard to control. Subsequently, creating procedural textures based on noise is a challenging and persistent problem, that recent noise "by-example" approaches [Galerne et al. 2012; Gilet et al. 2012] attempt to tackle. Control is improved by generating noise with an arbitrary power spectral density (PSD). But strong limitations remain. First, latest approaches exclusively build upon the power spectrum. Corresponding patterns are *Gaussian*, which represents a narrow subset of irregular textures. Second, these methods use a high quality but computationally expensive sparse convolution, involving a large number of band-pass filter kernels. Despite parallel GPU implementations, high noise evaluation costs are reported [Galerne et al. 2012; Gilet et al. 2012], thus either limiting quality or framerates.

This paper aims at getting over these two limitations by introducing a novel procedural noise. It is defined as a blending of local noises centered on a regular spatial grid, each of them being a sum of cosines with random phase. Therefore, we call it local random-phase (LRP) noise. Separate sampling in the spatial and spectral domains provide versatility: our model can simulate Gabor noise by sparse convolution as well as noise by Fourier series. As for Galerne et al. [2012] and Gilet et al. [2012], the power spectrum can be controlled using a discrete example. A stratified sampling in the spectral domain provides efficient generation of stochastic func-

tions with different looks while controlling the PSD. The stratification is pre-processed while random sampling and noise generation is processed at runtime. By improving the trade-off between spectral accuracy and spatial sampling, it is faster than sparse convolution and filtering, and thus better adapted to real-time procedural texturing. We provide parameter estimates such that the process can be controlled by only one parameter: the budget of cosines, which defines the computation load.

In a second step, we address the automatic computation of procedural textures from examples containing not only random patterns but also structure. It is well known that structure can be found in phase [Oppenheim and Lim 1981]. Assuming that the region of the spectrum encoding structure is known, our model naturally allows for preserving the input phase in this spectral region only. We thus reproduce the input structure while still having a random look. We demonstrate this by selecting the region with highest energies, which is known to contain structure information for near regular textures [Nicoll et al. 2005]. When applying this idea to irregular or stochastic textures it creates a periodic structure which induces an unnatural regularity. Periodicity is broken by using turbulence [Perlin 1985] and random placement. Their parameters are obtained from the example spectrum itself, therefore keeping unspoiled the global spectral energy distribution of the example. The balance between structure (fixed phase) and noise (random phase) is easy to tune with a single parameter. Resulting textures preserve main visual characteristics and implicitly fulfill the four aforementioned properties. An example of structured irregular procedural texture from an input example is shown in figure 1(b).

The remaining parts of the paper are organized as follows: in the next section we propose a brief overview of works related to noise. Then, we describe our noise model using random phase functions (section 3) and how such a noise can be given an arbitrary PSD so that noise by example is made possible (section 4). Next, we describe how to synthesize procedural textures from example images that also contain structured features (section 5). Finally, before concluding, we show results and discuss limitations.

2 Related works

Procedural noise. Mainly two types of techniques have been proposed to compute "infinite" procedural noise in computer graphics (see survey of Lagae et al. [2010a]): 1) lattice-based approaches [Perlin 1985] and 2) sparse convolution [Lewis 1987] using spatial filter functions and uniform random point distributions (impulses) computed on-the-fly using pseudo random number generation (PRNG). Sparse convolution permits a direct spectral control because the power spectrum of the resulting noise is linked to the filter kernel function. The Gabor kernel is particularly well adapted [Lagae et al. 2009] as it unifies spatial and spectral characteristics. Lagae and Drettakis [2011] further introduce the use of a phase to process aliasing raised by solid noise. However, convolution noise demands high computational resources, especially when the power spectrum is complex. Many impulses become necessary to guarantee both good spatial and frequency coverage. Faster noises, like anisotropic noise [Goldberg et al. 2008], filter white noise images using an inverse FT. Such methods define discrete noises on a tiny period and use an explicit data array instead of PRNG.

Procedural textures by example. We are concerned with irregular patterns, which can be represented by noise. Only few techniques attempt to obtain noise parameters to match an input exemplar. Ghazanfarpour and Dischler [1995] apply filtering of white noise. The filtered noise is used to define spatial distortions with controlled spectral characteristics, but without aiming for visual similarity. Dischler and Ghazanfarpour [1997] compute noise-based displacement textures obtained from an example 1D profile. A combined spectral and histogram analysis is used to adjust the parameters of a sum of gradient noises. Bourque and Dudek [2004] uses a spectral metric to browse a database of procedural textures and to adjust their parameters. It is a database querying technique, not an automatic by example procedural texture synthesis technique. Gilet et al. [2010] show that noise, based on sparse convolution with Gabor filters [Lagae et al. 2009], can be applied to model some subsets of anisotropic color textures by example, since Gabor noise is characterized by Gaussian ellipses in the spectral domain. Lagae et al. [2010b] describes a method for generating isotropic noise-like textures by separating frequency bands using a sum of multi-scale wavelet noises [Cook and DeRose 2005]. Galerne et al. [2012] exploits the spectral characteristics of Gabor noise [Lagae et al. 2009]. The authors propose a robust and general method for creating patterns with arbitrary PSD from an example. Multiple band-pass Gabor filters are used. The parameters of the filters are obtained by expressing the spectral domain as a sum of multi-scale shifted Gaussian functions. Gilet et al. [2012] likewise propose to use band-pass filter kernels. In this case, box functions are used to approximate an arbitrary power spectrum.

Random patterns defined by Fourier synthesis. Fourier synthesis has a long history in texture generation. In some early work, Gardner [1985] uses a set of combined cosines with modulated phase to define natural stochastic textures, such as clouds. Ghazanfarpour and Dischler [1996] also use large amounts of cosines to define solid textures. In both cases, the problem is that too many cosines are necessary to generate non-periodic stochastic patterns. Other techniques have used an inverse FT with random phase or white noise filtering to define patterns [Saupe 1988; van Wijk 1991; Dischler et al. 1998; Galerne et al. 2010] according to some predefined power spectrum. But obtained textures are discrete, periodic and do not define noise functions.

Runtime tile-based procedural texturing. Tile or patch based techniques [Cohen et al. 2003; Lagae and Dutré 2006; Vanhoey et al. 2013] consist in random arrangements of pre-computed texture pieces generated from one or multiple exemplars. PRNG is used to select at runtime the tiles and / or contents, thus breaking periodicity. But these techniques repeat over and over again the same tiles / patches (i.e. rigorously identical contents) even for irregular textures, which is in contradiction with the random and non-repetitive nature of natural irregular patterns.

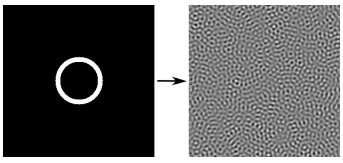
3 Noise model

A noise $n(x)$ in the spatial domain is characterized in the spectral domain by its power spectral density (PSD), i.e. the energy distribution $|\widehat{n}(f)|^2$, where $\widehat{n}$ denotes the Fourier transform. The challenge for a noise model consists in allowing for efficient computation and precise approximation for a given PSD.

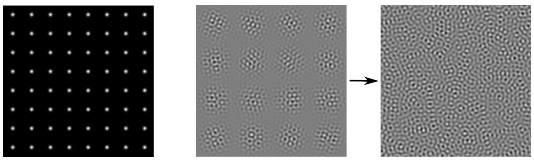
We define a new noise model by the following equation:

$$n(x) = \sum_{i=1}^{I} w\left(\frac{\|x - x_i\|}{\Delta}\right) \sum_{j=1}^{J} A_{i,j} \cos(2\pi f_{i,j} \cdot x + \varphi_{i,j}) \quad (1)$$

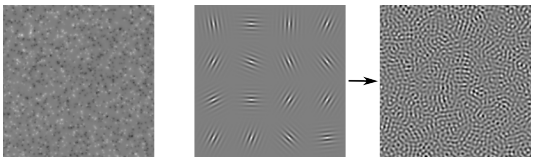
It is a mix of local noises with random phase, as illustrated in figure 2(b). The locality is controlled by a decreasing window w of width Δ centered at spatial samples x_i. Each local noise is a sum of cosines (sum over j) such that i) randomness is provided by random phases $\varphi_{i,j}$ and frequencies $f_{i,j}$, and ii) the spectrum is controlled by sampled frequencies $f_{i,j}$ and amplitudes $A_{i,j}$. Therefore we call it *local random-phase* (LRP) noise.



