IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2021–07–08.
Submitted on 2020–10–15, accepted on 2021–06–21.
ISSN 2105–1232 © 2021 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2021.324

The Portilla-Simoncelli Texture Model: towards Understanding the Early Visual Cortex

Jonathan Vacher[1,2], Thibaud Briand[3]

[1] Department of Systems and Computational Biology, Albert Einstein College of Medicine, Bronx, NY, 10461, USA
[2] Laboratoire des Systèmes Perceptifs (LSP), Département d'Études Cognitives,
École Normale Supérieure, PSL University, 75005 Paris, France
(jonathan.vacher@ens.fr)
[3] Université Paris-Saclay, CMLA (UMR CNRS 8536), France
(briand.thibaud@gmail.com)

Abstract

Texture synthesis is a prolific subarea in computer vision where statistical methods are often successful. The Portilla and Simoncelli (PS) texture algorithm is one of such methods that became very popular and has influenced visual perception studies. For many reasons it can still be considered as a state-of-the art texture synthesis algorithm: (i) it generates textures that are often indistinguishable from the original without scrutiny; (ii) it relies on few parameters compared to recent deep learning methods; (iii) recent algorithms often compare to it. Here, we review the scientific impact of this algorithm and give a detailed explanation. Briefly, the PS algorithm synthesizes a new texture by iteratively imposing to a Gaussian white noise image a set of high-order statistics of wavelet coefficients precomputed on a texture example. After few iterations the initial white noise image is transformed into a texture that is similar to the texture example. We provide a fast C++ implementation, evaluate the effect of the algorithm parameters and illustrate its capabilities with many synthesis examples. In addition, we propose two notable new features to the original implementation: (i) the possibility to interpolate between two textures; (ii) the possibility to handle non-periodicity using the "periodic+smooth" decomposition.

Source Code

The C++ source code and the online demo are accessible at the web page of this article[1]. Compilation and usage instruction are included in the `README.txt` file of the archive.

Keywords: texture synthesis; multi-scale pyramid decomposition; image filtering; summary statistics; texture perception

[1] https://doi.org/10.5201/ipol.2021.324

1 Introduction

The Portilla-Simoncelli (PS) algorithm performs texture synthesis by iteratively imposing high-order statistical constraints on local image features [41]. The set of statistical constraints is inspired by the primary visual cortex (area V1) architecture [25] and the long-standing Julesz hypothesis in psychophysics that textures with similar statistics are indistinguishable to humans [27] (without scrutiny or in peripheral vision).

From texture perception to texture synthesis algorithms. The Julesz hypothesis was originally about the second order statistics of pixels. After multiple falsifications of his initial hypothesis [29, 30, 12], Julesz proposed as an alternative that first order spatial statistics (densities) of deterministic features dubbed "textons" would be sufficient to explain texture discrimination [28]. This approach was later abandoned to understand texture perception (see [33] for a review). However, on the one hand, the idea of "textons" density led to a mathematical theory of micro-texture synthesis [19]. In this approach a texture is generated by convolution between a white noise image and a specific "texton", which leads to efficient texture synthesis algorithms [20] and multiple equivalent formulations of a parametric texture model to study vision [54]. On the other hand, the Julesz hypothesis led to the Heeger-Bergen (HB) [10, 24] and PS [41] algorithms, which rely on local feature statistics. In a way, this corresponds to statistics between sub-bands obtained by convolution with generic "textons" such as scaled and oriented Gabor functions, which are known to correctly model neuron receptive fields [26]. Thus, the two hypotheses (statistics and "textons"), which were seen as alternatives by Julesz, were finally combined successfully in the PS texture synthesis algorithm. Textures generated by the PS algorithm are shown to be almost indistinguishable from their original version [1].

Improvements of the PS algorithm. The PS algorithm can be viewed as an extension of the HB algorithm, which consists in adjusting higher-order wavelet sub-bands statistics. While the HB algorithm convergence can be refined using optimal transport theory [50], the PS algorithm still lacks a proper mathematical framework, and its convergence is still empirical. A variational approach, combining first-order statistics of decomposition coefficients in a sparse adapted feature dictionary and second order statistics of pixel values, provides visually similar synthesized textures [49]. Yet, this algorithm is somehow less generic than the PS algorithm because a different set of features is learned on each texture. From a neurally inspired perspective, convolutional neural networks (CNN) are successfully used to generate textures by simply matching second order statistics of each layer outputs [21]. The synthesized textures show an impressive quality and are often indistinguishable from their original version even when scrutiny is allowed [56]. In addition, CNN approaches allow for computational efficiency when trained on a single example texture [51]. However, it is likely that CNN performances are more due to the very high-dimensional decomposition than to the specificity of the learned filters. Indeed, synthesizing textures is also feasible using random convolutional units [23]. In comparison with most CNN-based approaches that use 50K parameters, the PS algorithm is still remarkably more parsimonious as it only requires around 1K parameters. However, we note that recent progresses have reduced the high parameter numbers of CNN-based texture synthesis methods [16].

Probing visual perception using texture synthesis algorithms. Explicitly inspired by biological vision, the PS algorithm soon became a computational model that served texture perception studies [1]. In particular, it is used to predict human categorization of textures both in attentive [2] and peripheral settings [3], visual crowding (impaired performances in peripheral vision in presence

of multiple visual objects) [4] and visual search [44]. The importance of the PS model increases specifically for the understanding of peripheral vision that is proposed to rely mainly on texture features [17, 55]. One conclusion is that texture features, i.e. PS statistics, might be represented in the area V2 of the visual cortex [18]. Characteristics of this representation, like robustness [60] and contextual modulation [61], have been further studied. Moreover, the higher-level visual cortex area V4 also plays a role in texture representation [38, 53]. While the hypothesis that peripheral vision relies on texture statistics generates great progress in understanding the visual cortex, it is still incomplete as it does not account for global scene features such as segmentation [57, 43]. Recent deep learning approaches are promising to bridge the gap between low-level texture perception and high-level tasks like scene segmentation and object recognition [21, 56, 13, 52]. Deep neural networks are now used to better probe neural sensitivity in the higher visual cortex [5, 40] and their reduced algorithmic load provides more practicable theoretical ground to further model vision [53].

Texture interpolation. Despite not being well-known, the PS algorithm can be used to perform interpolation of textures i.e. to continuously morph one texture into another one. Texture interpolation has been used previously in vision studies. First, to quantify texture naturalness [18] by interpolating between the micro-texture [19] and naturalistic [41] syntheses of textures. Second, to enrich a texture dataset used in neurophysiological recordings [38]. Texture interpolation is often well-formalized using optimal transport theory, which allows to define the barycenter of probability distributions [42, 58]. Recently, Gatys et al. approach has been corrected and extended to perform texture interpolations that are appropriate to study vision [53].

Algorithm overview. The PS algorithm can be summarized as follows. First, in order to mimic the phase invariant responses of "complex" cells in the primary visual cortex [36, 32], a texture example is decomposed into complex wavelet sub-bands at multiple orientations and scales. From these complex wavelets coefficients, hundreds of cross-statistics are calculated to form multiple statistical constraint sets. Second, an output texture is synthesized using an iterative scheme with a white noise initialization. At each step, the current output texture is first decomposed into complex wavelet sub-bands at the same orientations and scales. Then, from the coarsest scale and for each statistical constraint, the wavelet coefficients from which it depends are projected onto the set defined by the statistical constraint. When all wavelet coefficients at one scale have been projected onto the desired statistical constraint sets, they are recomposed together to obtain the upper scale low-pass sub-band. By iterating these steps, the output texture empirically converges towards a new texture sample that is qualitatively similar to the texture example.

Contributions. We propose a precise description of the Portilla and Simoncelli algorithm along with a C++ implementation that is faster than the original MATLAB implementation[2]. Our implementation also comprises the extension to perform texture interpolation that was used in vision studies [38, 18] and the possibility to handle non-periodicity using the *periodic+smooth* decomposition [35]. In particular, we describe the complex steerable pyramid decomposition as a template of the "complex" cells found in the primary visual cortex along with implementation details. Moreover, we explain how the authors proceeded to choose the relevant set of statistics and how they enforced these statistics on a white noise image to perform the synthesis. Finally, we show multiple texture synthesis results and we test the free parameters of the algorithm.

The remainder of this article is organized as follows. In Section 2, we present the steerable pyramid decomposition and reconstruction. Then, in Section 3, we describe in detail the PS algorithm. Finally,

[2] https://github.com/LabForComputationalVision/textureSynth

in Section 4, we show many synthesis results and test the impact of multiple features (free parameters, constraints) of the PS algorithm.

2 The Steerable Pyramid Decomposition and Reconstruction

The *steerable pyramid* decomposition is a wavelet transform, that linearly decomposes an image into multiple scaled and orientated sub-bands, proposed by E. Simoncelli and his co-authors (see [47, 46] and the references therein). An input image is first split into a high-frequency part and a low-frequency part, the latter is then split into oriented sub-bands and down-sampled. The output of the steerable pyramid decomposition is a collection of sub-bands (corresponding to filtering of the input image) of different sizes and referred to as a pyramid. Each sub-band corresponds to a specific scale and orientation, apart from the high-frequency and the low-frequency residuals. This decomposition is invertible and the initial image can be reconstructed from the pyramid. The filters involved are described in [41] and correspond to a complex version of the filters used in [10].

First, the filters are defined in Section 2.1. Then, the decomposition and reconstruction processes of continuous images are detailed in Section 2.2. The method for filtering discrete images is presented in Section 2.3. Finally, the practical algorithms for the decomposition and the reconstruction are given in Section 2.4.

2.1 Definitions of the Filters

The filters involved in the steerable pyramid computation are given in the frequency domain by continuous functions. Therefore, we need to define the Fourier transform.

Definition 1 (Fourier transform)**.** *The continuous Fourier transform applied to $L^1(\mathbb{R}^2)$ and its inverse, denoted $\mathcal{F}$ and $\mathcal{F}^{-1}$, are defined by*

$$\forall u \in L^1(\mathbb{R}^2), \quad \forall \mathbf{y} \in \mathbb{R}^2, \quad \hat{u}(\mathbf{y}) = \mathcal{F}(u)(\mathbf{y}) = \int_{\mathbb{R}^2} u(\mathbf{x}) e^{-\imath \mathbf{x}\cdot\mathbf{y}} \mathrm{d}\mathbf{x}, \tag{1}$$

$$\textit{and, if } \hat{u} \in L^1(\mathbb{R}^2), \quad \forall \mathbf{x} \in \mathbb{R}^2, \quad u(\mathbf{x}) = \mathcal{F}^{-1}(\hat{u})(\mathbf{x}) = \frac{1}{(2\pi)^2} \int_{\mathbb{R}^2} \hat{u}(\mathbf{y}) e^{\imath \mathbf{x}\cdot\mathbf{y}} \mathrm{d}\mathbf{y}. \tag{2}$$

The Fourier transform extended to $L^2(\mathbb{R}^2)$ and to tempered distributions [45, 48] is still denoted $\mathcal{F}$.

The steerable pyramid filters can be applied through convolution in the spatial domain as follows.

Definition 2 (Convolution)**.** *The convolution between two functions $(u, v) \in L^1(\mathbb{R}^2)$ is given by*

$$\forall \mathbf{x} \in \mathbb{R}^2, \quad u * v(\mathbf{x}) = \int_{\mathbb{R}^2} u(\mathbf{y}) v(\mathbf{y} - \mathbf{x}) \mathrm{d}\mathbf{y}. \tag{3}$$

The convolution extended to $L^2(\mathbb{R}^2)$ and, under stronger hypotheses, to tempered distributions [45, 48] is still denoted $*$.

Equivalently, the filters can be applied in the Fourier domain by multiplication thanks to the famous property of convolution reminded in the following proposition.

Proposition 1 (Fourier transform and convolution)**.** *Let $(u, v) \in L^1(\mathbb{R}^2)$, then*

$$\mathcal{F}(u * v) = \mathcal{F}(u)\mathcal{F}(v). \tag{4}$$

This result holds on $L^2(\mathbb{R}^2)$ and on the space of tempered distributions under stronger hypotheses.

Since the filters will be defined in terms of polar coordinates on the plane, let us define the standard polar transformation ρ to go from Cartesian to polar coordinates.

Definition 3 (Polar transformation)**.** *The polar transformation ρ is defined by*

$$\begin{array}{rccl} \rho: & \mathbb{R}^2 & \longrightarrow & \mathbb{R}^+\times]-\pi,\pi] \\ & (x,y) & \longmapsto & (r,\theta)=\begin{cases}(|x|,\pi) & \text{if } y=0 \text{ and } x\leq 0,\\ \left(\sqrt{x^2+y^2}, 2\arctan\left(\frac{y}{x+\sqrt{x^2+y^2}}\right)\right) & \text{otherwise.}\end{cases}\end{array} \tag{5}$$

Its inverse ρ^{-1} is defined by

$$\begin{array}{rccl} \rho^{-1}: & \mathbb{R}^+\times]-\pi,\pi] & \longrightarrow & \mathbb{R}^2 \\ & (r,\theta) & \longmapsto & (r\cos\theta, r\sin\theta). \end{array} \tag{6}$$

For clarity we first introduce the real version of the steerable pyramid filters and the complex version later on. Let $Q\geq 3$ be the number of orientations of the pyramid.

Definition 4 (Filters in polar coordinates)**.** *For any polar coordinate $(r,\theta)\in\mathbb{R}^+\times]-\pi,\pi]$, we define*

$$L(r,\theta)=L(r)=\begin{cases}1 & \text{if } r\leqslant\frac{\pi}{4},\\ \cos(\frac{\pi}{2}\log_2(\frac{4r}{\pi})) & \text{if } \frac{\pi}{4}\leqslant r\leqslant\frac{\pi}{2},\\ 0 & \text{if } r\geqslant\frac{\pi}{2},\end{cases} \tag{7}$$

and

$$H(r,\theta)=H(r)=\begin{cases}0 & \text{if } r\leqslant\frac{\pi}{4},\\ \cos(\frac{\pi}{2}\log_2(\frac{2r}{\pi})) & \text{if } \frac{\pi}{4}\leqslant r\leqslant\frac{\pi}{2},\\ 1 & \text{if } r\geqslant\frac{\pi}{2},\end{cases} \tag{8}$$

which correspond respectively to a low-pass filter and a high-pass filter. For $q\in\{0,\dots,Q-1\}$, we also define the following cone-shaped filters

$$G_q(r,\theta)=G_q(\theta)=\alpha_Q\left|\cos\left(\theta-\frac{q\pi}{Q}\right)\right|^{Q-1}\left(\mathbb{1}_{|\theta-\frac{\pi q}{Q}|\leqslant\frac{\pi}{2}}+\mathbb{1}_{|\theta-\frac{\pi(q-Q)}{Q}|\leqslant\frac{\pi}{2}}\right), \tag{9}$$

where the normalizing constant α_Q is given by

$$\alpha_Q=2^{Q-1}\frac{(Q-1)!}{\sqrt{Q(2(Q-1))!}}. \tag{10}$$

Then, the high-pass, the low-pass and the oriented symmetrical filters are defined by

$$L_0(r,\theta) = L_0(r)=L\left(\frac{r}{2}\right), \tag{11}$$

$$H_0(r,\theta) = H_0(r)=H\left(\frac{r}{2}\right), \tag{12}$$

$$B_q(r,\theta) = H(r)G_q(\theta),\quad q\in\{0,\dots,Q-1\}. \tag{13}$$

The corresponding filters in the spatial domain are then defined as follows (see Figure 1).

Definition 5 (Filters in spatial domain)**.** *The low-pass filters l_0 and l, the high-pass filter h_0 and the oriented symmetrical filters b_q are defined by*

$$l_0 = \mathcal{F}^{-1}(L_0\circ\rho)\quad\text{and}\quad l=\mathcal{F}^{-1}(L\circ\rho), \tag{14}$$

$$h_0 = \mathcal{F}^{-1}(H_0\circ\rho), \tag{15}$$

$$b_q = \mathcal{F}^{-1}(B_q\circ\rho),\quad q\in\{0,\dots,Q-1\}. \tag{16}$$

The low-pass filters l_0 and l are $L^2(\mathbb{R}^2)$ functions. The high-pass filter h_0 and the oriented filters $(b_q)_{q\in\{0,\dots,Q-1\}}$ are tempered distributions.

The specificity of the PS algorithm is the use of complex wavelets that are obtained using a type of 2D Hilbert transform [11]. The choice of the Hilbert transform naturally follows the reference direction $\frac{\pi q}{Q}$ of the oriented symmetrical filter b_q as described in the following definition.

