Additional material to "Local Random-Phase Noise for Procedural Texturing"

G. Gilet, B. Sauvage, K. Vanhoey, J.-M. Dischler, D. Ghazanfarpour

paper published at SIGGRAPH ASIA 2014

Contents

1 Spectrum, PSD, and energy calculations

We are interested in computing the spectrum, the PSD, and the energy of LRP noise defined by

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

Since the phases are random the local noises are statistically independent, so we study one single local noise. Moreover the spatial shift $(x - x_i)$ does impact neither the PSD nor the global energy, so we center the local noise at $x_i = 0$. Thus we are interested in the simplified noise signal

$$s(x) = w\left(\frac{\|x\|}{\Delta}\right) \sum_{j=1}^{J} A_j \cos(2\pi f_j \cdot x + \varphi_j) = W\left(\frac{x}{\Delta}\right) \sum_{j=1}^{J} A_j \cos(2\pi f_j \cdot x + \varphi_j) \tag{2}$$

where W is the 2D radial window defined by $W(x) = w(\|x\|)$.

1.1 Spectrum of s

The spectrum can be derived from equation (2) using standard Fourier formulas:

$$\widehat{s}(f) = \frac{\Delta^2}{2} \sum_{j=1}^{J} A_j e^{i2\pi\varphi_j} \widehat{W}\left(\Delta(f - f_j)\right) + A_j e^{-i2\pi\varphi_j} \widehat{W}\left(\Delta(f + f_j)\right) \tag{3}$$

1.2 PSD of s

To approximate the PSD $|\widehat{s}(f)|^2$ we assume that the windows $\widehat{W}$ in equation (3) have disjoint support, such that one can bring the modulus and the square inside the sum. This is only an approximation which holds in our case because: $\widehat{W}$ decays rapidly; the samples f_j are quite evenly distributed; the sampling is sparse; Δ is set in inverse proportion to the sampling sparsity $1/\sqrt{J}$. So the PSD can be approximated by

$$|\widehat{s}(f)|^2 \approx \frac{\Delta^4}{4} \sum_{j=1}^{J} A_j^2 \left|\widehat{W}\left(\Delta(f - f_j)\right)\right|^2 + A_j^2 \left|\widehat{W}\left(\Delta(f + f_j)\right)\right|^2 \tag{4}$$

Thus, regardless of the symmetry around $f = 0$, in the paper's section 4 "Noise by example" we state the problem as an approximation by a weighted sum of $\widehat{W}(\Delta f)$ centered at f_j.

1.3 Energy of s

The energy

$$E(s) = \iint |s(x)|^2 dx = \iint |\widehat{s}(f)|^2 df \tag{5}$$

of the noise can be approximated from equation (4) using integration in polar coordinates:

$$E(s) \approx \Delta^2 \pi \left(\sum_{j=1}^{J} A_j^2\right) \int_0^\infty r w^2(r) dr \tag{6}$$

By targeting a constant value for the magnitude A_j on gets the formula of the paper's section 4.3.

2 Color

All our examples show color whereas our formula operate on scalar values. Synthesizing texture with noise in RGB space is tedious: unseen colors can easily arise when channels are treated independently. The solution usually consist 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 [1]. It works well when the input texture exhibits a few dominant colors and it avoids histogram equalization.

First a "principal variation color space" is computed as explained in [1]. It results in a few dominant colors D_i and principal variations V_i representing clusters of pixels in RGB space. Each pixel p is represented by an index $i(p)$ and a coordinate $v(p)$ such that its color is approximated by $D_{i(p)}+v(p)V_{i(p)}$.

Then the indices i are sorted by increasing luminance of D_i and mapped onto $[-1;1]$ by a piecewise linear function I. So a value $I(i(p))$ is associated to each pixel p.

All algorithms are applied independently on $v(p)$ and $I(i(p))$ to get $v(x)$ and $I(i(x))$ at any position x. The index is recovered as $i(x) = \left[I^{-1}(I(i(x))\right]$. The final color is given by $D_{i(x)} + v(x)V_{i(x)}$.

Note that independent applications on $v(p)$ and $I(i(p))$ actually requires $2J$ cosine evaluations (instead of J) for each local noise. In practice, we lowered the number of evaluations for $v(p)$ to $J/2$. This renders satisfying results because v requires less precision. Thus when J cosines are mentioned in the paper, color textures actually do $1.5J$ evaluations. All examples in the paper use exactly five dominant colors.

References

[1] Kenneth Vanhoey, Basile Sauvage, Frédéric Larue, and Jean-Michel Dischler. On-the-fly multi-scale infinite texturing from example. *ACM Trans. Graph.*, 32(6):208:1–208:10, November 2013. 3

3 Texturing by example recap

The input image is provided. The cosine budget J and the proportion of structure r are defined by the user. Then texturing by example proceeds as follows. First a set of pre-processings occur on the CPU:

Compute the input PSD using Welch's method as explained in the paper's section 4.4.

Define A and φ by choosing any of the blocks used in Welch's method.

Compute the region R that contains the structure by iteratively adding the highest-amplitude frequencies as explained in the paper's section 5.1.

Distribute the sampling budget as $J_R = rJ$ for the structure and $(1-r)J$ for the random-phase noise.

Pre-process the random-phase noise on the PSD as explained in the paper's section 4.4 (items "Stratification" and "Sampling pre-processings" with $J_S = (1-r)J/L$).

Pre-process the structure as explained in the paper's section 5.3. It results in a subdivision into square blocks and a corresponding Δ_R. To each block are associated J_R highest-amplitude frequencies f with corresponding $A(f)$ and $\varphi(f)$.

Then the following data are transfered to the GPU:

- For the random-phase noise: the values of J_S, Δ_S and A_S, as well as the tables of sub-strata frequencies, as explained in the paper's section 4.4.
- For the structure: the value of J_R and Δ_R, the number or size of blocks, and for each block the J_R selected triplets $(f, A(f), \varphi(f))$.

Finally for every position x the value $n(x)$ is computed as follows:

Sampling for random-phase noise as explained in the paper's section 4.4 (item "Sampling"): a single sample $f_{i,j}$ is uniformly drawn per sub-stratum. The amplitude $A_{i,j}$ is given by A_S. Phase $\varphi_{i,j}$ is random in $[0; 2\pi]$.

Evaluation of random-phase noise. The noise $\sum_S n_S(x)$ is evaluated at x by applying formula (1) for each stratum S.

Random shifting. If periodicity breaking is required, replace the position by

$$x \leftarrow x + \frac{1}{2} t\left(\lfloor 2x \rfloor\right)$$

where t is a hash-code applied on both coordinates of x, as explained in the paper's section 5.2.