(a) Fourier series (PSD and result)



(b) LRP noise (spatial lattice, local noises, and result)



(c) Gabor noise (random impulses, kernels, and result)

Figure 2: *LRP noise (b) encompass noise by Fourier series (a) and Gabor noise (c). It is a mix of local noises with fixed PSD and random phase. It enables an optimized sampling in the spatial and spectral domains.*

Equation (1) applies to any dimension. In this paper we consider 2D positions x and frequencies f. The product $f \cdot x$ in the cosine denotes the dot product.

This model can be seen as a generalization of two standard techniques: noise by Fourier series and Gabor noise. Local random-phase noise provides a tunable compromise between the two. In the following, we explain its relation to them and discuss the parameters of equation (1) in detail.

3.1 Noise by Fourier series

One classical way of computing a noise with a given PSD is by inverse FT of its spectrum: amplitude is given by the PSD and phase is random (see figure 2(a)). It amounts to defining the noise by Fourier series (FS) as $\sum_{j=1}^{J} A_j \cos(2\pi f_j \cdot x + \varphi_j)$ with random φ_j. The approach by inverse FT has major drawbacks: i) the noise is periodic, ii) it is computationally intensive, and iii) discrete implementation by inverse FFT is resolution dependent.

Note that FS noise equals the interior sum in equation (1): it can be seen as the limit of our model when the window size Δ tends to infinity. Stated otherwise, our model locally blends FS noises using windows w, thereby breaking periodicity. Moreover, the complexity drops because the number J of cosines is low. So the computation becomes tractable in the spatial domain and the noise is resolution independent.

3.2 Gabor noise

Another popular way consists in filtering a white noise approximation with a Gabor kernel [Lagae et al. 2009]. The filtering is processed by a sparse convolution in the spatial domain (see figure 2(c)): $\sum_{i=1}^{I} A_i w\left(\frac{\|x-x_i\|}{\Delta}\right) \cos(2\pi f_i \cdot (x - x_i))$, where w is a truncated Gaussian. The power spectrum is controlled through f_i and A_i while randomness is provided by random positions x_i of the impulses. At a given position x, the sum runs over a neighborhood proportional to Δ. Since the spectral resolution is $1/\Delta$, the more precise the PSD approximation, the more impulses are necessary.

Our model can simulate Gabor noise by fixing one cosine per impulse ($J = 1$) and defining the phase as a function of random impulse positions ($\varphi_{i,j} = -f \cdot x_i$). However, by decorrelating positions x_i and phases $\varphi_{i,j}$, the computational efficiency is improved because i) the spatial sampling can be regular, and ii) one can increase the spectral sampling J independently.

3.3 Local random-phase noise

The LRP noise model, defined by equation (1) is controlled by a series of parameters in the spatial domain (I, x_i, Δ, w) and in the spectral domain (J, $f_{i,j}$, $A_{i,j}$, $\varphi_{i,j}$). In the following, we explain how to fix the window w with respect to a regular spatial lattice x_i such that, for any position x, the outer sum runs over $I = 9$ spatial samples only. Defining values for $f_{i,j}$, $A_{i,j}$ and $\varphi_{i,j}$ is the problem of spectral sampling that we address in sections 4 and 5. The two remaining parameters Δ and J emphasize a key property of this model: the spatial and spectral sampling densities are controlled independently (resp. by Δ and J). First, this property provides efficiency because it allows for balanced tuning of the sampling between space and frequency. Second, it provides versatility: the model can resemble sparse convolution when the noise has a simple spectrum, or Fourier series when fine spectral discrete control is needed. Third, it allows for capturing spatial structures by phase control (see section 5). Figure 3 illustrates the influence of spatial and spectral sampling densities on an anisotropic noise.

3.4 Spatial sampling

Since the random phase is de-correlated from the window positions, we can fix x_i on an regular integer lattice. w is chosen to have a value $w(0) = 1$ and to vanish outside $[-1.5; 1.5]$. By doing so, we get rid of random sparse sampling, thus optimizing spatial coverage. This spatial sampling fixedly depends on the window size controlled only by Δ from now on. Furthermore, we control the computational complexity: the first sum in equation (1) runs over a constant amount of only $I = 9$ neighboring spatial samples.

As a counterpart to the regular sampling, LRP noise is not strictly stationary. However, thanks to the shape of w (see section 3.6), and to its size related to the sampling rate, the envelope varies less than 5%. Thus no grid artifacts are visible in practice.

3.5 Control of the spectral leakage

The quality of a noise n depends on how precisely we control its spectrum. To explain the leakage problem, let us generate an anisotropic noise of fixed frequency f by equation (1). Cosines of frequency f are windowed with w, which causes $\widehat{n}$ to develop non-zero values (leaks) around f according to $\widehat{w}$. We want as much energy as possible to be concentrated around f, which is known as the spectral concentration problem. The leaks are impacted by the shape of the window (see section 3.6) and by its size Δ. Actually, they are proportional to $1/\Delta$. Stated differently, the larger w, the narrower $\widehat{w}$ and vice versa. This is well known from Heisenberg's uncertainty theorem: there is a lower bound to the product of variances in the spatial and spectral domains. Thus a trade-off is necessary.

The smaller Δ, the more leaks around the frequency f (figure 3 top row). Conversely, the larger Δ, the more precise the power spectrum. However, a too large Δ reduces the visual quality of noise,

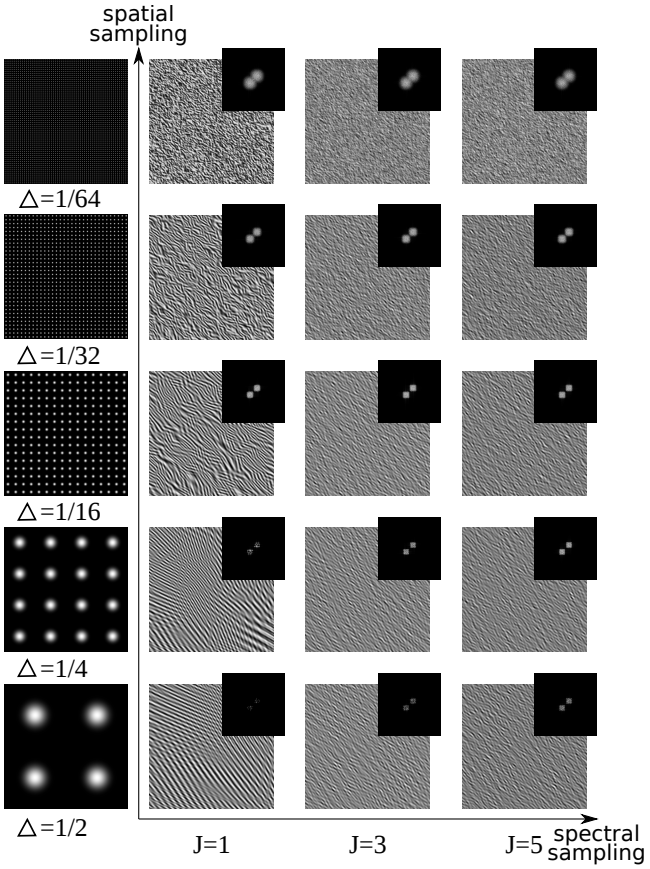


Figure 3: *Anisotropic noise generated for increasing spatial sampling (bottom up) and spectral sampling (from left to right). Increasing spectral sampling quickly improves visual quality (bottom lines from left to right). Fine spatial sampling fails to compensate insufficient spectral sampling (top left). The spectral leakage increases as the window becomes narrow (spectra, bottom up).*

because the regularity of the cosines becomes visible ($1/f$ period) and the result "lacks randomness" (figure 3 bottom left). One can be relieved of this constraint by increasing the spectral sampling, *i.e.* increasing the number J of cosines and random phases (figure 3 left to right). A good compromise is given by $\Delta \propto \sqrt{J}/F$, where F is the resolution in the spectral domain, *i.e.* the maximum frequency of the signal. A more precise relation is given in section 4 for the discrete case.