Definition 6 (Hilbert filter and transform)**.** *For $q \in \{0, \dots, Q-1\}$ denote by $\mathbf{n}_q$ the normal vector to the orientation $\frac{\pi q}{Q}$ given by*

$$\mathbf{n}_q = \left(-\sin\left(\frac{\pi q}{Q}\right), \cos\left(\frac{\pi q}{Q}\right)\right). \tag{17}$$

The Hilbert filter m_q is a tempered distribution defined through its Fourier transform $M_q \circ \rho$,

$$\forall \mathbf{x} \in \mathbb{R}^2, \quad (M_q \circ \rho)(\mathbf{x}) = 1 + \operatorname{sign}(\langle \mathbf{n}_q, \mathbf{x}\rangle) \tag{18}$$

where $\langle\ ,\ \rangle$ denotes the Euclidean scalar product. In polar coordinates it is defined by

$$M_q(r,\theta) = \begin{cases} 2 & \text{if } |\theta - \frac{\pi q}{Q}| < \frac{\pi}{2}, \\ 1 & \text{if } |\theta - \frac{\pi q}{Q}| = \frac{\pi}{2}, \\ 0 & \text{if } |\theta - \frac{\pi q}{Q}| > \frac{\pi}{2}. \end{cases} \tag{19}$$

The Hilbert transform with orientation $\frac{\pi q}{Q}$ of a function $u \in L^2(\mathbb{R}^2)$ is defined as its convolution with the Hilbert filter m_q or equivalently by the inverse Fourier transform of a multiplication in the Fourier domain

$$\mathcal{H}_q(u) = m_q * u = \mathcal{F}^{-1}((M_q \circ \rho)\mathcal{F}(u)) \in L^2(\mathbb{R}^2).$$

Such a choice of 2D Hilbert transform corresponds in fact to successive classical 1D Hilbert transforms along the lines that are orthogonal to the line passing through the origin with orientation $\frac{\pi q}{Q}$. See [31] for details.

Noting that the Fourier transforms of b_q and m_q are bounded functions, the oriented filter $m_q * b_q$ is a well defined tempered distribution. It can be applied directly, as in Algorithm 2, using its Fourier transform that is given in the following proposition.

Proposition 1. *The Fourier transform $(M_q B_q) \circ \rho$ of the tempered distribution $m_q * b_q$ is given in polar coordinates by*

$$(M_q B_q)(r,\theta) = 2\alpha_Q H(r) \cos\left(\theta - \frac{\pi q}{Q}\right)^{Q-1} \mathbb{1}_{|\theta - \frac{\pi q}{Q}| \leqslant \frac{\pi}{2}}. \tag{20}$$

Proof. This is obtained by combining (9), (13) and (19). The absolute value in (9) can be removed because the cosine function is non-negative on $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The case $|\theta - \frac{\pi q}{Q}| = \frac{\pi}{2}$ in (19) can be simplified because $\cos\left(\pm\frac{\pi}{2}\right) = 0$. □

The resulting wavelet $m_q * b_q$ is a complex-valued function whose real part is equal to the original real wavelet b_q and whose imaginary part is a quadrature phase wavelet with similar orientation $\frac{\pi q}{Q}$ (see Figure 2 left). One relevant feature for vision neurosciences is the property of phase invariance of this complex filter. In other words, the norm of the projection of a plane wave image (also known as a drifting grating in neurosciences) onto this wavelet does not depend on its phase (see Figure 2 right). For that reason such a complex wavelet is a good model of the "complex" cells that are present in the primary visual cortex [36, 32].

Finally, because the steerable pyramid decomposition is a multiscale transform, we introduce the zoom-in and zoom-out operators.

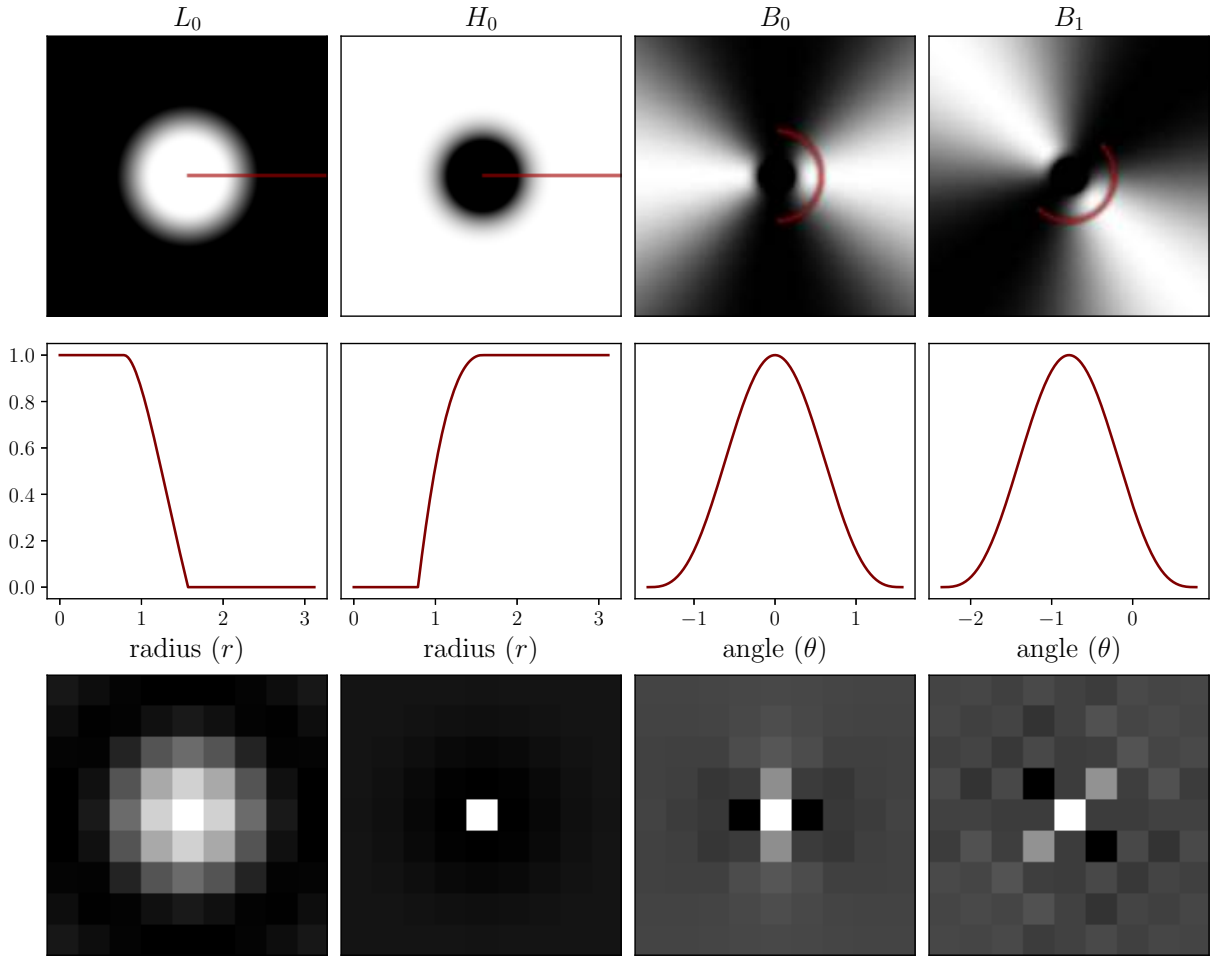

Figure 1: First and second rows: the filters given in Definition 4 in the spectral domain. The second row displays the profile of the section indicated in red in the first row. Third row: the filters given in Definition 5 in the spatial domain. The oriented filters are represented with $Q = 4$. The gray scale is arbitrary but increasing from black to white. The figure is best seen in color.

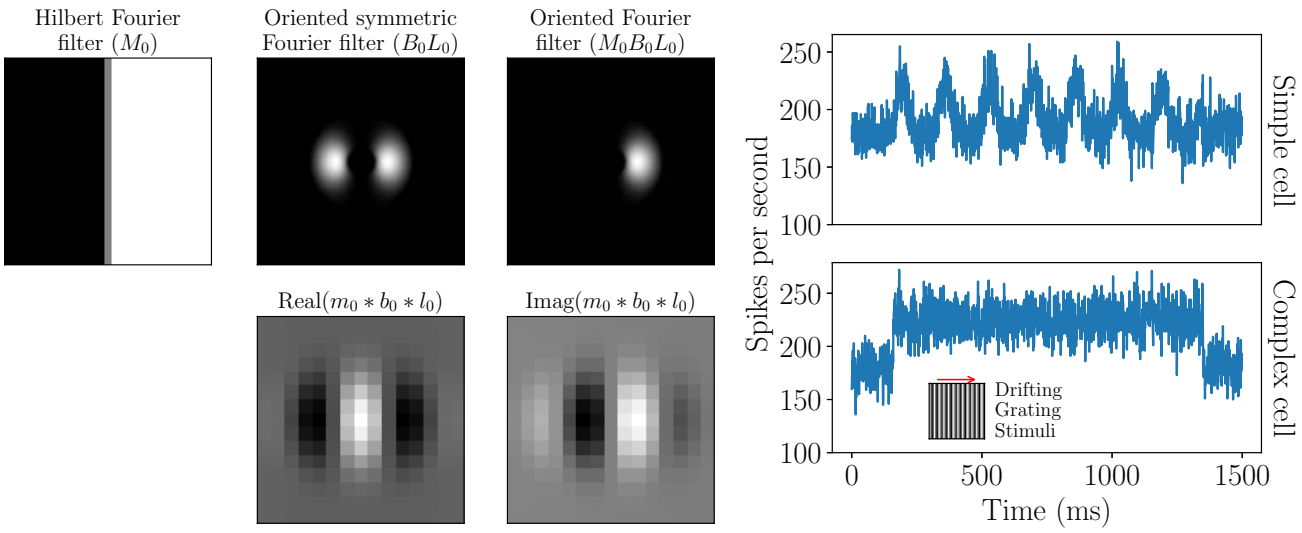

Figure 2: Left: Hilbert transform $m_0 * b_0$ of b_0. Real and imaginary parts are in phase quadrature. Gray scale is arbitrary but increasing from black to white. Right: Simulation of the response of simple and complex cell to a drifting grating stimuli. The simple cell is sensitive to the phase while the complex cell is phase-invariant.

Definition 7 (Zoom-in and zoom-out)**.** *Let $u \in L^2(\mathbb{R}^2)$. The zoom-in operator $\uparrow_2$ is defined by*

$$\forall \mathbf{x} \in \mathbb{R}^2, \quad \uparrow_2 (u)(\mathbf{x}) = u\left(\frac{\mathbf{x}}{2}\right). \tag{21}$$

The zoom-out operator $\downarrow_2$ is defined by

$$\forall \mathbf{x} \in \mathbb{R}^2, \quad \downarrow_2 (u)(\mathbf{x}) = u\left(2\mathbf{x}\right). \tag{22}$$

These operators extended to tempered distributions are still denoted $\uparrow_2$ and $\downarrow_2$.

Note that $L^2(\mathbb{R}^2)$ is stable by the application of the zoom-in and zoom-out operators, and of all the filters defined above (because their Fourier transforms are bounded functions). This property insures that the linear operators of the following section can be defined.

2.2 Image Decomposition and Reconstruction

The steerable pyramid filters are built so that they allow one to decompose an image into sub-bands and to reconstruct an image from sub-bands. In other words, the filters have properties that make the steerable pyramid decomposition a pseudoinvertible operator [6]. These properties are detailed in the original publication [46]. In the following section we explain how to decompose and reconstruct an image $u \in E = L^2(\mathbb{R}^2)$.

The steerable pyramid operator has two parameters: the number of scales P (default value 4) and the number of orientations $Q \geq 3$ (default value 4).

Decomposition. The complex steerable pyramid decomposition can be viewed as the composition of two linear operators: the first corresponds to the real steerable pyramid decomposition and the second to taking the Hilbert transforms of the oriented sub-bands. To be more precise, we define the linear operators $\mathcal{W}$ and $\mathcal{W}_0$ by

$$\begin{array}{rccc} \mathcal{W}: & E & \longrightarrow & E^{Q+1} \\ & u & \longmapsto & (b_0 * u, \ldots, b_{Q-1} * u, \downarrow_2 (l * u)) \end{array} \tag{23}$$

and

$$\begin{array}{rccc} \mathcal{W}_0: & E & \longrightarrow & E^{Q+2} \\ & u & \longmapsto & (h_0 * u, \mathcal{W}(l_0 * u)) \end{array} \tag{24}$$

Then, for all integers $n \geq 1$, we define the linear operator $R^n_{\mathcal{W}}$ associated to the linear operator $\mathcal{W}$,

$$\begin{array}{rccc} R^n_{\mathcal{W}}: & E^{n+1} & \longrightarrow & E^{n+Q+1} \\ & (u_0, \ldots, u_n) & \longmapsto & (u_0, \ldots, u_{n-1}, \mathcal{W}(u_n)). \end{array} \tag{25}$$

This allows one to define the recursive application of $\mathcal{W}$. Hence, the linear operator $\mathcal{P}_{\mathcal{R}}$ corresponding to the real steerable pyramid decomposition previously described in [10] is

$$\mathcal{P}_{\mathcal{R}} = R^{(P-1)Q+1}_{\mathcal{W}} \circ \cdots \circ R^{Q+1}_{\mathcal{W}} \circ \mathcal{W}_0. \tag{26}$$

A total of $PQ + 2$ real sub-bands are created during the real decomposition. In order to define the complex decomposition, we only need to define the additional linear operator $\tilde{\mathcal{H}}$ that consists in computing the Hilbert transforms of the PQ oriented sub-bands while leaving the high-pass (first position) and low-pass (last position) residuals unchanged. This is given by

$$\begin{array}{rccc} \tilde{\mathcal{H}}: & E^{PQ+2} & \longrightarrow & E^{PQ+2} \\ & & & (u_0, \quad m_0 * u_1, \quad \ldots, \quad m_{Q-1} * u_Q, \\ & (u_0, \ldots, u_{PQ+1}) & \longmapsto & \vdots \quad \ldots \quad \vdots \\ & & & m_0 * u_{(P-1)Q}, \quad \ldots, \quad m_{Q-1} * u_{PQ}, \quad u_{PQ+1}). \end{array} \tag{27}$$

Finally, the complex steerable pyramid operator $\mathcal{P}_{\mathcal{C}}$ is given by

$$\mathcal{P}_{\mathcal{C}} = \tilde{\mathcal{H}} \circ \mathcal{P}_{\mathcal{R}}. \tag{28}$$

The complex steerable pyramid decomposition is illustrated in the left-hand side of Figure 3 and the corresponding algorithm is detailed in Algorithm 2.

Reconstruction. The complex steerable pyramid operator admits a pseudoinverse, which, using (28), comes down to

$$\mathcal{P}_{\mathcal{C}}^{\dagger} = (\tilde{\mathcal{H}} \circ \mathcal{P}_{\mathcal{R}})^{\dagger} = \mathcal{P}_{\mathcal{R}}^{\dagger} \circ \tilde{\mathcal{H}}^{\dagger}. \tag{29}$$

The pseudoinverse of $\tilde{\mathcal{H}}$ is

$$\begin{array}{rccc} \tilde{\mathcal{H}}^{\dagger}: & E^{PQ+2} & \longrightarrow & E^{PQ+2} \\ & (u_0, \dots, u_{PQ+1}) & \longmapsto & (u_0, \mathrm{Re}(u_1), \dots, \mathrm{Re}(u_{PQ}), u_{PQ+1}) \end{array} \tag{30}$$

where Re denotes the real part operator. This follows from an ad hoc property of the Hilbert transform [31]. In order to define the pseudoinverse of $\mathcal{P}_{\mathcal{R}}$, we need to introduce the pseudoinverses of $R_{\mathcal{W}}^n$, $\mathcal{W}$ and $\mathcal{W}_0$. The pseudoinverse of $\mathcal{W}$ is the reconstruction of an image from its sub-bands at a single scale

$$\begin{array}{rccc} \mathcal{W}^{\dagger}: & E^{Q+1} & \longrightarrow & E \\ & (u_0, \dots, u_Q) & \longmapsto & l * \uparrow_2 (u_Q) + \sum_{q=0}^{Q-1} (b_q * u_q). \end{array} \tag{31}$$

From the pseudoinverse $\mathcal{W}^{\dagger}$ we can define the pseudoinverses of $R_{\mathcal{W}}^n$ and $\mathcal{W}_0$ as

$$\begin{array}{rccc} {R_{\mathcal{W}}^n}^{\dagger}: & E^{n+Q+1} & \longrightarrow & E^{n+1} \\ & (u_0, \dots, u_{n-1}, u_n, \dots, u_{n+Q}) & \longmapsto & (u_0, \dots, u_{n-1}, \mathcal{W}^{\dagger}(u_n, \dots, u_{n+Q})), \end{array} \tag{32}$$

and

$$\begin{array}{rccc} \mathcal{W}_0^{\dagger}: & E^{Q+2} & \longrightarrow & E \\ & (u_0, u_1, \dots, u_{Q+1}) & \longmapsto & h_0 * u_0 + l_0 * \mathcal{W}^{\dagger}(u_1, \dots, u_{Q+1}). \end{array} \tag{33}$$

Finally, the pseudoinverse of $\mathcal{P}_{\mathcal{R}}$ writes

$$\mathcal{P}_{\mathcal{R}}^{\dagger} = \mathcal{W}_0^{\dagger} \circ {R_{\mathcal{W}}^{Q+1}}^{\dagger} \circ \cdots \circ {R_{\mathcal{W}}^{(P-1)Q+1}}^{\dagger}. \tag{34}$$

The complex steerable pyramid reconstruction is illustrated in the right-hand side of Figure 3 and the corresponding algorithm is detailed in Algorithm 3.

2.3 Filtering of Discrete Images

Because the steerable pyramid filters are defined by continuous functions in the Fourier domain, we have described the steerable pyramid decomposition and reconstruction in terms of operators over $E = L^2(\mathbb{R}^2)$ functions. In practice, images are a finite collection of samples corresponding to gray level intensities. Applying filters defined by continuous functions in the Fourier domain to discrete images is usually performed using computations based on the discrete Fourier transform (DFT). These computations are theoretically justified in [9].

