J Math Imaging Vis
DOI 10.1007/s10851-014-0547-7

Variational Texture Synthesis with Sparsity and Spectrum Constraints

Guillaume Tartavel · Yann Gousseau · Gabriel Peyré

Received: 9 November 2013 / Accepted: 3 November 2014

Abstract This paper introduces a new approach for texture synthesis. We propose a unified framework that both imposes first order statistical constraints on the use of atoms from an adaptive dictionary, as well as second order constraints on pixel values. This is achieved thanks to a variational approach, the minimization of which yields local extrema, each one being a possible texture synthesis. On the one hand, the adaptive dictionary is created using a sparse image representation rationale, and a global constraint is imposed on the maximal number of use of each atom from this dictionary. On the other hand, a constraint on second order pixel statistics is achieved through the power spectrum of images. An advantage of the proposed method is its ability to truly synthesize textures, without verbatim copy of small pieces from the exemplar. In an extensive experimental section, we show that the resulting synthesis achieves state of the art results, both for structured and small scale textures.

Keywords Exemplar-based synthesis · Non-convex optimization · Statistical image modeling · Sparse decomposition · Dictionary learning · Random phase textures · Gaussian random fields

G. Tartavel (✉) · Y. Gousseau
Telecom ParisTech/LTCI CNRS, 46 rue Barrault,
75634 Paris Cedex 13, France
e-mail: tartavel@telecom-paristech.fr

Y. Gousseau
e-mail: gousseau@telecom-paristech.fr

G. Peyré
CNRS/Ceremade, Université Paris-Dauphine, Pl. Maréchal De Lattre De Tassigny, 75775 Paris Cedex 16, France
e-mail: peyre@ceremade.dauphine.fr

1 Introduction

1.1 Texture Synthesis

Texture synthesis is the process of generating new texture images from a given image sample. Being able to perform such a synthesis is of practical interest both for computer graphics, where it is needed to give a realistic aspect to 3D models, or for image restoration, where it may help the reconstruction of missing parts through inpainting.

One of the most difficult aspect of texture synthesis is the ability to deal with the wide variety of texture images. In particular, it is relatively difficult to synthesize textures at all observation scales using a given model. Before being more specific in the next section, we can roughly split texture synthesis methods in two groups. On the one hand, non-parametric patch-based methods, initiated by the seminal works from Efros and Leung [10] and Wei and Levoy [39], have proven their efficiency to synthesize highly structured textures, such as brick walls or pebbles seen at a close distance. On the other hand, methods relying on statistical constraints, initiated by the work of Heeger and Bergen [16], provide efficient methods for the synthesis of textures with small scales oscillations, or micro-textures, such as sand or stone.

In this work, we propose to rely on a variational approach to build a synthesis method that allows at the same time for the reconstruction of geometric textures, thanks to the use of sparsity in a patch-based dictionary, and the reconstruction of small scale oscillations, thanks to constraints on the spectrum of images. Before presenting in more details the contributions of this paper, and in order to explain both its inspiration and its originality, we start with a short presentation of previous works in this field.

1.2 Previous Works

1.2.1 Patch-Based Approaches

The first methods yielding realistic texture synthesis rely on a Markovian hypothesis: the distribution of a pixel given the rest of the texture only depends on the values in a (small) neighborhood. These methods are mostly parametric, following the work in [7]. The law of a pixel given its neighborhood is chosen *a priori* and its parameters can be learned from an exemplar using a maximum-likelihood estimation.

Non-parametric approaches [10,39] were introduced later to handle different kind of textures without designing the law of each of them by hand. For a given pixel, its value is sampled directly from an exemplar image, by seeking pixels whose square neighborhood (called a *patch*) is similar to the one of the pixel to be synthesized. This approach was a clear breakthrough in the field of texture synthesis, producing results that are visually almost perfect on difficult and structured examples. It was followed by a large body of works and many extensions were proposed, to perform real-time [23], dynamic, or volumetric texture synthesis, for instance. In particular, it is common to sample a small square area (a patch) at a time, instead of a single pixel [9]. The reader may refer to the state-of-the-art [38] for an overview of these approaches applied in different contexts.

One common drawback of these approaches, however, is that the resulting texture is often a plain juxtaposition of small pieces from the exemplar, even when sampling pixels one at a time [1]. In this sense they have a limited innovation capacity. An efficient way to avoid this drawback is to perform the synthesis from a dictionary of patches that is learned from the exemplar. Sparse decomposition methods such as [11] are classically applied to image denoising or enhancement. They rely on the assumption that each patch of an image can be decomposed as a sum of a small number of elements from a dictionary. The dictionary is usually learned from the image to be restored. These approaches were introduced in [28] as an efficient coding of natural images, inspired by the human vision system. More efficient algorithms for dictionary learning were proposed later: for instance the method of optimal direction (MOD) [12], and the K-SVD algorithm [2]. Recently, one author of the present paper has shown that this framework is well suited to texture modeling [30]. In particular, sparse dictionary learning has the remarkable ability to adaptively extract basic elements, or *textons*, from a texture. In [30], it is shown that new texture patches can be created from a learned dictionary, simply by imposing a sparsity constraint on their use of the dictionary atoms.

1.2.2 Statistical Approaches

Another set of approaches is based on statistical constraints. A set of statistics is identified and the synthesis process consists in generating a new image whose selected statistics match those of the original texture. The basic principle of the approach is in agreement with early works on texture discrimination [18], that spotted statistical constraints as a fruitful tool to investigate the human perception of textures.

In 1995, Heeger and Bergen [16] also proposed to rely on some statistical constraints to perform texture synthesis. Starting from an exemplar image, a new image is generated by imposing the marginal of wavelet coefficients, separately at each scale and orientation, as well as the image color distribution. The resulting method, although limited to relatively weakly structured textures, has the ability to produce realistic results in a computationally efficient manner. Later, it has been proposed [33] to impose some second order statistics on wavelet coefficients. More precisely, the constraints are based on the correlations between coefficients of atoms that are neighbor, either at a given scale or across scales. This allows for a better preservation of structured textures. To the best of our knowledge, this relatively old method is the one permitting to synthesize the largest class of textures, without simply copy-pasting pieces from an exemplar.

In a different direction, one may synthesize a texture by imposing constraints on its Fourier transform. In [14], it is shown that an efficient synthesis method is achieved by imposing the spectrum of synthesized images, through *random phase* textures. The resulting synthesis is fast and reliable, but limited to non-structured textures with small scales oscillations. Several works have extended this approach, either simplifying the reference from which the synthesis is performed [8,15] or developing video-texture synthesis methods [41].

In the field of computer graphics, procedural noises are widely used to yield a realistic aspect for materials. Contrary to other methods, a procedural noise is generated on-the-fly in any point of $\mathbb{R}^n$, and thus has a very low memory requirement and can be evaluated at any scales with any resolution. The Perlin noise [29] is the most classical example of such a noise. It is obtained as a linear combination of colored noises at different scales, resulting in a multi-scale noise whose spectrum is controlled by the weights of the combination. Sparse convolutive noise is another class of procedural noise, defined as the convolution of a compact kernel with a sparse convolution map. The Gabor noise is a particular case, using Gabor kernels: the distribution of scales and frequencies of the kernels can be either imposed by hand [22] or learned from a sample texture [15]. The state-of-the-art [21] provides a complete review of procedural noise functions.

1.2.3 Variational Formulation for Texture Synthesis

In this work, we advocate a generic variational approach to texture synthesis, as presented in detail in Sect. 2. This is not the first such approach in the context of texture manipulations. Some variational approaches define a function to measure the similarity between the patches of two images. This similarity measure is used in [20] to optimize an image so that all its patches look similar to a texture given as an example. Other variational approaches like [34] aim at transferring the texture of an image onto the content of another image. In this case, the functional to be minimized takes into account both the fidelity to the texture image and the fidelity to the content image.

1.3 Contributions of the Paper

In this paper, we introduce a generic texture synthesis method that is both suited to highly structured textures and textures that exhibit small scale oscillations.

First, we rely on adaptive sparse dictionary for the synthesis, following the ideas presented in [30]. As in this original approach, we impose a sparsity constraint on a decomposition of the patches into a dictionary. In addition, we also impose that atoms from the dictionary are used in the same proportions as in the original texture samples. This allows for a faithful reproduction of the structured parts (edges, corners, etc.) of the input texture. Second, we impose spectrum constraints on the synthesis, in the manner of [14]. This allows for the reproduction of high frequency components of the texture, but also, more surprisingly, for the low frequency regularity of the texture, as demonstrated in the experimental section. We also impose a constraint on the global color content of the image, that has a strong visual impact on the results. In order to combine the strength of the adaptive patch dictionary to reproduce geometry and the fidelity of frequency reproduction offered by spectrum constraints, we rely on a variational approach. Each constraint is enforced by controlling the distance of the synthesis to a set of compatible images. The minimization of the corresponding energy is performed using an averaged projections method. The approach is compared to the more classical alternating projection method proposed in [33], and the benefit of the proposed constraints is demonstrated. The resulting synthesis approach achieves, to the best of our knowledge, the best results obtained so far that truly generate a new texture without performing copy-pasting of the input. Another interesting asset of the proposed approach is that it only relies on first order statistical constraints between atoms from a dictionary. On the contrary, methods relying on a fixed dictionary necessitates second order statistics to synthesize structured textures [33]. This somehow accredits the second psychophysical theory proposed by B. Julesz [18], stating that texture discrimination is related to first order statistics of *textons* (in the presented work, textons may be identified with atoms from the dictionary). Observe nevertheless that the synthesis also relies on second order statistics between pixel values (through the power spectrum), therefore establishing a link between Julesz first [17] and second [18] theory for texture discrimination.

The plan of the paper is as follows. We define in Sect. 2 the energy function whose minimization leads our synthesis method and present in details the three constraints we impose for the synthesis: on the histogram, on the spectrum, and on the sparse decomposition of image patches. We then propose in Sect. 3 an algorithm to find a critical point of this energy function. Numerical results are shown, parameters are discussed and comparisons with the state of the art are provided in Sect. 4. Preliminary results of our work were presented in the conference paper [36]. More results can be found at:
http://perso.enst.fr/~gousseau/jmiv2014synthesis/,
and the Matlab code used for the experiments is publicly available at:
https://bitbucket.org/gtartavel/variational_synthesis/.

1.4 Notations

A matrix $A = (A^i_j)_{i,j}$ is made of columns A_j and rows A^i; its 2-norm is $\|A\|_{2,2}$ and its Frobenius norm is $\|A\|_F$. The ℓ^0 pseudo-norm of a vector b is $\|b\|_0 = \#\{i \setminus b_i \neq 0\}$ and its ℓ^p norm is $\|b\|_p$ for $p > 0$.

The indicator function $\iota_{\mathscr{C}}$ of a set $\mathscr{C}$ is equal to 0 on $\mathscr{C}$ and $+\infty$ outside. The distance of a point $x \in \mathbb{R}^m$ to a compact set $\mathscr{C}$ is $\mathscr{D}(x, \mathscr{C}) = \min_{y\in\mathscr{C}} \|x - y\|_2$ and a projection $\mathscr{P}_{\mathscr{C}}(x)$ of x on $\mathscr{C}$ is a minimizer of this distance.

The orthogonal discrete Fourier transform of an image u defined on a rectangular domain $\Omega \subset \mathbb{Z}^2$ of size $M_1 \times M_2$ is the image $\hat{u}$ made of the coefficients

$$\hat{u}(m) = \frac{1}{\sqrt{M_1 M_2}} \sum_{x\in\Omega} u(x) \ \exp\left\{-2\mathrm{i}\pi\left(\frac{x_1 m_1}{M_1} + \frac{x_2 m_2}{M_2}\right)\right\}. \tag{1}$$

The patches of size $\tau \times \tau$ in u are defined as

$$p_k = \big(u(x_k + t)\big)_t \quad \text{for} \quad t \in \{0, \dots, \tau - 1\}^2 \,. \tag{2}$$

Their locations x_k lie on a regular grid $x_k \in \Delta\mathbb{Z}^2$ of step $\Delta \in \mathbb{N}^*$. We denote by $\Pi(u) = \big(p_k\big)_k \in \mathbb{R}^{L\times K}$ the matrix, made of the K patches of u, where $L = d\tau^2$ is their dimension, with $d = 1$ for gray images and $d = 3$ for color. The adjoint operator of Π is denoted by Π^*.