Turbulence. If repetition breaking is required, set σ to the typical size of a feature in the texture and apply the turbulence on the position:

$$x \leftarrow x + \sigma n_T(x)$$

as explained in the paper's section 5.2.

Evaluation of structure. Find the block B in which x lies. Then evaluate the structure by

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

where the inside sum runs over the J_R frequencies f selected for this block B, as explained in the paper's section 5.3.

Compute the final noise value at x by summation of the structure and the random phase noise.

4 Results

4.1 Texture by example

Figure 1 shows texture by example synthesis, respectively (from left to right): the input, n_R, n_R + random placement, and finally n. The values for J and r are provided for each example.

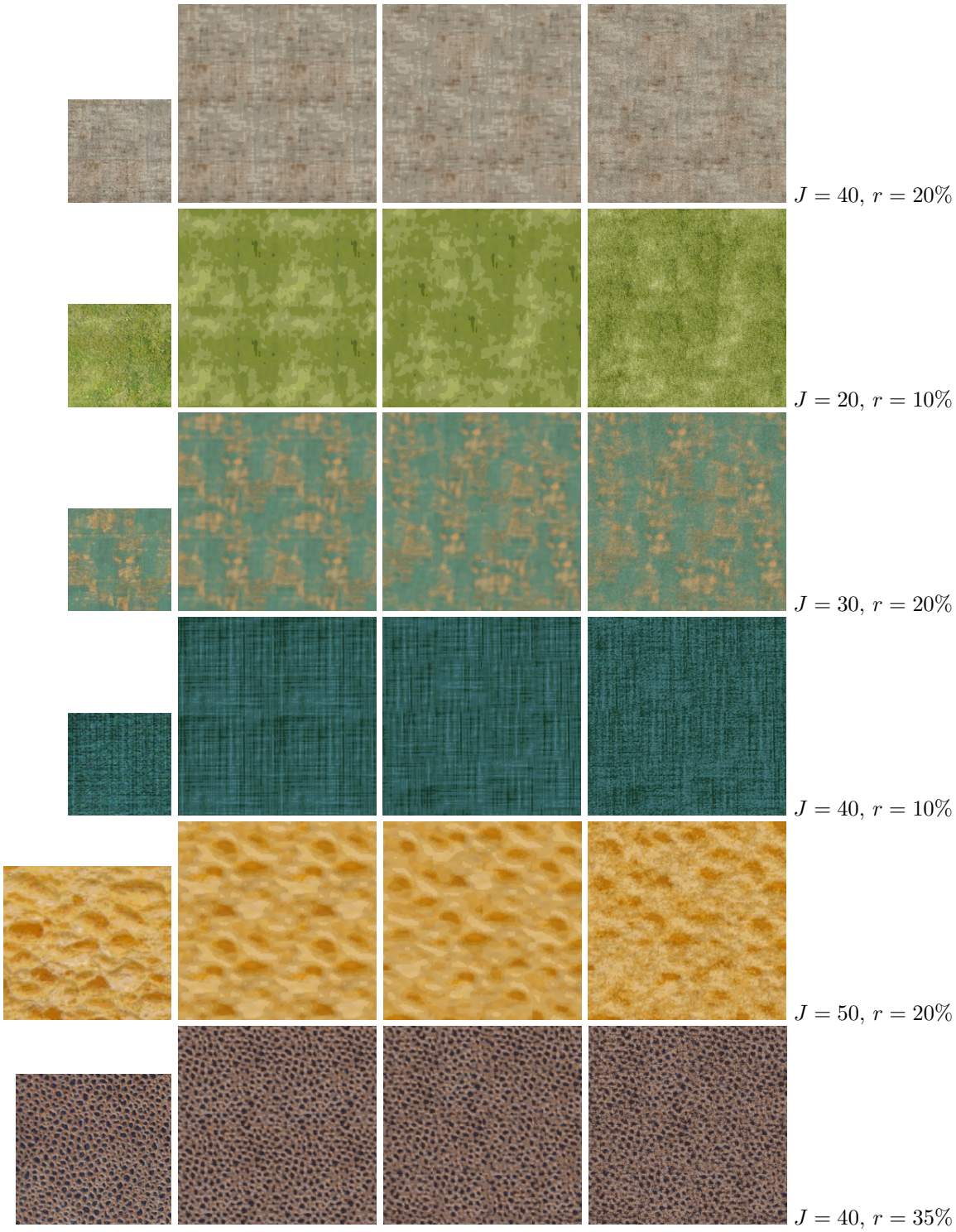


Figure 1: Texture by example.

4.2 Textured 3D object

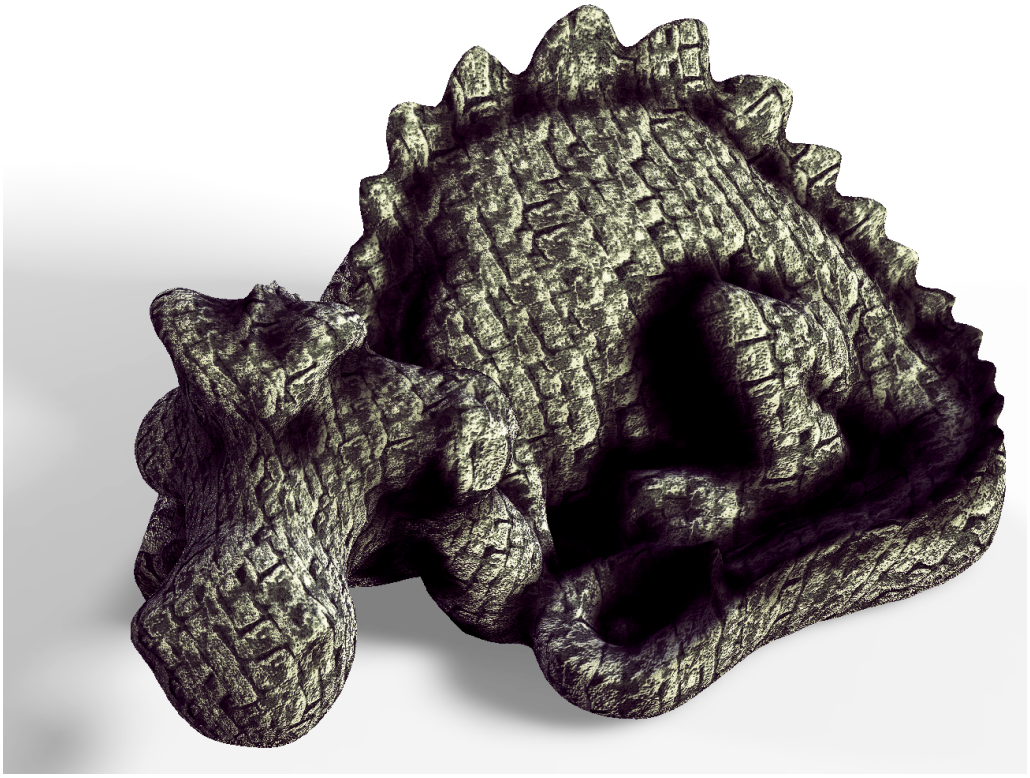


Figure 2: Example-based texturing and bump mapping of a 3D object ($J = 40$, $r = 35\%$). Rendering speed for this 1200^2 framebuffer is $58fps$ with texturing ($J = 50$ and $r = 30\%$), $22fps$ with bump mapping.