In this section we describe the practical method used to apply the filters to discrete images. In the following, M and N are two even numbers corresponding to the image size.

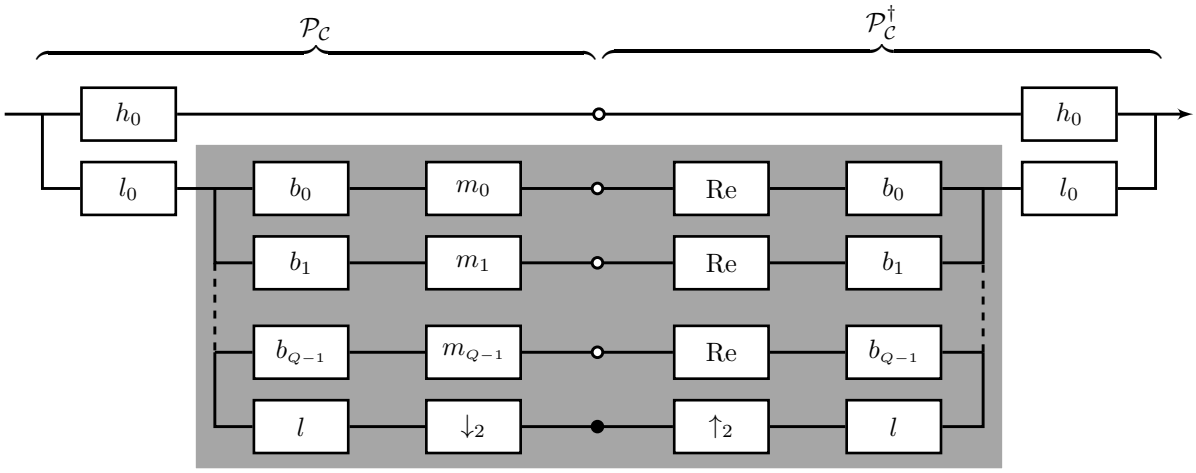

Figure 3: Block diagram representing the complex steerable pyramid decomposition (left-hand side) and reconstruction (right-hand side). The white dots represent single sub-bands while the black dots represent the sub-bands at the next scale. The symbol Re is the real part operator while $\downarrow_2$ and $\uparrow_2$ are respectively the zoom-out and zoom-in operators (see Definition 7).

Definition 8 (Discrete domains)**.** *The discrete spatial domain $\Omega_{M,N}$ is defined by*

$$\Omega_{M,N} = \{0, \dots, M-1\} \times \{0, \dots, N-1\}. \tag{35}$$

The discrete Fourier domain $\hat{\Omega}_{M,N}$, associated to $\Omega_{M,N}$, is defined by

$$\hat{\Omega}_{M,N} = \left\{-\frac{M}{2}, \dots, \frac{M}{2}-1\right\} \times \left\{-\frac{N}{2}, \dots, \frac{N}{2}-1\right\}. \tag{36}$$

Definition 9 (Discrete Fourier transform)**.** *The discrete Fourier transform (DFT) of $u \in \mathbb{C}^{\Omega_{M,N}}$ is denoted $\mathcal{F}_{M,N}(u) \in \mathbb{C}^{\hat{\Omega}_{M,N}}$ and is defined by*

$$\forall (m,n) \in \hat{\Omega}_{M,N}, \quad \mathcal{F}_{M,N}(u)_{m,n} = \sum_{(k,l)\in\Omega_{M,N}} u_{k,l} e^{-2\pi i (k\frac{m}{M} + l\frac{n}{N})}. \tag{37}$$

The inverse discrete Fourier transform (iDFT) of $v \in \mathbb{C}^{\hat{\Omega}_{M,N}}$ is denoted $\mathcal{F}^{-1}_{M,N}(v) \in \mathbb{C}^{\Omega_{M,N}}$ and is defined by

$$\forall (k,l) \in \Omega_{M,N}, \quad \mathcal{F}^{-1}_{M,N}(v)_{k,l} = \frac{1}{MN} \sum_{(m,n)\in\hat{\Omega}_{M,N}} v_{m,n} e^{2\pi i (m\frac{k}{M} + n\frac{l}{N})}. \tag{38}$$

Discrete filtering algorithm. Let u be an image of size $M \times N$ and f be a filter defined by a continuous function F in the Fourier domain. Following [9], the filtered image $f * u$ is computed by a component-wise multiplication in the Fourier domain as described in Algorithm 1.

During the steerable pyramid computations (see Algorithm 2 and Algorithm 3) the filters to be applied are the filters $f \in \{l_0, h_0, b_q, b_q * m_q, q \in \{0, \dots, Q-1\}\}$. The corresponding Fourier transforms are the continuous functions $F = \phi \circ \rho$ where $\phi \in \{H_0, L_0, B_q, B_q M_q, q \in \{0, \dots, Q-1\}\}$ so that Algorithm 1 can be applied.

2.4 Practical Decomposition and Reconstruction Algorithms

The practical algorithms for the steerable pyramid decomposition and reconstruction of discrete images are detailed in Algorithm 2 and Algorithm 3. Linear filtering is performed using Algorithm 1.

Algorithm 1: Application of a linear filter

Input : An image u of size $M \times N$ and a filter f defined by a continuous function F in Fourier domain

Output: The filtered image $f * u$

1 Compute $\mathcal{F}_{M,N}(u)$ the DFT of u

2 **for** $(m,n) \in \hat{\Omega}_{M,N}$ **do**

3 $\quad \widehat{f * u}_{m,n} \leftarrow \mathcal{F}_{M,N}(u)_{m,n} \cdot F\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right)$

4 Compute $f * u = \mathcal{F}^{-1}_{M,N}(\widehat{f * u})$ with an iDFT

Image sizes. The zoom-out and zoom-in operators of the theoretical decomposition and reconstruction (see Section 2.2) are respectively replaced by down-sampling and up-sampling by factor 2. This is also done using DFT computations: this corresponds, for the down-sampling, to DFT coefficients cropping and, for the up-sampling, to the so-called zero-padding. More details about image resampling are given in [8].

Because of the down-sampling, the pyramid decomposition is well-defined if and only if the width M and the height N of the input image u are proportional to 2^{P+1}. This will be assumed throughout the paper. Otherwise, a preliminary crop should be performed to ensure this condition.

The sizes of the $PQ + 2$ sub-bands computed during the decomposition are given in Table 1.

DFT computations. In practice, DFTs and iDFTs are computed using the FFT algorithm [15]. Computations are saved by avoiding the unnecessary DFT and iDFT computations between successive steps.

Optimizations. In Algorithm 2, the low-frequency band v is first down-sampled and then filtered by l_0 (see Line 6 and Line 7) while, according to (23), v should be first filtered by l and then down-sampled. The two possibilities are equivalent but here computations are saved as the filtered images are smaller. The corresponding optimization is also used in Algorithm 3 (see Line 3 and Line 4).

Using the linearity of the real part operator, computations are saved in Algorithm 3. Instead of applying it to all oriented sub-bands $s_{p,q}$, it is applied only once in Line 8.

3 The Portilla and Simoncelli Algorithm

The PS algorithm consists of two steps described as follows.

Step 1: Analysis. Compute the complex steerable decomposition of the texture example from which new texture samples will be generated, see details in Section 2.2. Compute the summary statistics of the pyramid sub-bands as given in Section 3.1.

Step 2: Synthesis. The output texture is synthesized using the following iterative scheme. Initialize with a white noise image. At each iteration, compute the complex steerable pyramid of the current output texture. Then, from the coarsest to the finest scale and for each summary statistic, impose it to the wavelet coefficients as detailed in Section 3.2. When all wavelet coefficients at one scale have the desired statistics, recompose them together to obtain the upper scale low-pass sub-band. The reconstruction is performed as presented in Section 2.2 (operator (31)).

	Notation	Number of images	Size
High-frequency band	h_{band}	1	$M \times N$
Oriented bands	$s_{p,q}$	PQ	$\frac{M}{2^{p-1}} \times \frac{N}{2^{p-1}}$ at scale $p \in \{1, \dots, P\}$
Low-frequency residual	l_{band}	1	$\frac{M}{2^P} \times \frac{N}{2^P}$

Table 1: Summary of the images computed during the steerable pyramid decomposition with P scales and Q orientations (see Algorithm 2). A total of $PQ + 2$ images is computed. The oriented bands are complex-valued images while the high-frequency band and the low-frequency residual are real-valued images.

Algorithm 2: Complex steerable pyramid decomposition

Input : A number of scales P, a number of orientations Q, a discrete image u of size $M \times N$ such that M and N are multiples of 2^{P+1}

Output: The steerable pyramid of u: sequence of $PQ + 2$ images $\{h_{\text{band}}, (s_{p,q}), l_{\text{band}}\}$ (see Table 1)

1. Compute the high-frequency residual $h_{\text{band}} = h_0 * u$ and store it in the pyramid
2. Compute the low-frequency band $v = l_0 * u$
3. **for** scale $p = 1$ **to** P **do**
4. | **for** orientation $q = 0$ **to** $Q - 1$ **do**
5. | | Compute the oriented high-frequency band $s_{p,q} = (m_q * b_q) * v$ and store it in the pyramid
6. | Down-sample v by a factor of 2
7. | Update $v \leftarrow l_0 * v$
8. Store v in the pyramid as the low-frequency residual l_{band}

Algorithm 3: Complex steerable pyramid reconstruction

Input : A steerable pyramid $\{h_{\text{band}}, (s_{p,q}), l_{\text{band}}\}$ having P scales and Q orientations

Output: The reconstructed image u

1 Initialize u as the low-frequency residual l_{band}
2 **for** scale $p = P$ **to** 1 **do**
3 | Update $u \leftarrow l_0 * u$
4 | Up-sample u by a factor of 2
5 | **for** orientation $q = 0$ **to** $Q - 1$ **do**
6 | | Compute $b_q * s_{p,q}$
7 | | Update $u \leftarrow u + b_q * s_{p,q}$
8 | Update $u \leftarrow \text{Re}(u)$
9 Update $u \leftarrow l_0 * u$
10 Compute $h_0 * h_{\text{band}}$
11 Update $u \leftarrow u + h_0 * h_{\text{band}}$

3.1 The Summary Statistics

In order to find the relevant statistical constraints, Portilla and Simoncelli have used an empirical approach based on a large texture dataset composed of artificial and natural textures covering most of the features found in textures. By only considering marginal and cross-statistics, they added different statistical constraints on the sub-bands one by one while testing the algorithm on the texture dataset. After adding a constraint, the results were subjectively classified according to the type of missing features. A new statistical constraint was then chosen whether the features were appropriately synthesized or not. Finally, each constraint was verified to be necessary by exhibiting synthesis failures. Superfluous constraints were removed. We exemplify in Section 4 how each statistical constraint impacts the synthesis.

Definitions. Now, we define the statistics of discrete images of size $M \times N$ that are required by the PS algorithm. First, marginal statistics are given in terms of central moments that have the following definition for an image v and an integer $n \geq 1$,

$$\mu_n(v) = \begin{cases} \dfrac{1}{MN} \displaystyle\sum_{(k,l)\in\Omega_{M,N}} v_{k,l} & \text{if} \quad n = 1, \\ \dfrac{1}{MN} \displaystyle\sum_{(k,l)\in\Omega_{M,N}} (v_{k,l} - \mu_1(v))^n & \text{if} \quad n > 1. \end{cases} \tag{39}$$

Then, the mean $\mu(v)$ and variance $\sigma^2(v)$ of u are simply $\mu(v) = \mu_1(v)$ and $\sigma^2(v) = \mu_2(v)$. Assuming $\mu_2(v) \neq 0$, its skewness $\eta(v)$ and kurtosis $\kappa(v)$ are given by

$$\eta(v) = \frac{\mu_3(v)}{\mu_2(v)^{3/2}} \quad \text{and} \quad \kappa(v) = \frac{\mu_4(v)}{\mu_2(v)^2}. \tag{40}$$

The maximum and minimum of v are given by

$$\max(v) = \max_{(k,l)\in\Omega_{M,N}} v_{k,l} \quad \text{and} \quad \min(v) = \min_{(k,l)\in\Omega_{M,N}} v_{k,l}. \tag{41}$$

The auto-correlation of an image v in a neighborhood of the origin $\mathcal{N}$ of size $|\mathcal{N}| = N_a^2$ is given by

$$\forall (i,j) \in \mathcal{N}, \quad R(v)_{i,j} = \frac{1}{MN} \sum_{(k,l)\in\Omega_{M,N}} (v_{k,l} - \mu(v))(v_{k-i,l-j} - \mu(v)), \tag{42}$$

where an implicit periodic boundary extension is used. For a collection of $L > 0$ images of the same size $(v^{(1)}, \dots, v^{(L)})$, the matrix C of pairwise cross-correlations has coefficients

$$\forall (i,j) \in \{1,\dots,L\}^2, \quad C_{i,j} = \mu\Big(\big(v^{(i)} - \mu(v^{(i)})\big) \cdot \big(v^{(j)} - \mu(v^{(j)})\big)\Big), \tag{43}$$

where the $\cdot$ symbol denotes the element-wise product. For another collection of $L' > 0$ images of the same size $(w^{(1)}, \dots, w^{(L')})$, the matrix D of cross-correlations has coefficients

$$\forall (i,j) \in \{1,\dots,L\} \times \{1,\dots,L'\}, \quad D_{i,j} = \mu\Big(\big(v^{(i)} - \mu(v^{(i)})\big) \cdot \big(w^{(j)} - \mu(w^{(j)})\big)\Big). \tag{44}$$

In the following the images $(w^{(i)})_i$ have the same size as or half the size of the images $(v^{(i)})_i$. If they have half the size, these images are upsampled to the appropriate size using zero-padding in the Fourier domain. Specifically, this is done when computing the summary statistics (v) and (vi) introduced below.

Summary statistics. Using the empirical approach described above, Portilla and Simoncelli found the following set of summary statistics to be "necessary and sufficient" to synthesize a wide class of natural textures:

(i) Marginal statistics:
  (a) Skewnesses and kurtoses of the low-frequency bands at each scale,
  (b) Variance of the high-frequency band,
  (c) Skewness, kurtosis, variance, mean, maximum and minimum of the reconstructed texture,

(ii) Auto-correlations of the low-frequency bands at each scale,

(iii) Auto-correlations of the modulus of each oriented sub-band,

(iv) Pairwise cross-correlations of the modulus of all oriented sub-bands at the same scale (size $Q \times Q$),

(v) Cross-correlations of the modulus of all oriented sub-bands with all oriented sub-bands at the coarser scale (size $Q \times Q$),

(vi) Cross-correlations of the real part of each oriented sub-band with both the real and imaginary part of all phase-doubled oriented sub-bands at the next coarser scale (Q matrices of size $1 \times 2Q$, which are computed and stored in a matrix of size $Q \times 2Q$).

Remarks. In addition, the means of the modulus of each oriented sub-band are computed and stored. These values are not technically summary statistics but they are required since the auto-correlation and cross-correlation adjustments are performed on zero-mean images.

The term "low-frequency bands" in (i)a and (ii) actually refers to low-frequency bands filtered by l_0. For instance, for the low-frequency residual, the summary statistics are computed on $l_0 * l_{\text{band}}$.

Note that the phases of the oriented sub-bands at the next coarser scale are doubled in (vi). Indeed, the local phase varies twice slower at the coarser scale. See [41, Section 2.2] for more details.

3.2 Constraints Adjustment

Random texture sampling. Once the summary statistics are established, the goal is to generate a new texture sample v from a texture example u. The summary statistics of the texture example are constraints that the new texture sample must match as closely as possible. To achieve this, an iterative scheme is used and the texture sample is initialized with a white noise image. At each step, the texture sample is passed to the complex steerable pyramid operator and its sub-bands are successively modified to satisfy the statistical constraints while inverting the complex steerable pyramid operator (i.e. reconstructing an input image with the modified sub-bands). One iteration of the synthesis algorithm is therefore a modified version of the reconstruction algorithm described in Algorithm 3 where sub-band statistics are adjusted while reconstructing an input image. The adjustment of the summary statistics is detailed in Algorithm 4. The full texture synthesis is detailed in Algorithm 5. Extension to color texture synthesis is explained in Appendix B.

Remarks. In Line 10 of Algorithm 4, the cross-correlation adjustment of the magnitudes is performed simultaneously for all the sub-bands while for the cross-correlation adjustment of the real parts, in Line 17, only one sub-band is modified at a time (only one row of (vi) is used at once).

In Line 19, the Hilbert transform of $\text{Re}(s_{p,q})$ is computed in order to recover an analytic signal. Although the Fourier transform of the filter m_q is discontinuous, Algorithm 1 is still used to compute $m_q * \text{Re}(s_{p,q})$.

As in [41], a convergence accelerator is used in Line 8 of Algorithm 5. This is a simpler version of the heavy-ball method [39] (or the Nesterov method [37]). The momentum term $0.8(v - v_{\text{old}})$ is added to the "ball" to mitigate zigzagging.

Note that the size of the synthesized texture is not specified in Algorithm 5. Any multiple of 2^{P+1} can be chosen. By default, it has the same size as the input texture example (possibly cropped).