3.6 Choice for the window w

The shape of the window w influences spectral leakage. An ideal solution to the spectral concentration problem is provided by the discrete prolate spheroidal sequence. But numerical computations are complex and cannot be implemented easily on the GPU for runtime computation. A fair approximation is provided by the Kaiser-Bessel window function

$$w(y) = b_3\left(3\pi\sqrt{1-(y/1.5)^2}\right)/b_3(3\pi)$$

where $b_{\alpha=3}(z) = \sum_{m=0}^{\infty} \frac{1}{m!(m+2)!}\left(\frac{z}{2}\right)^{2m+3}$ is the modified Bessel function with parameter $\alpha = 3$, which we approximate by 5 terms ($m \leq 4$). Normalization by $b_3(3\pi)$ ensures $w(0) = 1$. Normalization of y by 1.5 and clamping beyond ± 1.5 make the window fit within the desired interval $[-1.5, 1.5]$.

This window tends towards a Gaussian shape when α is increased. Using $\alpha = 3$, it resembles a Gaussian, which could be used alternatively. However, the spectral concentration properties of a Gaussian are less interesting and it has the drawback of requiring itself yet another zero-ended window, because it extends to infinity.

4 Noise by example

In this section, we explain our strategy to meet an arbitrary PSD. The phase φ is completely random and the cosine budget J is user-fixed. The challenge consists in sparsely sampling the frequency domain $(f_{i,j})$ with corresponding amplitudes $(A_{i,j})$, such that the resulting noise approximates a given PSD $A^2(f)$.

If the sampling were dense, one could use the target PSD as probability density function for $f_{i,j}$ (to give more probability to high energy regions) and one could set $A_{i,j} = A(f_{i,j})$. This does not work in our context of sparse sampling (J is typically a few tens to keep computation time low). Thus features in the spectral domain would be missed, such as narrow peaks that concentrate a lot of energy.

We state this problem as a compromise between spectral coverage and power approximation as follows. Each local noise (fixed i in equation (1)) is a sum of cosines of frequencies $f_{i,j}$ localized by the 2D window $W(x/\Delta) = w(\|x\|/\Delta)$. So its spectrum is a weighted sum of $\widehat{W}(\Delta f)$ centered at $f_{i,j}$ (see additional material for detailed calculations). Since $\widehat{W}$ is a decreasing radial function, one sample $(f_{i,j}, A_{i,j})$ contributes to the PSD with a power $A_{i,j}^2$ in a neighborhood of size $1/\Delta$ around $f_{i,j}$. In this light the frequencies should be well distributed to cover the spectrum with a minimum budget. On the other hand high power regions should be given more samples because they strongly influence the spatial characteristics of noise. We solve this problem by using stratified sampling.

In the following we first explain the stratified sampling strategy. Then we show how to build and to sample the strata. The target PSD may be continuous or discrete and we provide details for a discrete $T \times T$ spectrum. Finally we recap the practical implementation that provides noise generation from an input example texture.

4.1 Stratified sampling

The principle of stratified sampling is to partition the spectral domain into disjoint regions S called *strata*. Each stratum is then sampled independently. The first advantage is a better coverage of the domain because each stratum is ensured of being sampled. The second one is to control the distribution of the sampling budget among the strata. In equation (1), we would like each stratum to have its own spectral sampling density J_S and corresponding spatial sampling period Δ_S. It amounts to defining the noise

$$n(x) = \sum_S n_S(x) \tag{2}$$

as a sum of several noises n_S (one per stratum S), with disjoint spectra, each of them being governed by equation (1). The challenge is to define L strata and distribute the sampling budget J such that the generated noise spectrum amplitude $|\widehat{n}|$ approximates a target amplitude A as precisely as possible.

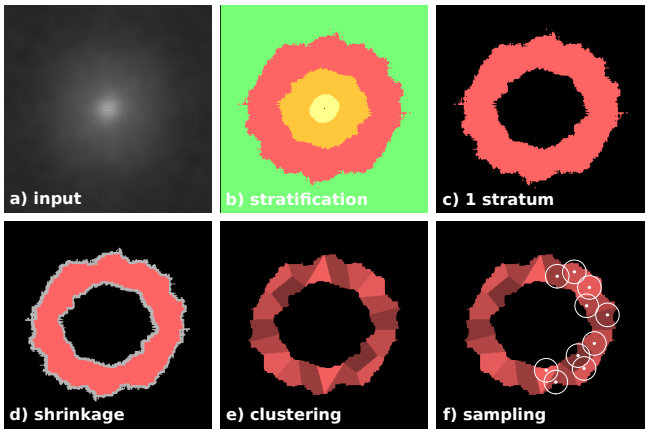


Figure 4: *Intra-stratum sampling. The input spectrum (a) is partitioned into strata (b) with the same amount of energy. Each stratum (c) is shrunk (d) and broken into sub-strata by k-means (e). One random frequency is drawn in each sub-stratum such that the spectral windows roughly cover the stratum (f).*

4.2 Strata building

A partition of the spectral domain independent of the PSD, such as a regular grid, well covers the domain but is not efficient in our case because of two reasons. First it equally allocates samples to low and high energy regions. We solve this by building strata that have the same energy. Second the power density may vary a lot within a stratum, so a few windows $\widehat{W}$ cannot approximate precisely the spectrum. We solve this by building strata that have homogeneous power densities.

To achieve this we partition the PSD range (*i.e.* the image of A^2) into L intervals I_S and we define the strata $S = \{f | A^2(f) \in I_S\}$. In this way strata have homogeneous power densities. In the discrete setting we uniquely define the intervals such that the stratum energies $E_S = \sum_{f \in S} A^2(f)$ are equal, which ends up in small strata of high PSD and large strata of low PSD (figure 4b). We then distribute the sampling budget uniformly $J_S = J/L$ such that high power strata are given more precision relatively to their size.

When applied on complex or noisy spectra such as example based spectra (section 4.4), a low number L of strata must be chosen in order to avoid too fragmented strata that would be difficult to sample. We actually use $L = 4$ in all our examples. It therefore produces large strata which may still have complex shapes.

4.3 Intra-stratum sampling

Here we explain how to draw the J_S samples $(f_{i,j}, A_{i,j})$ in a given stratum S. As explained above it amounts to approximating the spectrum in S by a sum of $\widehat{W}(\Delta_S f)$ centered at $f_{i,j}$ and weighted by $A^2_{i,j}$. The sampling is illustrated by figure 4. It must meet three goals: i) avoid spectral leakage, *i.e.* avoid diffusing too much energy outside S; ii) optimize the spectral coverage of S; iii) approximate the amplitude in S.

To avoid spectral leakage, we do not pick the frequencies in the entire stratum. S is shrunk by a constant δ defined such that at least 75% of the scaled window energy $|\Delta_S \widehat{w}(\Delta_S f)|^2$ is within $[-\delta; \delta]$. The 75% threshold is empirical. Calculating δ is solved numerically for any value of Δ_S. Discrete shrinking is performed by morphological erosion, modified so that strata smaller than δ are not completely swept out.

