Understanding and controlling contrast oscillations in stochastic texture algorithms using Spectrum of Variance

Fabrice Neyret, Eric Heitz

Understanding and controlling contrast oscillations in stochastic texture algorithms using Spectrum of Variance

Fabrice NEYRET *(LJK / Grenoble University & INRIA)*, Eric HEITZ *(Unity Technologies)*

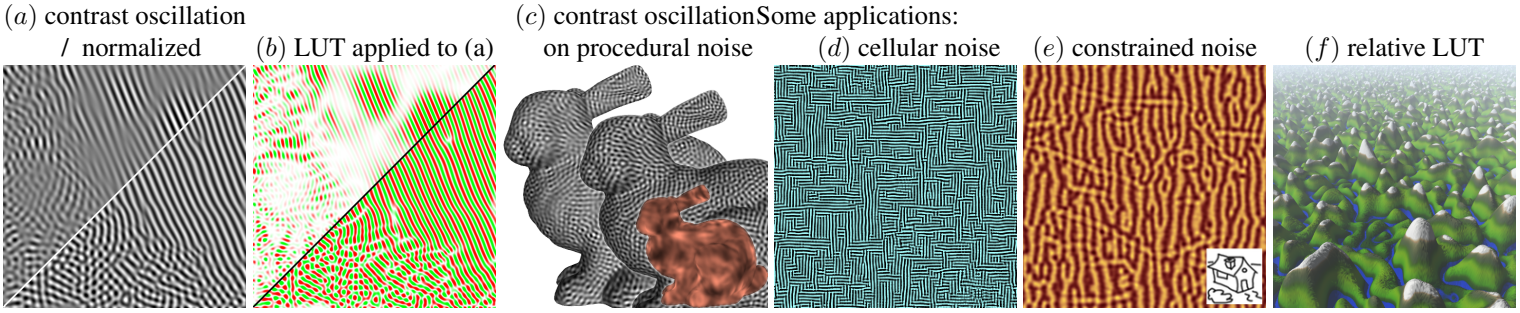

Figure 1: *Power-spectrum based texturing algorithms (e.g., Gabor, Fourier synthesis) suffer from unexpected low frequency contrast variations* (a,b,c top) *even when the spectrum has no low frequency (the contrast field is display in red in* (c)*). This prevents precise authoring with non-linear transform, like color LUT* (b top)*. Our renormalization method allows to control the stationarity* (a,b,c bottom)*. It also opens many doors for noise authoring such as the generation of reaction-diffusion-like strips and spots* (b bottom)*, cellular-like patterns* (d)*, content constraints* (e)*, or the parametrization of height maps relative to local extrema* (f)*. — Zoom or see supplemental for larger images.*

Abstract

We identify and analyze a major issue pertaining to all power-spectrum based texture synthesis algorithms – from Fourier synthesis to procedural noise algorithms like Perlin or Gabor noise – , namely, the *oscillation of contrast* (see Figures 1,2,3,7).

One of our key contributions is to introduce a simple yet powerful descriptor of signals, the *Spectrum of Variance* (not to be confused with the PSD), which, to our surprise, has never been leveraged before. In this new framework, several issues get easy to understand measure and control, with new handles, as we illustrate.

We finally show that fixing oscillation of contrast opens many doors to a more controllable authoring of stochastic texturing. We explore some of the new reachable possibilities such as constrained noise content and bridges towards very different families of look such as cellular patterns, points-like distributions or reaction-diffusion.

Keywords: noise, Gabor, procedural texture, Fourier, signal processing

1 Introduction and motivations

Interesting texture aspects can be obtained by using various mathematical operators. Many procedural noises[1] or Fourier synthesis approaches [Anjyo 1988; Bracewell 1999; Saupe 1988; Voss 1988] are based on Power Spectrum characterization. They produce desirable base patterns for color texture, bump-map and height fields, with convenient handles and good performance, able to populate large fields with high resolution details.[2]

However, all these methods intrinsically come with a visual issue that has never be mentioned before – e.g., it is not reported in the rich survey [Lagae et al. 2010a] although the issue is clearly visible in its Fig 10.1, with differences among the methods. They produce spatial oscillations in the contrast, i.e., low frequency modulation of the envelope of the noise, which were never explicitly required or specified, and are especially visible if the power spectrum contains no low frequencies (e.g., bi-lobe or blue noise as in Figures 2,3,7). These variations may induce pleasant heterogeneities in the resulting texture, but they are currently totally out of any control: What if the user does not want them ? What if he wants different scales, patterns or intensity of contrast variations ?

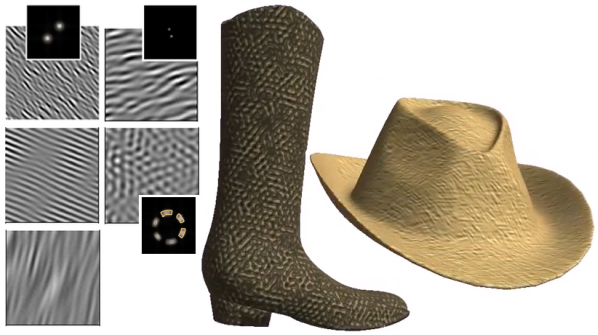

Figure 2: *Typical results from Figure 7 of [Lagae et al. 2009] show contrast oscillation* (left) *despite no low frequencies are present in the power spectrum* (snippets)*, and pleasant but totally uncontrolled heterogeneities in the resulting pattern* (middle and right).