Imposing sub-bands statistics. In the following paragraph we denote by v a generic image. In practice, the image v is the optimized image or any function of its sub-bands at a given iteration of the algorithm. The goal is to adjust the image v so that $K \in \mathbb{N}^*$ statistical estimators $\phi = \{\phi_1, \dots, \phi_K\}$ have the desired values $s = \{s_1, \dots, s_K\} \in \mathbb{R}^K$. A standard approach consists in minimizing the squared error

$$L(v) = \|\phi(v) - s\|^2 \overset{\text{def.}}{=} \sum_{k=1}^{K} (\phi_k(v) - s_k)^2. \tag{45}$$

Instead, the PS algorithm requires to consider the family of constraint sets $(S_{\phi_1}, \dots, S_{\phi_K})$ where

$$\forall k \in \{1, \dots, K\}, \quad S_{\phi_k} = \{v \,|\, \phi_k(v) = s_k\}, \tag{46}$$

such that the constraint is satisfied if and only if $v \in S = \cap_{k=1}^{K} S_{\phi_k}$. In order to project v onto S, the authors use the alternate projections method [34]. Although the hypotheses for convergence are not verified in general, the proposed algorithm empirically converges (see Section 4).

Projection on a set S_{ϕ_k} is performed by minimizing the squared error $L_k(v) = (\phi_k(v) - s_k)^2$. To achieve the minimization, a gradient descent with a single optimal step is used

$$\bar{v} = v + \lambda_k \phi_k'(v). \tag{47}$$

The step is optimal in the sense that λ is chosen such that $L_k(\bar{v})$ is minimized. If an image must satisfy $K' < K$ constraints, the successive gradient steps lead to a single step that writes

$$\bar{v} = v + \sum_{k=1}^{K'} \lambda_k \phi_k'(v). \tag{48}$$

This is the case for the auto-correlations (ii) and (iii) and for the cross-correlations (iv), (v) and (vi). The details of the constraints adjustment are given in Appendix A.

Texture interpolation. Interpolation between two textures is achieved by linearly interpolating the vectors of summary statistics $s_1, s_2 \in \mathbb{R}^{n_1 \times \cdots \times n_K}$ of two textures examples u_1 and u_2. Synthesis of the interpolation of u_1 and u_2 with respective weights t and $1-t$ with $t \in [0, 1]$ is achieved using the same algorithm but with the constraints $s_t = ts_1 + (1-t)s_2$. Such interpolation have been used in vision studies in restricted settings (gray level textures, somehow similar textures). We have extended the algorithm to color texture interpolations. Examples are shown in Section 4.

4 Experiments

As described in the first paragraph of Section 3.1, Portilla and Simoncelli use a large dataset of textures to determine the relevant statistical constraints. To do so, they create six categories of textures: two categories gather periodic and aperiodic artificial textures; three categories gather quasi-periodic, random and structured natural textures and one last category is composed of natural images, which are not textures. Some examples of textures for each category both with their synthesis are shown in Figure 6. Those examples were synthesized using the default parameters of our algorithm i.e. $P = 4$ scales, $Q = 4$ orientations and central auto-correlation size $N_a = 7$ but with *periodic+smooth* decomposition [35]. In the following paragraphs, we check first the empirical convergence of the algorithm.

Algorithm 4: Adjustment of the summary statistics for grayscale images

Input : A pyramid decomposition $\{h_{\text{band}}, (s_{p,q}), l_{\text{band}}\}$ with P scales and Q orientations, and the target summary statistics

Output: A discrete texture sample v with approximately adjusted statistics

Initialize $v \leftarrow l_{\text{band}} - \mu(l_{\text{band}})$
Update $v \leftarrow l_0 * v$
// Adjustment of the low-frequency constraints (i)a and (ii)
Adjust the auto-correlation of v (see Appendix A.2)
Adjust the skewness of v (see Appendix A.1.3)
Adjust the kurtosis of v (see Appendix A.1.4)
for scale $p = P$ **to** 1 **do**
    Up-sample v by a factor of 2
    // Adjustment of the magnitude of sub-bands (iii), (iv) and (v)
    for orientation $q = 0$ **to** $Q - 1$ **do**
        Initialize $a_{p,q} \leftarrow |s_{p,q}| - \mu(|s_{p,q}|)$ and, **if** $p < P$, update $a_{p+1,q} \leftarrow |s_{p+1,q}| - \mu(|s_{p+1,q}|)$
    Adjust the cross-correlation of the magnitudes $(a_{p,q})_q$, including the cross-correlation with their parent magnitudes $(a_{p+1,q})_q$ **if** $p < P$ (see Appendix A.3)
    for orientation $q = 0$ **to** $Q - 1$ **do**
        Adjust the auto-correlation of $a_{p,q}$ (see Appendix A.2)
        Adjust the mean of $a_{p,q}$ (see Appendix A.1.2) and take the positive part $\max(0, a_{p,q})$
        Adjust the modulus of $s_{p,q}$ to $a_{p,q}$ without changing its phase (see Appendix A.4)
    // Adjustment of the real part of sub-bands (vi)
    if $p < P$ **then**
        for orientation $q = 0$ **to** $Q - 1$ **do**
            Adjust the cross-correlation of the real part $\text{Re}(s_{p,q})$ with its phase-doubled parents' real and imaginary parts $(\text{Re}(s^2_{p+1,q}/|s_{p+1,q}|), \mathcal{I}\text{m}(s^2_{p+1,q}/|s_{p+1,q}|))_q$ (see Appendix A.3)
    // Combine the oriented sub-bands into the low-frequency band
    for orientation $q = 0$ **to** $Q - 1$ **do**
        Compute the Hilbert transform $s_{p,q} \leftarrow m_q * \text{Re}(s_{p,q})$ to get an analytic signal
        Update $v \leftarrow v + b_q * s_{p,q}$
    Update $v \leftarrow \text{Re}(v)$
    Update $v \leftarrow l_0 * v$
    // Adjustment of the low-frequency constraints (i)a and (ii)
    Adjust the auto-correlation of v (see Appendix A.2)
    Adjust the skewness of v (see Appendix A.1.3)
    Adjust the kurtosis of v (see Appendix A.1.4)
// Adjustment of the high-frequency constraints (i)b
Adjust the variance of h_{band} (see Appendix A.1.2)
Update $h_{\text{band}} \leftarrow h_0 * h_{\text{band}}$
Update $v \leftarrow v + h_{\text{band}}$
// Adjustment of the image constraints (i)c
Adjust the variance of v and set the mean to 0 (see Appendix A.1.2)
Adjust the skewness of v (see Appendix A.1.3)
Adjust the kurtosis of v (see Appendix A.1.4)
Adjust the mean of v (see Appendix A.1.2)
Adjust the range of v (see Appendix A.1.1)

Algorithm 5: Portilla-Simoncelli texture synthesis algorithm for grayscale images

Input : A number of scales P, a number of orientations Q, a number of iteration N_{it}, a discrete texture example u of size $M \times N$ such that M and N are multiples of 2^{P+1}

Output: A synthesized discrete texture sample v

// Input analysis

Compute the steerable pyramid of u using Algorithm 2

Compute from the pyramid and from u the summary statistics listed in Section 3.1

// Output synthesis

Initialize v with a Gaussian white noise image with the same mean and variance as u

Initialize $v_{\text{old}} \leftarrow v$

for iteration $i = 1$ **to** N_{it} **do**

  Compute the steerable pyramid of v using Algorithm 2

  Update v by imposing the summary statistics to the pyramid decomposition using Algorithm 4

  // Convergence accelerator

  Update $v \leftarrow v + 0.8(v - v_{\text{old}})$

  Update $v_{\text{old}} \leftarrow v$

Then, we analyze the influence the constraint sets and of two parameters of the algorithm. Finally, we show several examples of color texture interpolations. We do not compare the results to other algorithms because this is already extensively done in the literature (see Section 1 and references therein).

Running time comparison. In Table 2, a running time comparison shows that our C++ implementation is much faster than the existing MATLAB[3] implementation. The running times correspond to the syntheses of a single 256×256 color texture with default parameters. For the MATLAB implementation, all `.mex` files were pre-compiled. Syntheses were run on a Dell XPS 15 9500 equipped with Intel Core i7-10750H CPU at 2.60GHz $\times$ 12.

	MATLAB	C++ (ours)
Running time (s)	42.3 ± 0.02	10.5 ± 0.08

Table 2: Running time comparison of the MATLAB and C++ implementations for a single 256×256 color texture with default parameters (estimated from the synthesis of 139 textures). The time after $\pm$ corresponds to one standard error of the mean. For the MATLAB implementation, all `.mex` files were pre-compiled. Syntheses were run on a Dell XPS 15 9500 equipped with Intel Core i7-10750H CPU at 2.60 GHz $\times$ 12.

Empirical convergence. As explained in Section 3.2, the PS algorithm uses an alternate projection method to minimize the loss L defined by (45). In addition, the constraints are adjusted while reconstructing the image from its sub-bands. Therefore, the adjustments at a finer scale or of another constraint could compromise the adjustments at a coarser scale or of other constraints. In order to empirically verify those possible issues, we computed the evolution of the loss L and of the partial loss functions (i.e. the sum of L_k for subsets of the constraint set corresponding to constraints (i)a, (ii), (iii), (iv), (v) and (vi) at each scale) with the number of iterations. The results for 177 textures are reported in Figure 4 and Figure 5 where we plotted the log-loss functions. The average global loss is a non-increasing function of the iterations. This is also the case for all partial loss

[3] http://www.cns.nyu.edu/~lcv/texture/

functions except for the marginal statistics (i)a of the residual (right of the second row in Figure 4). We conclude that neither the alternating between constraint adjustments and the reconstruction nor the alternating between constraint adjustments are preventing the loss from decreasing at all scales and for all constraint types.

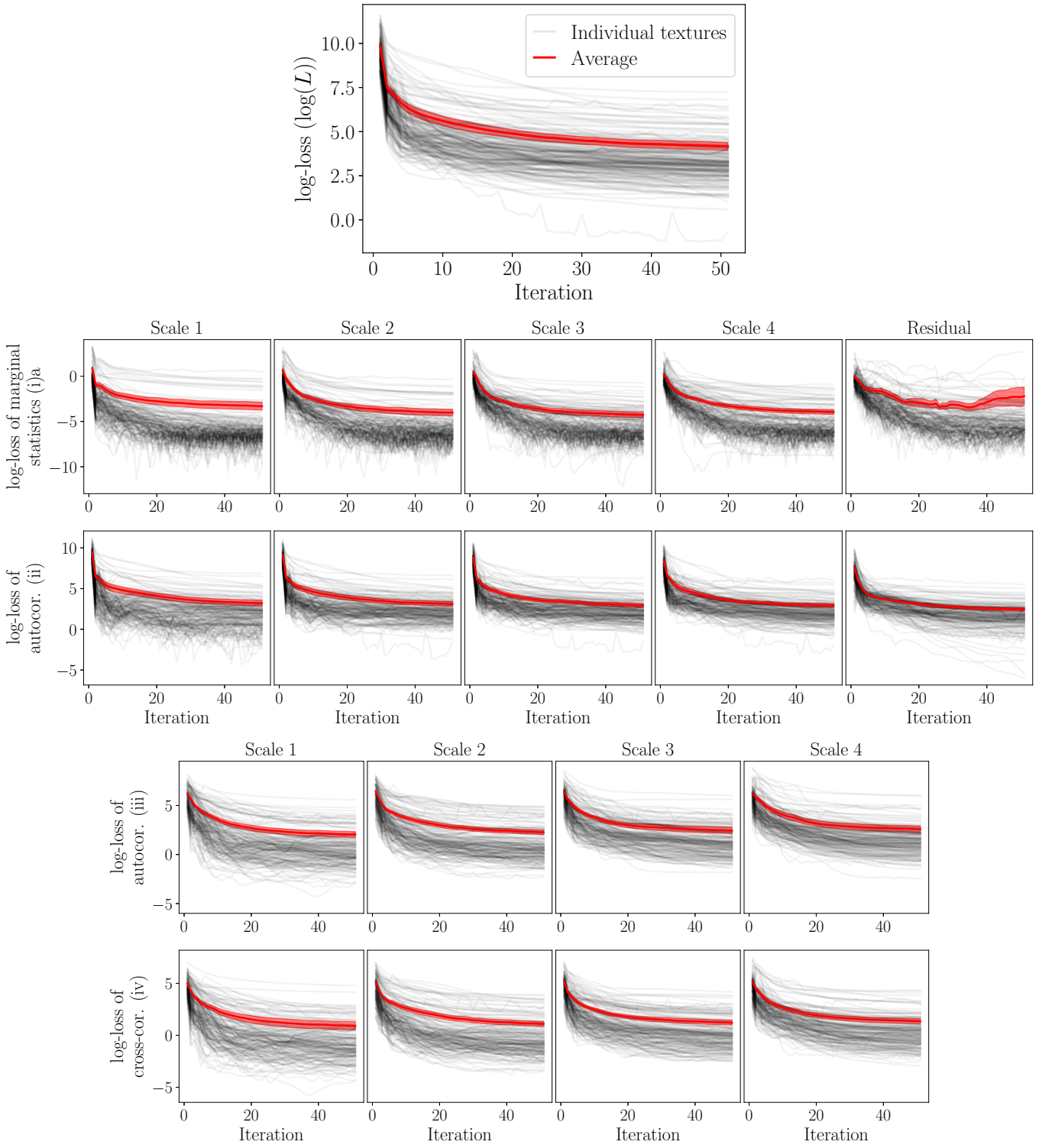

Figure 4: From top to bottom: logarithm of the loss function (45) and logarithm of the partial loss function for the group of constraints (i)a, (ii), (iii), (iv). Error bars in red indicate 3 standard error of the mean computed over 177 textures. Results for individual textures are represented in faint gray. The figure is best seen in color.

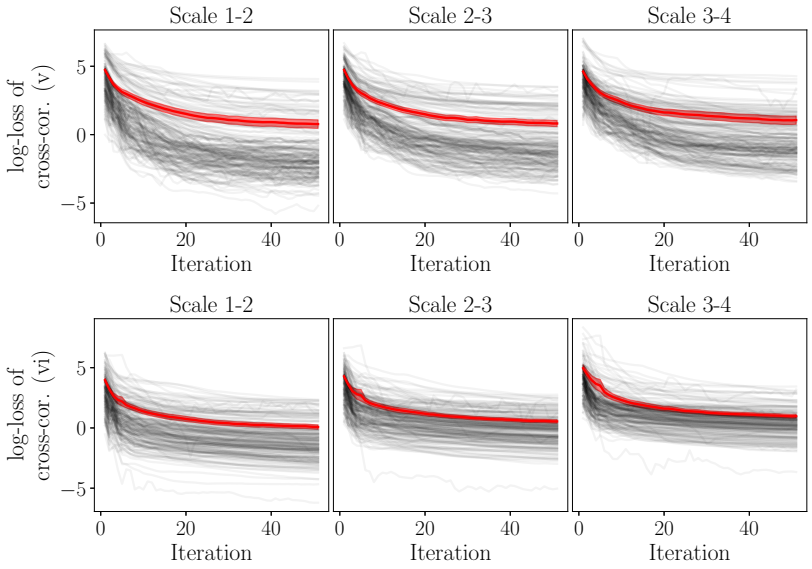

Figure 5: Top and bottom: logarithm of the partial loss function for the group of constraints (v) and (vi). Error bars in red indicate 3 standard error of the mean computed over 177 textures. Results for individual textures are represented in faint gray. The figure is best seen in color.

Influence of the constraint type. In order to illustrate the trial and error methodology of Portilla and Simoncelli, we use four groups of statistical constraints: C_1 the group of marginal statistics (see (i)); C_2 the group of auto-correlations (see (ii) and (iii)); C_3 the group of magnitude correlations (see (iv) and (v)) and C_4 the group of phase correlations (see (vi)). These groups have been used in vision studies [1, 38]. Figure 7 displays two textures from each texture category and their synthesis with all statistical constraints, and all statistical constraints but one of each of the 4 groups. Qualitatively, the absence of marginal statistic constraints (C_1) affects the luminance and color of the textures (rows 3, 4, 5, 9 and 11) while the absence of auto-correlation constraints (C_2) affects their spatial regularity (rows 1, 2, 4 and 5). The lack of magnitude correlations (C_3) favours the high spatial frequency compared to the low spatial frequency (rows 4, 5, 6, 8, 9 and 10). Finally, the lack of phase correlations (C_4), albeit subtle, creates wrong phases compared to the original texture (opposite shading in row 4, dark oblique lines in row 5).

Influence of central autocorrelation size and edge handling. We did not test the pyramid parameters i.e. the number of scales and orientations because they have already been tested in [10] and have a similar effect on the synthesis. Yet, we test two aspects of the synthesis algorithm. First, the size of the auto-correlation neighborhood N_a and second, the effect of the *periodic+smooth* decomposition [35]. All other parameters were set to default. Figure 8 displays two textures from each texture category and their synthesis with varying neighborhood size (columns 2 to 5). The main effect of increasing the neighborhood size is to better capture the quasi-periodicity/stationarity of low-frequencies, yet the synthesized texture often gets worse or does not improve above $N_a = 7$ except for few structured textures (rows 9 and 10). By comparing the second and last columns, we observe that the *periodic+smooth* decomposition reduces the amount of sharp edges due to boundary discontinuities (e.g. rows 6 and 12).