5 Parameter tuning

In this section, we show the influence of several parameters. Two of them are tunable by the user: the cosine budget J and r, the proportion of structure to be preserved. The other parameters are fixed in our framework. We show them here to illustrate their influence on the result.

5.1 Cosine budget J

Five figures (numbers 3 to 7) illustrate the influence of the cosine budget that we grow from $J = 30$ up to $J = 60$. All examples are computed with 4 strata.

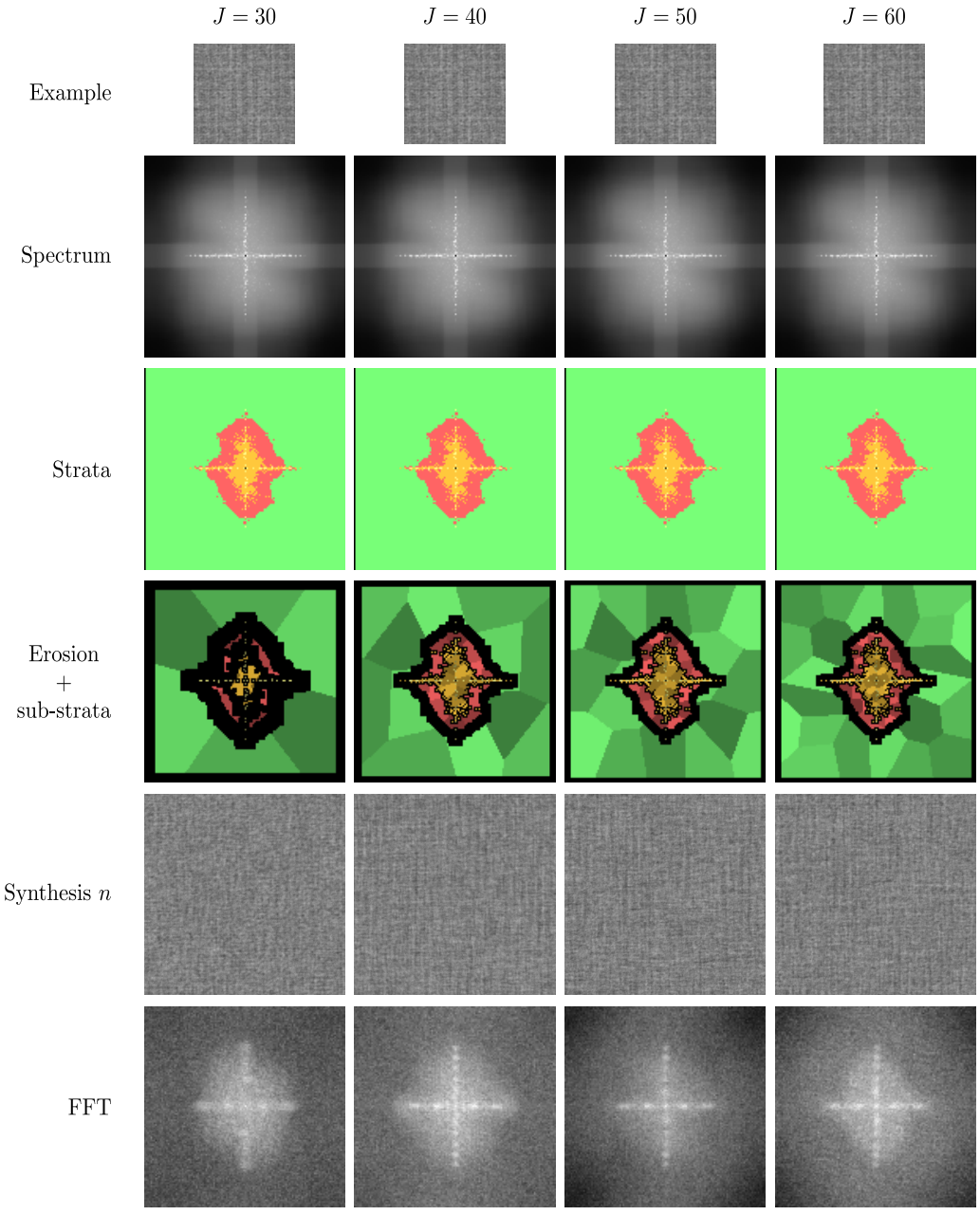


Figure 3: Variable cosine budgets for the "Fabric" texture.

