APPENDIX

Anonymous authors
Paper under double-blind review

A PROOF OF PROPOSITION 1

Using the fact that

$$\begin{aligned}\rho(\text{Real}(ze^{i\alpha})) &= \rho(\text{Real}(|z|e^{i(\varphi(z)+\alpha)})) \\ &= |z|\rho(\cos(\alpha+\varphi(z)),\end{aligned}$$

and taking the Fourier transform of $\rho_\alpha(z)$ in the variable α, we obtain

$$\begin{aligned}\mathcal{F}(\rho_\alpha(z))(k) &:= \frac{1}{2\pi}\int_{[0,2\pi]}\rho_\alpha(z)e^{-ik\alpha}d\alpha \\ &= |z|\frac{1}{2\pi}\int_{[0,2\pi]}\rho(\cos(\alpha+\varphi(z))e^{-ik\alpha}d\alpha \\ &= |z|e^{ik\varphi(z)}c_k \\ &= [z]^k c_k,\end{aligned}$$

where c_k is the Fourier transform of $h(.) := \rho(\cos(.))$ at the frequency k. The function $\alpha \mapsto \rho_\alpha(z)$ being periodic in α, we have its decomposition in Fourier series

$$\begin{aligned}\rho_\alpha(z) &= \sum_{k\in\mathbb{Z}}\mathcal{F}(\rho_\alpha(z))(k)e^{ik\alpha} \\ &= \sum_{k\in\mathbb{Z}} c_k[z]^k e^{ik\alpha}.\end{aligned}$$

We can then write, for any $z, z' \in \mathbb{C}$, and $\alpha, \alpha' \in [0, 2\pi]$,

$$\rho_\alpha(z)\rho_{\alpha'}(z')^* = \sum_{k,k'\in\mathbb{Z}^2} c_k c_k^*[z]^k[z']^{-k'}e^{i(k\alpha-k'\alpha')}.$$

Replacing z and z' by any two wavelet coefficients $x\star\psi_{j,\theta}(u)$ and $x\star\psi_{j',\theta'}(u-\tau)$, we thus obtain the relation in Proposition 1.

B PROOF OF PROPOSITION 2

Let $z \in \mathbb{C}$, and recall from eq. (5), that $\rho_\alpha(z) = \rho(\text{Real}(ze^{i\alpha}))$. Note that we have the following relation

$$z = \rho_0(z) - \rho_\pi(z) - i(\rho_{\frac{\pi}{2}}(z) - \rho_{\frac{3\pi}{2}}(z)). \tag{9}$$

We can then write

$$\begin{aligned}zz'^* &= \big(\rho_0(z) - \rho_\pi(z) - i(\rho_{\frac{\pi}{2}}(z) - \rho_{\frac{3\pi}{2}}(z))\big)\big(\rho_0(z') - \rho_\pi(z') - i(\rho_{\frac{\pi}{2}}(z') - \rho_{\frac{3\pi}{2}}(z'))\big) \\ &= \sum_{\alpha,\alpha'\in I^2} w_{\alpha,\alpha'}\rho_\alpha(z)\rho_{\alpha'}(z'),\end{aligned}$$

with $I = \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. Replacing z with $x\star\psi_{j,\theta}(u)$, z' with $x\star\psi_{j',\theta'}(u-\tau)$, and injecting this relation in eq. (1) gives us the desired result.

C INFLUENCE OF THE CHOICE OF THE WAVELET TRANSFORM

C.1 INFLUENCE OF THE WAVELET FAMILY

In Section 3.2, we illustrated the importance of the set of indices Υ that define the wavelet coefficients being correlated. Another important role is played by the choice of the wavelets used in equation 2. As illustrated in Figure 5, this choice can have a visible impact on the quality of the textures. We observe that, while on the first example, the coherence of the structures appear similar for the three wavelet families, the second example shows that the wavelets used in Portilla & Simoncelli (2000) are less efficient in reproducing the contours of the objects (pebbles). While in our experiments, we chose to use the classical Morlet wavelets, an optimal choice for the wavelet family remains an open problem.