Interpolation of two textures. To conclude and as an original contribution of this paper, we show many examples of interpolation between arbitrarily chosen color textures in Figure 9. All interpolations were synthesized using default parameters but with the *periodic+smooth* decomposition to avoid boundary artifacts. The interpolations are perceptually smooth and seem therefore suitable

to study visual perception [18, 38]. Yet, other approaches using a combination of optimal transport and CNN also provide examples that are suitable to study visual perception [59].

5 Conclusion

In this work, we reviewed the impact of the Portilla and Simoncelli texture synthesis algorithm from mathematical imaging to the neurosciences of vision. We provided a precise description of the mathematical framework both with pseudo-codes and a C++ implementation. Our code is much faster than the existing MATLAB implementation (see Table 2). Our implementation also comprises the extension to perform texture interpolation that was used in vision studies [38, 18] and the possibility to handle non-periodicity using the *periodic+smooth* decomposition [35]. Additionally, we illustrated the empirical approach used to design the set of statistical constraints by gathering textures into 6 categories as proposed in the original article [41]. We analyzed the effect of the central autocorrelation size and of the *periodic+smooth* decomposition. Finally, we showed several examples of interpolation between textures.

Appendix A Details of the Constraints Adjustment

The following sections describe the adjustments, used in Algorithm 4, of each statistic mentioned in Section 3.1. We denote by u the image with the target statistics and by v the image of size $M \times N$ to be adjusted. In addition, $\mathbb{1}$ is the image with all pixels equal to 1.

A.1 Marginal Statistics Adjustment

The marginal statistics listed in Section 3.1 (see (i)) are adjusted as follows.

A.1.1 Range Adjustment

The maximum and minimum of v are imposed by clamping

$$\forall (k,l) \in \Omega_{M,N}, \quad \tilde{v}_{k,l} = \begin{cases} \max(u) & \text{if} \quad v_{k,l} > \max(u), \\ \min(u) & \text{if} \quad v_{k,l} < \min(u), \\ v_{k,l} & \text{otherwise.} \end{cases} \tag{49}$$

A.1.2 Mean and Variance Adjustment

Assume that $\sigma(v) \neq 0$. Then, the mean $\mu(v)$ and the variance $\sigma^2(v)$ of v are jointly imposed by the affine transform

$$\tilde{v} = \frac{\sigma(u)}{\sigma(v)}\big(v - \mu(v)\mathbb{1}\big) + \mu(u)\mathbb{1}. \tag{50}$$

Particular case. Assume that $\mu(v) = 0$. Then, its mean can be imposed by a simple shift

$$\tilde{v} = v + \mu(u)\mathbb{1}. \tag{51}$$

Similarly, its variance can be imposed by a simple multiplication

$$\tilde{v} = \frac{\sigma(u)}{\sigma(v)} v. \tag{52}$$

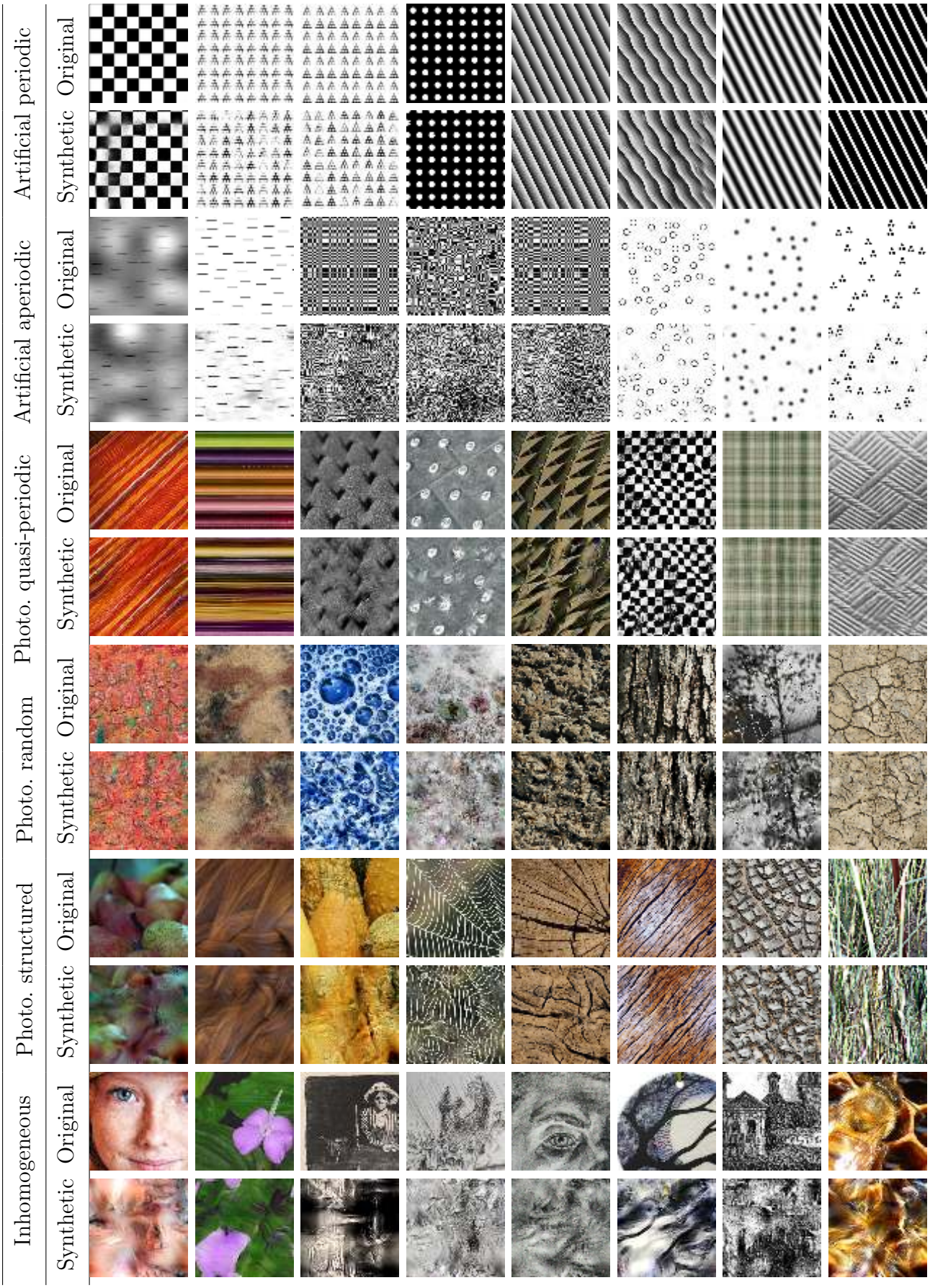

Figure 6: Examples of textures and their synthesis from each category. The figure is best seen in color.

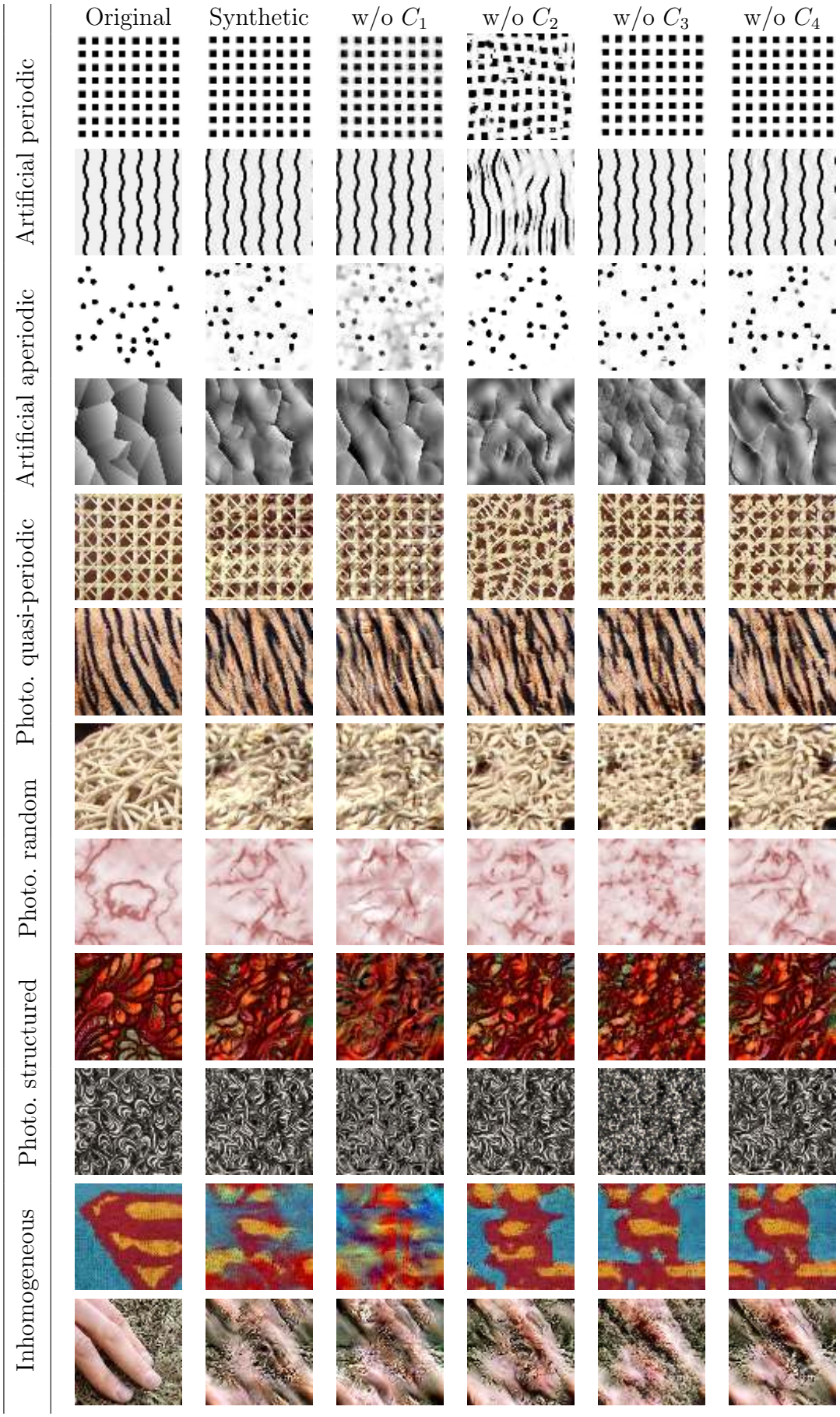

Figure 7: Textures from each category synthesized without statistical constraints defined by C_1, C_2, C_3 and C_4 (see text). The figure is best seen in color.

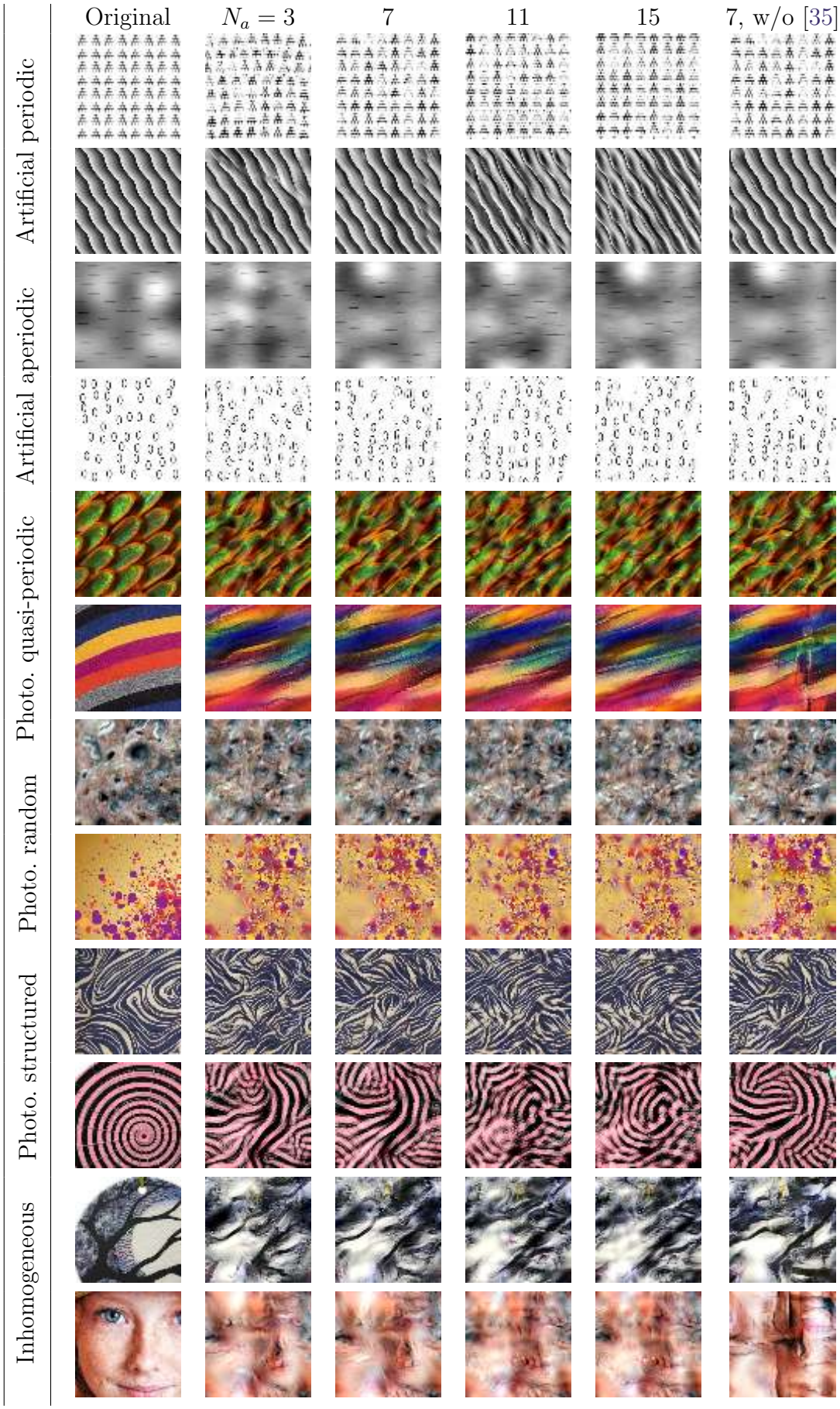

Figure 8: Textures from each category synthesized with varying auto-correlation neighborhood sizes (columns 2-5) and without using *periodic+smooth* decomposition (last column). The figure is best seen in color.

Figure 9: Examples of interpolation between natural textures. The figure is best seen in color.

A.1.3 Skewness Adjustment

The skewness of v is adjusted to $\eta(u)$ by a single optimal step in the gradient direction. In the following, we describe the method proposed in [41]. To simplify, we note $\mu_n(v) = \mu_n$ for $n \geq 2$.

Gradient descent. Assume without loss of generality that $\mu(v) = 0$ (otherwise the mean can be subtracted). Then, the gradient $\nabla\eta(v)$ at location v is proportional to

$$d = v^2 - \mu_2^{\frac{1}{2}}\eta(v)v - \mu_2\mathbb{1}. \tag{53}$$

Following (47), the adjusted image $\tilde{v} = v + \lambda d$ must satisfy the equation

$$\eta(\tilde{v}) = \frac{\mu_3(v+\lambda d)}{\mu_2(v+\lambda d)^{3/2}} = \eta(u). \tag{54}$$

Computation of λ. Noting that $\mu(v + \lambda d) = 0$ when $\mu(v) = 0$ and substituting (53) into (54), we obtain

$$\frac{\sum_{n=0}^{3} p_n\lambda^n}{\left(\sum_{n=0}^{2} q_n\lambda^n\right)^{\frac{3}{2}}} = \eta(u), \tag{55}$$

where

$$\begin{aligned}
p_0 &= \eta(v)\mu_2^{\frac{3}{2}},\\
p_1 &= 3\left(\mu_4 - \mu_2^2(1+\eta(v)^2)\right),\\
p_2 &= 3\left(\mu_5 - 2\mu^{\frac{1}{2}}\eta(v)\mu_4 + \mu_2^{\frac{5}{2}}\eta(v)^3\right),\\
p_3 &= \mu_6 - 3\mu_2^{\frac{1}{2}}\eta(v)\mu_5 + 3\mu_2(\eta(v)^2-1)\mu_4 + \mu_2^3\left(2+3\eta(v)^2-\eta(v)^4\right),\\
q_0 &= \mu_2,\\
q_1 &= 0,\\
q_2 &= \mu_4 - (1+\eta(v)^2)\mu_2^2.
\end{aligned} \tag{56}$$

Squaring both sides of (55), λ finally solves

$$\sum_{n=0}^{6} a_n\lambda^n = 0 \tag{57}$$

where

$$\begin{aligned}
a_0 &= p_0^2 - \eta(u)^2 q_0^3,\\
a_1 &= 2p_0p_1,\\
a_2 &= p_1^2 + 2p_0p_2 - 3\eta(u)^2q_0^2q_2,\\
a_3 &= 2\left(p_0p_3 + p_1p_2\right),\\
a_4 &= p_2^2 + 2p_1p_3 - 3\eta(u)^2q_0q_2^2,\\
a_5 &= 2p_2p_3,\\
a_6 &= p_3^2 - \eta(u)^2q_2^3.
\end{aligned} \tag{58}$$

The algebraic equation in (57) has six complex roots. To avoid too large modifications during the adjustment, the chosen solution λ for the gradient descent is a real solution in the interval $[\lambda_{\min}, \lambda_{\max}]$ around $\lambda = 0$ where $\lambda \mapsto \eta(v+\lambda d)$ has positive slope. Assuming $d \neq 0$, this interval exists because d is the gradient direction. If there is no such solution, one of the extreme value $\lambda_{\min}$ or $\lambda_{\max}$ is chosen instead. If there are multiple candidates, the selection is done in two steps:

1. The numerator in (54) must have the same sign as $\eta(u)$. Indeed, the squaring between (54) and (55) may introduce undesired symmetrical solutions.