2 Variational Formulation

As explained above, we define an energy functional E to assert the quality of a synthesis. The synthesis process then consists in finding local minima with low values of this functional.

The functional E is designed to account for three constraints: one on the first order statistics on the use of atoms from an adaptive, sparse dictionary, one on the spectrum of images, that is on second order pixel values statistics, and one on color histograms, that is on first order pixel values statistics. This is achieved through the distances to 3 sets: the set $\mathscr{C}_\text{p}$ of patches being sparse in a dictionary D_0 learned from u_0 and whose atoms frequency matches the ones of the decomposition of u_0 in this dictionary, the set $\mathscr{C}_\text{s}$ of images whose Fourier power spectrum is the same as in u_0, and the set $\mathscr{C}_\text{h}$ of images whose color histogram is the same as u_0.

We therefore define $E(u)$ equals to

$$\frac{\alpha_p}{2}\mathscr{D}^2(\Pi(u), \mathscr{C}_\text{p}) + \frac{\alpha_s}{2}\mathscr{D}^2(u, \mathscr{C}_\text{s}) + \frac{\alpha_h}{2}\mathscr{D}^2(u, \mathscr{C}_\text{h}), \qquad (3)$$

where $(\alpha_p, \alpha_s, \alpha_h)$ are weighting terms.

The adjoint operator Π^*, involved in the optimization process, reverberates the constraints on the patches to the image. The sets $(\mathscr{C}_\text{p}, \mathscr{C}_\text{s}, \mathscr{C}_\text{h})$ are detailed hereafter. The projection operators $\mathscr{P}_\mathscr{C}$ on these sets are also defined and presented in the following paragraphs since they are involved in the iterative optimization process detailed in Sect. 3. We start with the most straightforward histogram constraint, then proceed with the Fourier constraint, and eventually present the most involved, dictionary-based constraint.

2.1 Histogram Constraint

The constraint $\mathscr{C}_\text{h}$ imposes the histogram to match the one of the exemplar. In other words, we want to preserve the gray-level or color distribution of the texture. This requirement is common in texture synthesis: it is used in [16] or [30] to ensure that the synthesized image has a similar dynamic and color distribution as the original image.

2.1.1 Definition

We define the set of images whose color histogram is the same as the one of u_0 as:

$$\mathscr{C}_\text{h} = u_0\big(\Sigma_{|\Omega|}\big) = \left\{u_0 \circ \sigma \setminus \sigma \in \Sigma_{|\Omega|}\right\}, \qquad (4)$$

where $\Sigma_{|\Omega|}$ is the set of permutations of Ω. Two images have the same histogram if they only differ by a permutation of their pixels.

2.1.2 Projection

The projection on $\mathscr{C}_\text{h}$ is called "histogram transfer" [32]. In the case of grey-level images ($d = 1$), the projection $u_0 \circ \sigma^*$ of an image u on the set $\mathscr{C}_\text{h}$ is given by

$$\sigma^* = \sigma_{u_0} \circ \sigma_u^{-1} \qquad (5)$$

where we denote by σ_v the permutation sorting the pixel values of an image v:

$$v_{\sigma_v(0)} \leq \cdots \leq v_{\sigma_v(i)} \leq v_{\sigma_v(i+1)} \leq \ldots$$

When u and u_0 have different numbers of pixels, the sorted sequence of values $u_0 \circ \sigma_{u_0}$ is oversampled using a nearest-neighbor approximation. Note that a linear or cubic interpolation is not adapted since it creates new gray level values. This may create artifacts if the histogram to be interpolated has a gap (as for example with a mixture of Gaussians).

The color case is more involved: algorithms to compute the exact optimal assignment (such as the Hungarian method of [19]) have roughly cubic complexity, and can thus not be used for our problem. [5] proposes to replace the multi-dimensional Wasserstein distance by a summation of 1D Wasserstein distances of projections of the distributions. We follow this approach but with only 3 fixed projection directions. For $i = 1 \ldots 3$, let $c^{(i)} \in \mathbb{R}^3$ be the colors obtained by a PCA on the RGB components of u_0.

We denote respectively by $u_0^{(i)}$ and $u^{(i)}$ the components of u_0 and u in this color basis. We replace the set $\mathscr{C}_\text{h}$ by $\mathscr{C}_\text{h}{}'$ by allowing a different permutation for each component:

$$\mathscr{C}_\text{h}{}' = \left\{u \setminus \forall i, \exists \sigma_i \in \Sigma_{|\Omega|} : u^{(i)} = u_0^{(i)} \circ \sigma_i\right\}. \qquad (6)$$

The projection is thus obtained by solving 3 independent histogram transfers between each $u_0^{(i)}$ and $u^{(i)}$.

2.2 Fourier Spectrum Constraint

The Fourier transform of the empirical covariance of the image u_0 is $|\hat{u}_0|^2$, where $\hat{u}_0$ denotes the discrete Fourier transform of u_0, defined in (1). The Fourier spectrum thus encodes the correlations between the pixels in the image. As explained in the introduction, these correlations are an important characteristic of textures [14,18], which motivates their inclusion in the functional (3).

2.2.1 Definition

The set of images whose spectrum is the same as u_0 is

$$\mathscr{C}_\text{s} = \left\{u \in \mathbb{R}^{\Omega \times d} \setminus \forall m, \exists \varphi(m) : \hat{u}(m) = e^{\mathrm{i}\varphi(m)}\hat{u}_0(m)\right\} \qquad (7)$$

where the multiplication by $e^{\mathrm{i}\varphi(m)}$ preserves the amplitude but changes by $\varphi(m)$ the phase of the coefficient $\hat{u}_0(m) \in \mathbb{R}^d$. Note that φ must be antisymmetric modulo 2π since u and u_0 are real images. In the gray-level case ($d = 1$), the condition can be rewritten as $|\hat{u}(m)| = |\hat{u}_0(m)|$. In the color case ($d = 3$), it is important for color coherency that the multiplication of $\hat{u}_0(m) \in \mathbb{C}^3$ by $e^{i\varphi(m)}$ preserves the phase differences between the R, G, and B channels [14]. Therefore, texture synthesis is performed in [14] by preserving the spectrum of

u_0 while generating phases $\varphi(m)$ randomly. It is equivalent to draw at random an element from $\mathscr{C}_s$.

To reduce the boundary artifacts caused by the inherent periodicity assumption of the discrete Fourier transform, we use the periodic decomposition of [27], so that the image u_0 is decomposed as the sum of a smooth image and a (circularly) periodic image. We drop the smooth component and only keep the periodic one, which is well suited to the discrete Fourier transform.

2.2.2 Projection

The projection of an image u on the set $\mathscr{C}_s$ consists in putting together the modulus of $\hat{u}_0$ and the phases of $\hat{u}$. The solution, as detailed in Appendix 1.1, is the image u_s defined by

$$\hat{u}_s(m) = \frac{\hat{u}(m) \cdot \hat{u}_0(m)}{\left|\hat{u}(m) \cdot \hat{u}_0(m)\right|} \hat{u}_0(m) \quad \forall m \tag{8}$$

where $x \cdot y = xy^*$ denotes the hermitian product of $\mathbb{C}^d$ ($d = 1$ for gray level images and $d = 3$ for color images). The case where u and u_0 have not the same dimensions is treated by windowing and padding u_0 as presented in [14].

Remark As shown in the experimental section (Sect. 4), the spectrum constraint $\mathscr{C}_s$ handles a drawback of the patch sparsity constraint $\mathscr{C}_p$ described in the following paragraph: patches cannot handle the low frequencies of the image because they are too small to capture them. Patches also reduce high frequencies because of the sparse decomposition based on the ℓ^2 norm which promotes smooth approximations. The spectrum constraint is thus a good candidate to go along with the patch constraint.

2.3 Sparse Decomposition of the Patches

Decomposing the patches $\Pi(u)$ of an image u as a sparse linear combination $D_0 W$ of elements in an adaptive dictionary D_0 has first been proposed by [30] in the context of texture synthesis. As explained in the introduction, this framework enables us to efficiently synthesize highly structured aspects of textures, such as edges or regular patterns.

The set of patches $\Pi(u_0)$ of u_0 is first factorized into a dictionary D_0 and a sparse matrix W_0 of coefficients. In this first case, both the dictionary and the coefficients are unknown: the K-SVD algorithm [11] computes these two matrices D_0 and W_0. Then, during the synthesis, any set of patches P is decomposed into this fixed dictionary D_0 and leads to a weight matrix W. The weights of this decomposition must satisfy two constraints: each patch can use only a few number of atoms from D_0, and each atom of D_0 must be used as often in W as it is in W_0.

2.3.1 Learning Stage

The learning stage consists in building an over-complete dictionary D_0 from the exemplar u_0. This stage is performed only once before the synthesis process.

A dictionary D_0 of N atoms is obtained by minimizing

$$(D_0, W_0) = \underset{D,W}{\arg\min} \|\Pi(u_0) - DW\|_F^2 \quad \text{s.t.} \begin{cases} \|D_n\|_2 \leq 1 & \forall n = 1 \ldots N, \\ \|W_k\|_0 \leq S & \forall k. \end{cases} \tag{9}$$

This non-convex combinatorial problem is NP-hard [37]. An approximated solution can be computed using the K-SVD [2] algorithm for instance, or the MOD [12] algorithm.

The number N of elements of the dictionary must be chosen so that D_0 has some redundancy. We choose, for instance, N to be approximately equal to $2\tau^2$ to have a redundancy factor of 2.

The sparsity constraint $\|W_k\|_0 \leq S$ imposes to each patch p_k to be approximated using at most S atoms from the dictionary. This constraint enforces the dictionary to represent the patterns and features of the texture, as shown in [30].

2.3.2 Definition of the Constraint

The dictionary D_0 being learned, we control both the sparsity of the decomposition and the number of occurrences of the atoms. As explained in the introduction, this is an algorithmic interpretation of the second Julesz principle, stating that first order statistics of *textons* (in the present context, of atoms) are important for the visual discrimination of textures. Precisely, the second constraint is that each atom d_n should be used at most F^n times. The bound

$$F^n = \frac{K}{K_0} \|W_0^n\|_0 \tag{10}$$

is learned from the decomposition W_0 of the exemplar u_0 and is normalized according to the numbers K and K_0 of patches in u and u_0 respectively (the exemplar and the synthesis may have different dimensions).

The resulting set of image patches satisfying these constraints is

$$\mathscr{C}_p = \left\{ D_0 W \;\middle\backslash\; \|W_k\|_0 \leq S \text{ and } \left\|W^n\right\|_0 \leq F^n \quad \forall k, n \right\}. \tag{11}$$

Observe that this constraint results in a constraint on the number of non-zero coefficients both on the rows and the columns of the weight matrix W.

2.3.3 Projection

Computing the projection $D_0 W$ on $\mathscr{C}_p$ is a combinatorial problem quite similar to [11] which is known to be a

NP-hard problem [37]. We approximate a solution of this problem using the greedy Algorithm 1. The result is then improved using a back-projection step, as detailed hereafter. This algorithm is inspired from the Matching Pursuit (MP) algorithm [26].