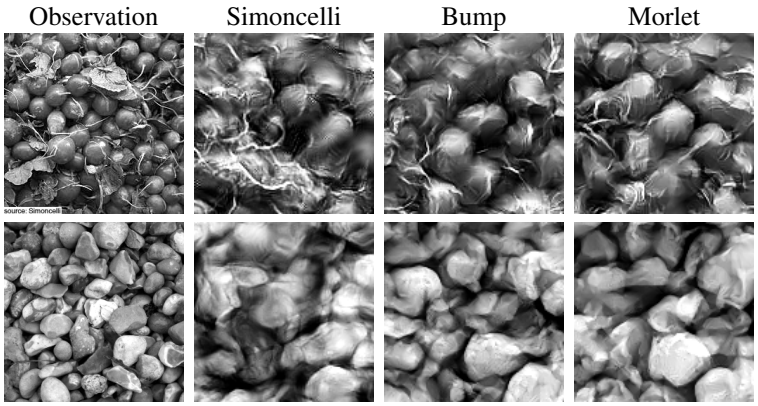

Figure 5: Comparision between different wavelets. Central zooms of syntheses using the same covariance model, with three different wavelet families. From left to right: observation, Simoncelli steerable wavelets, bump steerable wavelets (Mallat et al., 2020), and Morlet wavelets.

C.2 INFLUENCE OF SCALE PARAMETER

Recall from Section 2.2.1, the wavelet transform of an image x is defined by

$$\{x \star \psi_{j,\theta},\ x \star \phi_J\}_{0 \leq j < J,\, \theta \in \frac{\pi}{L}\{0,\cdots,L-1\}}.$$

The maximal scale parameter J also plays an important role in the definition of the wavelet transform. It determines the scales of the structures being captured by the transform. If this parameter is too small, large structures in the observation image might not be captured and reproduced in the model syntheses. Conversely, if J is too large, then the large scale statistics may have a high variance, inducing a memorization effect in the syntheses. Figure 6 illustrates this point on two examples from Section 4.2. By setting $J = 4$ (i.e. the maximal range of structures captured by the wavelets is of size $2^4 = 16$), we observe on the first example that the larger structures (bubbles) are not well reproduced. When J is set to 6, the observation is almost identically reproduced by the synthesis. Similarly on the second examples, several parts of the synthesis appear very similar to ones in the observation. We found that a suitable trade-off consists in setting $J = 5$ for images of size $N = 256$.

D MODEL AND ALGORITHMIC SPECIFICATION

We provide additional information needed to reproduce the numerical results of the models (for both gray-scale and color textures) considered in this paper. First, we detail the models of PS, RF, VGG. We then give algorithmic parameters to obtain the synthesis images. For natural textures, we also propose a strategy to synthesize non-periodic images in our models[1].

[1] an image is non-periodic if a periodic extension of the image to the domain outside Ω_N create discontinuities at the contour of Ω_N

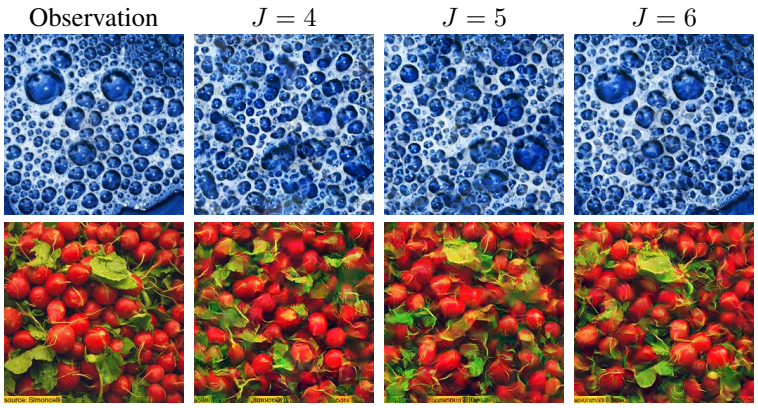

Figure 6: Syntheses from the $\mathrm{ALPHA}_{\mathrm{C}}$ model defined with three different maximal scale parameter $J \in \{4, 5, 6\}$.

Sources of textures Our natural texture examples were obtained from the following three sources: CNS NYU[2], Textures.com[3], Describable Textures Dataset model[4] and the Github page of Berger & Memisevic (2017)[5].

D.1 Model parameters

We specify the model parameters to synthesize both gray-scale and color textures. We also discuss how to extend the RF and the VGG model, originally designed for color textures, to model gray-scale textures.