2. The remaining solution with minimal magnitude is chosen.

Interval with positive slope. We find an interval where $f : \lambda \mapsto \eta(v+\lambda d)$ has a positive slope by finding the values $\lambda_{\min}$ and $\lambda_{\max}$ of the negative and positive roots of f' with the smallest magnitudes. From (54) and (55), we can write

$$f(\lambda) = \eta(v + \lambda d) = \frac{\sum_{n=0}^{3} p_n \lambda^n}{\left(\sum_{n=0}^{6} b_n \lambda^n\right)^{\frac{1}{2}}}, \tag{59}$$

where

$$\begin{aligned} b_0 &= q_0^3, \\ b_1 &= 0, \\ b_2 &= 3q_0^2 q_2, \\ b_3 &= 0, \\ b_4 &= 3q_0 q_2^2, \\ b_5 &= 0, \\ b_6 &= q_2^3. \end{aligned} \tag{60}$$

The derivative f' is then given by

$$f'(\lambda) = \frac{\sum_{n=0}^{7} d_n \lambda^n}{\left(\sum_{n=0}^{6} b_n \lambda^n\right)^{\frac{3}{2}}}, \tag{61}$$

where

$$\begin{aligned} d_0 &= p_1 b_0, \\ d_1 &= -p_0 b_2 + 2p_2 b_0, \\ d_2 &= 3p_3 b_0, \\ d_3 &= -2p_0 b_4 + p_2 b_2, \\ d_4 &= -p_1 b_4 + 2p_3 b_2, \\ d_5 &= -3p_0 b_6, \\ d_6 &= -2p_1 b_6 + p_3 b_4, \\ d_7 &= -p_2 b_6. \end{aligned} \tag{62}$$

Finally, finding the roots of f' is equivalent to solving the algebraic equation

$$\sum_{n=0}^{7} d_n \lambda^n = 0. \tag{63}$$

Algorithm. The adjustment of the skewness is detailed in Algorithm 6. Note that the minimal and maximal reachable skewness in the interval $[\lambda_{\min}, \lambda_{\max}]$ are $\eta_{\min} = \eta(v + \lambda_{\min} d)$ and $\eta_{\max} = \eta(v + \lambda_{\max} d)$. Therefore, there is no point in computing the roots of (57) if $\eta(u)$ does not belong to $]\eta_{\min}, \eta_{\max}[$.

A.1.4 Kurtosis Adjustment

Similarly to the skewness, the kurtosis of v is adjusted to $\kappa(u)$ by a single optimal step in the gradient direction. We summarize in the following the method proposed in [41]. To simplify, we note $\mu_n(v) = \mu_n$ for $n \geq 2$.

Gradient descent. Assume that $\mu(v) = 0$. Then, the gradient $\nabla\kappa(v)$ is proportional to

$$d = v^3 - \frac{\mu_4}{\mu_2} v - \mu_3 \mathbb{1}. \tag{64}$$

Algorithm 6: Skewness adjustment

Input : An image v such that $\mu(v) = 0$ and the target skewness $\eta(u)$
Output: An image $\tilde{v}$ with adjusted skewness

// Computation of the coefficients
Compute the moments $(\mu_n)_{2\leq n\leq 6}$
Compute the input skewness $\eta(v)$
Compute the coefficients $(p_n)_{0\leq n\leq 3}$, $(q_n)_{0\leq n\leq 2}$, $(a_n)_{0\leq n\leq 6}$, $(b_n)_{0\leq n\leq 6}$ and $(c_n)_{0\leq n\leq 7}$ using (56), (58), (60) and (69)
// Computation of the minimal and maximal reachable skewness
Compute the roots of (63)
Set $\lambda_{\min}$ and $\lambda_{\max}$ as the negative and positive roots with smallest magnitudes
Compute $\eta_{\min} = \eta(v + \lambda_{\min} d)$ and $\eta_{\max} = \eta(v + \lambda_{\max} d)$ using (59)
// Computation of λ
Initialize $\lambda = 0$
if $\eta(u) \leq \eta_{\min}$ **then**
    Update $\lambda \leftarrow \lambda_{\min}$
else if $\eta(u) \geq \eta_{\max}$ **then**
    Update $\lambda \leftarrow \lambda_{\max}$
else
    Compute the roots of (57)
    Keep the real roots
    if *there is only one root left* **then**
        Set λ as the remaining root
    else if *there are multiple roots left* **then**
        Keep the roots such that the numerator in (54) has the same sign as $\eta(u)$
        Set λ as the remaining root with minimal magnitude
// Gradient descent
Compute $\tilde{v} = v + \lambda d$ using (53)

Following (47), the adjusted image $\tilde{v} = v + \lambda d$ must satisfy the equation

$$\kappa(\tilde{v}) = \frac{\mu_4(v + \lambda d)}{\mu_2(v + \lambda d)^2} = \kappa(u). \tag{65}$$

Computation of λ. Noting that $\mu(v + \lambda d) = 0$ when $\mu(v) = 0$ and substituting (64) into (65), we obtain

$$\frac{\sum_{n=0}^{4} p_n \lambda^n}{(\sum_{n=0}^{2} q_n \lambda^n)^2} = \kappa(u), \tag{66}$$

where

$$\begin{aligned}
p_0 &= \mu_4,\\
p_1 &= 4(\mu_6 - \alpha^2\mu_2 - \mu_3^2),\\
p_2 &= 6(\mu_8 - 2\alpha\mu_6 - 2\mu_3\mu_5 + \alpha^2\mu_4 + (\mu_2 + 2\alpha)\mu_3^2),\\
p_3 &= 4(\mu_{10} - 3\alpha\mu_8 - 3\mu_3\mu_7 + 3\alpha^2\mu_6 + 6\alpha\mu_3\mu_5 + 3\mu_3^2\mu_4 - \alpha^3\mu_4 - 3\alpha^2\mu_3^2 - 3\mu_4\mu_3^2),\\
p_4 &= \mu_{12} - 4\alpha\mu_{10} - 4\mu_3\mu_9 + 6\alpha^2\mu_8 + 12\alpha\mu_3\mu_7 + 6\mu_3^2\mu_6 - 4\alpha^3\mu_6 - 12\alpha^2\mu_3\mu_5 + \alpha^4\mu_4\\
&\quad - 12\alpha\mu_3^2\mu_4 + 4\alpha^3\mu_3^2 + 6\alpha^2\mu_3^2 - 3\mu_3^4,\\
q_0 &= \mu_2,\\
q_1 &= 0,\\
q_2 &= \tfrac{p_1}{4}.
\end{aligned} \tag{67}$$

with $\alpha = \frac{\mu_4}{\mu_2}$. Multiplying (66) by the denominator of its left-hand side, we obtain that λ is a root of

$$\sum_{n=0}^{4} a_n \lambda^n = 0, \tag{68}$$

where

$$\begin{aligned}
a_0 &= p_0 - \kappa(u) q_0^2,\\
a_1 &= p_1,\\
a_2 &= p_2 - 2\kappa(u) q_2 q_0,\\
a_3 &= p_3,\\
a_4 &= p_4 - \kappa(u) q_2^2.
\end{aligned} \tag{69}$$

The algebraic equation in (68) has four complex roots. To avoid too large modifications during the adjustment, the chosen solution λ for the gradient descent is the real solution with minimal absolute value in the interval $[\lambda_{\min}, \lambda_{\max}]$ around $\lambda = 0$ where $\lambda \mapsto \kappa(v + \lambda d)$ has positive slope. Assuming $d \neq 0$, this interval exists because d is the gradient direction. If there is no such solution, one of the extreme values $\lambda_{\min}$ or $\lambda_{\max}$ is chosen instead.

Interval with positive slope. We find an interval where $g : \lambda \mapsto \kappa(v + \lambda d)$ has a positive slope by finding the values $\lambda_{\min}$ and $\lambda_{\max}$ of the negative and positive roots of g' with the smallest magnitudes. The derivative of g is

$$g'(\lambda) = \frac{\sum_{n=0}^{4} d_n \lambda^n}{(\sum_{n=0}^{2} q_n \lambda^n)^3}, \tag{70}$$

where

$$\begin{aligned}
d_0 &= p_1 q_0,\\
d_1 &= 2p_2 q_0 - 4p_0 q_2,\\
d_2 &= -3p_1 q_2 + 3p_3 q_0,\\
d_3 &= -2p_2 q_2 + 4p_4 q_0,\\
d_4 &= -p_3 q_2.
\end{aligned} \tag{71}$$

Finding the roots of g' comes down to solving the algebraic equation

$$\sum_{n=0}^{4} d_n \lambda^n = 0. \tag{72}$$

Algorithm. The adjustment of the kurtosis is detailed in Algorithm 7. Note that the minimal and maximal reachable kurtosis in the interval $[\lambda_{\min}, \lambda_{\max}]$ are $\kappa_{\min} = \kappa(v + \lambda_{\min} d)$ and $\kappa_{\max} = \kappa(v + \lambda_{\max} d)$. Therefore, there is no point in computing the roots of (68) if $\kappa(u)$ does not belong to $]\kappa_{\min}, \kappa_{\max}[$.

Algorithm 7: Kurtosis adjustment

Input : An image v such that $\mu(v) = 0$ and the target kurtosis $\kappa(u)$
Output: An image $\tilde{v}$ with adjusted kurtosis
// Computation of the coefficients
Compute the moments $(\mu_n)_{2\leq n\leq 12}$
Compute the input kurtosis $\kappa(v)$
Compute the coefficients $(p_n)_{0\leq n\leq 4}$, $(q_n)_{0\leq n\leq 2}$, $(a_n)_{0\leq n\leq 4}$ and $(d_n)_{0\leq n\leq 5}$ using (67), (69) and (71)
// Computation of the minimal and maximal reachable kurtosis
Compute the roots of (72)
Set $\lambda_{\min}$ and $\lambda_{\max}$ as the negative and positive roots with smallest magnitudes
Compute $\kappa_{\min} = \kappa(v + \lambda_{\min} d)$ and $\kappa_{\max} = \kappa(v + \lambda_{\max} d)$ using the left-hand side of (66)
// Computation of λ
Initialize $\lambda = 0$
if $\kappa(u) \leq \kappa_{\min}$ **then**
    Update $\lambda \leftarrow \lambda_{\min}$
else if $\kappa(u) \geq \kappa_{\max}$ **then**
    Update $\lambda \leftarrow \lambda_{\max}$
else
    Compute the roots of (68)
    Keep the real roots
    Set λ as the remaining root with minimal magnitude
// Gradient descent
Compute $\tilde{v} = v + \lambda d$ using (64)

A.2 Auto-correlation Adjustment

The auto-correlation $R(v)$ of an image v (see (42)) is approximately adjusted to the desired auto-correlation $R(u)$ in a central neighborhood $\mathcal{N}$ as follows. Assuming that $\mu(v) = 0$, the partial derivatives of the auto-correlation write

$$\forall (i,j) \in \Omega_{M,N}, \quad \forall (m,n) \in \mathcal{N}, \quad \frac{\partial R(v)_{m,n}}{\partial v_{i,j}} = \frac{1}{MN}(v_{i+m,j+n} + v_{i-m,j-n}), \tag{73}$$

with an implicit periodic boundary extension. Thus, accounting for the symmetries of the partial derivatives, the successive gradient steps write

$$\forall (i,j) \in \Omega_{M,N}, \quad \tilde{v}_{i,j} = v_{i,j} + \sum_{(m,n)\in\mathcal{N}'} \lambda_{m,n} \frac{\partial R(v)_{m,n}}{\partial v_{i,j}}, \tag{74}$$

where $\mathcal{N}'$ is the subset of $\mathcal{N}$ that accounts for the symmetries of $(\lambda_{m,n})_{(m,n)\in\mathcal{N}}$. For instance, if the central neighborhood $\mathcal{N}$ is of size $N_a \times N_a$ then $\mathcal{N}'$ contains the $\frac{N_a^2+1}{2}$ indices so that there is no redundancy. Note that (74) is the result of adjusting each auto-correlation coefficient successively by a step in the gradient direction as described in (48).

Because the partial derivatives of $R(v)$ write as in (73), (74) can be written as a circular convolution

$$\tilde{v} = v * h_\lambda, \tag{75}$$

where h_λ is a symmetric filter with support included in $\mathcal{N}$ and verifying in $\mathcal{N}$ the constraint

$$R(u) = R(v) * R(h_\lambda). \tag{76}$$

There is *a priori* no unique solution h_λ to (76). However, assuming that the coefficients of $R(h_\lambda)$ are the unknowns and using the fact that $R(h_\lambda)$ is symmetric, then (76) can be seen as a linear system of $|\mathcal{N}'|$ equations with $|\mathcal{N}'|$ unknowns. Note that this system involves coefficients of $R(v)$ that belong to the central neighborhood of size $(2N_a-1)\times(2N_a-1)$.

Assuming that the solution $R(h_\lambda)$ does have a non-negative DFT, $R(h_\lambda)$ can then be interpreted as the auto-correlation of a filter h_λ satisfying the constraint (76) and given by

$$h_\lambda = \mathcal{F}_{M,N}^{-1}\left(\sqrt{\mathcal{F}_{M,N}(R(h_\lambda))}\right). \tag{77}$$

As in general $\mathcal{F}_{M,N}(R(h_\lambda))$ is not non-negative, an absolute value is added in the square root and we use the following approximate filter

$$h_\lambda = \mathcal{F}_{M,N}^{-1}\left(\sqrt{|\mathcal{F}_{M,N}(R(h_\lambda))|}\right). \tag{78}$$

In practice, the circular convolution in (75), which gives the updated image $\tilde{v}$, is performed in the Fourier domain using

$$\mathcal{F}_{M,N}(\tilde{v}) = \mathcal{F}_{M,N}(h_\lambda)\cdot\mathcal{F}_{M,N}(v) = \sqrt{|\mathcal{F}_{M,N}(R(h_\lambda))|}\cdot\mathcal{F}_{M,N}(v). \tag{79}$$

The proposed auto-correlation adjustment is not exact but, according to [41], it becomes sufficiently accurate after a few iterations of the synthesis loop.

Algorithm. The adjustment of the auto-correlation is detailed in Algorithm 8.

Algorithm 8: Auto-correlation adjustment

Input : An image v and the target auto-correlation $R(u)$ in a central neighborhood $\mathcal{N}$
Output: An image $\tilde{v}$ with (approximately) adjusted auto-correlation

1. Compute $\mathcal{F}_{M,N}(v)$ the DFT of the input
2. Compute the input auto-correlation $R(v) = \frac{1}{MN}\mathcal{F}_{M,N}^{-1}(|\mathcal{F}_{M,N}(v)|^2)$
3. Solve for $R(h_\lambda)$ in the equation $R(u) = R(v) * R(h_\lambda)$
4. Compute $\mathcal{F}_{M,N}(R(h_\lambda))$ the DFT of $R(h_\lambda)$
5. Compute the DFT of the updated image $\mathcal{F}_{M,N}(\tilde{v}) = \sqrt{|\mathcal{F}_{M,N}(R(h_\lambda))|}\cdot\mathcal{F}_{M,N}(v)$
6. Compute $\tilde{v} = \mathcal{F}_{M,N}^{-1}(\mathcal{F}_{M,N}(\tilde{v}))$

A.3 Cross-correlation Adjustment

The cross-correlation adjustment is performed as follows. First, this is detailed in the one scale case and then for the two scales case.

One scale adjustment. The goal is to modify a list of sub-bands $(v^{(1)},\dots,v^{(L)})$ of size $M\times N$ so that its pairwise cross-correlation matrix C (see (43)) becomes the target matrix $\tilde{C}$.