Algorithm 1: approximation of the projection on $\mathscr{C}_\mathrm{p}$

```
**Data**: patches $P = \Pi(u)$, dictionary $D_0$.
**Input**: sparsity $S$, # of occurrences $F^n$, iteration factor $\lambda \approx 1.5$.
**Output**: coefficients $W$.
*Initialization:* set $W = 0$ and $R = P$; compute $\Phi = D_0^T P$.
**for** $\ell = 1$ **to** $\lambda SK$ **do**
    – find the best indices $(k^*, n^*)$ defined by (15).
    – compute the best weight $w^*$ using (16).
    – update $W$, $R$, and $\Phi$ using (14), (19), and (20).
*Back-projection:* update $W$ by solving the linear systems (22).
```

2.3.4 Algorithm Details

The algorithm is iterative. The coefficients W_k^n are updated one by one until the constraints $\mathscr{C}_\mathrm{p}$ detailed in (11) are saturated. At each step, the choice of the couple patch/atom (k, n) to be updated is optimal. The non-zero coefficients of the resulting weight matrix W are then refined (during the back-projection step).

Greedy algorithm. We denote by $E_{n,k}$ the elementary matrix whose only non-zero coefficient is 1 at position (n, k). At step ℓ of Algorithm 1, the current estimation of W is denoted by $W^{(\ell)}$. Both a patch index k^* and an atom index n^* are chosen according to

$$(k^*, n^*, w^*) = \operatorname*{arg\,min}_{k,n,w} \left\| P - D_0\big(W^{(\ell)} + wE_{n,k}\big) \right\|^2 \quad (12)$$

under the constraint

$$W^{(\ell)} + wE_{n,k} \in \mathscr{C}_\mathrm{p}. \quad (13)$$

The coefficient (k^*, n^*) of $W^{(\ell)}$ is updated while the others are left unchanged:

$$W^{(\ell+1)} = W^{(\ell)} + w^* E_{n^*,k^*}. \quad (14)$$

As shown in Appendix 1.2, the solution of (12) is

$$(k^*, n^*) = \operatorname*{arg\,max}_{(k,n)\in\mathscr{I}_{W^{(\ell)}}} \left|\langle R_k^{(\ell)}, D_n\rangle\right| \quad (15)$$

$$w^* = \langle R_k^{(\ell)}, D_n\rangle \quad (16)$$

where $R_k^{(\ell)}$ is the kth column of the residual $R^{(\ell)}$ defined at step ℓ by

$$R^{(\ell)} = P - D_0 W^{(\ell)}, \quad (17)$$

and the set $\mathscr{I}_{W^{(\ell)}}$ of available indices is

$$\mathscr{I}_W = \left\{(k, n) \setminus \left\|W_k^{\sim n}\right\|_0 < S \quad \text{and} \quad \left\|W_{\sim k}^n\right\|_0 < F^n\right\}, \quad (18)$$

where we denote $W_k^{\sim n} = \big(W_k^{n'}\big)_{n'\neq n}$ and $W_{\sim k}^n = \big(W_{k'}^n\big)_{k'\neq k}$. In the very particular case where D_0 is orthogonal, this algorithm converges in at most KS iterations because the resulting (k^*, n^*) are different at each iteration: the K constraints $\|W_k\|_0 \leq S$ are saturated after KS iterations. In general, the algorithm does not converge in a finite number of iterations. Nevertheless we decide to stop after λKS iterations with $\lambda = 1.5$. The reader can refer to [26] to derive more properties about convergence.

Efficient computation. To save computation time, residuals $R^{(\ell)}$ and inner products $\Phi^{(\ell)} = \big(\langle R_k^{(\ell)}, D_n\rangle\big)_{n,k}$ can be precomputed and updated by

$$R^{(\ell+1)} = R^{(\ell)} - w^* D_0 E_{n^*,k^*} \quad (19)$$

$$\Phi^{(\ell+1)} = \Phi^{(\ell)} - w^* D_0^T D_0 E_{n^*,k^*}. \quad (20)$$

Using a max-heap search for (15), precomputing $D_0^T D_0$, and assuming that $S \ll L \leq N \ll K$, the time complexity of this algorithm is in $\mathscr{O}\big(KN(L + \lambda S \log K)\big)$, as shown in Appendix 1.3. Note as a reference that the computation of all the inner products $\langle P_k, D_n\rangle$ already requires $\Theta(KNL)$ operations.

Back-projection. The iterative greedy algorithm described above provides weights $\tilde{W}$ to decompose P as $D_0 W \in \mathscr{C}_\mathrm{p}$. A drawback of the greedy approach is that the weights are estimated one by one: the approximation error may be quite large. The back-projection step —as introduced in [26]— improves this solution by refining all the non-zeros coefficients of $\tilde{W}$ while the constraints from $\mathscr{C}_\mathrm{p}$ are still satisfied.

The support of $\tilde{W}_k$ is denoted by

$$I_k = \left\{n \setminus \tilde{W}_k^n \neq 0\right\}. \quad (21)$$

Its cardinal satisfies $\#(I_k) \leq S$ since $\tilde{W} \in \mathscr{C}_\mathrm{p}$. The back-projection step consists in computing the projection of P_k on $\mathrm{Span}\,\big((D_n)_{n\in I_k}\big)$. This is done by solving a linear system in dimension at most S: the back-projected weights W_k are

$$W_k^{I_k} = \operatorname*{arg\,min}_{w} \left\|P_k - D_{I_k} w\right\|_F^2 \quad (22)$$

and 0 elsewhere, where $W_k^I = \big(W_k^n\big)_{n\in I}$ and $D_I = \big(D_n\big)_{n\in I}$.

Patch extraction and image reconstruction. The patch extracting operator Π involved in the function (3) extracts the patches $P = \Pi(u)$ from any image u.

Its adjoint operator Π^* is involved in the optimization process described in Sect. 3. The effect of Π^* on a set of patches P is to merge all of them into an image $\tilde{u} = \Pi^*(P)$ by summing the overlapping parts of the patches.

In our previous work [36], we replaced this operator by a non-linear operator Π_{NL}, because the linear operator Π^* introduced some blur when summing the overlapping parts of the patches. In the current paper, we introduced the spectrum constraint $\mathscr{C}_{\mathrm{s}}$ which ensures the synthesized image to be sharp enough. As a result, we do not need the non-linear operator any longer. During the optimization process in Sect. 3, we use the true adjoint operator Π^* involved in the gradient of the function (3).

3 Optimization

In the previous section, we have defined and justified the variational framework we propose for texture synthesis. The corresponding energy is expressed as a weighted sum of distances to constraint sets. For each of these sets, a projection operator has been defined. In this section, we present a general optimization procedure that is suited to such an energy, resulting in the texture synthesis Algorithm 2 and its multiscale version Algorithm 3. Before proceeding, let us observe that the presented algorithm only yields local and non unique solutions to the proposed non-convex variational approach. From an optimization point of view, this is a pain. But from a synthesis point of view, this is good news: several (local) minimizers correspond to as many possible texture syntheses.

3.1 General Formulation

The penalty function E defined by (3) can be rewritten as

$$E(u) = \frac{1}{2} \sum_i \alpha_i \mathscr{D}^2(A_i u, \mathscr{C}_i) \tag{23}$$

where the indices i of the sum are (p, s, h), the linear operators A_i are (Π, Id, Id), and $\mathscr{C}_i$ and α_i are respectively the constraints and the weighting terms defined in Sect. 2.

This function measures how close some linear measurements $A_i u$ are to the subsets $\mathscr{C}_i$. We want these distances to be as small as possible and therefore look for a local minimizer of the energy E.

3.2 Minimization

We use a gradient descent scheme to minimize the energy E. As explained in the following paragraph, there is no theoretical guaranty about the convergence of this algorithm. However, local properties of the gradient are favorable for the cluster points of the algorithm to be critical points of the functional E, as desired.

Let $\mathscr{K}_i$ be the cut locus of $\mathscr{C}_i$, that is, the points on which the distance $\mathscr{D}_{\mathscr{C}_i}$ is not differentiable. Apart from certain of these points, the projection $\mathscr{P}_{\mathscr{C}_i}$ is uniquely defined. Let $\mathscr{K} = \bigcup_i A_i^{-1}(\mathscr{K}_{\mathrm{p}})$ be the union of their reciprocal images. In any $u \notin \mathscr{K}$, the functional E is $\mathscr{C}^1$. Its gradient is

$$\nabla E(u) = \sum_i \alpha_i A_i^* \left(A_i u - \mathscr{P}_{\mathscr{C}_i}(A_i u)\right), \tag{24}$$

where the projectors $\mathscr{P}_{\mathscr{C}_i}$ are given in Sect. 2. On the contrary, for $u \in \mathscr{K}$, the projections and thus the gradient are not uniquely defined; however, any projection provides a descent direction for such u. Observe also that $\mathscr{K}$ is a measure-zero set.

In order to find a local minimum of (23), we perform a gradient descent from a random point $u^{(0)}$. The resulting sequence is defined by

$$u^{(\ell+1)} = u^{(\ell)} - \tau \nabla E(u^{(\ell)}) \tag{25}$$

where ∇E is the gradient (24) of E, resulting in Algorithm 2. This gradient is not globally nor locally Lipschitz, mainly because of the spectrum constraint $\mathscr{C}_{\mathrm{s}}$. The same problem is encountered in phase retrieval problems [4]. In order for the gradient descent to converge, the step size should decrease to 0 or be chosen adaptively by line-search. In practice, we choose a constant step size $\tau = 1/c$ with $c = \sum_i \alpha_i \left\| A_i^* A_i \right\|_{2,2}$ and motivate this choice in the next paragraph. Even if there is no theoretical convergence guaranty, numerical experiments show that the gradient descent converges in practice to a stationary point of E (see Sect. 4).

Algorithm 2: synthesis algorithm, monoscale

```
Input: exemplar image $u_0$.
Data: synthesis initialization $u$, e.g. white noise.
Output: synthesized image $u$.
Learning Step:
– Estimate $(D_0, W_0)$ defined in (9), e.g. using [11].
– Compute $F^n$ defined in (10)
Synthesis Step:
for $\ell = 1$ to $\ell_{max}$ do
    – histogram transfer: set $u_h = \mathscr{P}_{\mathscr{C}_{\mathrm{h}}}(u)$ defined after (6).
    – spectrum transfer: set $u_s = \mathscr{P}_{\mathscr{C}_{\mathrm{s}}}(u)$ defined in (8).
    – sparse coding: compute $P = \mathscr{P}_{\mathscr{C}_{\mathrm{p}}}(\Pi(u))$ using Algo. 1
    and reconstruct $u_p = \Pi^* P$, i.e. merge the patches into an
    image by summing overlapping values of the patches
    – averaging: $u = (\alpha_h u_h + \alpha_s u_s + \alpha_p u_p)/(\alpha_h + \alpha_s + \alpha_p Z)$
```

3.3 Averaged Projections

This paragraph points out that the averaged projection algorithm is a particular case of the proposed gradient descent scheme.

We consider the case of a perfect tiling, i.e. when all the pixels belong to Z patches exactly where $Z \triangleq \lceil \tau/\Delta \rceil^2 = \|\Pi^* \Pi\|_{2,2}$, which is the case for integer ratio $\tau/\Delta \in \mathbb{N}^*$ of patch size over spacing. In this case, $\Pi^* \Pi = Z \cdot \mathrm{Id}$. Gradient descent step (25) becomes in this case

$$u^{(\ell+1)} = c^{-1} \sum_i \alpha_i A_i^* \mathscr{P}_{\mathscr{C}_i}(A_i u^{(\ell)}), \tag{26}$$

where the constant c previously defined is simply $c = \tilde{\alpha_p} + \alpha_s + \alpha_h$ with $\tilde{\alpha_p} = \alpha_p Z$. Relation (26) is a step of averaged projections: the transforms $A_i u^{(\ell)}$ are projected on the respective sets $\mathscr{C}_i$ and are then averaged together.