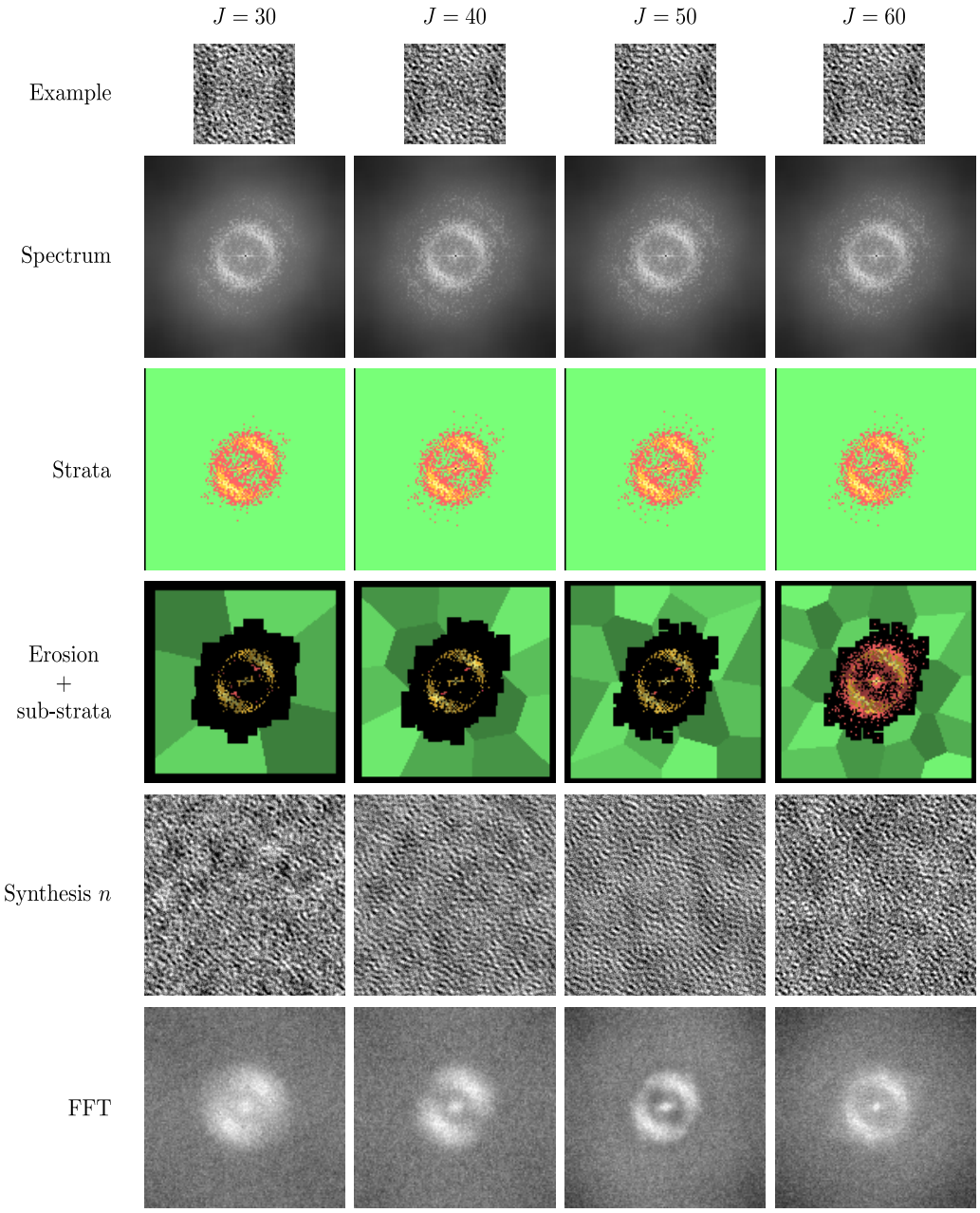


Figure 4: Variable cosine budgets for the “Coral” texture.

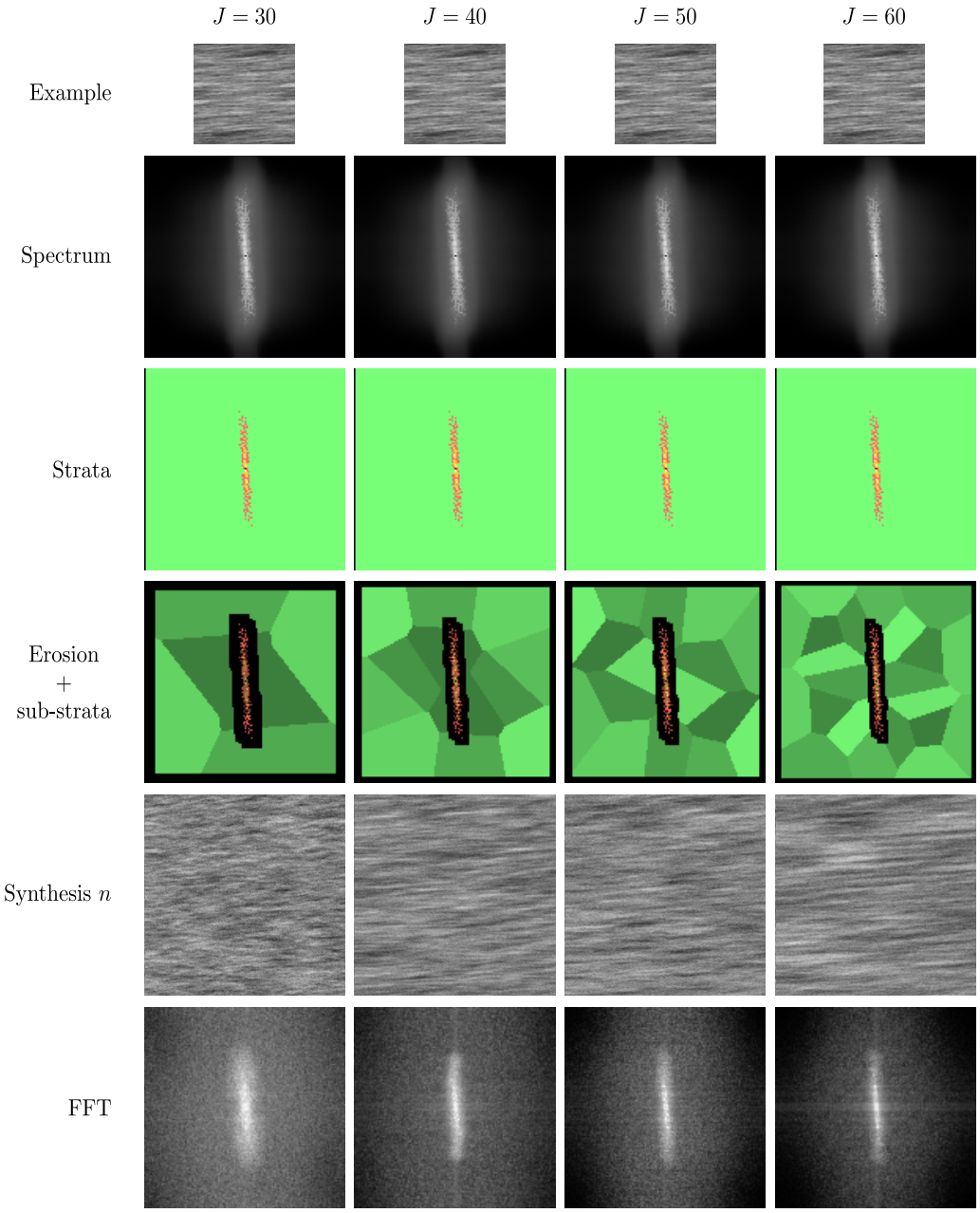


Figure 5: Variable cosine budgets for the “Ocean” texture.