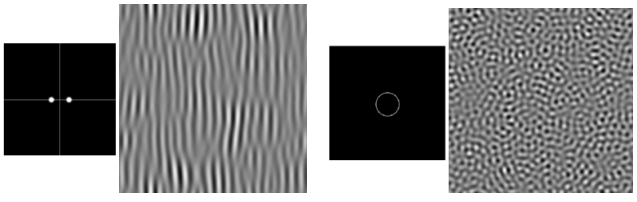

Figure 3: *Bi-lobe* (left) *and blue noise* (right)*. Despite no low frequencies are present in the PSD* (black snippets)*, the contrast of the generated texture varies, at low frequency.*

Another – directly related – ubiquitous problem of noises is that the dynamics of their value is impossible to control as desired: one has to choose between saturating the target range and suffering

[1] Gradient noise (Perlin, etc) [Perlin 1985; Perlin 2002; Kensler et al. 2008; Spjut et al. 2009], sparse [van Wijk 1991; Lewis 1984] or dense convolution noise (Gabor noise) [Lagae et al. 2009; Lagae et al. 2010b; Bénard et al. 2010; Galerne et al. 2012].

[2] Example based synthesis is another interesting approach, but it targets different families of images than stochastic textures and have different properties than noise algorithms. It is thus out of the scope of this paper.

clamping[3], or not clamping at the price of low contrast.
Worse: noise is usually not the end result; various transforms are applied to it before obtaining the final color, bump or displacement texture. But contrast oscillations drastically limit the use of non-linear noise shaping post-transformations – e.g., color lookup tables – since local extrema values are distributed widely and do not concentrate at range extremities. This makes uneasy for the artist the creation of homogeneous *features* from the crests and valleys of the signal (see Figure 1,b).
As mentioned in the survey [Lagae et al. 2010a], all of this illustrates how the current authoring workflow for noise textures is tedious and limited.

Moreover, the survey reminds that noise is expected to be "*a stationary* (and normal) *random process*", i.e., there is generally some *stationarity scale* L beyond which the generated texture must have stationary statistics[4]. Consequently, the presence of contrast and bounding oscillations means that these methods can severely fail[5] to fulfill the stationarity requirement. Or at least, that they don't provide any control on the *stationarity scale*. This indicates that those methods are *incomplete* in terms of texture generators.

In this paper we explain where these low frequency variations come from (Section 2) and how this can be fixed (Section 3). We propose a simple model which is compatible with procedural noise on 3D surfaces, as well as an extended model relying on the Fourier transform in image space.

Our approach opens even broader avenues beyond controlled heterogeneity of noise textures, which we explore through some experiments:

- We show in section 4.1 that our approach permits to bridge to patterns usually corresponding to very different classes of texturing algorithms, such as those based on simulation, vectors & graphs, dart throwing or grammars. This includes for instance the patterns of reaction-diffusion [Witkin and Kass 1991; Turk 1991] – which is costly to generate and difficult to control – , points-like distributions, or cellular patterns. Even if the full richness of the pure methods cannot be equaled, this allows to ease and unify authoring workflow and the (possibly space-varying) interpolation between these looks.
- We illustrate another generalization: how noise normalization can be used to reparametrize color lookup table so as to fit local extrema instead of absolute height in height fields.
- In Section 4.2 we show how normalization allows us to introduce content constraints in the generated noise via the influence of a target image or interactive painting over the noise features. The lack of content constraints has long been a classical authoring limit of noise algorithms.

2 Understanding contrast variations

Let's define the contrast on a given region as the RMS contrast [Wikipedia c], i.e. the standard deviation of the texture within the region. The contrast on a moving window of given scale gives the signal envelope around its mean value.

[3]Or rely on poor "smooth-clamp" compromises like sigmoid post-transform (e.g., $tanh()$).

[4]At least the canonical one. Explicit space-varying parameters or post-transform can of course be combined to it.

[5]Especially for pure Fourier synthesis and convolution approaches. Gradient noise (e.g., Perlin noise) and stochastic subdivision [Fournier et al. 1982; Lewis 1986; Lewis 1987] do contain local handles that can or could be used to influence local statistics – with limited accuracy.

Power spectrum (PSD) captures the overall aspect of many unstructured textures. But the unintuitive point we want to stress in this paper is that it does not yield stationarity when the user usually expect it would. Even if the spectrum has no low frequency (LF), the *contrast* does, which is strongly perceived (see Figures 1,2,3,7). This is the exact equivalent of the LF sound beat for close high frequency (HF) tones[6].

Indeed, there are two (related) problems providing interpretations, to which we add a new understanding:

2.1 Image PSD doesn't tell about *sub-windows* PSD

Just as for sound, the overall spectrum modulus of an image gives wrong intuitions about the local ones – the windowed spectrum, which is the one that matters for the textural appearance. The latter happens to do vary in terms of shape, energy and statistical moments: This is where stationarity requirement breaks (for window size $\geq L$). Very low frequencies would obviously change the average of local windows, but even when the spectrum has no wavelength larger than the window radius, all the frequencies that were harmonic in the large image and are no longer in the local window – e.g., $\sin(\frac{3}{2}2\pi\frac{x}{L})$ – are no longer explicitly representable in the local spectrum and will inject their energy through all the other frequencies (see Figure 5,left). Conversely, two windows with opposite phase for a same given wavelength would contribute as zero overall energy at this frequency in the overall spectrum, despite there is energy locally.

Conforming to a *stationarity scale* L thus requires to enforce constant (or similar) PSD in all windows of that scale, which an overall PSD cannot do[7]: this is indeed a constraint on *phases*. (Same for constraints like windowed bounds or histogram.)

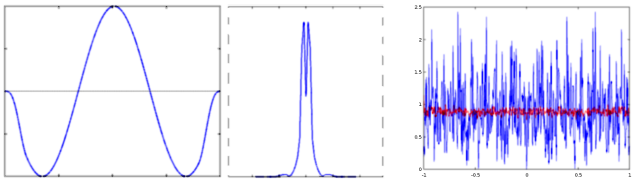

Figure 5: (Left): *Inharmonic sinus (windowed by a smooth kernel to ensure continuous-derivable wrapping) and its Fourier modulus.* (Right): *PSD of a realization of white noise by points distribution and Bernoulli's weights.* Red: *average of 100 realizations converges towards flat PSD.*

2.2 *Process* PSD doesn't tell about image PSD

The texture often results from a random process with an (implicit or explicit) PSD *in probability space*. But no realization's PSD ever reaches this ideal PSD, even for infinite size or resolution. E.g., Gabor noise convolves a kernel with white noise generated by a points distribution. But the modulus of the Fourier transform of a given points distribution (even an infinite one) cannot be flat. That is, no realization of white noise process is truly white.