To optimize the coverage while saving randomness in the frequencies, the shrunk stratum is broken into J_S sub-strata using k-means with random seeding. Then one single frequency $f_{i,j}$ is uniformly drawn in each sub-stratum. Since the spectrum is symmetric it is applied on half of it (figure 4f). Since sub-strata may have an arbitrary shape (see figure 4e), practical implementation consists in storing the sub-stratum frequencies into lookup tables, which are then randomly accessed.

A good spectral coverage also requires the spectral windows $\widehat{W}(\Delta_S f)$ to be large enough so that J_S windows can cover S. As shown by white circles in figure 4 the windows' footprints are disks of radius $1/\Delta_S$. In the discrete setting we define $\Delta_S = T\sqrt{J_S}/\sqrt{2|S|}$, where $|S|$ is the number of discrete frequencies in S. Thus the area covered by J_S footprints is proportional to the stratum area $|S|/T^2$. The constant factor, here 2, depends on the actual window shape: we fix it such that the windows slightly overlap.

For each frequency $f_{i,j}$ we now have to define an amplitude $A_{i,j}$ such that the target PSD is well approximated over S. Since S contains homogeneous amplitudes and is sparsely sampled, we target a constant value $A_{i,j} = A_S$. We aim at the energy over the stratum to be preserved: we define A_S such that the noise energy $E(n_S) = \int |\widehat{n_S}|^2$ is equal to $E_S = \int_S A^2$. In the discrete setting we actually define

$$A_S = \left(\sum_{f \in S} A^2(f) \right)^{1/2} \left(\Delta_S^2 J_S \pi \int_0^\infty r w^2(r) dr \right)^{-1/2} \quad (3)$$

where $E_S = \sum_{f \in S} A^2(f)$ is computed by discrete summation. The constant $\int_0^\infty r w^2(r) dr \approx 0.1126$ for the Kaiser-Bessel window. Formula (3) is based on an analytical derivation of $\widehat{n_S}$ which is detailed in the additional material.

4.4 Noise by example with discrete PSD

We now explain in practical terms how the sampling is implemented for an arbitrary power spectrum, derived from a user-supplied "example-noise" image. Figure 5 summarizes the procedure. Only one parameter is tunable by the user: the cosine budget J which defines the computation load.

A first series of computations are pre-processed on the CPU:

Computing the input PSD. The FFT of the noise image is computed. To get a better approximation of the power spectral density, we use Welch's method [1967]. It consists in averaging the power spectra of multiple FFTs computed on crops of size $T \times T$. This defines the frequency resolution $F = 1/T$ (see section 3.5). T is chosen as the largest power of two strictly lower than the input image size.

Stratification. The obtained power spectrum is partitioned into $L = 4$ strata as explained in section 4.2. To each stratum S, we associate $J_S = J/L$ cosines.

Sampling pre-processing. As explained in section 4.3, for each stratum S we fix $\Delta_S = T\sqrt{J_S}/\sqrt{2|S|}$ and compute A_S using equation (3). Each stratum is shrunk by morphological erosion and broken into J_S sub-strata using k-means.

Then we transfer to the GPU the values of J_S, Δ_S and A_S, as well as the tables of sub-strata frequencies. The following computations are all done on-the-fly, to evaluate the noise at continuous positions x:

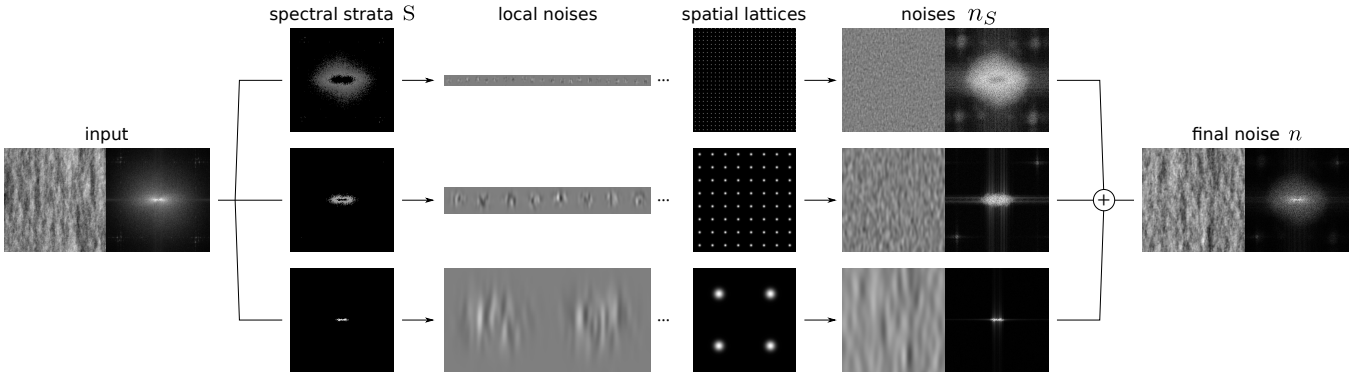


Figure 5: *Noise by example workflow. The input example is analyzed by Fourier transform. It's spectrum is decomposed into strata S according to energy levels. Each stratum has its own frequency sampling and spatial lattice, leading to LRP noises n_S with disjoint spectra that are finally summed to get the final noise n.*

Sampling. A single sample $f_{i,j}$ is uniformly drawn per substratum. The amplitude $A_{i,j}$ is given by A_S. Phase $\varphi_{i,j}$ is random in $[0; 2\pi[$. This is done using a linear congruential pseudo-random number generator, initialized by a hash-code computed from $\lfloor x/\Delta_S \rfloor$.

Evaluation. The noise is evaluated by formulas (2) and (1).

5 Textures by example

In previous sections, we have shown how our LRP noise model can generate Gaussian patterns with arbitrary PSD. In this section, we target a new goal that is made possible by our model: reproducing example textures containing not only Gaussian patterns but also visually structured features.

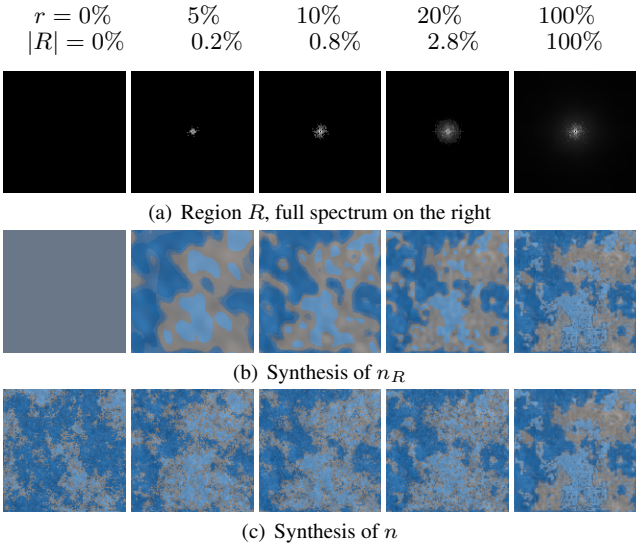
Structure in an image defines the most noticeable features like stripes, wood veins or color patterns (*e.g.* in figures 8 and 13). In the spectral domain, they are characterized not only by the PSD, but also by the phase [Oppenheim and Lim 1981]. Our model (equation (1)) allows to fix phases independently from the spatial sampling, unlike previous procedural methods. We will thus fix the phases of a (spectral) region composed of the frequencies encoding the structure. The remaining frequencies are left random-phased using the stratified sampling of section 4, as to keep the advantage of procedural noise. Regular structure being characterized by high energy [Nicoll et al. 2005], this region will be built by assembling highest-energy frequencies until it assembles an arbitrary proportion of the total energy.