A cluster point $\tilde{u}$ of the iterates (26) is a stationary point of the energy (23) and satisfies

$$\tilde{u} = c^{-1} \sum_i \alpha_i A_i^* \mathscr{P}_{\mathscr{C}_i}(A_i \tilde{u}). \tag{27}$$

This provides a geometrical interpretation of the solutions: each solution is the barycenter of the projections of its transforms on the sets $\mathscr{C}_i$ weighted by α_i.

Alternated projections. Instead of using averaged projections (26), it is possible to use alternated projections, which gives the following iterations:

$$\begin{aligned} u^{(3\ell+1)} &= Z^{-1} \Pi^* \mathscr{P}_{\mathscr{C}_\mathrm{p}}(\Pi u^{(3\ell)}), \\ u^{(3\ell+2)} &= \mathscr{P}_{\mathscr{C}_\mathrm{s}}(u^{(3\ell+1)}), \\ u^{(3\ell+3)} &= \mathscr{P}_{\mathscr{C}_\mathrm{h}}(u^{(3\ell+2)}), \end{aligned}$$

in the case of $\tilde{\alpha_p} = \alpha_s = \alpha_h = 1$. The convergence properties of both averaged and alternated non-convex projections on smooth manifolds are analyzed in [24].

Observe however that this scheme has no chance to converge since the sets $\mathscr{C}_i$ are distinct in general. It leads to 3 cluster points, namely 1 point in each set $\mathscr{C}_i$. Recall that averaged projections lead to a cluster point $\tilde{u}$ (27), which is a compromise between the 3 constraints $\mathscr{C}_i$.

The experimental results in the next section (Fig. 11 in particular) show that the alternated projections algorithm is more likely to produce artifacts than the averaged projections algorithm. Note that several texture synthesis methods, like [16] and [33], are implicitly based on an alternated projections scheme. As explained in [16], it is better not to iterate alternated projections too many times because the results may suffer from artifacts.

3.4 Multi-scale Procedure

Since the energy (3) is non-convex, the choice of the initialization $u^{(0)}$ has a strong influence on the result. In order to avoid visually unsatisfying local minima, we perform the synthesis through J scales using a classical multi-scale scheme summarized in Algorithm 3.

At scale $j \in \{J-1, \dots, 0\}$, the original image u_0 is decimated by a factor 2^j to give an image u_j. The synthesis is performed using u_j as an exemplar and a dictionary D_j learned on it. The result $u_j^{(\infty)}$ is oversampled by a factor 2 and used as initialization $u_{j-1}^{(0)}$ for the next scale. The roughest scale is initialized with a uniform white noise[1] $u_{J-1}^{(0)}$. The resulting image is the synthesis $u_0^{(\infty)}$ obtained at the finest scale.

Decimation and oversampling operations are performed as follows. The image to be decimated is first filtered by a 2×2 box kernel and then decimated by a factor 2. The oversampling step is performed with a bi-cubic interpolation. The shift of half a pixel is obtained using the cubic convolution kernel $h = (-1, 9, 9, -1)/16$.

Algorithm 3: final synthesis algorithm, multi-scale

```
Input: exemplar image $u_0$.
Output: synthesized image $u$.
Initialization: generate a white noise $\tilde{u}_{J-1}^{(0)}$.
Multi-scale Synthesis:
for $j = J-1$ to $1$ do
    – compute decimation $u_j$ of $u_0$ by a factor $2^j$
    – synthesize $\tilde{u}_j$ from exemplar $u_j$ with initialization $\tilde{u}_j^{(0)}$
      using Algo. 2
    – compute upsampling $\tilde{u}_{j-1}^{(0)}$ of $\tilde{u}_j$ by a factor 2
Final Scale:
– synthesize $\tilde{u}_0$ from exemplar $u_0$ with initialization $\tilde{u}_0^{(0)}$ using
  Algo. 2
Post Processing:
– spectrum transfer: set $u_s = \mathscr{P}_{\mathscr{C}_s}(\tilde{u}_0)$ defined in (8).
– histogram transfer: set $u = \mathscr{P}_{\mathscr{C}_\mathrm{h}}(u_s)$ defined after (6).
```

3.5 Implementation Details

Periodicity of the synthesis. During the synthesis process, because of the spectrum constraint, the image is assumed to be periodic. The histogram constraint and the white noise initialization are consistent with this assumption since all pixels are considered independent. The oversampling step is made periodic by using a circular convolution. The patch processing is made consistent by defining the patch extracting operator $\Pi : u \mapsto \Pi(u)$ on the periodic image. Some patches thus contain pixels both on the very left side and on the very right side of the image. As a result, the synthesized image is also periodic. It can be tiled or shifted circularly.

Spacing between patches. As defined in the notation section (Sect. 1.4), the patches are located on a grid of step Δ. To avoid mosaic artifacts, the value of Δ must be as small as possible, 1 or 2 pixels for instance. But the lower the slower. A higher value, for instance 4 pixels, provides a trade-off between synthesis quality and computation time.

Note that the offset of the grid of patches can be freely chosen (recall that the image is assumed periodic). Choosing

[1] Whose range is the range of the pixels' values.

a random offset for each iteration reduces the mosaic artifact caused by a large Δ. In practice, a step size of $\Delta = 4$ pixels with patches of size $\tau = 12$ pixels does not produce mosaic artifacts with this technique and increases the speed of the algorithm (a few minutes for a 256×256 px image).

Post processing. To increase the grain of the resulting texture, we add an extra projection step at the very end of the multi-scale synthesis process, resulting in the final Algorithm 3. The image $u_0^{(\infty)}$ synthesized at the finest step is projected on the spectrum constraint and then on the histogram constraint. The final image is $u^{(\infty)} = \mathscr{P}_{\mathscr{C}_\mathrm{h}} \circ \mathscr{P}_{\mathscr{C}_\mathrm{s}}\big(u_0^{(\infty)}\big)$.

4 Numerical Results

This section provides numerical results of our synthesis algorithm. We illustrate the influence of each term of the energy function (3) and of the parameters. We compare our results to the most classical synthesis methods relying on statistical constraints. We also compare the innovation capacity of the approach to that of exemplar-based methods.

When not explicitly specified, the extracted patches are of size $\tau = 12$ and are centered on a grid with a step size of $\Delta = 4$ pixels. The synthesis is performed over $J = 3$ scales and the terms of the functional are given the same weight[2] $\alpha_h = \alpha_s = \tilde{\alpha_p} = 1$. The dictionary is made of $N = 384$ atoms and the sparsity is set to $S = 4$ coefficients per patch.

Under these settings with images of size 256×256 using a quad-core CPU at 2.3 GHz, the preliminary learning stage performed by the SPAMS toolbox [25] takes roughly a minute, and the synthesis using our unoptimized Matlab code also takes about a minute. As a comparison, exhaustive patch-searching algorithms such as the one from [10] take several minutes, while fast improvements like those from [23] or [3] run in a couple of seconds.

Figure 1 shows several results of our algorithm for textures from the VisTex [31] database. The sand and water textures are well reproduced thanks to the spectrum constraints. The regularity of quasi-periodic textures is ensured by both the spectrum constraint and the multi-scale scheme. The sharpness of the edges and the geometrical patterns are handled by the sparse decomposition of the patches into the adaptive dictionary.

The last row of Fig. 1 shows difficult examples to illustrate the limitations and the failure cases of our algorithm. Repetition of small sharp objects —like the pills texture— cannot be represented by our approach: the sparsity of the patches cannot handle several objects in a patch, and the spectrum constraint is not able to create sharp objects. A mixture of large scale patterns and small details —like the tiles or the pumpkins— are also difficult to generate because of the patch constraint: small patches cannot deal with large structures, whereas large patches cannot handle small details because of the sparsity constraint.

[2] We recall the normalization $\tilde{\alpha_p} = \alpha_p Z$ with $Z = \lceil \tau/\Delta \rceil^2$.

4.1 Comparison

4.1.1 Statistical Methods

We first show a comparison with other statistical methods that we now recall.

The approach of Heeger and Bergen [16] imposes the histogram of the image and also the distributions of the coefficient of a multi-scale decomposition [35], scale by scale. An alternated projection method is used: the constraints are imposed turn by turn. As explained in [16], this should not be iterated more than 4 or 5 times because of convergence issues. The decomposition being based on gabor-like filters, this method is not adapted to synthesize edges or structures. This method may be tested online using the *IPOL* demo from [6].

The method from Portilla and Simoncelli [33] is the state of the art among statistical methods, to the best of our knowledge. Several constraints on the coefficients of the multi-scale decomposition [35] are imposed: mean, variance, and other moments. The main difference with [16] is that correlations are imposed between neighbor coefficients, whereas [16] only considers marginals at each scale and orientation. Some edges and sharp structures can be synthesized thanks to these dependencies. Let us emphasize that, in contrast, the method presented in this paper does not impose dependency between coefficients corresponding to different atoms in the dictionary. However, we used an *adaptive* dictionary learned on the exemplar, whereas [16,33] use a non-adaptive pyramid transform.

In [14], synthesis is performed by preserving the amplitude of the Fourier spectrum and by randomly shuffling the phases of the Fourier Transform. This is strictly equivalent to generating a white noise and projecting it on our spectrum constraint $\mathscr{C}_\mathrm{s}$. This approach is fast and is well adapted to smooth textures such as sand or a cloudy sky, but cannot produce edges. This method may be tested online using the *IPOL* demo from [13].

The algorithm of [30] is the first applying sparse decomposition of patches to texture synthesis. A redundant dictionary is learned beforehand on an exemplar of the texture. The synthesis is then performed from a white noise image by alternating histogram transfer and sparse coding into the dictionary. This is equivalent to alternated projection on our sets $\mathscr{C}_\mathrm{h}$ and $\mathscr{C}_\mathrm{p}{}'$ where $\mathscr{C}_\mathrm{p}{}'$ has no constraint on the number of usage F^n of the elements in the dictionary.

In Fig. 2, we show several synthesis examples using our approach and the statistical methods recalled in this section.