Proof: Assuming binomial weights $W_j \in \{-1, 1\}$ at each of N points x_j, the sum $\sum_{j=1}^{N} W_j e^{-2i\pi f x_j}$ giving the Fourier coefficient for a given frequency f can be seen as an isotropic random walk in the complex 2D plane. Its average position is zero but its distance to origin – i.e., modulus – follows the Rayleigh distribution $dP_N(r) = \frac{2r}{N} e^{-\frac{r^2}{N}}$ [Hughes et al. 2005] and thus has

[6]Which can even create melodies out of the blue using pathological constructions, see [Wikipedia a].

[7]Apart via strict periodicity, by canceling out all but the $\frac{j}{L}$ frequencies.

Figure 4: Left: *Two procedural Gabor noises (bi-lobe strips and blue-noise spots) showing unspecified contrast variations (explicitly displayed in snippet), and its correction. Here, the variations even misleads relief perception via false shading. Note that this example is implemented using non-parametric real-time GPU Gabor noise, with contrast evaluated on mesh vertices in the vertex shader.* Middle: *The boot of [Lagae et al. 2009] (cf Figure 2) without the artifacts, resp., with controlled variance applied to the renormalized Gabor before the color LUT.* Right: *The hat of [Lagae et al. 2009] without and with correction. — Zoom or see supplemental for larger images.*

average $E(r) = \frac{\sqrt{\pi N}}{2}$, second moment $E(r^2) = N$ and standard deviation $\sqrt{N(1-\frac{\pi}{4})}$ for N unit steps. The values $PSD(f)$ – i.e., the squared modulus – through the spectrum of the realization can thus be seen as the realizations of a random variable along the "probability space" f. I.e., the PSD of realizations of white noise oscillates within the spectrum by the constant relative amount $\frac{stddev}{mean} = \sqrt{\frac{4}{\pi} - 1}$, and it thus not flat – see Figure 5, right. That's why the PSD of a random process[8] (which is flat for a white noise) is estimated from data by averaging many periodograms, which does converge (in red).
Similarly, the total energy $\int_f PSD(f)$ for a window has a standard deviation and thus varies through realized windows, which gives another interpretation of LF contrast variations.

2.3 A new tool, the *Spectrum of Variance*

We propose a more powerful and convenient way to characterize the contrast oscillations of a signal $s(x)$: the *Spectrum of Variance*, that we define as $\mathcal{F}((s-\bar{s})^2)$, where $\mathcal{F}()$ denotes the Fourier transform. This is a spectral decomposition of $(s-\bar{s})^2$ which integral is the total variance (not to be confused with the variance of spectrum $|\mathcal{F}(s)|^2$, i.e., the PSD, which integral also gives the total variance and is thus another decomposition). To lighten notations, in the following we assume a zero-mean signal and a unit integration length.

Let's illustrate this with the case of a Gaussian lobe (with random phases) spread around f in Fourier space – in 2D the corresponding spacial texture is a strip pattern (see Figure 3, left). Since spectra of real signals are symmetric, this is a bi-lobe noise. The spectrum of the squared signal is the auto-convolution of the signal spectrum: $\mathcal{F}(s^2) = \bar{\mathcal{F}}(s) * \mathcal{F}(s)$. As shown in Figure 6, a low-frequency lobe appears as the result of the convolving of the left and right $\mathcal{F}(s)$ lobes, while the two HF lobes are twice higher and larger than the original ones. Indeed, this spectrum is the multiscale decomposition of variance, thus our naming "Variance Spectrum":

- $\mathcal{F}(s^2)(0)$ is the total texture variance (on the integrated domain).
- Given a window scale L, the part of the spectrum within frequency range $[-\frac{1}{L}, \frac{1}{L}]$ corresponds to contrast oscillations larger than this window scale.

Proof: Let $w(x)$ be the windowing weighting function associated to a window centered around x_i – e.g., a Gaussian $\mathcal{G}(x_i, L)$. $w(x)$ is assumed to be smooth and L larger than the strips

[8]Note that for a white noise generated by binomial values on a regular grid, in practice the result is the same.

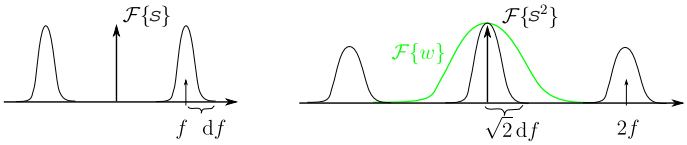

Figure 6: *Spectrum of a bi-lobe noise* (left) *and of its square* (right). *The windowed second moment of the signal corresponds to the filtering of the latter by* $\mathcal{F}(w)$ (in green)*: it results in a LF lobe in Fourier, thus corresponding to LF contrast variations in image space.*

wavelength $\frac{1}{f}$ (since by definition the expected stationarity scale is larger than the pattern elements). The windowed signal variance ${\sigma_L}^2(x_i)$ is the windowed second moment equal to $\int s(x)^2\, w(x)\, dx = (s^2 * w)(x_i)$. In Fourier space this gives $\mathcal{F}(s^2) \cdot \mathcal{F}(w)$, where $\mathcal{F}(w)$ is LF. This results in the LF lobe (or part of it, if $\frac{1}{L}$ is very small), which in image space characterizes the LF spatial variations of the windowed variance. Figure 9 shows in image space the full *(d)* and LF part *(e)* of the spectrum of variance of texture *(a)*. Figure 7,bottom illustrates the effect of different window sizes.

3 Control using the *Variance Spectrum*

3.1 Basic solution

The simplest way to remove contrast oscillation is to renormalize the signal by the contrast, i.e., the windowed standard deviation: $s'(x) = s(x)\frac{\sigma'}{\sqrt{{\sigma_L}^2(x)}}$ where σ' is the target standard deviation, e.g., $max(\sigma_L(x))$. If the chosen $\frac{1}{L}$ covers the full central lobe of $\mathcal{F}(s^2)$, one can verify that the normalization totally removes it[9], while quite preserving the spectrum $\mathcal{F}(f)$: see Figure 9,f.