5.1 Structure in fixed phases

The LRP noise model allows for free phase selection in equation (1). When evaluating like in section 4, random phase generates random features. At the other end an exact periodic reproduction of the input image can be achieved by fixing the phase $\varphi(f)$ and the amplitude $A(f)$ to those of the example's FT (it amounts to inverse FT). We define a trade-off between these two ends by fixing the phase and the amplitude for a region R that "contains the structure" while the complement is still computed as in section 4:

$$n = n_R + \sum_S n_S \quad \text{where}$$

$$n_R(x) = \sum_{i=1}^{I} w\left(\frac{\|x - x_i\|}{\Delta_R}\right) \sum_{f \in R} A(f) \cos(2\pi f \cdot x + \varphi(f)) \quad (4)$$



(a) Region R, full spectrum on the right

(b) Synthesis of n_R

(c) Synthesis of n

Figure 6: *Fixed phases for high-energy regions ($J = 50$). From left to right the region R grows while its energy interval gets larger. The corresponding structure n_R better captures structures until being equal to the input (right). The final noise n exchanges randomness for faithfulness.*

also follows equation 1 with no randomness (fixed A and φ), and finest sampling $J_R = |R|$. To have the same $T \times T$ resolution as the PSD we get A and φ from any of the blocks used in Welch's method.

To determine R in practice, we introduce a new tunable parameter $r \in [0; 1]$ such that the proportion of total energy contained within R is r. To build R with this respect, we iteratively add the highest-amplitude frequencies to R.

Figure 6 shows R growing from void (left) to full energy (right): n_R gradually captures the input texture while n looses randomness. At the extremities, n varies from a fully procedural ($r = 0$) to a copy of the original sample ($r = 1$). The irregular structure is well preserved, for instance when $r \approx 20\%$.

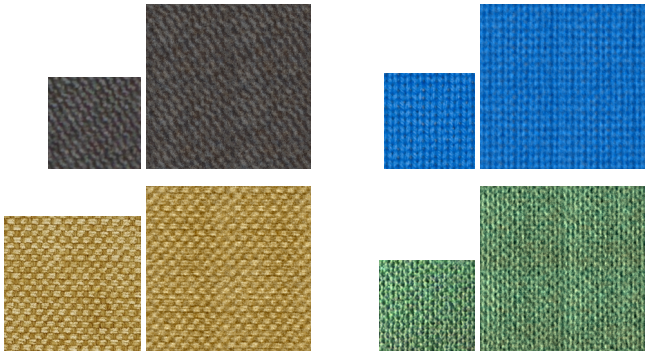
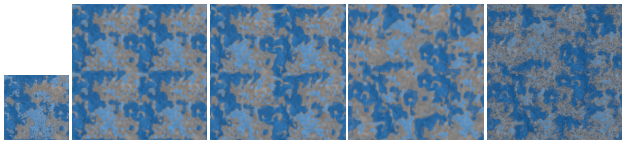


Figure 7: *Near regular textures are faithfully reproduced with structure preservation ($J = 50$, $r = 10\%$). Neither turbulence nor random placement is required.*



(a) In (b) n_R (c) + turb. (d) + rand. pl. (e) n

Figure 8: *Texture by example. The main features of the input (a) are reproduced in (b) thanks to the model of equation* (4)*, but introduce periodicity and repetition. Feature repetition is discarded in (c) by turbulence: features are spatially distorted. Periodicity is broken in (d) by random placement. Final result n is obtained by adding random phase strata n_S.*

5.2 Repetition and periodicity

The counterpart to fixing the spectrum in R is that we have to deal with repetition (the same pattern appears at multiple positions) and periodicity (similar patterns appear at regularly spaced positions). For near regular textures like in figure 7 this is not a problem. However, for less regular textures, it provides visual artifacts when r grows (see figure 8(b)). We remove them by using two widely-used techniques.

Breaking repetition. In order to break repetition of major features' shapes, we distort space like in Perlin's "image synthesizer" [1985]. We evaluate the structure function n_R at a perturbed position $x + \sigma\ n_T(x)$ where σ is a scaling factor and n_T is a turbulence function. It is defined by adding absolute values in equation (2): $n_T = \sum_S |n_S|$ is computed from the (whole) spectrum itself. This introduces a break of gradient (thus a sudden change of direction) making a turbulence-based deformation well matching natural brownian motion-like phenomena. Since it is based on the very spectrum of the noise the features are distorted but not broken, as shown in figure 8(c). The scaling factor σ is easy to tune: it must match the size of typical features of the texture. A rough stratification and a low cosine budget is actually sufficient. Some more examples and details are given in additional material.

Breaking periodicity. When a lot of structure is preserved features appear at regular spatial intervals. In figures 8(b) and 8(c) for example, one can guess a 2×2 regular grid, regardless of turbulence. This is classically solved by randomizing spatial distribution. Similarly to quilting [Efros and Freeman 2001] or Chaos Mosaics [Xu et al. 2002] we randomly shift blocks of size $1/2$. This is done

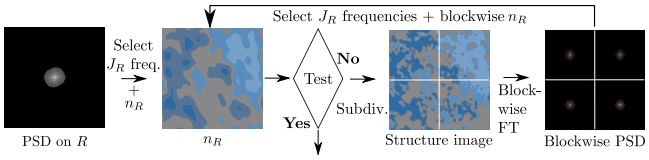


Figure 9: *Computation load is reduced by a pre-computed iterative subdivision process. n_R has to be evaluated using only J_R cosines. When J_R frequencies cannot reproduce the structure image, the latter is regularly subdivided in always smaller blocks. Block-wise FT are computed and J_R frequencies per block are selected (red pixels) for synthesis of n_R by inverse FT. When the result is satisfying (below an error threshold w.r.t. the structure image), subdivision stops.*

by evaluating n_R at the shifted position $x + \frac{1}{2}t\,(\lfloor 2x \rfloor)$ where t is a hash-code applied on both coordinates of x. This way, periodicity is avoided: see figure 8(d). The size $1/2$ is an arbitrary compromise between a sufficient randomization and preservation of local spatial coherence.

In practice, partial structure preservation works well for $r \in [0, 20\%]$. When r grows, turbulence and random placement become more and more useful. Note that when both are used, random placement must be applied first in order to ensure continuity of the spatial distortion all over space. It would make no sense to apply turbulence independently to individual texture blocks, because discontinuities would become visible on the block borders.

5.3 Complexity reduction

Once R has been pre-computed, $|R|$ amplitudes and phases have been stored, and thus evaluation is achieved using equation (4) for R and (2) for the remaining PSD. However, evaluating $|R|$ cosine functions may be intractable. To define a constant computation load respecting the user-provided cosine budget J, we reduce the number of cosine evaluations for R to a proportion of the cosine budget, namely $J_R = rJ$. Stratification for the remaining part of the PSD will then benefit from $(1 - r)J$ samples to distribute over its spectrum.

Figure 9 shows how we proceed to reduce the number of cosine evaluations. Let's call the *structure image* the inverse Fourier transform of the spectrum on R only, which is what we want to approximate by n_R with J_R cosines only. Firstly we select the J_R highest-amplitude frequencies. Then n_R is evaluated with these J_R frequencies at the positions of the pixels of the structure image. We define the result as satisfying if the average color deviation (compared to the structure image) is below the color precision ($1/256$ in general). If it is not, an iterative process begins. The structure image is subdivided by a regular grid and a block-wise FFT is computed. Then, the J_R highest-amplitude frequencies are selected for each block. Finally, n_R is computed by block and re-assembled for evaluation. Δ_R is set to the block size so as to handle transitions. The algorithm stops when the result is satisfying, otherwise it iterates with a new subdivision step.

This way, global low-frequency structure can be reconstructed at a reduced cost. We are actually trading continuity for computational efficiency here. Depending on the input image as well as parameters J and r, the subdivision process may iterate more or less. At the extreme, one pixel forms a block and its color is then exactly encoded by the only zero frequency. Tests are shown in the additional material as well as a step-by-step description of the complete procedure.