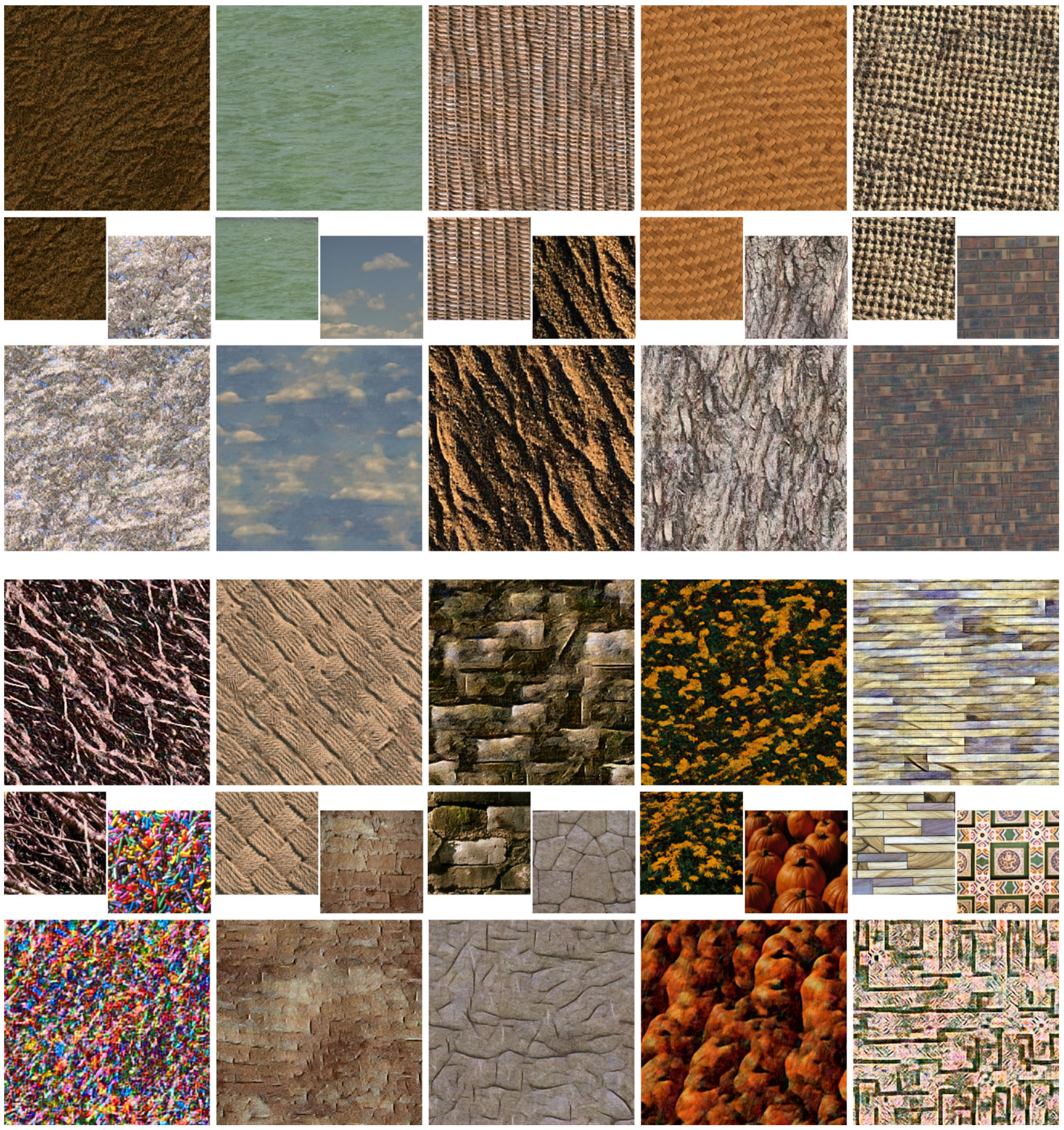


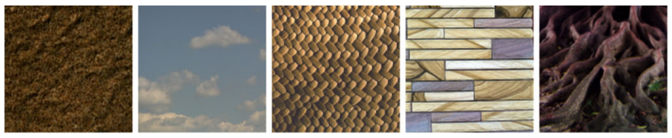
Fig. 1 Result of our algorithm (*big images*) for several exemplar (*small images*). The last examples are the most difficult because they simultaneously exhibit many small sharp details and large geometrical structures

As we can see, the proposed approach has the best ability to reproduce both structures and fine details of textures.

4.1.2 Copy-Paste Methods

Figure 3 shows two synthesis results obtained with the Efros and Leung algorithm [10], using the IPOL accelerated implementation [1]. The synthesis is performed by choosing and copying pixels one by one from the exemplar. The key idea of this algorithm may be encountered in a large number of more recent algorithms detailed in the start-of-the-art [38], including [20] illustrated on Fig. 4.

Figure 3 also displays on the bottom the map of coordinates provided by the IPOL implementation [1]. Figure 4



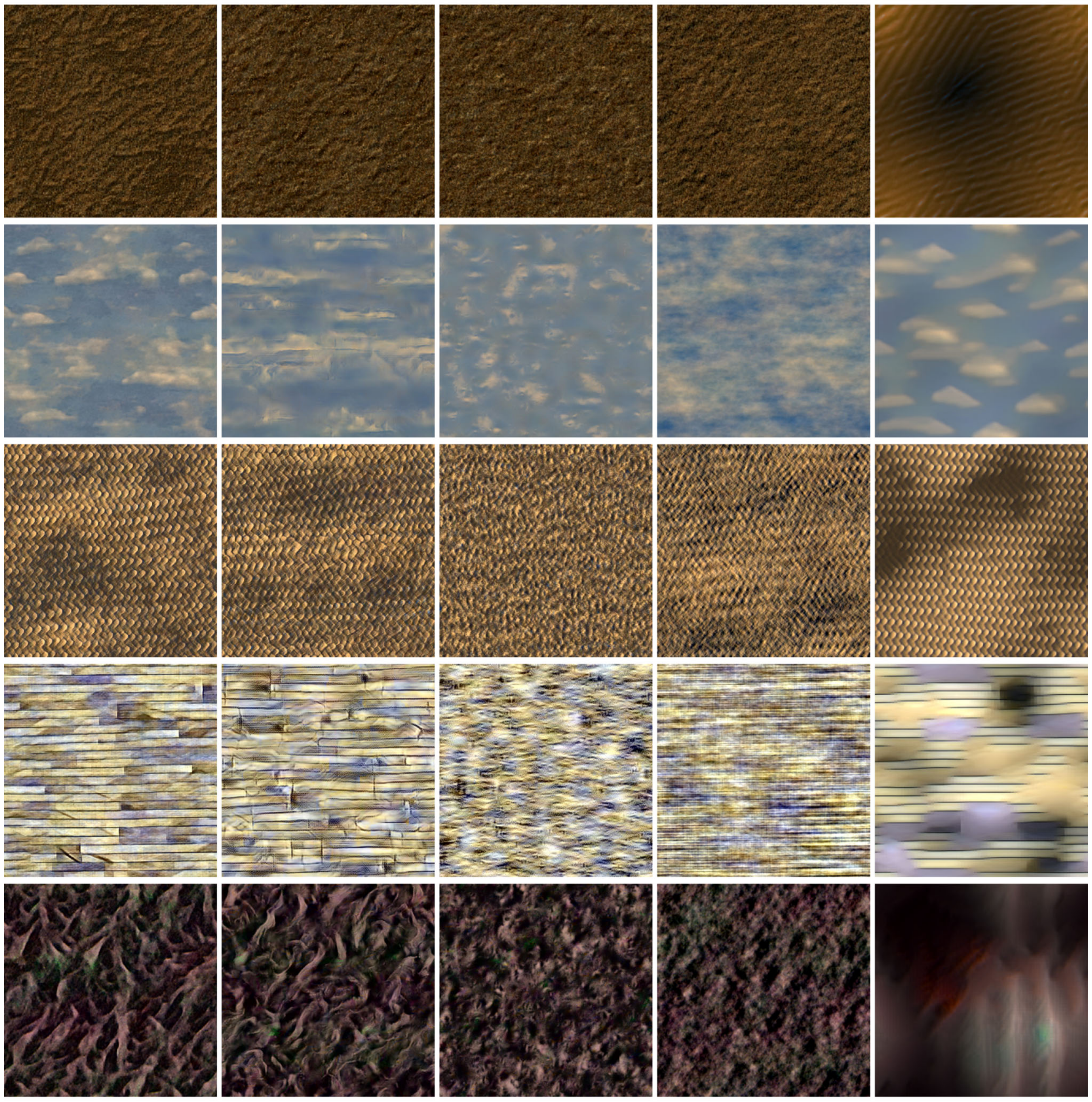


Fig. 2 We present synthesis results using our approach and other statistical methods. From left to right: our approach penalizes a deviation of histogram, spectrum, and sparse approximation of patches; [33] imposes statistics including correlations in a wavelet frame; [16] imposes the image histogram and wavelet coefficient distributions (*scale by scale*); [14] preserves only the spectrum modulus; [30] imposes the image histogram and patch sparsity

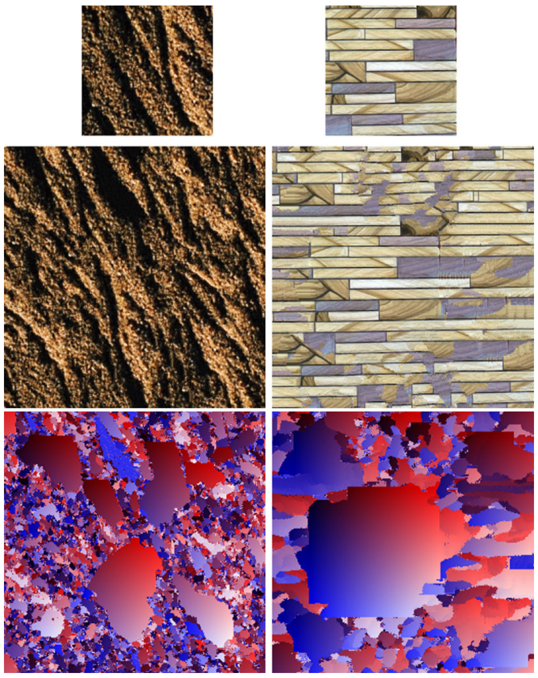


Fig. 3 Results from the pixel-by-pixel copy-paste algorithm [10] (*middle*); the coordinate maps (*bottom*) show that the results are tilings from the exemplars (*top*), even if the synthesis is performed pixel after pixel

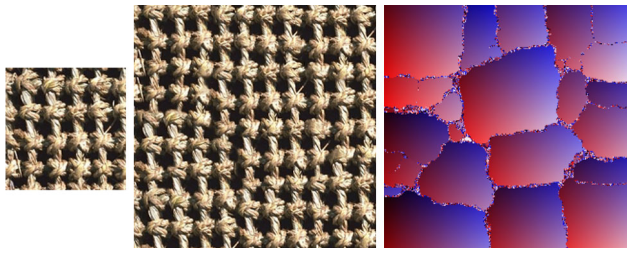


Fig. 4 Results from the more recent copy-paste algorithm [20] (*center*); like in Fig. 3, the coordinate map (*right*) shows that the result is tilings from the exemplars (*left*)

shows a map of coordinates computed *a posteriori* from a result of [20]. This map represents the location in the exemplar of the synthesized pixels: pure black (resp. red, blue, white) means that the synthesized pixel comes from the top-left (resp. top-right, bottom-left, bottom-right) corner of the exemplar image. The structure of the synthesized image is clearly visible on these maps: the synthesis is made of a tiling from the exemplar even if pixels are copied one by one. The results of such copy-paste methods are often visually better than those obtained using statistical methods, but their innovation capacity is much lower. In contrast, results from our approach (see Fig. 1) include no repetition from the input image.

The next subsection provides experiments about the innovation properties of synthesis algorithms.

4.1.3 Innovation Capacity

We carried out the following experiment to demonstrate the innovation capacity of several texture synthesis algorithms including the proposed one.

Given an input image u_0 and a corresponding synthesis u using any algorithm, we find the closest neighbor in u_0 of each patch of u. Figure 5 shows, from left to right: the synthesized images using different algorithms, the locations of the nearest-neighbor of each 3×3 patch (using the color map described above), a zoom on the central part of this map, and the histogram of the distances from each patch to its nearest-neighbor. The synthesis images are obtained using our algorithm, the random phase algorithm from [14], the statistical pyramid-based approach from [33], and the pixel-based approach from [10]. The random phase algorithm from [14] is a reference in term of innovation capacity since it generates a new image at random from a gaussian model.

The location maps (middle columns in Fig. 5) show that our algorithm and the one from [33] produce an image which is locally different from the input image, in the same way as the random phase texture obtained using [14]. On the contrary, the location map corresponding to [10] has many homogeneous regions (see the map and its zoom in Fig. 5): these regions show that the synthesized image is actually a tiling made from the input image. The distance histograms (right column in Fig. 5) show that the images synthesized using our approach has no 3×3 regions which are exactly the same as in the input image, and neither have the results using the algorithm of [33] or the random phase from [14]. On the opposite, one third of the pixels synthesized by the method from [10] has exactly the same 3×3 neighborhood as in the input image, as shown by the first bin of the histogram: this bin has been made wider and smaller to fit on the page, its area represents the proportion of distance that are almost zero.

The location maps and the distance histograms show that our algorithm and the one from [33] perform a genuine synthesis, while the pixel-based approach of [10] generate a texture which is a piecewise copy of the exemplar image.

4.2 Detailed Analysis of the Synthesis Functional

The influence of each of the three terms of the function (3) is illustrated in Fig. 6. The effect of each constraint is demonstrated by suppressing one of them while preserving the two others. This experiment yields the following observations.

The patch decomposition term relies on the adaptive dictionary, which is good at representing geometrical features. The syntheses produced without this constraint have little geometrical content: edges and sharp details are completely lost.