Now, the user does have a choice: he might target a totally *homogeneous texture* with black or white patterns and gray only at transitions – e.g., strip and spots – , local extrema all being global extrema and texture dynamics being saturated in every waves. Or he might just want to control the size L over which stationarity is enforced and under which he allows complex gray-level variations. Totally removing the LF lobe would produce an homogeneous texture. To keep some local contrast variations, L should be chosen so that only the most central part of it is canceled out. Figure 7 illustrates the effect of L.

[9]Which means that a subtle phase conspiracy has been settled in the signal so that the Fourier module of the squared signal cancels out for all f in the central region.

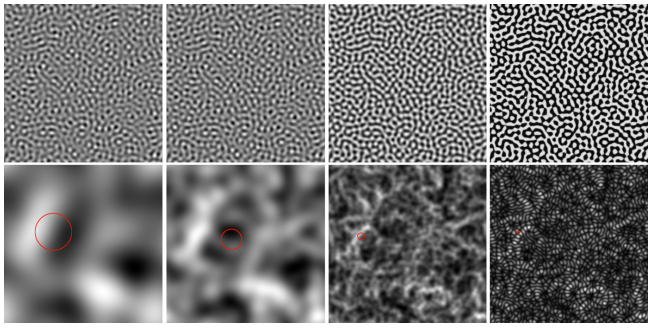

Figure 7: top: *Blue noise normalized using different stationarity scales L.* bottom: *corresponding normalization field, i.e., variance filtered at the target window size (figured in red). NB:* $[min, max]$ *is remapped to* $[0, 1]$. *The lower the frequency, the lower the amplitude of variations. On the third column the stationarity scale is the same as the wavelength of waves: each wave saturates the dynamics. On the fourth column the window scale is just a few pixels: the texture is forced to produce regions of min vs max value, with a sudden transition width equal to the stationarity scale.*

Note that the bi-lobe case is somewhat peculiar, with a totally separated LF lobe. In the blue noise case the variance spectrum shows a "HF circular lobe" partly mixed with the LF lobe (see Figure 8, top), so there is no "natural threshold" for total renormalization. (If approaching the blue noise ring by a set of lobes is acceptable, then we can obtain a well separate LF lobe; cf bottom row).

If local contrast variations are allowed on this way, the stationarity scale is controlled but not the other characteristics of the contrast distribution. If the user wants full control on the local contrast variations, the simplest solution is to renormalize totally, then to multiply the resulting homogeneous texture by an explicit contrast texture, as illustrated Figure 16,a and 4, middle.

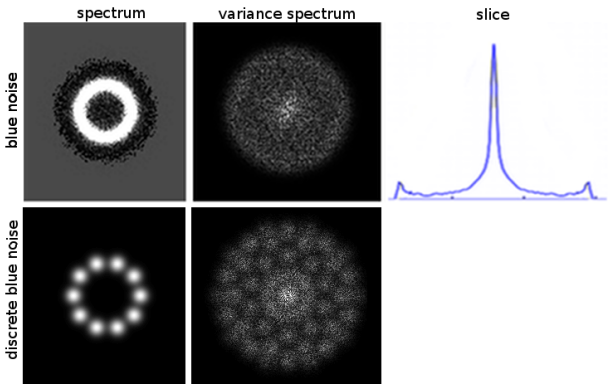

Figure 8: Top*: With blue noise spectrum* (left)*, the auto-convolution of the ring gives a central lobe not well separated to the "HF" ring in the variance spectrum* (middle and right)*. if "full renormalization" is aimed at, the exact stationarity scale to choose is arbitrary.* Bottom*: If the ring is approximated by a series of lobes, we recover a separate central lobe.*

Implementation:

This base correction is simple to insert in existing texturing workflows:

For applications allowing image processing in stored textures, Fourier tools can be used (cf first half of the pseudo-code in Figure 10):

1: The source texture s to normalize is computed[10];
2: Its spectrum of variance $\hat{\sigma^2}$ is explicitly calculated using $\mathcal{F}(s^2)$;
3: This spectrum is LF-filtered by multiplying it by the target windowing filter (e.g., box[11] or Gaussian in the spectral domain);
4: The signal is normalized by $\sigma_L(x) = \sqrt{\mathcal{F}^{-1}(LF \cdot \hat{\sigma^2})}$

For on-the-fly procedural noise on surface such as Perlin or Gabor, one first needs to evaluate the first and second statistical moments – accounting for the weight $w(x)$ – within the neighborhood of radius L so as to obtain the windowed standard deviation to be used for the normalization. Even if only a few dozen of samples are used for that, this multiplies the overall cost of the procedural texture by a huge amount. But since the resulting normalization value is LF, it can easily be optimized: the sum can be evaluated in a first pass at coarse density and stored in a temporary acceleration structure (in texture space, screen space, 3D space, or even surface space as simply as via values at vertices – assuming the mesh is dense enough).
In our mesh-based implementation with procedural Gabor noise on GPU (see Figure 4, left) we computed the filtered variance with 25 Gaussian weighted noise evaluations in the vertex shader (after ensuring the mesh tessellation was appropriate). In the fragment shader, the filtered variance was reconstructed by linear interpolation of the vertex values. Then, the pixel noise was evaluated and divided by the reconstructed standard deviation. The evaluation of the filtered variance at vertices plus renormalization was roughly the same cost than evaluation of the regular Gabor noise to be renormalized.

For texture on-surface painting applications, most of the above applies (i.e., space-wise operations instead of Fourier), without the demanding constraints of no-storage and high frame-rates.