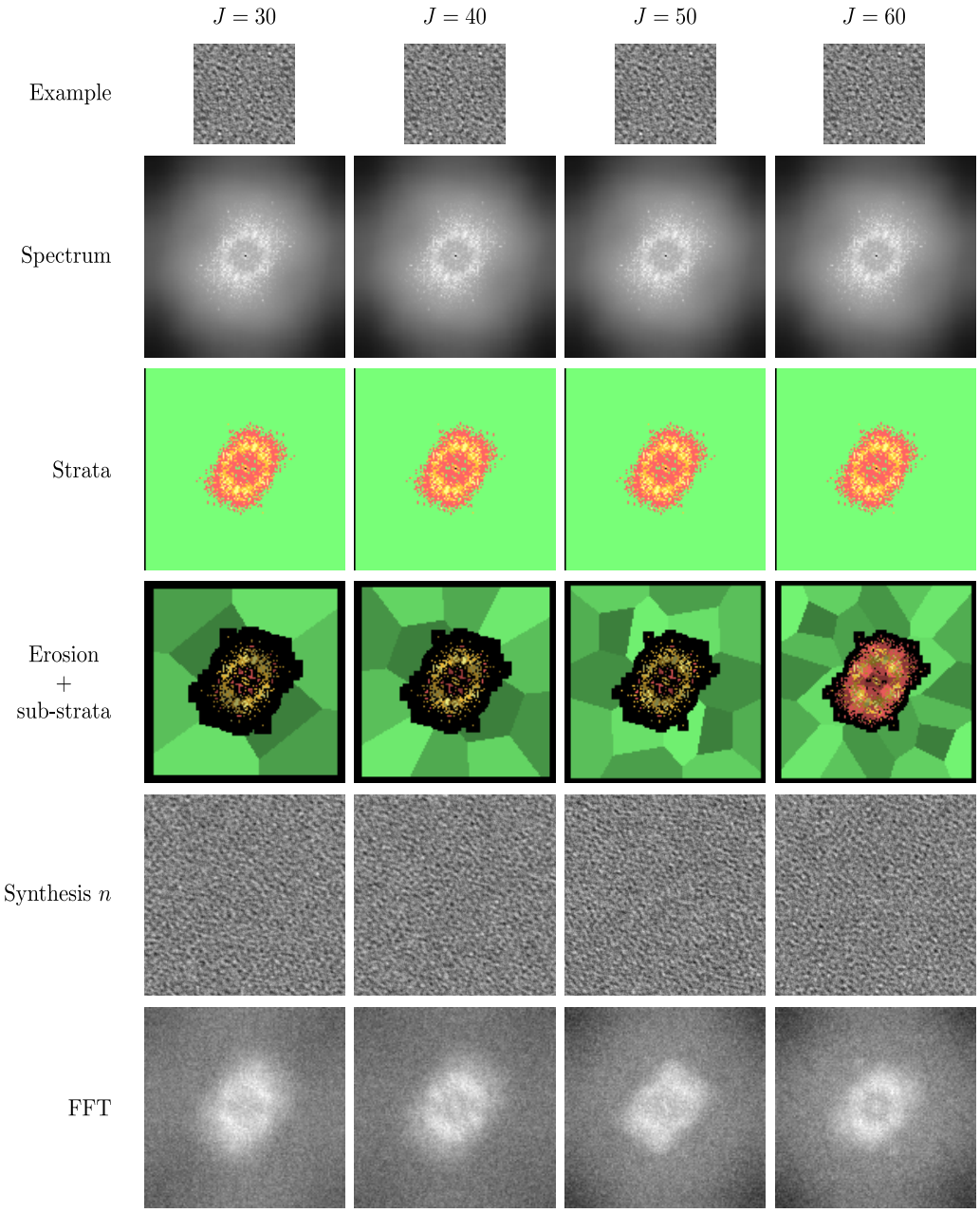


Figure 6: Variable cosine budgets for the “Peaua” texture.

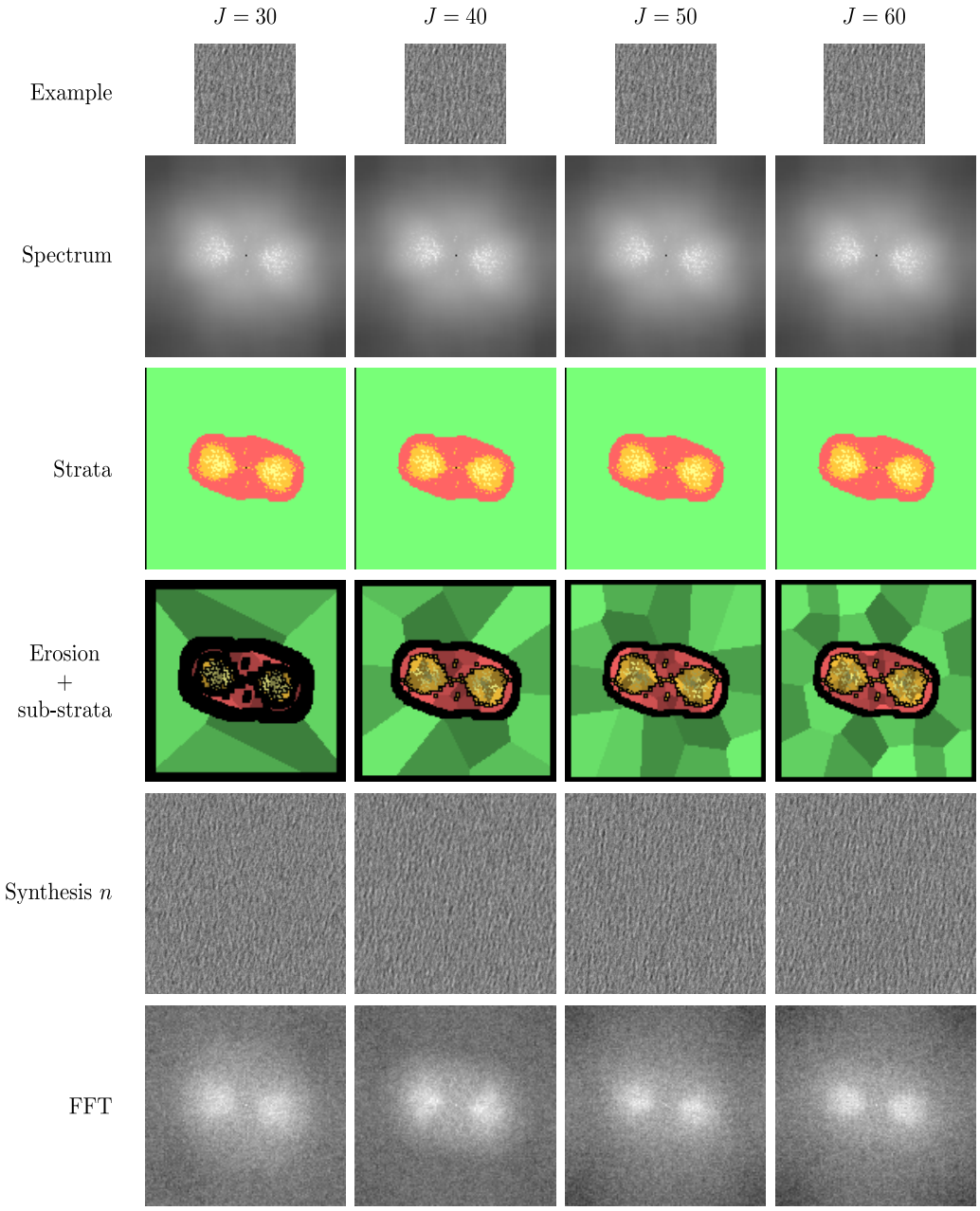
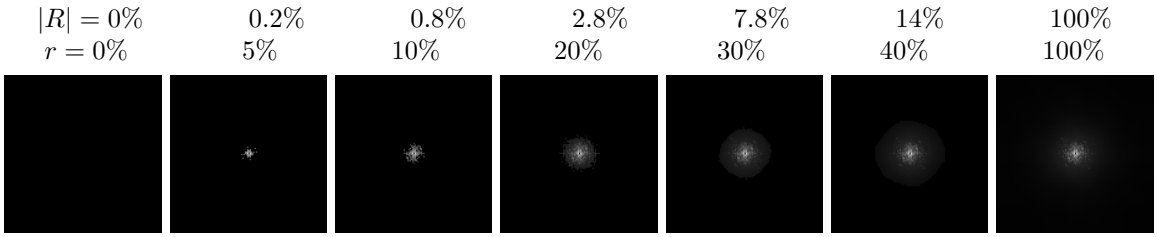