We assume that for all $i\in\{1,\dots,L\}$, $\mu(v^{(i)})=0$ and we denote by V the matrix with rows equal to the pixel values of $(v^{(1)},\dots,v^{(L)})$. The partial derivatives of $C_{i,j}$ write

$$\forall(i,j,k,m,n)\in\{1,\dots,L\}^3\times\Omega_{M,N},\quad \frac{\partial C_{i,j}}{\partial v_{m,n}^{(k)}} = \frac{1}{MN}\begin{cases} v_{m,n}^{(i)} & \text{if } k=j, i\neq j,\\ v_{m,n}^{(j)} & \text{if } k=i, i\neq j,\\ 2v_{m,n}^{(i)} & \text{if } k=i=j,\\ 0 & \text{otherwise.}\end{cases} \tag{80}$$

Therefore, for all $k \in \{1, \dots, L\}$, the updated sub-bands $\tilde{v}^{(k)}$ write

$$\tilde{v}^{(k)} = v^{(k)} + \sum_{l=1}^{L} \bar{\Lambda}_{k,l} v^{(l)}, \tag{81}$$

where $\bar{\Lambda}$ is a real symmetric matrix of size $L \times L$. Similarly to (74), (81) is the result of adjusting each correlation cross-coefficient successively by a step in the gradient direction. Then, (81) can be written in a matrix form

$$\tilde{V} = \Lambda V \quad \text{where} \quad \Lambda = I + \bar{\Lambda}. \tag{82}$$

To determine the matrix Λ, we use the constraint $\tilde{C} = \frac{1}{MN}\tilde{V}\tilde{V}^{\mathrm{T}}$, which can be expanded as

$$\tilde{C} = \Lambda C \Lambda^{\mathrm{T}}. \tag{83}$$

A solution to (83) can be computed by evaluating the eigenvector decompositions of C and $\tilde{C}$, which are symmetric matrices assumed to be positive definite. This gives

$$C = BDB^{\mathrm{T}} \quad \text{and} \quad \tilde{C} = \tilde{B}\tilde{D}\tilde{B}^{\mathrm{T}}, \tag{84}$$

where D and $\tilde{D}$ are diagonal matrices with positive entries and, B and $\tilde{B}$ are orthogonal matrices. Injecting (84) into (83) leads to

$$\tilde{D}^{-1/2}\tilde{B}^{\mathrm{T}}\Lambda B D^{1/2}(D^{1/2}B^{\mathrm{T}}\Lambda^{\mathrm{T}}\tilde{B}\tilde{D}^{-1/2}) = I. \tag{85}$$

In other words, $\tilde{D}^{-1/2}\tilde{B}^{\mathrm{T}}\Lambda B D^{1/2}$ is an orthogonal matrix. Therefore, a matrix Λ solution to (83) verifies

$$\Lambda = \tilde{B}\tilde{D}^{1/2} O D^{-1/2} B^{\mathrm{T}}, \tag{86}$$

where O is an orthogonal matrix. As suggested in the original article [41], we use $O = \tilde{B}^{\mathrm{T}}B$ since this leads to an exact cross-correlation adjustment. With this choice, the matrix Λ can be written as

$$\Lambda = \tilde{C}^{\frac{1}{2}} C^{-\frac{1}{2}}. \tag{87}$$

Finally, the update is done using (82).

Two scales adjustment with fixed sub-bands. Let $(v^{(1)}, \dots, v^{(L)})$ and $(w^{(1)}, \dots, w^{(L')})$ be two lists of sub-bands of size $M \times N$. The goal is to modify $(v^{(1)}, \dots, v^{(L)})$ so that its pairwise cross-correlation matrix C (see (43)) and its cross-correlation matrix D with $(w^{(1)}, \dots, w^{(L')})$ (see (44)) become the target matrices $\tilde{C}$ and $\tilde{D}$.

We assume that for all $i \in \{1, \dots, L\}$, $\mu(v^{(i)}) = 0$ and for all $j \in \{1, \dots, L'\}$, $\mu(w^{(j)}) = 0$. We denote by V and W the matrices with rows equal to pixel values of respectively $(v^{(1)}, \dots, v^{(L)})$ and $(w^{(1)}, \dots, w^{(L)})$. The partial derivatives of $C_{i,j}$ are given in (80) and the ones of $D_{i,j}$ write

$$\forall (i,k,j,m,n) \in \{1,\dots,L\}^2 \times \{1,\dots,L'\} \times \Omega_{M,N}, \quad \frac{\partial D_{i,j}}{\partial v^{(k)}_{m,n}} = \begin{cases} w^{(j)}_{m,n} & \text{if } k = i, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, for all $k \in \{1, \dots, L\}$, the updated sub-bands $\tilde{v}^{(k)}$ write

$$\tilde{v}^{(k)} = v^{(k)} + \sum_{l=1}^{L} \bar{\Lambda}_{k,l} v^{(l)} + \sum_{n=1}^{L'} \Sigma_{k,n} w^{(n)}, \tag{88}$$

where $\bar{\Lambda}$ is a real symmetric matrix of size $L \times L$ and Σ is a real matrix of size $L \times L'$. Again, (88) follows from the successive adjustments of each correlation coefficient by a step in the gradient direction. Then, (88) can be written in matrix form

$$\tilde{V} = \Lambda V + \Sigma W \quad \text{where} \quad \Lambda = I + \bar{\Lambda}. \tag{89}$$

To determine the matrices Λ and Σ, we use the two constraints $\tilde{C} = \frac{1}{MN}\tilde{V}\tilde{V}^{\mathrm{T}}$ and $\tilde{D} = \frac{1}{MN}\tilde{V}W^{\mathrm{T}}$, which can be expanded as

$$\tilde{C} = \Lambda C \Lambda^{\mathrm{T}} + \Lambda D \Sigma^{\mathrm{T}} + \Sigma D^{\mathrm{T}} \Lambda^{\mathrm{T}} + \Sigma E \Sigma^{\mathrm{T}}, \tag{90}$$

$$\tilde{D} = \Lambda D + \Sigma E, \tag{91}$$

where $E = \frac{1}{MN} W W^{\mathrm{T}}$. Assuming that E is invertible, we have

$$\Sigma = (\tilde{D} - \Lambda D) E^{-1}. \tag{92}$$

Injecting (92) into (90) gives

$$\Lambda F \Lambda^{\mathrm{T}} = \tilde{F}, \tag{93}$$

where

$$F = C - D E^{-1} D^{\mathrm{T}} \quad \text{and} \quad \tilde{F} = \tilde{C} - \tilde{D} E^{-1} \tilde{D}^{\mathrm{T}}. \tag{94}$$

A solution Λ to (93) is obtained as previously for the similar problem in (83). Assuming that the symmetric matrices F and $\tilde{F}$ are positive definite, this is given by

$$\Lambda = \tilde{F}^{\frac{1}{2}} F^{-\frac{1}{2}}. \tag{95}$$

Finally, the update is done using (89).

Algorithm. The adjustments of the cross-correlation for the one scale and two scales cases are detailed respectively in Algorithm 9 and Algorithm 10.

Algorithm 9: Cross-correlation adjustment (one scale)

Input : A list of sub-bands $(v^{(1)}, \ldots, v^{(L)})$ and the target cross-correlation matrix $\tilde{C}$
Output: A list of sub-bands $(\tilde{v}^{(1)}, \ldots, \tilde{v}^{(L)})$ with adjusted cross-correlation matrix

1 Compute the input pairwise cross-correlation matrix $C = \frac{1}{MN} V V^{\mathrm{T}}$ where V is the matrix with rows equal to the pixel values of $(v^{(1)}, \ldots, v^{(L)})$
2 Compute $\Lambda = \tilde{C}^{\frac{1}{2}} C^{-\frac{1}{2}}$
3 Compute $\tilde{V} = \Lambda V$ where $\tilde{V}$ is the matrix with rows equal to the pixel values of $(\tilde{v}^{(1)}, \ldots, \tilde{v}^{(L)})$

A.4 Modulus Adjustment

The modulus of the complex-valued image v is adjusted without changing its phase by a simple multiplication

$$\tilde{v} = \frac{|u|}{|v|} v. \tag{96}$$

Algorithm 10: Cross-correlation adjustment (two scales)

Input : Two lists of sub-bands $(v^{(1)}, \dots, v^{(L)})$ and $(w^{(1)}, \dots, w^{(L')})$, and the target cross-correlation matrices $\tilde{C}$ and $\tilde{D}$

Output: A list of sub-bands $(\tilde{v}^{(1)}, \dots, \tilde{v}^{(L)})$ with adjusted cross-correlation matrices

1 Compute the input cross-correlation matrices $C = \frac{1}{MN} V V^{\mathrm{T}}$ and $D = \frac{1}{MN} V W^{\mathrm{T}}$ where V and W are the matrices with rows equal to the pixel values respectively of $(v^{(1)}, \dots, v^{(L)})$ and $(w^{(1)}, \dots, w^{(L')})$

2 Compute $E = \frac{1}{MN} W W^{\mathrm{T}}$

3 Compute $F = C - D E^{-1} D^{\mathrm{T}}$ and $\tilde{F} = \tilde{C} - \tilde{D} E^{-1} \tilde{D}^{\mathrm{T}}$

4 Compute $\Lambda = \tilde{F}^{\frac{1}{2}} F^{\frac{1}{2}}$

5 Compute $\Sigma = (\tilde{D} - \Lambda D) E^{-1}$

6 Compute $\tilde{V} = \Lambda V + \Sigma W$ where $\tilde{V}$ is the matrix with rows equal to the pixel values of $(\tilde{v}^{(1)}, \dots, \tilde{v}^{(L)})$

Appendix B Extension to Color Images

In 2013, Portilla and Simoncelli proposed on their web page[4] a MATLAB implementation of the color version of the algorithm. Following the idea of [24] of using a color-space transformation for decorrelating the color bands, color is mainly handled thanks to a principal component analysis (PCA). However, unlike [24], the adjustments of the summary statistics are not entirely performed independently on the decorrelated color bands, namely the PCA bands. Additional statistics are added to the list of summary statistics and some adjustments are done using jointly all color information.

Section B.1 details how to apply the forward and reverse conversions from RGB to PCA colorspace. The differences between the color and the grayscale versions are listed in Section B.2.

B.1 Principal Component Analysis (PCA)

In this section we explain how the principal component analysis (PCA) is applied to a color image. Note that similar computations are performed in [10] for the Heeger-Bergen algorithm.

Let u be a color image of size $M \times N$. First, the means of the color bands are subtracted and u is embedded in a matrix U of size $3 \times MN$ where each row corresponds to a color band. Then, the color covariance matrix C of size 3×3 is given by

$$C = \frac{1}{MN} U U^{\mathrm{T}}. \tag{97}$$

The symmetrical matrix C can be factorized as

$$C = V D V^{\mathrm{T}} \tag{98}$$

where V is an orthogonal matrix of size 3×3 containing the eigenvectors of C and D is a diagonal matrix of size 3×3 containing the eigenvalues of C.

Direct PCA. To get an image $\tilde{u}$ with decorrelated color bands, the direct PCA can be applied. The image $\tilde{u}$ is obtained by computing the matrix $\tilde{U}$ given by

$$\tilde{U} = D^{-\frac{1}{2}} V^{\mathrm{T}} U. \tag{99}$$

[4] https://www.cns.nyu.edu/~lcv/texture/

Inverse PCA. The direct PCA can be inverted and the original color-space can be recovered. Indeed, u can be computed from $\tilde{u}$ using

$$U = V D^{\frac{1}{2}} \tilde{U}. \tag{100}$$

Note that the means of the color bands must also be added to recover the original image.

B.2 Differences between the Color and the Grayscale Versions

The differences between the color and the grayscale versions are the following.

Summary statistics. The summary statistics of the grayscale version (see Section 3.1) are computed independently for each PCA band with the following two exceptions.

- Firstly, the marginal statistics in (i)c are computed from the original image and not from the PCA bands.
- Secondly, the cross-correlations in (iv), (v) and (vi) are jointly built from the three color bands. If a matrix was of size $L \times L'$ in the grayscale version, then the corresponding matrix is of size $3L \times 3L'$ in the color version.

In addition, the following statistics are added to the list of summary statistics:

(vii) Color covariance matrix (see (97)) of the input texture,
(viii) Auto-correlations of each PCA band,
(ix) Skewness and kurtosis of each PCA band,
(x) Pairwise cross-correlations of the real part of all oriented sub-bands at the same scale (size $3Q \times 3Q$),
(xi) Pairwise cross-correlation of the low-frequency residual l_{band} zoomed by a factor 2 and its shifted versions by 2 pixels along the 4 possible directions (size 15×15),
(xii) Cross-correlation of the real part of all oriented sub-bands at the coarser scale P with the 5 color images defined in (xi) (size $3Q \times 15$).

Note that the zoom in (xi) is done in the Fourier domain using zero-padding while the shifts are done in the spatial domain with a periodic boundary condition.

Texture synthesis algorithm. The texture synthesis algorithm, described in Algorithm 5 for grayscale images, is modified for color images as follows. All direct or inverse PCAs are done using the color covariance matrix of the input texture.

- In Line 1 and Line 6, the steerable pyramid decompositions are applied to the PCA bands of the corresponding images, which requires to previously apply direct PCAs.
- In Line 2, the computations of the summary statistics are made from the pyramid, the PCA bands and the input texture.
- In Line 3, the color noise is initialized with a zero mean Gaussian white noise having the same color covariance matrix (vii) as the input texture.
- In Line 7, the pyramid decompositions of the PCA bands are used as input for the constraints adjustment.
- In Line 8, the convergence accelerator is not used.

Constraints adjustment. The adjustment of the summary statistics, described in Algorithm 4 for grayscale images, is modified for color images as follows.

- As stated before, the adjustment algorithm takes the pyramid decompositions of the PCA bands as input.
- Before the filtering in Line 2, the cross-correlation of the low-frequency residual l_{band} (see (xi)) is adjusted. First, l_{band} is zoomed by a factor 2 and its 4 shifted versions are computed. The pairwise cross-correlation matrix of the 15 components is then adjusted as described in Appendix A.3. For each color band, the average of the 5 unshifted components is computed. Finally, l_{band} is updated as the down-sampled version by a factor 2 [8] of the resulting color image.
- In Line 17, the adjustment of the real part of all the oriented sub-bands is performed simultaneously for the Q orientations (instead of one at a time). The target cross-correlation matrices are the pairwise cross-correlation matrix of (x) and the full cross-correlation matrix of (vi) (instead of a single row). In addition, an adjustment is made for the coarser scale P. As target cross-correlation matrices, the pairwise cross-correlation matrix of (x) is still used along with the cross-correlation matrix of (xii).
- Between Line 28 and Line 29, color handling steps are added. It consists in the following:
  1. The auto-correlations of the PCA bands (viii) are adjusted (see Appendix A.2).
  2. The mean and the variance of each PCA band are adjusted respectively to 0 and 1 (see Appendix A.1.2).
  3. The skewness and kurtosis of each PCA band (ix) are adjusted (see Appendix A.1.3 and Appendix A.1.4).
  4. The covariance matrix of the PCA bands is adjusted to the identity (using similar computations as in Appendix A.3).
  5. The inverse PCA is applied to the PCA bands. Note that, thanks to the previous step, the color covariance matrix of the resulting image is the one of the input (vii).

Texture interpolation. The extension of texture interpolation to color images is straightforward except for the choice of the color covariance matrix. We arbitrarily choose to use the average $C_t = tC_1 + (1-t)C_2$ between the covariance matrices C_1 and C_2 of the two input textures.

Appendix C Algorithm and Implementation Details

In this section we precise details and tweaks that cannot be discussed elsewhere.

C.1 Input Handling and Options

Details concerning the input handling and options of the algorithm are presented below.

Input image crop. In order to compute the steerable pyramid with P scales, the image sizes must be multiples of 2^{P+1}. When necessary the input images are cropped and the output sizes are adjusted.

Number of scales. The sizes of the low-frequency residual l_{band} must be greater than the auto-correlation neighborhood N_a. The number of scales in the pyramid decomposition is adjusted (before the input image crop) to verify this condition.

Handling constant content. The PS algorithm suffers from instability if the input texture is close to a constant. For a grayscale image, no computations are made if the input variance is smaller than 10^{-2}. For a color image, the color input image is replaced by the grayscale average of its color bands if the color covariance matrix has an eigenvalue smaller than 10^{-2}.

Handling artificial textures. Following the original MATLAB implementation, a small noise is added to the sample texture before the pyramid decomposition in Line 1 of Algorithm 5. For artificially generated textures, this is supposed to "avoid instability created by symmetric conditions at the synthesis stage". Note that the marginal statistics of (i)c are computed from the noise-free texture. This trick is not used for color images.

Noise initialization. The synthesized texture is initialized in Line 3 of Algorithm 5. We provide to the user the possibility to specify the seed of the random number generator or to directly specify the noise image.

Boundary handling. The steerable pyramid decomposition relies on DFT computations (see Section 2.2). To avoid artifacts introduced by the implicit periodic assumption we propose to the user the possibility to use the *periodic+smooth* decomposition of [35]. As in [10], this boundary handling consists in applying the algorithm to the periodic component of the input texture. The user can also add the smooth component to the output texture. If the input and output sizes differs, the smooth component is zoomed using bilinear interpolation [22]. The steps for computing the *periodic+smooth* decomposition of an image are detailed in [7, Section 3.2].

C.2 Constraints Adjustment

Details concerning the constraints adjustment are presented below.

Adjustment of the low-frequency constraints. In Algorithm 4, the low-frequency constraints (i)a and (ii) are only adjusted if the variance of the corresponding input low-frequency band is large enough. Otherwise, only the variance of the low-frequency band is adjusted. More precisely, the adjustments are made if the ratio between the variance of the input low-frequency band with:

- the variance of the input is greater than 10^{-4} for grayscale images,
- the eigenvalue of the corresponding color band is greater than 10^{-3} for color images.

In our implementation the variance of the low-frequency band is multiplied by 16^{p-1} where p is the scale in the pyramid. This is done to replicate the normalization convention of the original MATLAB implementation.

Skewness and kurtosis. Concerning the skewness and kurtosis adjustments (see Appendix A.1.3 and Appendix A.1.4), the following is done.

- The adjustment is not performed if the initial value v_{ini} and the target value v_{tar} are too close. More precisely, the signal-to-noise ratio, noted SNR, is defined by
$$\text{SNR} = 20 \log_{10} \left(\left| \frac{v_{\text{tar}}}{v_{\text{tar}} - v_{\text{ini}}} \right| \right). \tag{101}$$
The adjustment is only performed if SNR ≤ 60.
- For stability purposes, the mean and the variance are adjusted after the adjustment of the skewness and kurtosis.
- A complex root is considered as a real root if the modulus of the ratio between its imaginary and real parts is smaller than 10^{-6}.