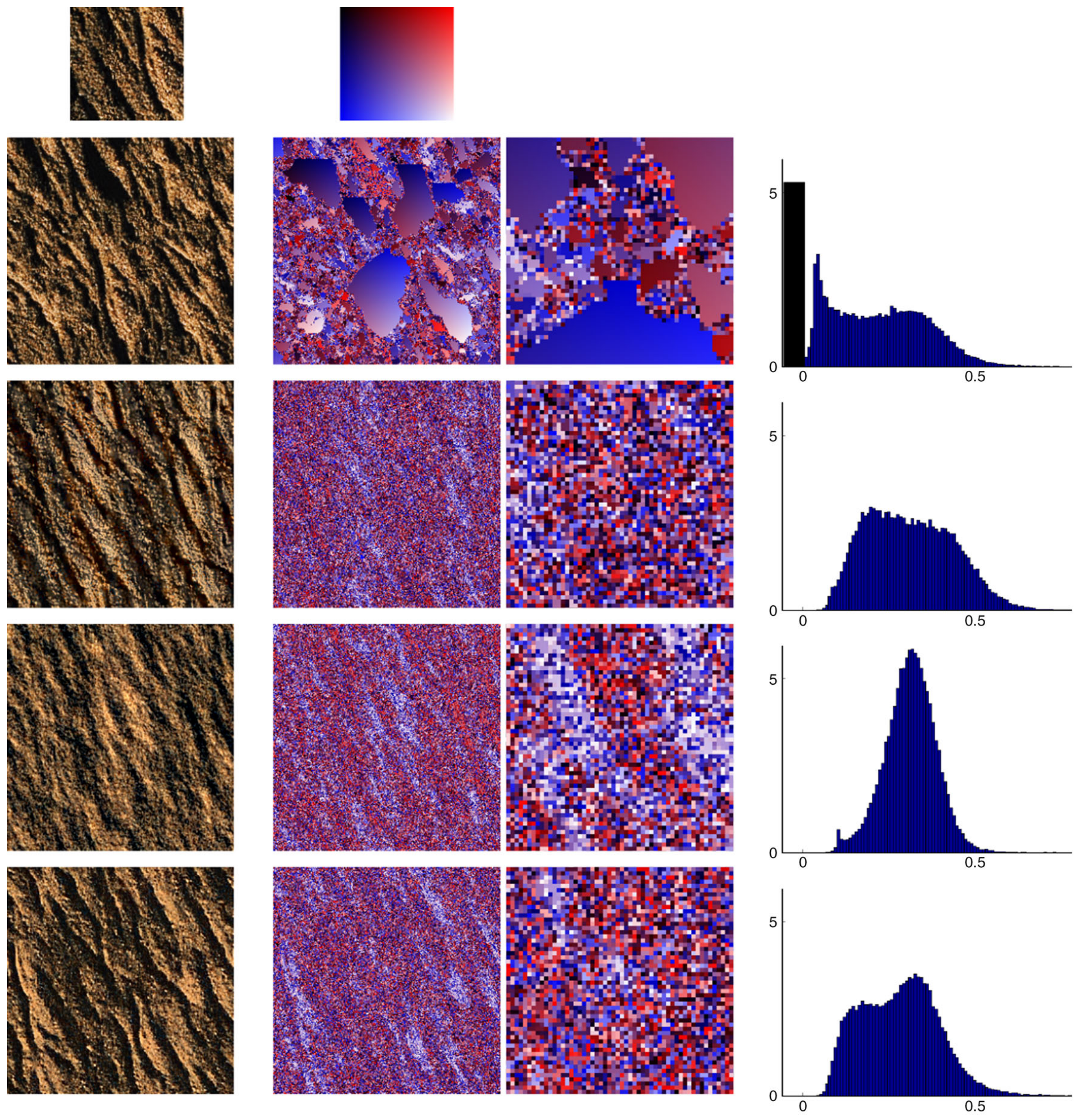


Fig. 5 Analysis of the innovation capacity of several algorithms. For each patch of the synthesized image, we find its nearest patch in the input image. From left to right: synthesis image obtained from different algorithms, map of location of the nearest patch, 4× zoom on its central part, and histogram of the distances to the nearest patch. From top to bottom: input image, synthesized images using [10], using [33], using [14], and using our algorithm. Both location map and distances histogram show that the pixel-based synthesis algorithm from [10] mostly create a tiling, whereas our algorithm and the one from [33] are able to generate novel images. Note that the wide black bar on the top-right histogram correspond to the zero bin and has been made wider and smaller to fit on the page

On the other hand, a sparse decomposition cannot represent granularities of textures like sand or rocks. This is due to the noisy aspect of such textures which cannot be sparsely represented. It is approximated by an almost-constant value and the texture generated is too smooth when using only histogram + patch sparsity terms.

The spectrum term acts as a complement of the patch sparsity term. It is powerful to represent granularities, as may be seen on the sand example. However, the spectrum cannot produce any edge, as illustrated by the synthesis with only the spectrum and histogram terms. Preserving only the spectrum is proposed and illustrated in [14].

We also remark that the low frequencies imposed by the spectrum enhance the regularity of the texture. Without this constraint, the results have some brighter or darker regions, whereas the results using the spectrum are more stationary.

The histogram term deals with the contrast and the color faithfulness of the result. Note that the patch decomposition alone has no control on the patch dynamic.

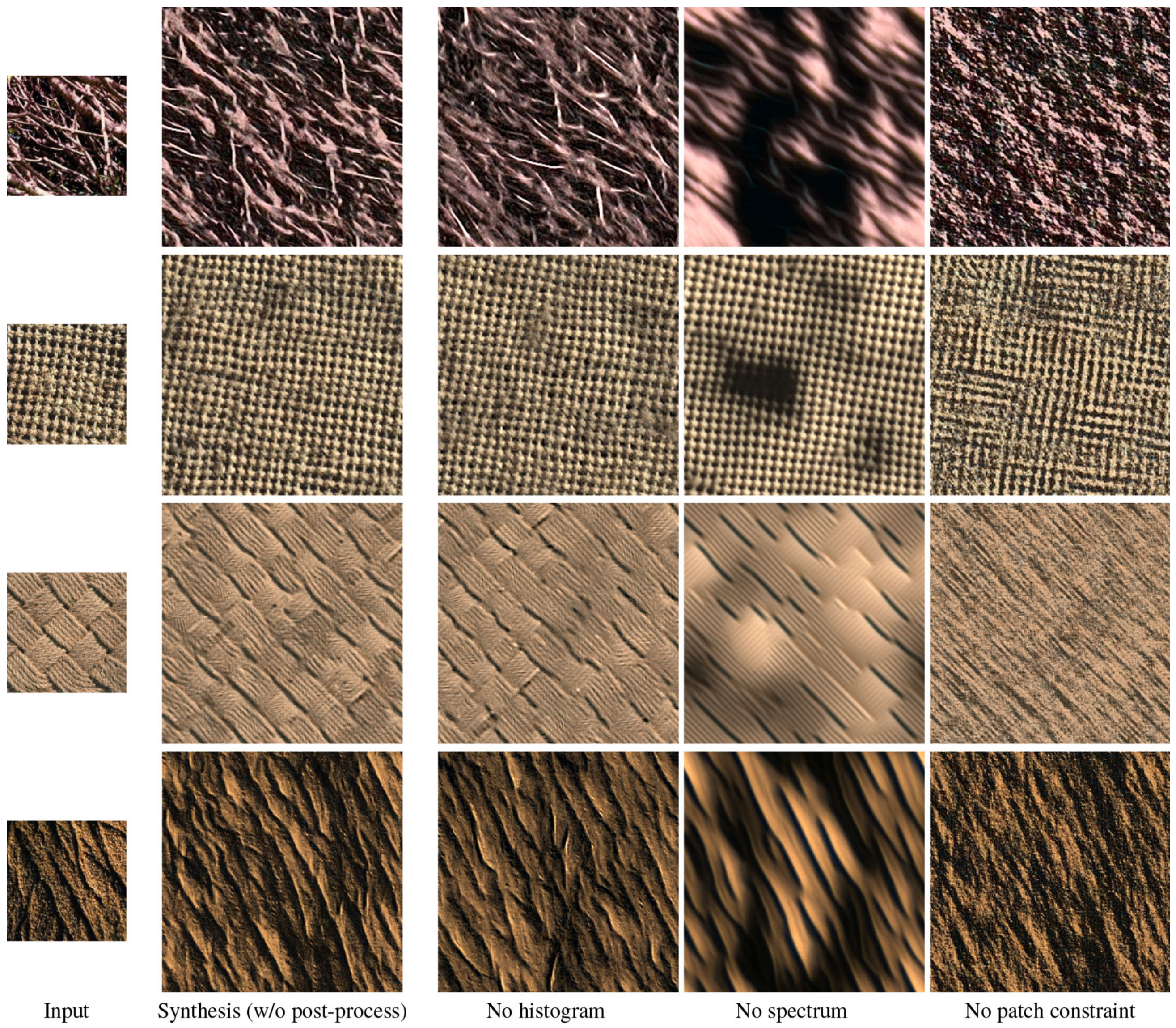


Fig. 6 Synthesis results when dropping one of the three terms. The histogram term prevents from a loss of contrast. The spectrum term spatially regularizes the synthesis and generates granularity. The patch decomposition term handles sharp edges. Note that the post-process is not performed to allow a fair comparison of each term

4.3 Influence of the Parameters

This section illustrates the influence of the parameters of our synthesis algorithm: the size $\tau \times \tau$ of the patches, the number J of scales, the weights α of each term of the function, and the sparsity factor S.

4.3.1 Patch Size and Multi-scale

Figure 7 illustrates the influence of the multi-scale process and of the size of the patches.

The effect of the multi-scale is clearly visible on all the results: it increases the spatial coherence of the texture. In the case of quasi-periodic textures with small patterns, it increases the regularity of the texture: the patterns are more regularly distributed. In the case of textures with large patterns, it produces large structures such as long and continuous edges. This is not possible with the single scale synthesis.

The size of the patches must be roughly the size of the elements in the texture. A patch size of 12×12 pixels is a good compromise on images of size 128×128 pixels from the VisTex database.