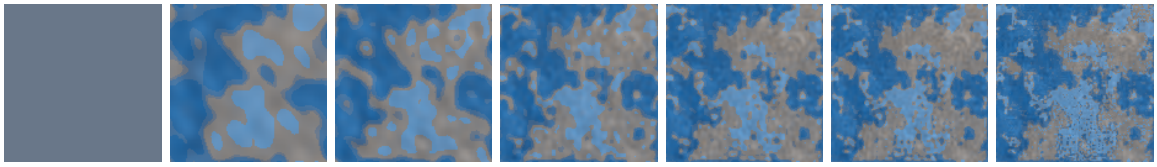
Figure 7: Variable cosine budgets for the “Skin” texture.

5.2 Proportion of structure r

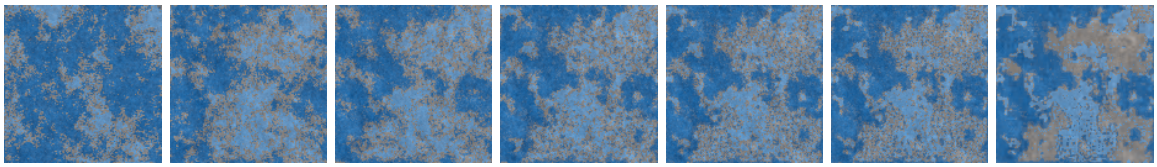
Figure 8 is an extension of figure 6 in the paper. It shows additional steps in defining the proportion of structure. The accompanying video shows some interactive tuning of this parameter.



(a) Region R. Full power spectrum on the right



(b) Synthesis of n_R



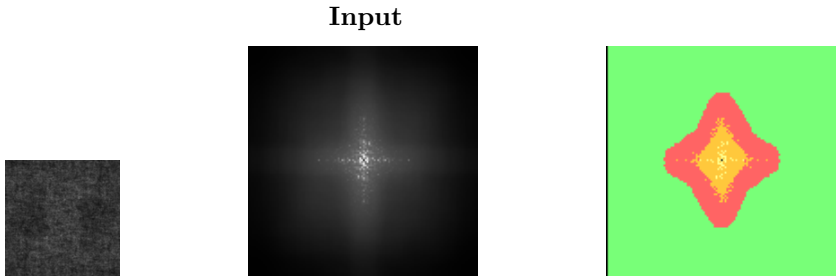
(c) Synthesis of n

Figure 8: Fixed phases for high-energy regions. From left to right the region R grows while its energy interval gets larger. r is the percentage of preserved energy. $|R|$ denotes the percentage of frequencies in R w.r.t. the full spectrum. The corresponding structure n_R better captures structures until being equal to the input (right). The final noise n exchanges randomness for faithfulness. The cosine budget is $J = 50$.

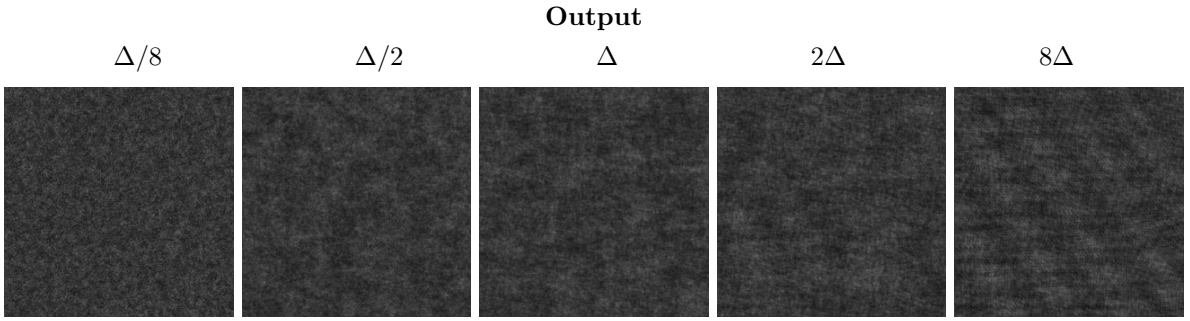
5.3 Window size Δ

Random-phase LRP noise based on discrete spectra resulting from input images are shown in figures 9, 10 and 11. All examples were computed with a cosine budget of $J = 50$ and 4 strata. In these examples, Δ is computed according to the formula of section 4.4 and then modified artificially by different factors (see figures).

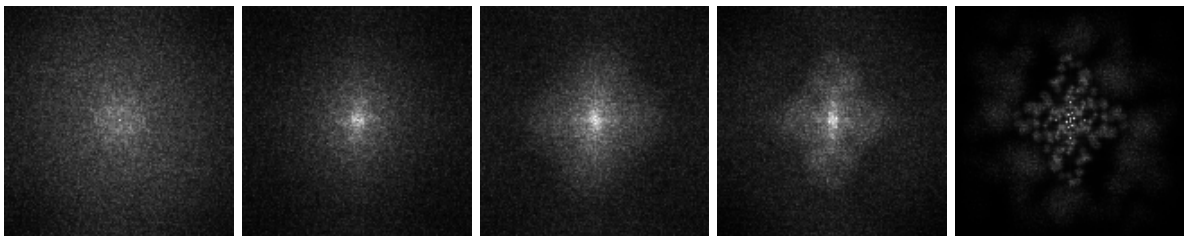
One can notice that enlarging Δ results in bad spectral coverage and a synthesized texture with artifacts. Conversely, shrinking Δ results in spectral leakage. Using Δ according to our formula is a good compromise in all our examples.