- PS: We set the number of scales $J = 5$, and the number of orientations $L = 8$ for the Simoncelli steerable wavelets. The spatial shift $\tau = (\tau_1, \tau_2) \in \Omega_N$ is chosen to be in the range of $\max(\tau_1, \tau_2) \leq 4$. The rest of the model parameters are given by default in the software available at `https://www.cns.nyu.edu/~lcv/texture/`.
- RF: For gray-scale textures, we consider $J \times L$ random convolutional filters $(\psi_f)_{1 \leq f \leq JL}$. Let $f = (j, \ell)$, the index j representing the scale of each filter, whose size is (W_j, W_j) for $1 \leq j \leq J$. For $\Gamma = \{f = (j, \ell) | j \leq J, \ell \leq L\}$, the representation is

$$R^{\mathrm{RF}} x(\gamma, u) = \rho(x \star \psi_{j,\ell}(u)), \quad \gamma = (j, \ell) \in \Gamma.$$

  For a color image $x = \{x^c\}_{1 \leq c \leq 3}$, we use $3 \times J \times L$ random convolutional filters. The representation of x is

$$R^{\mathrm{RF}} x(\gamma, u) = \rho\big(\sum_{c=1}^{3} x^c \star \psi_{c,j,\ell}(u)\big), \quad \gamma = (j, \ell) \in \Gamma.$$

  The correlations $C^{\mathrm{RF}} x$ are defined for all pairs $(\gamma, \gamma') \in \Gamma \times \Gamma$. In both the gray and color cases, it results in a correlation matrix C^{RF} with $J^2 L^2$ statistics.

  Following the default setting of the RF model, we set $J = 8$ and $L = 128$ for filters whose sizes are $W_1 = 3, W_2 = 5, W_3 = 7, W_4 = 11, W_5 = 15, W_6 = 23, W_7 = 37, W_8 = 55$. Each filter $\psi_{j,\ell}$ or $\psi_{c,j,\ell}$ is generated randomly according to `GlorotUniform` in Lasagne[6].
- VGG. For a color image $x = \{x^c\}$, the VGG model computes a correlation matrix $C^{\mathrm{VGG}} x$ between the features maps within different layers of a pre-trained CNN network. To adapt

[2] `http://www.cns.nyu.edu/~lcv/texture/`
[3] `https://textures.com/`
[4] `https://www.robots.ox.ac.uk/~vgg/data/dtd/index.html`
[5] `https://github.com/guillaumebrg/texture_generation`
[6] `https://lasagne.readthedocs.io/`

this model to gray-scale textures, we shall add one input layer which converts a gray-scale image y into a color image, by setting $x^c = y$ for each color channel c. This allows one to use the same $C^{\mathrm{VGG}}x$ to compute the gradient of the VGG loss with respect to y, and therefore to synthesise a gray-scale texture. For both gray-scale and color textures, we use only five layers 'conv1_1', 'pool1', 'pool2', 'pool3', 'pool4', as proposed in the original work.

- ALPHA. See the main text.

D.2 ALGORITHMIC PARAMETERS

We specify the optimization parameters used to synthesize both gray-scale and color textures.

- PS: It utilizes iterative projections onto constraint sets to generate textures. We set the number of iterations to 200.
- RF: It uses the L-BFGS procedure[7] with a memory size 20, and with a maximal number of iterations 2000. The initialization for each pixel value of a gray-image is Uniform between $[-1, 1]$. For the color image case, each RGB channel is initialized independently with a Uniform distribution between $[-1, 1]$. To address non-zero mean textures (i.e. $\mathbb{E}(X(u)) \neq 0$), the empirical mean of $\bar{x}$ is subtracted from the input x to compute the representation. It is added back to the output of the optimization to produce a synthesis.
- VGG: It uses the L-BFGS procedure[8] with a memory size of 20, and with a maximal number of iterations of 2000. The initialization of each pixel value is the standard normal distribution (zero mean, unit variance). To address non-zero mean textures (i.e. $\mathbb{E}(X(u)) \neq 0$), the VGG mean is subtracted from the input x of the representation[9]. It is added back to the output after the optimization to produce a synthesis (with an additional histogram matching post-processing).
- ALPHA: For all the models in ALPHA, we use the L-BFGS optimization algorithm with restarts. Starting from the standard normal distribution, with mean and standard deviation estimated from the observation, we use the L-BFGS procedure implemented in Pytorch. It runs for 500 iterations and then it is restarted with an initialization obtained from the previous L-BFGS result. This is repeated 10 times to obtain the synthesis (with an additional histogram matching post-processing).