6 Results

In this section we discuss qualitative results and performances. We compare our method to patch-based techniques and to procedural techniques. All performances result from a real-time implementation algorithm on an nVidia GeForce GTX 780 with 3GB video memory using a deferred shading pipeline. Amplitudes, frequencies and phases are stored into GPU memory using *Shader Storage Buffer Objects* for direct runtime access.

Color representation. Our examples show color whereas our formula operate on a single channel only. Synthesizing texture with noise in RGB space is tedious: unseen colors can easily arise when channels are treated independently. Solving this is a complex problem which is out of scope for this paper. The solution usually consists in either using an indirection in a color table, or using a color space with channels as independent as possible. Here we use a mix of these two approaches based on the color representation proposed by Vanhoey et al. [2013]. It works well when the input texture exhibits a few dominant colors and it avoids histogram equalization. Technical details are provided in additional material.

6.1 Noise with arbitrary PSD

The LRP-noise model can approximate an arbitrary PSD which may be continuous or discrete. Noise by example is made possible by computing a target discrete PSD from an input example.

We compare results to Lagae et al. [2009] for continuous spectra. In general, our method reports a better quality/complexity ratio: a computation time gain of about a factor of 2 is observed at equal quality. This is because of a better spectral/spatial sampling balance that optimizes coverage in the spectral domain and therefore allows for a regular sparse coverage in the spatial domain.

Equation (1) defines that a budget of J results in the computation of 9 window functions plus $9J$ cosines per evaluation. For a fair comparison with Lagae et al. [2009] in figure 10, we used $9J$ impulses, which corresponds to $9J$ cosines plus $9J$ Gaussian window evaluations. Figure 10 shows two different spectra that result in better local noise at equivalent cost when approximated with our technique. Our results for $J = 3$ are for example of equivalent or close quality to those obtained with $J = 5$ for Lagae et al. [2009].

6.2 Procedural texture by example.

In section 5, we presented a method to reproduce structured features. This is where our method achieves an unmet goal: procedurally reproducing an input texture, including its structural components. Our model blends between pure Gaussian textures and textures consisting of random placement and spatial distortions. Therefore, we compare to methods reproducing either one of them.

Patch-based techniques copy texture parts in order to generate larger textures. This includes structural elements and therefore structure preservation is optimal but repetition may generate unnatural textures. Figure 11 compares our result (last column) with patch-based techniques. In the first column, one can identify the same repeated structures, which become obvious on larger textures. Since our model adds randomness (noise, random placement, turbulence), we trade perfect resemblance and visual matching for more visual variety. In other words, our model may create texture features that were not present in the input example. This adds realism while structure preservation is still satisfying in figure 11.

One can observe our method for both color and displacement mapping on a 3D model in figure 12. It shows texture continuity, which

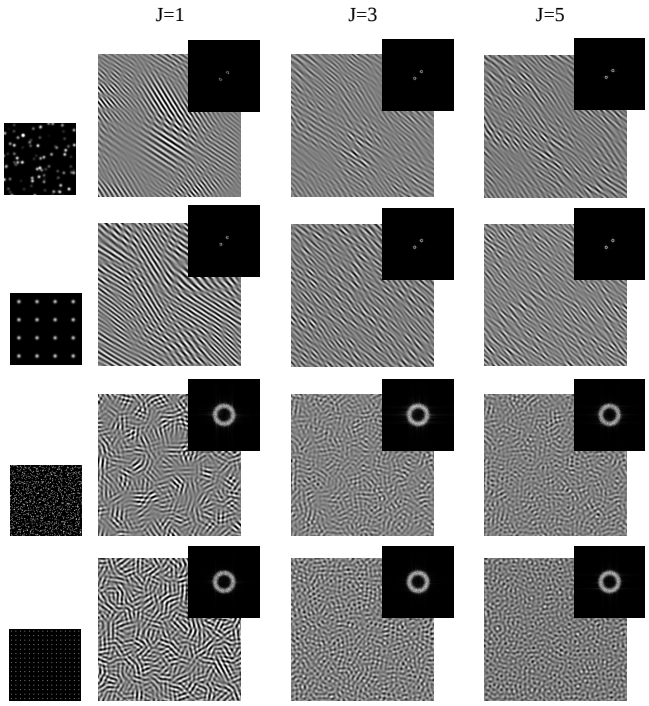


Figure 10: *Comparison of our LRP noise (2nd and 4th rows) with Gabor kernels (1st and 3rd rows) on continuous ring-shaped spectra. Evaluations for $J = 1, 3, 5$ (1 stratum) show that LRP noise converges more rapidly.*

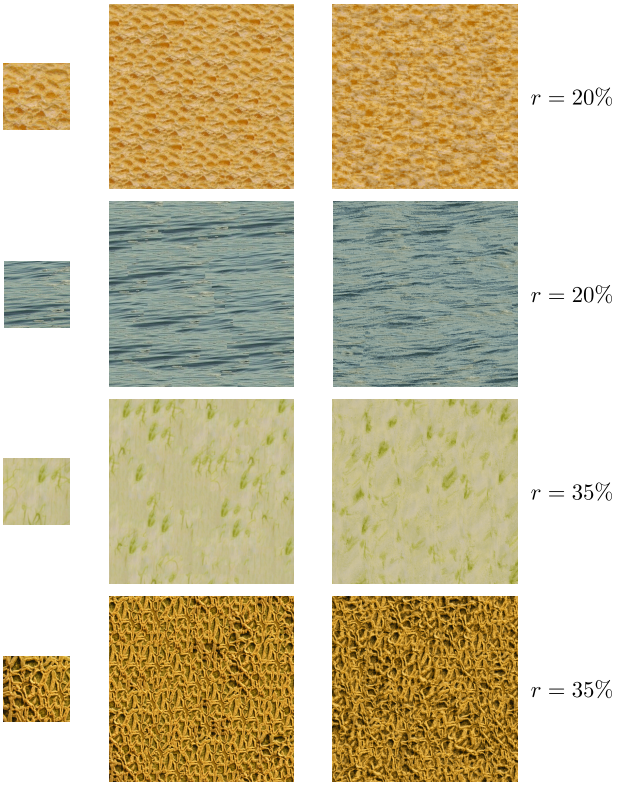


Figure 11: *Comparison of our LRP noise model (right) to patch-based methods (left): [Kwatra et al. 2003] (first two rows) and [Efros and Freeman 2001] (last two rows). LRP noise ($J = 50$) has less resemblance but more variety, and it is continuous.*

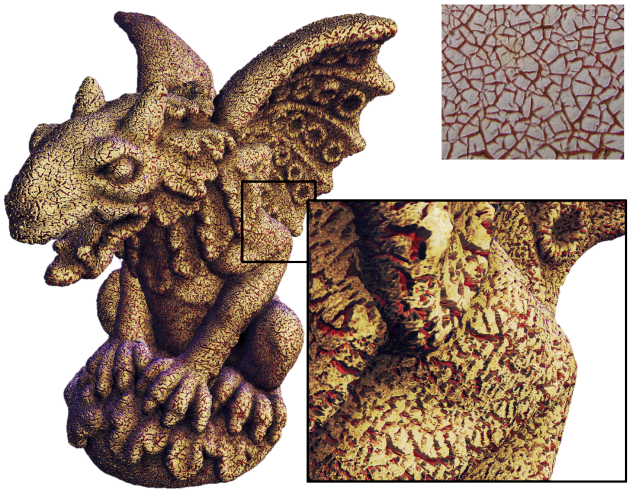


Figure 12: *Example-based texturing of a 3D object ($J = 50$, $r = 30\%$). No artifacts appear when zooming in because the model is continuous. Rendering speed for this 1200^2 framebuffer is $400\,fps$ without texturing, $59\,fps$ with texturing, $23\,fps$ with bump mapping, and $7\,fps$ with tessellation and displacement mapping.*

Texture	Rand phases	Fixed phases		+ rand pl.
	Fps / ms	Fps	Mem.	Fps / ms
ConcreteBare	1750 / 0.57	1780	$5.2K$	1477 / 0.68
Ground	2330 / 0.43	2315	$1.7K$	1983 / 0.50
ConcreteWorn	2050 / 0.49	1990	$3.9K$	1680 / 0.59
Fabric2004	1760 / 0.57	1760	$3.2K$	1500 / 0.67
Rera	1800 / 0.56	1770	$3.2K$	1420 / 0.70

Table 1: *Comparison of performances (speed and memory). Texture names refer to those of figure 13: the same parameters J and r were used. Measurements are for resolution 128×128 pixels and the memory designates the stored frequencies.*

is a property patch-based methods do not have: it extends to infinity and is scale-independent (*i.e.* one can zoom in without resolution loss).