(a) Example image, its PSD and 4 strata

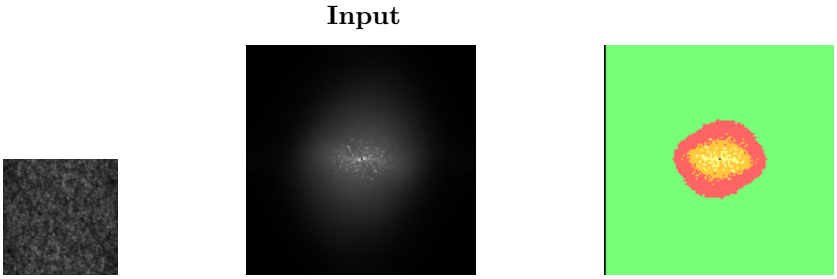


(b) Synthesis of n

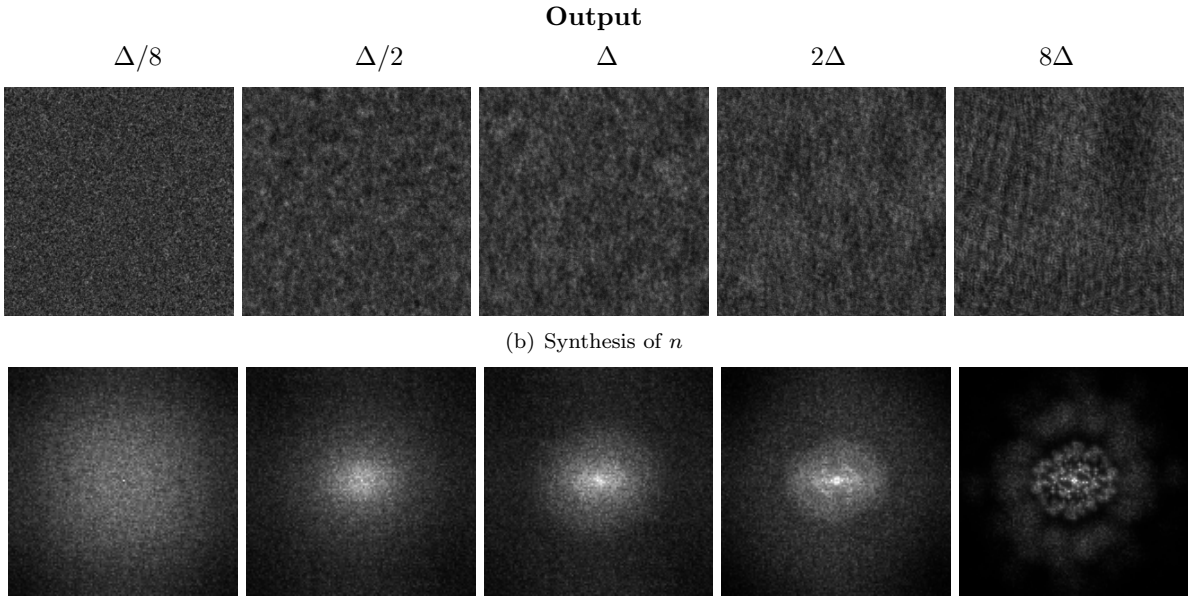


(c) FFT of n

Figure 9: Variable window sizes for the *Concrete* texture.



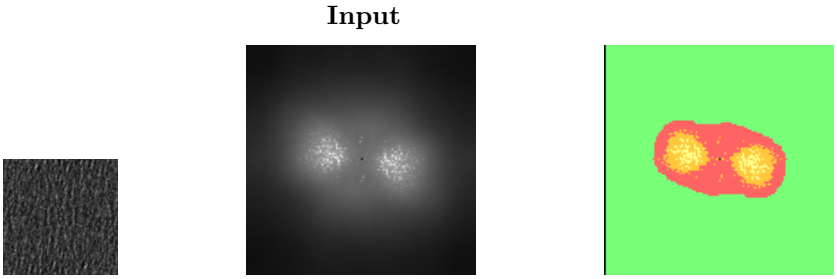
(a) Example image, its PSD and 4 strata



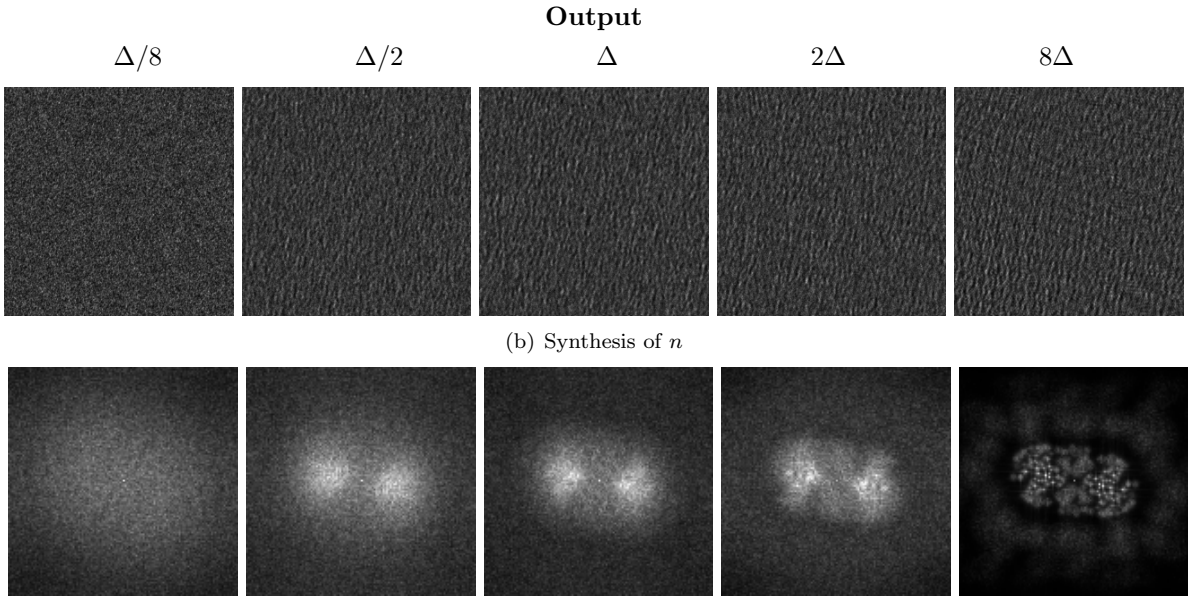
(b) Synthesis of n

(c) FFT of n

Figure 10: Variable window sizes for the *Granite* texture.