If the patches are too small compared to the patterns of the texture, the pattern cannot be well synthesized and suffer from geometrical distortions. On the contrary, using too big patches makes the sparse approximation into D_0 rougher and

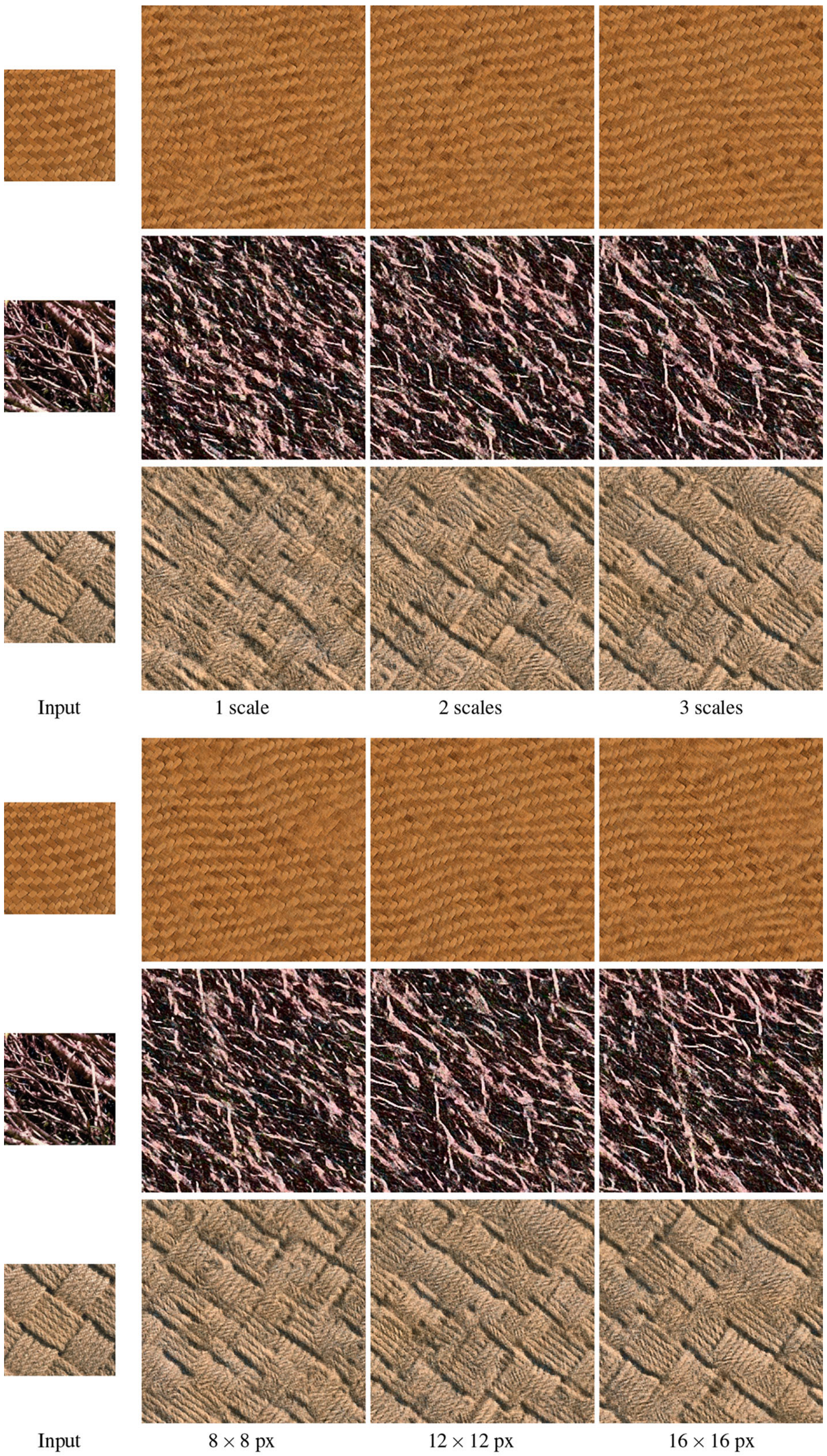


Fig. 7 Synthesis results with different numbers of scales (*top*) and different sizes of patches (*bottom*). Multi-scale ensures spatial coherency of the texture. The patches must roughly have the size of the patterns of the texture

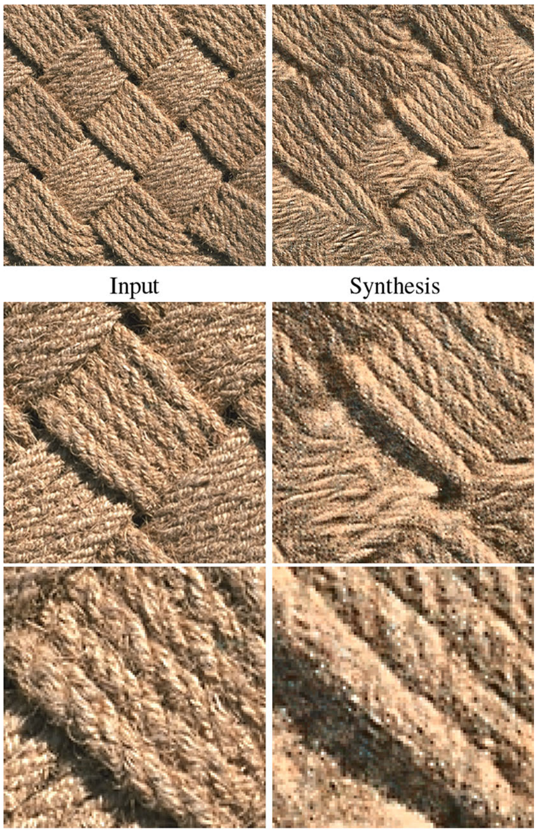


Fig. 8 Synthesis of a 512×512 image using 5 scales and patches of size 18×18. The synthesized image has the same appearance as in the 128×128 image. We see on the zoomed result that fine details are lost since bigger patches are used: a more evolved multi-scale scheme may be necessary to handle both very large and very thin patterns.

the small details are lost. Moreover, bigger patches imply more coefficients, a bigger dictionary, and less sparsity: the sparse decomposition algorithm becomes far slower and less accurate (we recall that we use a heuristic since the problem is NP-hard).

Figure 8 shows the synthesis of a higher resolution image of 512×512 pixels, which illustrates the limitations of our method. The size of the patches is set to 18×18 and 5 scales are used. The algorithm is able to synthesize an image with a similar aspect as in the lower resolution case (128×128 pixels). However the zoom in Fig. 8 shows that the result is less sharp, mainly because the patches are bigger. Synthesizing very large geometric structures as well as small details is challenging. A multi-scale scheme with a varying size of patches may be more appropriate for such a case.

4.3.2 Weights of the Function

Figure 9 shows results obtained when varying the weighting of the different terms in the functional (3) instead of choosing $\alpha_h = \alpha_s = \tilde{\alpha_p} = 1$.

This default choice provides reasonable results for most of the textures, but the weights can be tuned for each texture to be synthesized. We did not find an easy way to automatically compute a good set of weights for each texture, although this would be useful in practice.

The synthesis given in Fig. 9 are obtained, from left to right, with the following settings of $\alpha_s/\tilde{\alpha_p}$: $5/1$, $3/1$, $1/1$, $1/3$, and $1/5$, and always with $\alpha_h = (\alpha_s + \tilde{\alpha_p})/2$. Textures with large structures, long edges, or geometrical elements, are better synthesized with a higher weight for the patch sparsity term. On the contrary, a texture without sharp nor structured elements but with a granular aspect is better reproduced with more weight on the spectrum term. In the extreme case of a texture without any geometrical element, the patch term can be removed: see the examples of the sand texture in Fig. 2 with [14] (spectrum only) or in Fig. 6 without the patches sparsity term (spectrum and histogram).

The intermediate cases (a granular texture with some sharp patterns) are well synthesized with a balanced weighting $\alpha_s = \tilde{\alpha_p}$.

4.3.3 Sparsity

Figure 10 illustrates the effect of the sparsity parameter S.

A larger parameter S means that the patches are a linear combination of more elements from the dictionary D_0. In the case of texture synthesis,

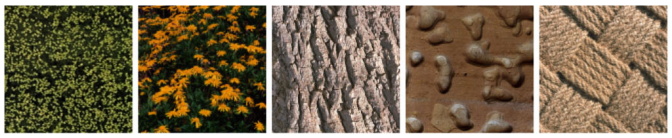
a large value of S allows the superposition of several atoms and create artifacts, particularly visible on the example in Fig. 10.

On the contrary, the smallest value $S = 1$ imposes each patch to be proportional to 1 atom from D_0. Imposing $S = 1$ and binary weights $W \in \{0, 1\}^{N \times K}$ is an interesting alternative. It forces each patch to be equal to an atom of the dictionary. The normalization constraint $\|d_n\|_2 \leq 1$ is no longer necessary and should be removed in this case.

Within this setting, the learning stage (9) becomes a K-means algorithm. The decomposition Algorithm 1 becomes a nearest-neighbor classification, with a constraint on the number F^n of use of each atom. The dictionary is in this case a resampled and weighted version of the set of all the patches in the original image. It is more compact because it contains far less patches than the image. The nearest-neighbor search would thus be far faster than an exhaustive search. Observe that in the case of an exhaustive patch search, the synthesis method is similar to the "Texture Optimization" algorithm [20] and to the video inpainting approach of Wexler et al. [40].

4.3.4 Other Settings

Figure 11 shows the effect of several options of our method.



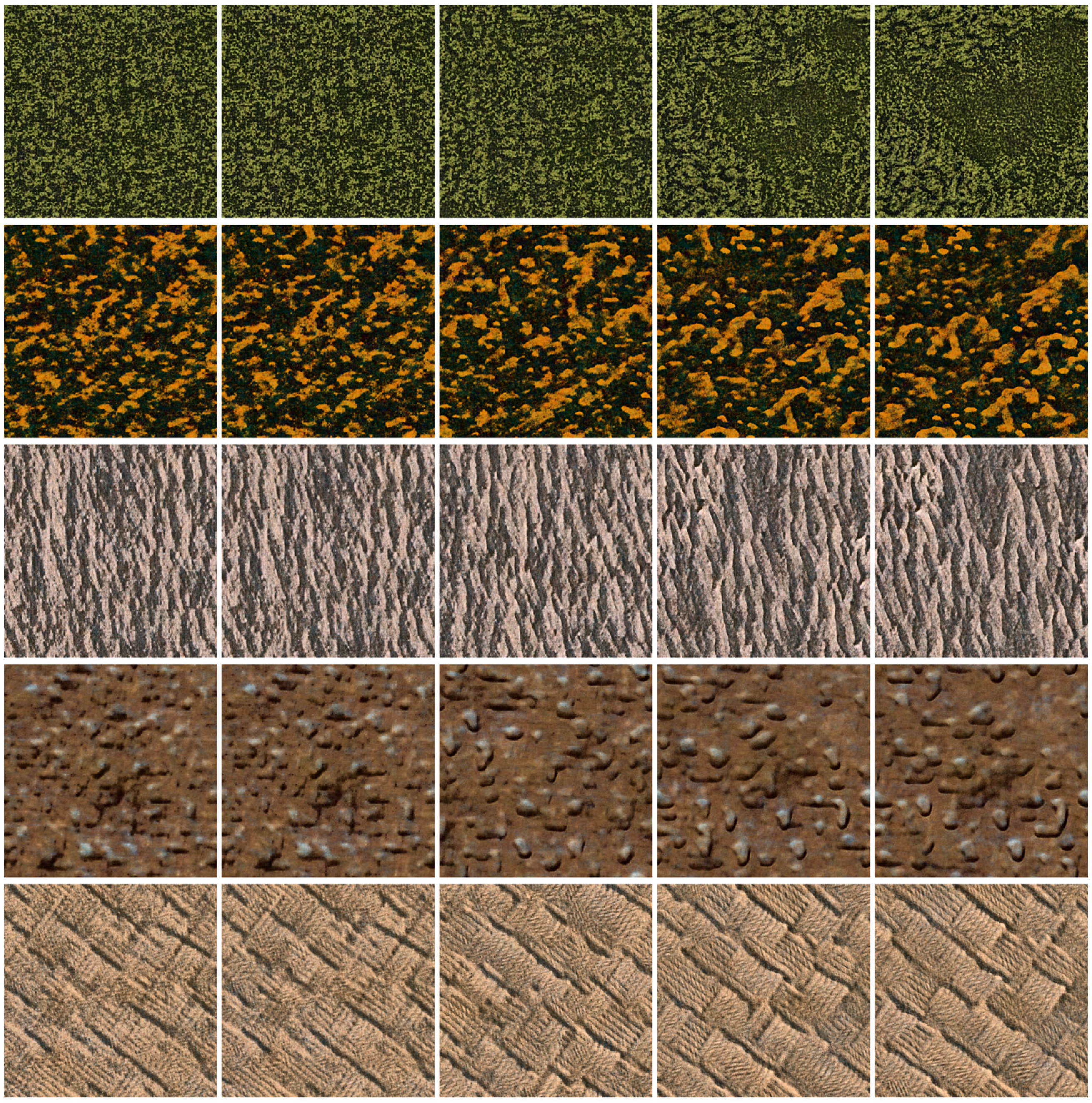


Fig. 9 Synthesis results with different weighting: more spectrum on the left, more patch sparsity on the right. Textures with few geometrical content (*top*) are better reproduced with more spectrum; textures with large structures (*bottom*) need more patches sparsity

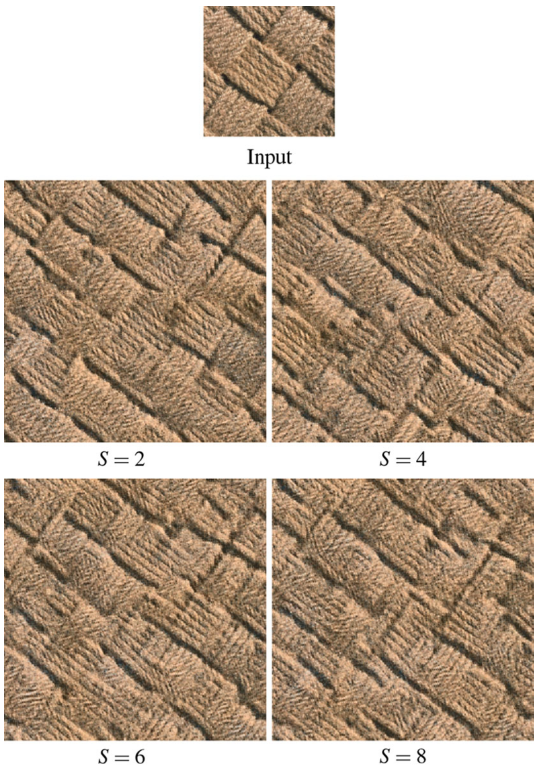


Fig. 10 Synthesis results with different sparsity $S = 2, 4, 6, 8$. This example illustrates the artifacts caused by the superposition of several atoms for larger S.