As opposed to patch-based methods, noise-based procedural textures can generate continuous diverse content. Figure 13 shows comparative results with state of the art method [Galerne et al. 2012]. The second and third column compare Gabor noise with LRP noise (with fully random phases): equivalent results are obtained. When fixing some phases, we overcome the major limitation of this technique: its inability to reproduce the overall structure of the input example. The last column shows that preserving structure is important to obtain a high quality resembling result.

Table 1 shows corresponding performances. The complexity is essentially related to the number of special function calls ($\cos$, w). That is, $9(J + s)$ calls where s is the number of strata. In practice, for LRP noise with simple PSD (*e.g.* figure 10), we obtain 36 to 54 calls (typically $J = 3$ to 5, $s = 1$). For noise by example, 306 to 486 calls are required (typically $J = 30$ to 50, $s = 4$). This induces that our texture evaluation is faster than Galerne et al. [2012]. Table 1 shows that fixed phases imply no overload because the same number of calls is required. Random placement penalizes the performance (10 to 20% in practice, depending on r) because blocks have to be blended.

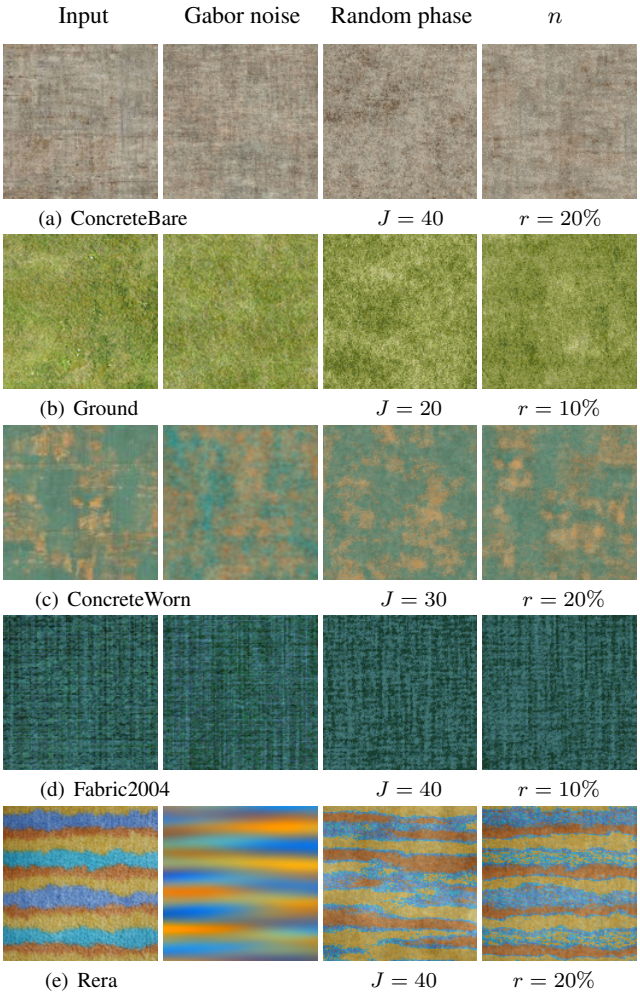


Figure 13: *An input texture and procedural textures generated respectively by Galerne et al. [2012], by our technique with $R = \emptyset$ (random phase) and with an arbitrary R (some fixed phases) plus random placement.*

6.3 Parameter management

The two main parameters of our model are the cosine budget J and the energy ratio r separating phase-fixed and phase-random components. The pre-processing takes less than 1 second for 256^2 input images making the tuning interactive, as shown in the video. The first parameter (J) can be set according to the quality/performance trade-off. A few cosines are sufficient for simple spectra (see figures 3 and 10). Noise or texture by example generally requires about 50 cosines. Examples of possible trade-offs are provided as additional material. The second parameter (r) allows us to smoothly blend between a pure Gaussian texture model and a texture exactly matching the input. In the latter case, our model reduces to applying turbulence and random placement. Figure 14 illustrates the influence of this ratio: structure may be lost if it is too low (middle column).

Other parameters can generally be fixed (as Δ and L) or are no specificity of our method (as σ). In the additional material, we provide many variants and tests about their sensitivity.

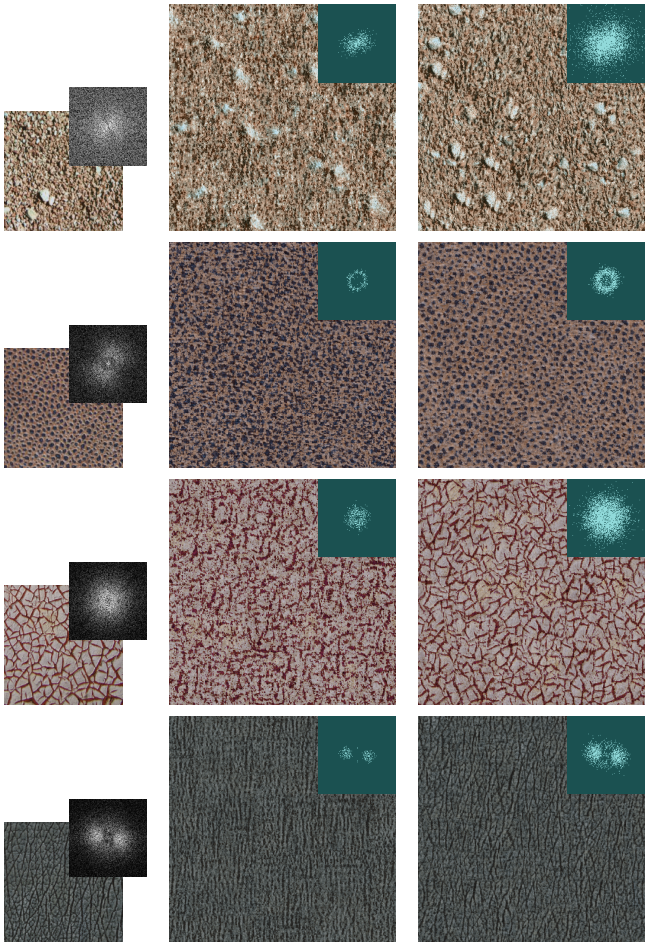


Figure 14: *Role of the energy ratio: some structures may be lost if it is set to low. Left: input. Middle:* $r = 20\%$. *Right:* $r = 35\%$ *(except bottom:* $r = 30\%$*).*

6.4 Limitations

Capturing structure. Our model assumes that structures can be captured in a region of the spectrum. Some types of texture however may have their structure spread over the entire frequency domain. This is the case for features with precise shape and very coherent color variations (induced by lighting for instance). Characterizing them is still a challenge.

The result is that we cannot capture structure on some examples, as in figure 15. In this case results are either too noisy (center) either not continuous due to random placement (right). The turbulence may also introduce too strong feature distortions.