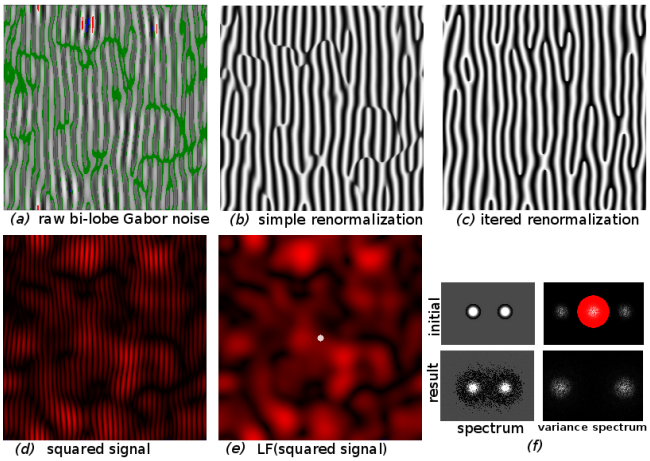

Figure 9: (a): *Raw bi-lobe Gabor noise. zero contrast locus are marked in green.* (b): *Simple normalization (showing locus of sudden inversion).* (c): *Our high quality scheme, iterating several normalization and spectrum reprofiling passes.* (d): *Squared signal.* (e): *Windowed variance (i.e., LF-filtered squared signal).* (f): *Signal spectrum (left) and variance spectrum (right) before (top) and after (bottom) normalization.*

[10] Here we assume it has zero-mean for simplicity.

[11] Note that using Fourier-box filter to threshold at mid-lobe yields some negative values in the spatial domain that must be clamped or `abs`ed before the evaluation of `sqrt`.

3.2 High quality solution

The simple corrective scheme above gives satisfying results in many cases, and is acceptable for real-time approximation in the worse cases.

In nasty cases like highly focused power spectra (bi-lobe Gabor happens to be such a nasty case) the windowed variance vanishes along extended 1D locus (in green in Figure 9,a). The resulting division by zero is easily avoided by normalizing with $\frac{\epsilon+\sigma'}{\epsilon+\sigma_L(x)}$ for some small ϵ constant, but this aspect of the problem is more numerical than analytical since both the numerator and denominator tend towards zero. The visual artifact rather lays in the very high slopes – i.e., sudden inversion – introduced in the pattern across this locus. It is perceptually worsen by the high LF coherency of the locus showing up as large parasitic features in the nasty cases, see Figure 9,b. Large ϵ limits this artifact at the price of less accurate canceling of contrast oscillation.

No normalization scheme can cancel out the effect of these intrinsic locus. Solving the problem requires to displace or erase these locus. This is a phase issue in the variance field, which cannot be fixed without acting on the ones of the signal. For this we propose an iterative scheme alternating the contrast normalization with a power spectrum *reprofiling*, the latter consisting in clamping frequencies that leaked out of the initial power spectrum footprint so as to remove parasitic high slopes. We have found out that in worst cases, 5 to 10 iterations were sufficient to reach high quality, as seen in Figure 9,c. We suspect that this efficiency of the phases rearrangement is probably related to the similarity of our iterative method with the Gerchberg–Saxton family of algorithms [Oppenheim and Lim 1981; Wikipedia b; Fienup 1978].

Implementation:

For applications where the Fourier transform can be used (explicit storage + operations in texture space + no space-varying texture tuning), operations are done easily in Fourier space – filtering is just multiplying by a mask – and converting back and forth with image space. We show the pseudocode in Fig.10. It implements the Fourier approach, but in our examples we have implemented both algorithms.

For other applicative contexts, we must rely on on-surface local operations for local variance evaluation, low-pass filtering and normalization. A random process like Gabor noise is the typical tool to rely on for Fourier-style generation on 3D surface without explicit use of the Fourier transform (which is not even defined in this case). Windowing operations such as filtering the multiscale variance – i.e., integrating $\omega(x) * s(x)^2$ – and other filtering operations are done by explicit convolution with a kernel.
Reprofiling the spectrum of the normalized noise requires storing the intermediate results and is likely to require a larger and thus costlier kernel with denser evaluation. It is thus probably not compatible with run-time procedural applications, as for the kernels with HF parts that will be used in the next section. But most is still affordable in the scope of softwares for texture painting on surfaces.

4 Extending the expressive space of noise textures

4.1 Extended use of normalized noise

Accurate post-transform of noise: Beyond the new reachable appearances of base noise (see Figure 4), control of the stationarity

```
im = real(ifft2(Kernel.*exp(2*I*pi*rand))); % Gabor noise
imf_footprint  = (Kernel>0);                % template for spectrum reprofiling
% imf_footprint = Gauss(0,KernelRadius);    % smooth variant
minC = 1e-2;                                % to avoid div0 when normalizing

for i = 1:N  % --- iterations (for Quality mode)

    % --- renormalization (for both modes) ----------------------
    immean = mean(im(:));
    imE = (im-immean).^2;
    imEf = fft2(imE);             % variance spectrum
    imEf .*= LF_filter;           % keep only central lobe of variance spectrum
    imE = real(ifft2(imEf));      % envelope (i.e. LF) of contrast variations
    imE = abs(imE);               % get rid of (rare & small) negative values

    imC = sqrt(imE);              % variance -> contrast ( = std-dev )
    imC /= max(imC(:));           % normalized contrast: max = 1 -> keep untouched

    imK = (minC+1)/(minC+imC);    % renormalization factor
    im = immean  + (im-immean).*imK;
  % im = newmean + (im-immean).*imK * newStd/max(imC(:)); % variant

    % --- spectrum reprofiling (for Quality mode) ---------------
    immean = mean(im(:));
    imf = fft2(im-immean);
    imf .*= imf_footprint;        % clamping spectrum to its initial footprint
    im = real(ifft2(imf));        % to smooth-out high slopes at degenerated locus
    im = .5+.5*im/max(abs(im)(:));% normalize image to [0,1]

end
```

Figure 10: *Octave/Matlab pseudo-code for quality normalization, relying on the FFT.*

scale guarantees the sharp bounding of values, i.e., the saturation of value range in each local window. This allows for an accurate mapping of noise to color through look-up tables (see Figure 1,a&b without/with normalization), and more generally, the best use of non-linear post-transforms to let the artist finely shape the resulting texture. An extreme example is the generation of points-like distributions by thresholding renormalized blue noise – see Figure 11 –, which can be used as a basis for spotty, cellular, or stipple and ink patterns.