D.3 NON PERIODIC BOUNDARIES IN NATURAL IMAGES

The convolution operation in the wavelet transform (equation 2) is performed using the Fast Fourier Transform. Additionally, recall from Section 2.1 that spatial shifts are defined with periodic boundary conditions. This implies periodicity of the input image x. However, natural texture images are not periodic, so one needs to adapt the computation of coefficients to take into account possible border effects. To that end, instead of averaging over all $u \in \Omega_N$ as in eq. (1), each correlation coefficient is averaged over a sub-window inside Ω_N, which size depends on the scales of the coefficients being correlated. More precisely, let $\gamma = (j, \theta, \alpha)$ and $\gamma' = (j', \theta', \alpha')$. Note $j_m := \max(j, j')$. We define $\Omega_{j_m} := \{u = (u_1, u_2) \in \Omega_N : \ 2^{j_m} \leq u_i < N - 2^{j_m}, \ i = 1, 2\}$. Then, for non periodic images, we compute

$$C^{\mathrm{ALPHA}}x(\gamma, \gamma', \tau) = \frac{1}{|\Omega_{j_m}|} \sum_{u \in \Omega_N} \mathbb{1}_{\Omega_{j_m}}(u) \mathbb{1}_{\Omega_{j_m}}(u - \tau) R^{\mathrm{ALPHA}}x(\gamma, u) R^{\mathrm{ALPHA}}x(\gamma', u - \tau), \tag{1}$$

where the spatial shifts are defined periodically. Note that the spatial averages μ_γ and $\mu_{\gamma'}$ are also performed on Ω_{j_m}.

[7] `scipy.optimize.fmin_l_bfgs_b` in Python

[8] `scipy.optimize.minimize` in Python

[9] For the color image whose pixel value is between zero and one, the mean of BGR is 0.40760392, 0.45795686, 0.48501961. For the gray-scale, we simply take the average of the BGR mean.

E VGG SCORE

In Ustyuzhaninov et al. (2017), the authors proposed to use the synthesis loss of the VGG model to evaluate the quality of syntheses from any model. The goal is to define a quantitative, and more objective evaluation method than mere visual inspection. Since the VGG model produces syntheses almost indistinguishable from real textures, it is natural to consider its loss to asses the quality of a synthesis. We computed this loss for the examples presented in Figure 3, and report it in Table 2. Note however that this loss is not exactly the same as the one used in Ustyuzhaninov et al. (2017), as the layers selected to compute the loss are different. In this work, we chose to use the layers suggested in Gatys et al. (2015), (i.e. 'conv1_1', 'pool1', 'pool2', 'pool3', and 'pool4' of the VGG-19 network (Simonyan & Zisserman, 2014)), and compute the relative VGG loss[10].

We notice that this score is not always consistent with visual inspection, as there are texture examples and models for which the syntheses do not look much like the observation image, yet produce a small VGG loss (see e.g. the first and last rows of Figure 3, the RF model syntheses have the smallest loss). It should also be noted that the VGG loss reported on the VGG syntheses is *not* the synthesis loss after optimization, as a histogram matching (HM) procedure is performed as post-processing after optimization. We observed that the VGG loss of the syntheses from the VGG model after HM was considerably higher than the one for syntheses before it, while being visually very similar as illustrated in Figure 7. These observations suggest that the VGG score suffers from instabilities after reaching a certain level (that is, if the VGG loss is small enough, small perturbations of the values of the image pixels might have a strong impact on the loss).

Data / Model	ALPHA_I	VGG	PS	RF
Radishes	5.02e-05	1.87e-05	2.37e-04	1.13e-05
Cherries	4.86e-05	1.47e-06	6.68e-04	9.65e-06
Gravel	5.97e-05	3.08e-06	7.25e-04	1.29e-05
Turbulence	5.59e-05	5.97e-05	2.42e-04	3.95e-05

Table 2: Relative VGG loss of each model on examples in Figure 3. Each row corresponds to the same row in Figure 3.

Figure 7: Visual comparison of syntheses from the VGG model, before and after histogram matching. Before HM, the relative VGG loss is 2.22e-08, while after HM, the loss is 5.38e-05.

[10]Using the code from `https://github.com/ivust/random-texture-synthesis/blob/master/vgg_loss.py` (function style_loss_relative).