Cross-correlation adjustment. The cross-correlation adjustment (see Appendix A.3) relies on a positive definite assumption that does not always hold. In practice, the square root computations of matrices are done as follows.

For the one scale adjustment, the eigenvector decompositions $C = BDB^{\mathrm{T}}$ and $\tilde{C} = \tilde{B}\tilde{D}\tilde{B}^{\mathrm{T}}$ are first computed. To avoid numerical instabilities, the eigenvalues in D smaller than 10^{-12} are set to 0 and the positive part of the eigenvalues in $\tilde{D}$ is taken. Finally, the matrices $C^{-\frac{1}{2}}$ and $\tilde{C}^{\frac{1}{2}}$ are computed using
$$C^{-\frac{1}{2}} = BD^{-\frac{1}{2}}B^{\mathrm{T}} \quad \text{and} \quad \tilde{C}^{\frac{1}{2}} = \tilde{B}\tilde{D}^{\frac{1}{2}}\tilde{B}^{\mathrm{T}}, \tag{102}$$
where $D^{-\frac{1}{2}}$ denotes the Moore-Penrose inverse of $D^{\frac{1}{2}}$.

For the two scales adjustment with fixed sub-bands, the handling is similar (replacing C and $\tilde{C}$ by F and $\tilde{F}$) except that the eigenvalues are allowed to be negative. The eigenvalues in D smaller than 10^{-12} in absolute value are set to 0 while $\tilde{D}$ is not modified. The complex-valued square roots are computed using the principal square root. Finally, the update matrix $\tilde{V}$ given in (89) may be complex-valued and its real part must be taken. In practice, the cross-correlation is only adjusted if the ratio between the imaginary part variance and the real part variance is smaller than 10^{-6} for grayscale images and 10^{-3} for color images.

Variance of the high-pass residual. In Line 26 of Algorithm 4, the variance of the high-pass residual is only adjusted if the initial variance is greater than the target one.

C.3 Saving Computations

Computations are saved thanks to the following points.

- The spectral samples of the filters and the plans for computing the FFTs and iFFTs are precomputed. For color images, the eigenvectors and the eigenvalues of the color covariance matrix are also precomputed.
- During the steerable pyramid decomposition (see Algorithm 2) and the constraints adjustment (see Algorithm 4), consecutive steps may rely on DFT computations. Unnecessary FFTs or iFFTs can then be avoided.
- In Line 1 of Algorithm 4, the mean of the low-frequency residual l_{band} is set to 0. In practice, this is done efficiently in the Fourier domain during the complex steerable pyramid decomposition (see Algorithm 2).
- During the input analysis, in Line 2 of Algorithm 5, the low-frequency bands of each scale are required to compute the summary statistics (i)a and (ii). In practice, these images are added to the pyramid decomposition in Line 1 of Algorithm 5.

Acknowledgment

The authors would like to thank Prof. Jean-Michel Morel for his support, suggestions, and many fruitful discussions.

T.B. was partly supported by Office of Naval research grant N00014-17-1-2552, DGA Astrid project "filmer la Terre" n° ANR-17-ASTR-0013-01, MENRT and Fondation Mathématique Jacques Hadamard.

J.V. was partly supported by National Institutes of Health Grant EY031166.

Image Credits

Artificial textures are the ones used by Portilla and Simoncelli in the original paper [41]. They are available at their web page dedicated to the article[5].

Colored textures are taken from the dataset of [14], which is available through the web page of the dataset[6].

References

[1] B. Balas, *Texture synthesis and perception: Using computational models to study texture representations in the human visual system*, Vision research, 46 (2006), pp. 299–309. https://dx.doi.org/10.1016/j.visres.2005.04.013.

[2] ——, *Attentive texture similarity as a categorization task: comparing texture synthesis models*, Pattern recognition, 41 (2008), pp. 972–982. https://dx.doi.org/10.1016/j.patcog.2007.08.007.

[3] B. Balas, C. Conlin, and D. Shipman, *Summary statistics and material categorization in the visual periphery*, ACM Transactions on Applied Perception (TAP), 14 (2017), p. 8. https://dx.doi.org/10.1145/2967498.

[4] B. Balas, L. Nakano, and R. Rosenholtz, *A summary-statistic representation in peripheral vision explains visual crowding*, Journal of Vision, 9 (2009), pp. 13–13. https://dx.doi.org/10.1167/9.12.13.

[5] P. Bashivan, K. Kar, and J. J. DiCarlo, *Neural population control via deep image synthesis*, Science, 364 (2019). https://doi.org/10.1126/science.aav9436.

[6] A. Ben-Israel and T. Greville, *Generalized inverses: theory and applications*, vol. 15, Springer Science & Business Media, 2003. http://dx.doi.org/10.1007/b97366.

[7] T. Briand, *Reversibility Error of Image Interpolation Methods: Definition and Improvements*, Image Processing On Line, 9 (2019), pp. 360–380. https://doi.org/10.5201/ipol.2019.277.

[8] ——, *Trigonometric Polynomial Interpolation of Images*, Image Processing On Line, 9 (2019), pp. 291–316. http://doi.org/10.5201/ipol.2019.273.

[9] T. Briand and J. Vacher, *How to Apply a Filter Defined in the Frequency Domain by a Continuous Function*, Image Processing On Line, 6 (2016), pp. 183–211. http://dx.doi.org/10.5201/ipol.2016.116.

[5] https://www.cns.nyu.edu/~lcv/texture/

[6] https://www.robots.ox.ac.uk/~vgg/data/dtd/

[10] T. Briand, J. Vacher, B. Galerne, and J. Rabin, *The Heeger & Bergen pyramid based texture synthesis algorithm*, Image Processing on Line, 4 (2014), pp. 276–299. https://doi.org/10.5201/ipol.2014.79.

[11] T. Bülow and G. Sommer, *A novel approach to the 2d analytic signal*, in International Conference on Computer Analysis of Images and Patterns, Springer, 1999, pp. 25–32. https://doi.org/10.1007/3-540-48375-6_4.

[12] T. Caelli and B. Julész, *On perceptual analyzers underlying visual texture discrimination: Part i*, Biological Cybernetics, 28 (1978), pp. 167–175. https://doi.org/10.1007/BF00337138.

[13] L-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 40 (2017), pp. 834–848. https://doi.org/10.1109/TPAMI.2017.2699184.

[14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, *Describing textures in the wild*, in Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014. https://doi.org/10.1109/CVPR.2014.461.

[15] J. W. Cooley and J. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of computation, 19 (1965), pp. 297–301. http://dx.doi.org/10.2307/2003354.

[16] V. De Bortoli, A. Desolneux, A. Durmus, B. Galerne, and A. Leclaire, *Maximum entropy methods for texture synthesis: Theory and practice*, SIAM Journal on Mathematics of Data Science, 3 (2020), pp. 52–82. https://doi.org/10.1137/19M1307731.

[17] J. Freeman and E. P. Simoncelli, *Metamers of the ventral stream*, Nature Neuroscience, 14 (2011), p. 1195. https://doi.org/10.1038/nn.2889.

[18] J. Freeman, C. M. Ziemba, D. J. Heeger, E. P. Simoncelli, and A. J. Movshon, *A functional and perceptual signature of the second visual area in primates*, Nature Neuroscience, 16 (2013), p. 974. https://doi.org/10.1038/nn.3402.

[19] B. Galerne, Y. Gousseau, and J-M. Morel, *Random phase textures: Theory and synthesis*, IEEE Transactions on image processing, 20 (2011), pp. 257–267. https://doi.org/10.1109/TIP.2010.2052822.

[20] B. Galerne, A. Leclaire, and L. Moisan, *Texton noise*, in Computer Graphics Forum, vol. 36, Wiley Online Library, 2017, pp. 205–218. https://doi.org/10.1111/cgf.13073.

[21] L. Gatys, A. S. Ecker, and M. Bethge, *Texture synthesis using convolutional neural networks*, in Advances in Neural Information Processing Systems, 2015, pp. 262–270. https://doi.org/10.1109/RoboMech.2016.7813173.

[22] P. Getreuer, *Linear Methods for Image Interpolation*, Image Processing On Line, 1 (2011), pp. 238–259. https://doi.org/10.5201/ipol.2011.g_lmii.

[23] K. He, Y. Wang, and J. Hopcroft, *A powerful generative model using random weights for the deep image representation*, in Advances in Neural Information Processing Systems, 2016, pp. 631–639. https://dl.acm.org/doi/10.5555/3157096.3157167.

[24] D. J. Heeger and J. R. Bergen, *Pyramid-based texture analysis/synthesis*, in Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM, 1995, pp. 229–238. http://dx.doi.org/10.1145/218380.218446.

[25] D. H. Hubel and T. N. Wiesel, *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex*, The Journal of Physiology, 160 (1962), pp. 106–154. https://doi.org/10.1113/jphysiol.1962.sp006837.

[26] J. P. Jones and L. A. Palmer, *An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex*, Journal of Neurophysiology, 58 (1987), pp. 1233–1258. https://dx.doi.org/10.1152/jn.1987.58.6.1233.

[27] B. Julész, *Visual pattern discrimination*, IRE Transactions on Information Theory, 8 (1962), pp. 84–92. https://doi.org/10.1109/TIT.1962.1057698.

[28] ——, *Textons, the elements of texture perception, and their interactions*, Nature, 290 (1981), p. 91. https://doi.org/10.1038/290091a0.

[29] B. Julész, E. N. Gilbert, L. A. Shepp, and H. L. Frisch, *Inability of humans to discriminate between visual textures that agree in second-order statistics–revisited*, Perception, 2 (1973), pp. 391–405. https://doi.org/10.1068/p020391.

[30] B. Julész, E. N. Gilbert, and J. D. Victor, *Visual discrimination of textures with identical third-order statistics*, Biological Cybernetics, 31 (1978), pp. 137–140. https://doi.org/10.1007/BF00336998.

[31] F. W. King, *Hilbert transforms*, vol. 2, Cambridge University Press Cambridge, 2009. https://doi.org/10.1017/CBO9780511721458.

[32] K. P. Körding, C. Kayser, W. Einhäuser, and P. König, *How are complex cell properties adapted to the statistics of natural stimuli?*, Journal of Neurophysiology, (2004). https://doi.org/10.1152/jn.00149.2003.

[33] M. S. Landy and N. Graham, *Visual perception of texture*, The Visual Neurosciences, 1 (2004), p. 1106. http://cognet.mit.edu/erefs/visual-neurosciences-volumes-1-and-2.

[34] A. S. Lewis and J. Malick, *Alternating projections on manifolds*, Mathematics of Operations Research, 33 (2008), pp. 216–234. https://doi.org/10.1287/moor.1070.0291.

[35] L. Moisan, *Periodic plus smooth image decomposition*, Journal of Mathematical Imaging and Vision, 39 (2011), pp. 161–179. https://doi.org/10.1007/s10851-010-0227-1.

[36] A. J. Movshon, I. D. Thompson, and D. J. Tolhurst, *Receptive field organization of complex cells in the cat's striate cortex.*, The Journal of Physiology, 283 (1978), pp. 79–99. https://dx.doi.org/10.1113/jphysiol.1978.sp012489.

[37] Y. Nesterov, *A method of solving a convex programming problem with convergence rate* $O\left(1/k^2\right)$, in Soviet Mathematics Doklady, vol. 27, 1983, pp. 372–376. http://mi.mathnet.ru/eng/dan46009.

[38] G. Okazawa, S. Tajima, and H. Komatsu, *Image statistics underlying natural texture selectivity of neurons in macaque V4*, Proceedings of the National Academy of Sciences, 112 (2015), pp. E351–E360. https://dx.doi.org/10.1073/pnas.1415146112.

[39] B. Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics, 4 (1964), pp. 1–17. https://doi.org/10.1016/0041-5553(64)90137-5.

[40] C. R. Ponce, W. Xiao, P. F. Schade, T. S. Hartmann, G. Kreiman, and M. S. Livingstone, *Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences*, Cell, 177 (2019), pp. 999–1009. https://doi.org/10.1016/j.cell.2019.04.005.

[41] J. Portilla and E. P. Simoncelli, *A parametric texture model based on joint statistics of complex wavelet coefficients*, International Journal of Computer Vision, 40 (2000), pp. 49–70. http://dx.doi.org/10.1023/A:1026553619983.

[42] J. Rabin, G. Peyré, J. Delon, and M. Bernot, *Wasserstein barycenter and its application to texture mixing*, in International Conference on Scale Space and Variational Methods in Computer Vision, Springer, 2011, pp. 435–446. https://doi.org/10.1007/978-3-642-24785-9_37.

[43] R. Rosenholtz, *Capabilities and limitations of peripheral vision*, Annual Review of Vision Science, 2 (2016), pp. 437–457. https://doi.org/10.1146/annurev-vision-082114-035733.

[44] R. Rosenholtz, J. Huang, A. Raj, B. Balas, and L. Ilie, *A summary statistic representation in peripheral vision explains visual search*, Journal of Vision, 12 (2012), pp. 14–14. https://dx.doi.org/10.1167/12.4.14.

[45] L. Schwartz, *Théorie des distributions*, Actualités Scientifiques et Industrielles, Institut de Mathématique, Université de Strasbourg, 1 (1966), p. 2. http://dx.doi.org/10.1090/S0002-9904-1952-09555-0.

[46] E. P. Simoncelli and W. T. Freeman, *The steerable pyramid: A flexible architecture for multi-scale derivative computation*, in Proceedings., International Conference on Image Processing, vol. 3, IEEE, 1995, pp. 444–447. https://doi.org/10.1109/ICIP.1995.537667.

[47] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, *Shiftable multiscale transforms*, IEEE Transactions on Information Theory, 38 (1992), pp. 587–607. https://doi.org/10.1109/18.119725.

[48] R. S. Strichartz, *A guide to distribution theory and Fourier transforms*, World Scientific, 2003. https://doi.org/10.1142/5314.

[49] G. Tartavel, Y. Gousseau, and G. Peyré, *Variational texture synthesis with sparsity and spectrum constraints*, Journal of Mathematical Imaging and Vision, 52 (2015), pp. 124–144. https://doi.org/10.1007/s10851-014-0547-7.

[50] G. Tartavel, G. Peyré, and Y. Gousseau, *Wasserstein loss for image synthesis and restoration*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 1726–1755. https://doi.org/10.1137/16M1067494.

[51] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, *Texture networks: Feed-forward synthesis of textures and stylized images*, in Proceedings of International Conference on Machine Learning (ICML), 2016, pp. 1349–1357. http://proceedings.mlr.press/v48/ulyanov16.html.

[52] J. Vacher and R. Coen-Cagli, *Combining mixture models with linear mixing updates: multilayer image segmentation and synthesis*, arXiv preprint arXiv:1905.10629, (2019). https://arxiv.org/abs/1905.10629.

[53] J. Vacher, A. Davila, A. Kohn, and R. Coen-Cagli, *Texture interpolation for probing visual perception*, Advances in Neural Information Processing Systems, (2020). https://arxiv.org/pdf/2006.03698.

[54] J. Vacher, A. I. Meso, L. U. Perrinet, and G. Peyré, *Bayesian modeling of motion perception using dynamical stochastic textures*, Neural computation, 30 (2018), pp. 3355–3392. https://doi.org/10.1162/neco_a_01142.

[55] T. S. A. Wallis, M. Bethge, and F. A. Wichmann, *Testing models of peripheral encoding using metamerism in an oddity paradigm*, Journal of Vision, 16 (2016), pp. 4–4. https://dx.doi.org/10.1167/16.2.4.

[56] T. S. A. Wallis, C. M. Funke, A. S. Ecker, L. A. Gatys, F. A. Wichmann, and M. Bethge, *A parametric texture model based on deep convolutional features closely matches texture appearance for humans*, Journal of Vision, 17 (2017), pp. 5–5. https://dx.doi.org/10.1167/17.12.5.

[57] T. S. A. Wallis, C. M. Funke, A. S. Ecker, L. A Gatys, F. A. Wichmann, and M. Bethge, *Image content is more important than Boumas Law for scene metamers*, eLife, 8 (2019), p. e42512. https://doi.org/10.7554/eLife.42512.

[58] G. Xia, S. Ferradans, G. Peyré, and J. Aujol, *Synthesizing and mixing stationary Gaussian texture models*, SIAM Journal on Imaging Sciences, 7 (2014), pp. 476–508. https://doi.org/10.1137/130918010.

[59] Z. Xue and Z. Wang, *Texture Mixing by Interpolating Deep Statistics via Gaussian Models*, IEEE Access, 8 (2020), pp. 60747–60758. https://doi.org/10.1109/ACCESS.2020.2981410.

[60] C. M. Ziemba, J. Freeman, A. J. Movshon, and E. P. Simoncelli, *Selectivity and tolerance for visual texture in macaque V2*, Proceedings of the National Academy of Sciences, 113 (2016), pp. E3140–E3149. https://doi.org/10.1073/pnas.1510847113.

[61] C. M. Ziemba, J. Freeman, E. P. Simoncelli, and A. J. Movshon, *Contextual modulation of sensitivity to naturalistic image structure in macaque V2*, Journal of Neurophysiology, 120 (2018), pp. 409–420. https://doi.org/10.1152/jn.00900.2017.