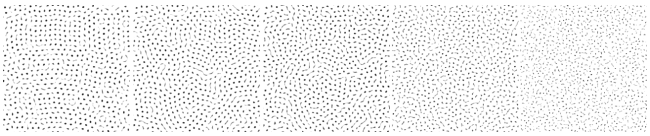

Figure 11: *Points-like distributions obtained with a LUT selecting the peaks of normalized blue noise. The better the normalization the smaller can be the spots (with no miss or excessive deformation).*

Indexing height-fields local extrema: In the case of height fields, the same base noise is often used to control the relief height and the color so as to have color correlated to the relief. But what if the user wants consistently colored peaks and valleys, i.e., relative to local extrema instead of absolute height ? It is now possible to do so by using the renormalized version only to reparametrize and saturate the index to the LUT, letting the heights unnormalized – cf Figure 1,f: all peaks are made white, whatever their height.

Reaction-Diffusion look: Homogeneous patterns, i.e., signals such that each local fluctuation saturates the extrema (local extrema are also global extrema), typically occur for strip patterns on sand, that can be found at centimetric (e.g., wet beach and sea floor) as well as hectometric (e.g., dunes) scale.
Indeed there is a whole family of patterns based on this constraint: spots, strips, and segments found on zebras, cheetahs, fishes and shells, corresponding to *reaction-diffusion* patterns [Witkin and Kass 1991; Turk 1991]. Reaction-diffusion algorithms are very costly – it's a time-varying PDE to be simulated numerically – and the look is difficult to control by the user. With our approach, total renormalization on noise produces homogeneous patterns: e.g., blue noise gives spots and bi-lobe Gabor noise gives strips, with

all possible intermediaries (see Figure 12).
The full richness of the true method cannot be equaled, but the simple cases can be generated and authored much more easily and efficiently. Moreover, unification allows interpolation between the different looks presented in this paper.

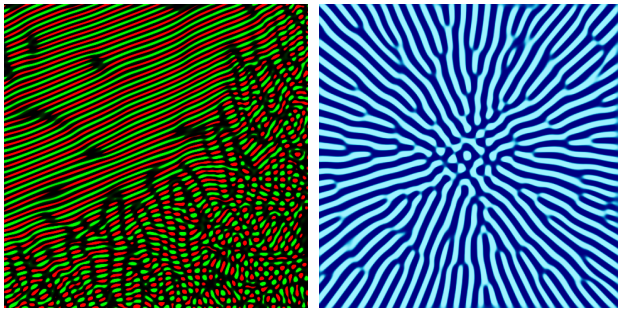

Figure 12: *Reaction-diffusion looking pattern obtained by applying a color look-up table on a totally normalized (stationarity scale = pattern size) spatially varying Gabor noise. In these figures, no spectrum reprofiling was done.*

Constrained noise: target image and interactive editing. In many situations the artist would like to constrain the noise at some locations (e.g., wood knots), or to fit boundary conditions (e.g., to connect the pattern to geometric features). Also, he might be unsatisfied with the location or shape of some generated details and be willing to modify the result without breaking the local look of the noise.

Indeed, our normalization scheme can apply to any signal, including non-noise algorithm, and even source images or drawings – as long as there is no zero-contrast on large sub-windows. Linearly mixing a target image to a base noise ensures this, and enforces the resulting normalized noise to conform to the features imposed by the drawing. The lerp parameter allows to control how separated vs merged the drawing is within the noise – see Figure 13. Similarly, we can let the user draw interactively on top of the noise during the normalization-reprofiling iterations and mix his input with the temporary renormalized noise, thus interactively influencing the location and shape of given noise features.

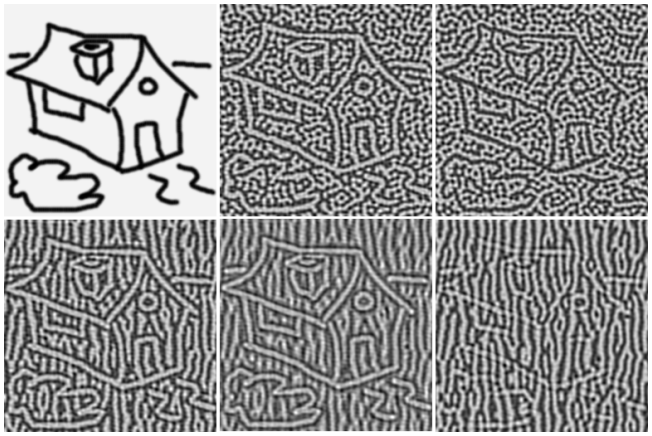

Figure 13: *Normalization of* `lerp`*(drawing,noise,w), with different noises and weights w. The drawing can be either an image or interactive. Here the drawing was given at once, but similar results are obtained by interactively painting it over the generated noise during the iterations loop.*

4.2 Going deeper in Variance Spectrum control

(a) src noise. Del: (b) contrast LF (c) LF+interfer. (d) LF+lobes

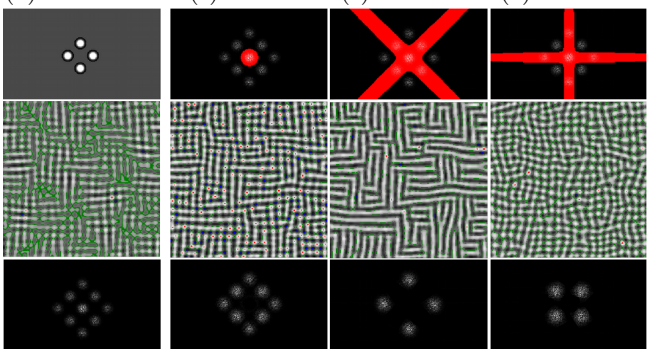

Figure 14: *The variance spectrum of a quadri-lobe Gabor noise shows 9 lobes: 4 correspond to the initial lobes, 4 to interferences between lobes, plus the LF lobe. Besides the simple filtering-out of the central lobe (b), we can choose to remove interferences (c), or to keep only them (d).* a, top and bottom*: initial spectrum and variance spectrum.* b,c,d top and bottom*: variance spectrum before and after selected normalization. In red: the filter selecting the features collected for renormalizing the signal.*

Up to now we simply used the spectrum of variance to delimit and filter out its central lobe. But there is a lot more we can see and do with this tool: E.g., for noise which power spectra has multiple lobes $l_i(f)$, variance spectrum shows a full grid of HF lobes $l_{i,j}(f)$ corresponding to "tensorial" convolution-product $l_i * l_j$. For $i \neq j$ they correspond to beats or interferences between two characteristic frequencies of the noise.