(a) Example image, its PSD and 4 strata



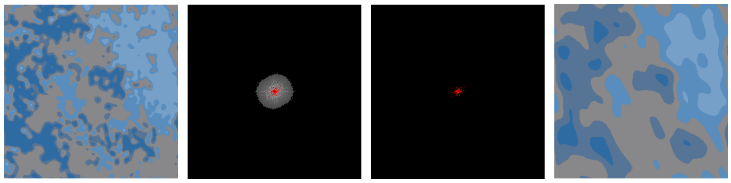
(b) Synthesis of n

(c) FFT of n

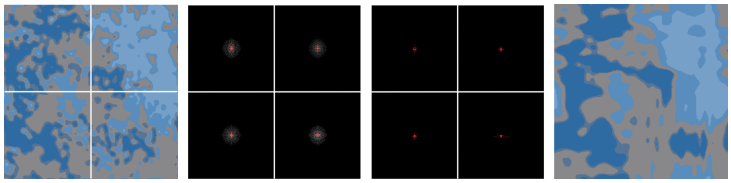
Figure 11: Variable window sizes for the *Skin* texture.

5.4 Number of subdivisions

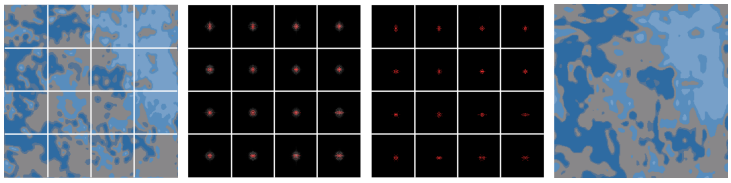
Figure 5.4 illustrates the iterative algorithm of section 5.3. After each subdivision step, block-wise FFTs are computed and the $J_R = 32$ highest-energy frequencies are stored (red) and used for evaluation of n_R. One can see that for the first iterations, this is not sufficient to reproduce the reference structure image (figure 12(e)). After roughly 3 iterations, the resulting image is sufficiently close to the expected result. $32 \times 8^2 = 2048$ amplitudes and phases have to be stored in this case.



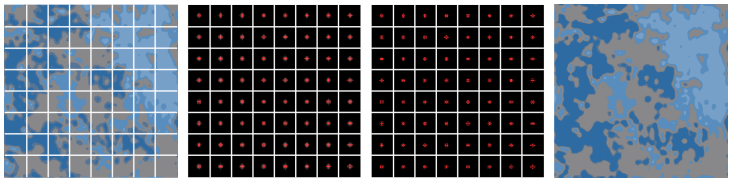
(a) Iteration 0



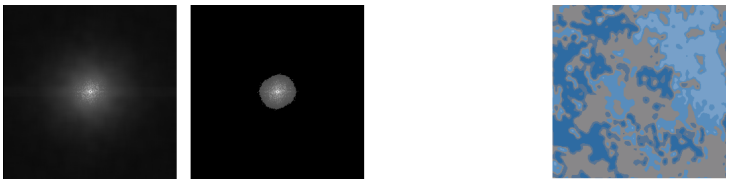
(b) Iteration 1



(c) Iteration 2



(d) Iteration 3

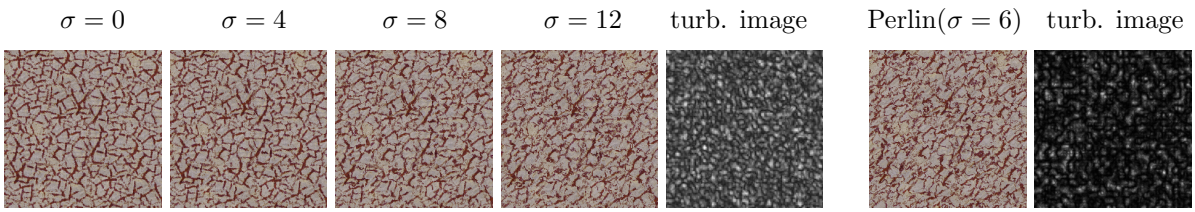


(e) Complete PSD (left), PSD on R (center) and structure image (right).

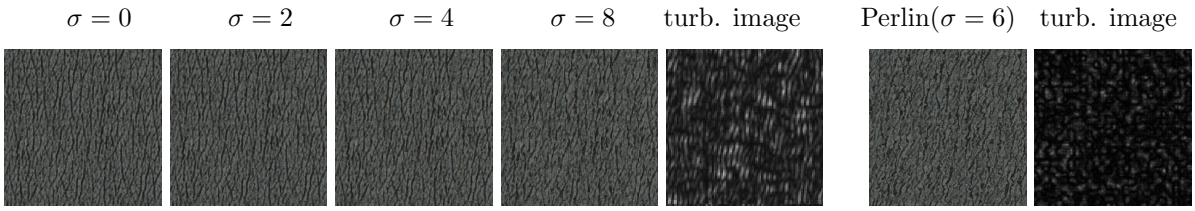
Figure 12: Example of subdivisions of an input texture of size 256^2. In figure 12(e), the structure image (right) is obtained by inverse FT of the spectrum of a region R (left). The first column in the full figure denotes the subdivided structure image. The second one the block-wise PSDs. The third one the $J_R = 32$ block-wise selected frequencies. The last one is composed of assembled block-wise n_R evaluated from the J_R stored frequencies.

5.5 Turbulence magnitude

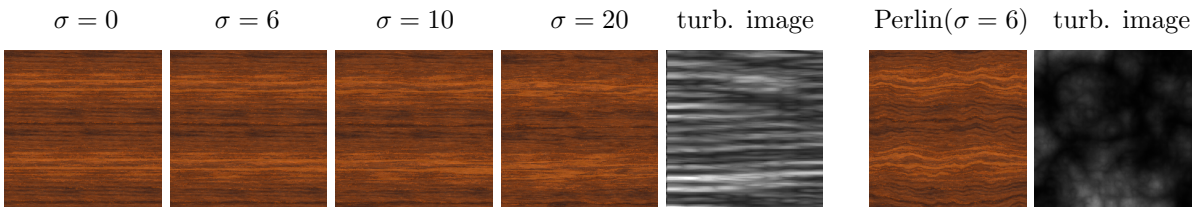
Our turbulence function is controlled by the spectrum. Therefore, it respects its anisotropy and can be controlled at will using the parameter σ (see the paper's section 5.2). In figure 13, we show several variants for σ which can all be valid depending on the users will. Note that compared to Perlin's turbulence, the anisotropy is better preserved, for example in figure 13(c). On the other examples, results are comparable.



(a) Paint Peel



(b) Skin



(c) Wood

Figure 13: Variable turbulence values.