Texture filtering. LRP noise is defined in the frequency domain and can thus be filtered at runtime using frequency clamping: cosines with too high frequencies (i.e. frequencies above a limit defined according to the screen-space sampling rate) are faded out. Since we encoded fixed phases on multiple blocks (section 5.3), there are also multiple 0-frequency values that cannot be faded out. These values represent the mean of each block. To perform anti-aliasing on the set of 0-frequencies, further mip-mapping is thus required.

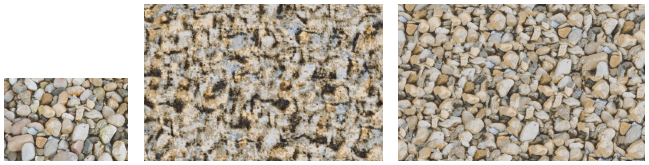


Figure 15: *On this example (left), selecting frequencies by thresholding based on amplitude fails to separate structure from noisy patterns. A too low* r *(center) results in lost structures whereas a higher* r *(right) generates visual discontinuities induced by the random placement.*

For color textures, filtering is known to be difficult and would require further investigation. When using a color table, indices are averaged instead of colors. Therefore we define a mipmap of the color table for each block, using color probabilities as weights.

7 Conclusions

In this paper we presented a new noise model for procedural texturing. It is defined as a blending of local noises centered on a regular spatial lattice. Each local noise is defined as a sum of cosines with random phase, while frequencies and amplitudes are sampled from a given power spectral density. A stratified spectral sampling strategy allows for efficient noise by example, outperforming state-of-the-art techniques. A key property of our model is the separate spectral and spatial sampling, that provides performance and versatility. It enjoys the nice properties of procedural approaches: no repetition, continuous texture, straightforward parallel implementation, on-the-fly parameter tuning.

Our model also allows for preserving structures in the texture, assuming that the structure information can be identified by a region in the spectrum. Structures are reproduced by saving the phases in the corresponding region, while the complement still has random phase. Thus we are able to generate a much broader range of textures than other procedural noises. We show the effectiveness of our approach on many examples whose structure is encoded in high energy regions of the spectrum. Thus the user can easily balance between structure and noise with a single ratio.

An interesting problem would be to characterize the uniformity of the r-slider w.r.t. the user-perceived "amount of structure" that is preserved in the resulting texture. This is however a complex problem requiring a thorough user-study.

A very promising perspective is a better selection of the region for structure in the spectrum. It would improve the range of textures yet not requiring any change in LRP noise model.

Our noise model can be straightforwardly extended to any higher dimension by using a n-D regular grid instead of a 2D grid. In particular, we would like to investigate solid texture applications.

Surface noise allows for computing noise directly on the surface. This allows to get rid of parameterization and distortions are avoided. The strategy generally consists in projecting 3D impulses in the tangent plane and computing the noise in this plane. In our context a problem arises because the projection of a regular grid is not a regular grid. In all our examples, we subsequently used a parameterization of the surface. An extension to surface noise would be useful.

Acknowledgements

We would like to thank Frédéric Larue for his technical assistance and the anonymous reviewers for their valuable comments.

References

BOURQUE, E., AND DUDEK, G. 2004. Procedural texture matching and transformation. *Computer Graphics Forum 23*, 3, 461–468.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Trans. Graph. 22*, 3 (July), 287–294.

COOK, R. L., AND DEROSE, T. 2005. Wavelet noise. *ACM Trans. Graph. 24*, 3 (July), 803–811.

DISCHLER, J.-M., AND GHAZANFARPOUR, D. 1997. A procedural description of geometric textures by spectral and spatial analysis of profiles. *Computer Graphics Forum 16*, 3, C129–C139.

DISCHLER, J., GHAZANFARPOUR, D., AND FREYDIER, R. 1998. Anisotropic solid texture synthesis using orthogonal 2d views. *Computer Graphics Forum 17*, 3, 87–95.

EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. 2002. *Texturing and Modeling: A Procedural Approach*, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '01, ACM Siggraph, 341–346.

GALERNE, B., GOUSSEAU, Y., AND MOREL, J.-M. 2010. Random phase textures: Theory and synthesis. *IEEE Transactions in Image Processing*.

GALERNE, B., LAGAE, A., LEFEBVRE, S., AND DRETTAKIS, G. 2012. Gabor noise by example. *ACM Trans. Graph. 31*, 4 (July), 73:1–73:9.

GARDNER, G. Y. 1985. Visual simulation of clouds. *SIGGRAPH Comput. Graph. 19*, 3 (July), 297–304.

GHAZANFARPOUR, D., AND DISCHLER, J. 1995. Spectral analysis for automatic 3-d texture generation. *Computers & Graphics 19*, 3, 413 – 422.

GHAZANFARPOUR, D., AND DISCHLER, J.-M. 1996. Generation of 3d texture using multiple 2d models analysis. *Computer Graphics Forum 15*, 3, 311–323.

GILET, G., DISCHLER, J.-M., AND SOLER, L. 2010. Procedural descriptions of anisotropic noisy textures by example. In *Eurographics*, Eurographics Association. Short paper.

GILET, G., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2012. Multiple kernels noise for improved procedural texturing. *Vis. Comput. 28*, 6-8 (June), 679–689.

GOLDBERG, A., ZWICKER, M., AND DURAND, F. 2008. Anisotropic noise. *ACM Trans. Graph. 27*, 3 (Aug.), 54:1–54:8.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. In *ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, SIGGRAPH '03, ACM Siggraph, 277–286.

LAGAE, A., AND DRETTAKIS, G. 2011. Filtering solid gabor noise. *ACM Trans. Graph. 30*, 4 (July), 51:1–51:6.

LAGAE, A., AND DUTRÉ, P. 2006. An alternative for wang tiles: Colored edges versus colored corners. *ACM Trans. Graph. 25*, 4 (Oct.), 1442–1459.

LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. 2009. Procedural noise using sparse gabor convolution. *ACM Trans. Graph. 28*, 3 (July), 54:1–54:10.

LAGAE, A., LEFEBVRE, S., COOK, R., DEROSE, T., DRETTAKIS, G., EBERT, D., LEWIS, J., PERLIN, K., AND ZWICKER, M. 2010. A survey of procedural noise functions. *Computer Graphics Forum 29*, 8, 2579–2600.

LAGAE, A., VANGORP, P., LENAERTS, T., AND DUTRÉ, P. 2010. Procedural isotropic stochastic textures by example. *Computers & Graphics 34*, 4, 312 – 321.

LEWIS, J. P. 1987. Generalized stochastic subdivision. *ACM Trans. Graph. 6*, 3 (July), 167–190.

NICOLL, A., MESETH, J., MÜLLER, G., AND KLEIN, R. 2005. Fractional Fourier texture masks: Guiding near-regular texture synthesis. *Computer Graphics Forum 24*, 3 (Sept.), 569–579.

OPPENHEIM, A. V., AND LIM, J. S. 1981. The Importance of Phase in Signals. *Proceedings of the IEEE 69*, 5 (May), 529–541.

PERLIN, K. 1985. An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3 (July), 287–296.

SAUPE, D. 1988. Algorithms for random fractals. In *The Science of Fractal Images*, H.-O. Peitgen and D. Saupe, Eds. Springer-Verlag New York, Inc., New York, NY, USA, ch. Algorithms for Random Fractals, 71–113.

VAN WIJK, J. J. 1991. Spot noise texture synthesis for data visualization. *SIGGRAPH Comput. Graph. 25*, 4 (July), 309–318.

VANHOEY, K., SAUVAGE, B., LARUE, F., AND DISCHLER, J.-M. 2013. On-the-fly multi-scale infinite texturing from example. *ACM Trans. Graph. 32*, 6 (Nov.), 208:1–208:10.

WELCH, P. D. 1967. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audia and Electroacoustics 15*, 2 (June), 70–73.

XU, Y.-Q., GUO, B., AND SHUM, H. 2002. Chaos Mosaic: Fast and Memory Efficient Texture Synthesis. Tech. rep., Microsoft Research, Apr.