We can filter out at will some or all of these inter-lobe beats the same way as we did for the LF lobe; or to the contrary, we might keep only the beats and filter out the "pure" lobes ${l_i}^{*2}$. For this, we just have to set accordingly the shape of the filter acting on the spectrum of variance, selecting what should be normalized out[12] or preserved.

In the first case (see Figure 14,c), the resulting texture shows "cellular" regions with clear dominance of one single frequency – i.e., direction, for the bi-directional quadrilobe example – at a time : lobes interferences are canceled out. In the second case (see Figure 14,d) we obtain a spots pattern which clearly exhibits the tensorial product of frequencies, but all directional dominance is avoided. In Figure 15 we show the same operations on aligned quadri-lobes (i.e., different scales instead of directions).

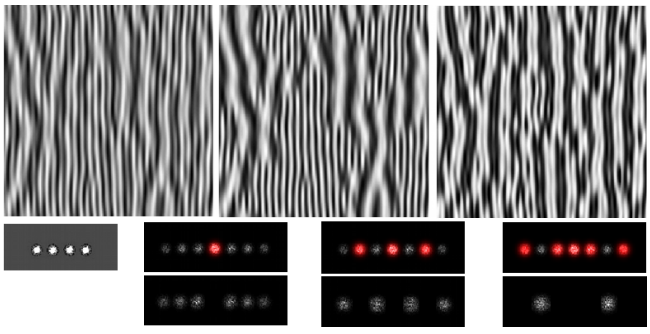

Figure 15: *Same as Fig. 14 with aligned quadri-lobes.*

[12] I.e., we still compute the signal normalization factor as $\sqrt{\mathcal{F}^{-1}(filter \cdot \mathcal{F}((s-\bar{s})^2))}$ – or the equivalent operation using signal-space filters.

The LF pattern of dominance randomly results from the relative weight of the LF contrast variation of each lobe (they do vary, since only their sum is normalized). If the artist want to explicitly control the pattern of dominance (as we did earlier for the pattern of contrast), we can separately normalize the signal corresponding to each lobe, mix their result[13] with lerp tuned by an artist-designed dominance map, then renormalize the result, see Figure 16,b&c.

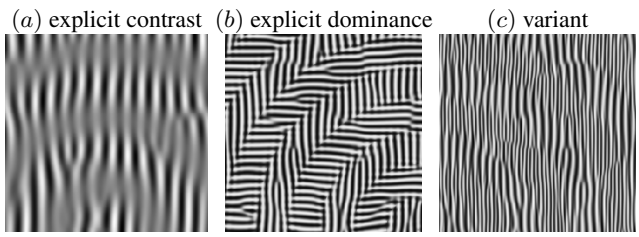

Figure 16: (a): *Explicit contrast control by multiplying totally renormalized signal by a target contrast map.* (b, c): *Explicit control of the lobe preference by separate pre-normalizing, lerping according to a dominance map, and re-normalizing. PSD = orthogonal vs aligned quadri-lobes.*

Note that no matter how different they are, these cellular regions always connect to each other as a continuous and well-contrasted pattern, which would not be possibly with a single mask layer selecting between two texture patterns.

When the lobes correspond to different scales rather than different directions, their perceptual role might be different: The user may mean that features of different scales piles up hierarchically, each normalized separately. In such a case we just have to split the noise into different channels as we did above, normalize them, then recombine them (see Figure 17).

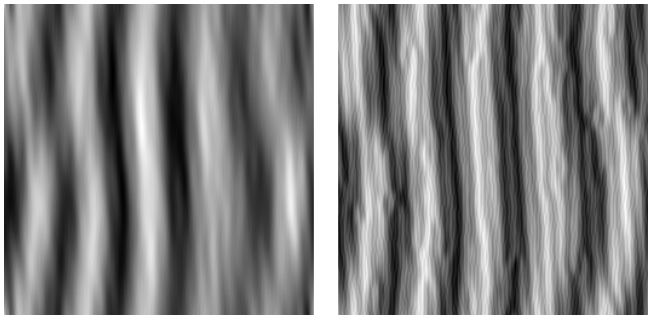

Figure 17: *For multiscale lobes (here, aligned hexa-lobes), we can normalize bi-lobe layers separately* (right) *rather than together* (left)*, to preserve the perceptual hierarchy.*

5 Conclusion and future work

We have introduced the *spectrum of variance*, a new powerful handle for understanding and controlling a previously undescribed aspect of power spectrum based stochastic textures, the *oscillations of contrast*. Thanks to it, we can now ensure *stationarity* in noise and Fourier textures, at controlled scale, which was not possible before. We have sketched some of the various possible applications and noise authoring modalities allowed by this new accuracy, extending the expressive space of texture synthesis. Each of these directions would deserve further developments – as does noise authoring in general itself –, but this is out of the scope of this paper.
In terms of implementation, we have illustrated both Fourier-based operators allowing the most complex results, and spatial operators compatible with real-time non-parametric procedural noise on curved surfaces.

Beyond pure synthesis, with this new mathematical tool we could also measure the contrast properties in stochastic examples image and reproduce them in noise synthesis, a dimension that Fourier resynthesis [Anjyo 1988; Bracewell 1999; Saupe 1988; Voss 1988] and "Gabor by example" [Ghazanfarpour and Dischler 1995; Ghazanfarpour and Dischler 1996; Galerne et al. 2012; Gilet et al. 2014] papers were all blind about.
Similarly, the various variants of Perlin noise have quite different contrast properties as visible in Fig 10.1 of the survey [Lagae et al. 2010a]. It would be interesting to analyze this more precisely.

More fundamentally, as we have illustrated in this paper, a key challenge for better texture synthesis workflows is to couple controls in several spaces which don't talk well to each others: Here we coupled power spectrum and windowed variance, but we needed iterations for accurate enforcement of both. Could this be done more directly ? Would it be possible to control not only the local range but also the histogram of values ? Could we control the slopes as well ?
Also, we have also illustrated that it is easy to get miss-leaded in specifications of target properties: windowed-spectrum differs from image-spectrum and process-spectrum. Strips patterns, as well as all stationary textures realizations, are not pure power spectra : the least constraint on local dynamics is indeed a phase property. Raw noise is not the final texture (so studies like anti-aliasing should include the non-linear post-transforms). What about color ? LUT-transform of grey value signals is very constraining (e.g., since deriving from a potential no color cycles can appear in the resulting image) while independent color channels miss colored structures (optimizing the color space [Galerne et al. 2012] is a slight improvement here). Would it be possible to directly generate a color noise ?

Noise-based textures offer more control and robustness than example-based resynthesis approaches, but they need to be enriched in many directions to better cover the space of desirable look. We hope to have shown that there is still a lot to be explored !

[13] We can also directly normalize the lerp of the un-normalized lobes to let some random variations in the pattern.
E.g., `sourceSignal = lerp(`$Gabor_1$`,`$Gabor_2$`,map)`.

References

ANJYO, K.-I. 1988. A Simple Spectral Approach to Stochastic Modelling for Natural Objects. In *EuroGraphics*.

BÉNARD, P., LAGAE, A., VANGORP, P., LEFEBVRE, S., DRETTAKIS, G., AND THOLLOT, J. 2010. NPR Gabor Noise for Coherent Stylization. In *ACM SIGGRAPH 2010 Talks*, 40.

BRACEWELL, R. N. 1999. *Fourier Transform and its Applications*. McGraw-Hill.

FIENUP, J. R. 1978. Reconstruction of an object from the modulus of its fourier transform. *Opt. Lett. 3*, 1 (Jul), 27–29.

FOURNIER, A., FUSSELL, D., AND CARPENTER, L. 1982. Computer rendering of stochastic models. *Commun. ACM 25*, 6 (June), 371–384.

GALERNE, B., LAGAE, A., LEFEBVRE, S., AND DRETTAKIS, G. 2012. Gabor noise by example. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 31*, 4 (July), 73:1–73:9.

GHAZANFARPOUR, D., AND DISCHLER, J.-M. 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*.

GILET, G., SAUVAGE, B., VANHOEY, K., DISCHLER, J., AND GHAZANFARPOUR, D. 2014. Local random-phase noise for procedural texturing. vol. 33 of *ACM Siggraph Asia 2014 Papers*, 195:1–195–11.

HUGHES, B. L., RYCROFT, C. H., AND BAZANT, M. Z. S., 2005. Introduction to random walks and diffusion (lecture 1). http://ocw.mit.edu/courses/mathematics/18-366-random-walks-and-diffusion-fall-2006/lecture-notes/. Department of Mathematics, MIT.

KENSLER, A., KNOLL, A., AND SHIRLEY, P. 2008. Better gradient noise. Tech. rep., Scientific Computing and Imaging Institute, University of Utah.

LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. 2009. Procedural noise using sparse gabor convolution. *ACM Trans. Graph., Siggraph'09 Proceedings 28*, 3 (July).

LAGAE, A., LEFEBVRE, S., COOK, R., DEROSE, T., DRETTAKIS, G., EBERT, D., LEWIS, J., PERLIN, K., AND ZWICKER, M. 2010. State of the art in procedural noise functions. In *EG 2010 - State of the Art Reports*.

LAGAE, A., LEFEBVRE, S., AND DUTRÉ, P. 2010. Improving gabor noise. *IEEE Transactions on Visualization and Computer Graphics*.

LEWIS, J.-P. 1984. Texture synthesis for digital painting. SIGGRAPH '84, 245–252.

LEWIS, J. P. 1986. Methods for stochastic spectral synthesis. In *Proceedings of Graphics Interface '86*, M. Green, Ed., 173–179.

LEWIS, J. P. 1987. Generalized stochastic subdivision. *ACM Transactions on Graphics 6*, 3, 167–190.

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.

PERLIN, K. 2002. Improving noise. *ACM Trans. Graph., Siggraph'02 Proceedings 21*, 3 (July), 681–682.

SAUPE, D. 1988. *The Science of Fractal Images*. Springer-Verlag New York, Inc., ch. Algorithms for random fractals, 71–113.

SPJUT, J. B., KENSLER, A. E., AND BRUNVAND, E. L. 2009. Hardware-accelerated gradient noise for graphics. In *Proceedings of the 19th ACM Great Lakes Symposium on VLSI*, 457–462.

TURK, G. 1991. Generating textures for arbitrary surfaces using reaction-diffusion. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol. 25, 289–298.

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

VOSS, R. 1988. *The Science of Fractal Images*. Springer-Verlag New York, Inc., ch. Fractals in nature: from characterization to simulation, 21—70.

WIKIPEDIA. Constant spectrum melody. https://en.wikipedia.org/wiki/Constant_spectrum_melody.

WIKIPEDIA. Gerchberg–saxton algorithm. https://en.wikipedia.org/wiki/Gerchberg%E2%80%93Saxton_algorithm.

WIKIPEDIA. Rms contrast. https://en.wikipedia.org/wiki/Contrast_(vision)#RMS_contrast.

WITKIN, A., AND KASS, M. 1991. Reaction-diffusion textures. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, vol. 25, 299–308.