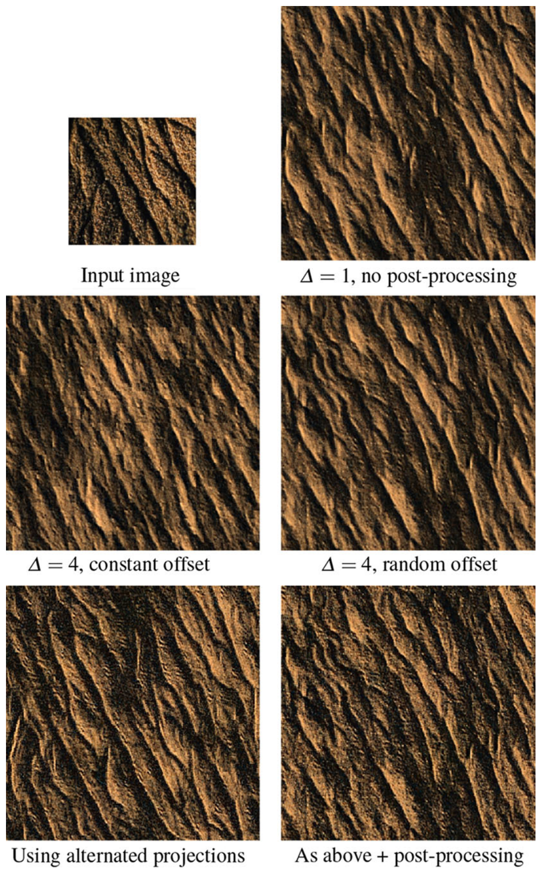

Input image
$\Delta = 1$, no post-processing
$\Delta = 4$, constant offset
$\Delta = 4$, random offset
Using alternated projections
As above + post-processing

Fig. 11 Synthesis results with different settings. Top right: synthesis using all the patches ($\Delta = 1$) without the post-processing step (the final projection on $\mathscr{C}_{\mathrm{s}}$ and then $\mathscr{C}_{\mathrm{h}}$); middle left: using only the patches on a grid of step $\Delta = 4$; middle right: adding random translations of the grid at each iteration; bottom right: adding the post-processing step; bottom left: comparison with alternated projections instead of our gradient descent scheme

The set of patches Πu can be subsampled to improve the speed of the algorithm as explained in Sect. 3.5.

We only consider the patches lying on a grid of step Δ with $\Delta > 1$. This leads to a mosaic artifact: it appears because the borders of the patches becomes visible. To avoid this artifact, we use random translations of the grid at each iteration: the border of the patches are not always at the same relative position, and the mosaic artifact disappears. Figure 11 shows that using $\Delta = 4$ with random offset gives a similar result than when using $\Delta = 1$, while being 16 times faster.

Alternated projections on the sets $\mathscr{C}_{\mathrm{h}}$, $\mathscr{C}_{\mathrm{s}}$, and $\mathscr{C}_{\mathrm{p}}$ (Sect. 3.3) leads to the images in Fig. 11 (right). The result is quite similar to our result (2nd image from the right) but has some artifacts: some small edges appear, and some grain is missing. More generally, the results obtained using this alternated projection algorithm are sometimes similar to ours, and sometimes suffer from several kinds of artifacts, which point out the bad convergence of that method.

The post-processing step, that is the final projection on $\mathscr{C}_{\mathrm{s}}$ and then $\mathscr{C}_{\mathrm{h}}$, removes residual smoothness of the texture, due to the patch sparsity constraint. This is visible in Fig. 11 (3rd and 4th images from the left).

5 Conclusion

In this paper, we proposed a texture synthesis framework which puts together both a Fourier spectrum constraint and a sparsity constraint on the patches. Thanks to these two constraints, our method is able to synthesize a wide range of textures without a patch-based copy-paste approach. The spectrum controls the amount of oscillations and the grain of the synthesized image, while the sparsity of the patches handles geometrical information such as edges and sharp patterns.

We propose a variational framework to take both constraints into account. This framework is based on a non-convex energy function defined as the sum of weighted distances between the image (or some linear transform of it) and these constraints. The synthesis consists in finding local minima of this function. The proposed framework is generic and can take into account other constraints: in addition to the sparsity and spectrum constraint, we use a histogram constraint to control the color distribution.

A numerical exploration of synthesis results shows that, on contrary to copy-based approaches, our method produces a high degree of innovation, and does not copy verbatim whole parts of the input exemplar.

Acknowledgments We would like to thank the reviewers for their valuable suggestions to improve and complete this paper. We thank the authors of the VisTex database [31] for the set of textures they publicly provide. Gabriel Peyré acknowledges support from the European Research Council (ERC project SIGMA-Vision).

Appendix 1: Proofs

Projection on $\mathscr{C}_s$

We prove expression (8) of the projection of u on $\mathscr{C}_s$. The projection is $u_s \in \mathscr{C}_s$ of the form $\hat{u}_s(m) = e^{i\varphi(m)}\hat{u}_0(m)$. Our goal is to minimize

$$\|u - u_s\|^2 = \sum_m \left\| \hat{u}(m) - e^{i\varphi(m)}\hat{u}_0(m) \right\|^2 \tag{28}$$

with respect to φ.

As the term of the sum are independent, the problem is to minimize

$$f(\psi) = \left\| x - y e^{i\psi} \right\|^2 \tag{29}$$

where $x = \hat{u}(m)$, $y = \hat{u}_0(m)$ and $\psi = \varphi(m)$ for any m. The hermitian product of $x, y \in \mathbb{C}^3$ is denoted by $x \cdot y = \sum_i x_i y_i^* \in \mathbb{C}$

The development of the expression of $f(\psi)$ gives

$$f(\psi) = \|x\|^2 - e^{i\psi} y \cdot x - e^{-i\psi} x \cdot y + \|y\|^2 .$$

The function f being continuous and 2π-periodic on $\mathbb{R}$, it admits (at least) a minimum and a maximum which are critical points ψ_c satisfying $f'(\psi_c) = 0$. Let's write $x \cdot y = Ae^{i\theta}$ with $A \geq 0$. The derivative

$$f'(\psi) = -ie^{i\psi} y \cdot x + ie^{-i\psi} x \cdot y$$

gives $e^{2i\psi_c} = e^{2i\theta}$ and the critical points ψ_c are thus characterized by $e^{i\psi_c} = \pm e^{i\theta}$.

The second derivative

$$f''(\psi) = e^{i\psi} y \cdot x + e^{-i\psi} x \cdot y .$$

provides more information: we know $e^{i\theta}$ is a minimum since $f''(e^{i\theta}) = 2A \geq 0$, and $-e^{i\theta}$ is a maximum since $f''(-e^{i\theta}) = -2A \leq 0$.

The case $x \cdot y = 0$ leads to $A = 0$ and f being constant. In other cases, $A > 0$: the minimums $\psi_{\min}$ of the functions are strict and satisfy $e^{i\psi_{\min}} = e^{i\theta} = \frac{x \cdot y}{|x \cdot y|}$, hence the expression of $\hat{u}_s(m)$ given in (8).

Projection on $\mathscr{C}_p$

We provide here the proof that (15) and (16) are the minimizers of (12).

Using (17), the expression (12) is $\left\| R^{(\ell)} - wD_0 E_{n,k} \right\|^2$. Expressing this norm as the sum of the norms of the columns leads to

$$\min_{k,n,w} \left\| R_k^{(\ell)} - wD_n \right\|^2 + \sum_{k' \neq k} \left\| R_{k'}^{(\ell)} \right\|^2 . \tag{30}$$

Let us fix k and n. The problem

$$\min_w \left\| R_k^{(\ell)} - wD_n \right\|^2 \tag{31}$$

is now an orthogonal projection. Since D_n is normalized, the solution is

$$w^* = \langle R_k^{(\ell)}, D_n \rangle \tag{32}$$

as stated in (16) and Pythagora's theorem gives the error

$$\left\| R_k^{(\ell)} - w^* D_n \right\|^2 = \left\| R_k^{(\ell)} \right\|^2 - \langle R_k^{(\ell)}, D_n \rangle^2 . \tag{33}$$

Substituting w in (30) by its optimal value w^* function of (k, n) simplifies the problem to

$$\min_{k,n} -\langle R_k^{(\ell)}, D_n \rangle^2 + \sum_{k'} \left\| R_{k'}^{(\ell)} \right\|^2 . \tag{34}$$

Hence the result (17) stating $(k^*, n^*) = \arg\max \left| \langle R_k^{(\ell)}, D_n \rangle \right|$ among the set $\mathscr{I}_{W^{(\ell)}}$ of admissible (k, n).

Complexity of the greedy Algorithm 1

Here we show that the complexity of the greedy Algorithm 1 without the back-projection step is

$$\mathscr{O}\big(KN(L + \lambda S \log K)\big). \tag{35}$$

Initialization. Computing the inner products $\Phi = D_0^T P$ between the K patches and the N atoms in dimension L requires $\mathscr{O}(KNL)$ operations using standard matrix multiplications. Precomputing the inner products $D_0^T D_0$ is in $\mathscr{O}(N^2 L)$ operations. Initializing W and R is in KN operation.

Building a max-heap from the absolute values of the KN inner products $\Phi_{n,k} = \langle R_k, D_n \rangle$ has a time-complexity of $\mathscr{O}\big(KN \log KN\big))$.

Loop for $\ell = 1$ *to* λSK. Finding the indices (k^*, n^*) and extracting the maximum $\left|\Phi_{n^*,k^*}\right|$ from the heap requires $\mathscr{O}(\log KN)$ operations.

Updating W with the optimal weight $w^* = \Phi_{n^*,k^*}$ is only one operation. Updating R is in $\mathscr{O}(L)$ operations since only the L coefficients of the k^*-th column is affected. Similarly, updating Φ is in $\mathcal{O}(N \log KN)$ operations since the N updated coefficients must be relocated in the heap, and we recall that $D_0^T D_0$ is precomputed.

Conclusion. Since we assume that $S \ll L \leq N \ll K$, previous bounds can be simplified, and in particular $\log(KN) \in \mathcal{O}(\log K)$. The initializations and precomputations are in $\mathcal{O}(KNL)$. Updating the heap is in $\mathcal{O}(\lambda SKN \log K)$; building and searching the heap are cheaper. Hence the complexity (35).

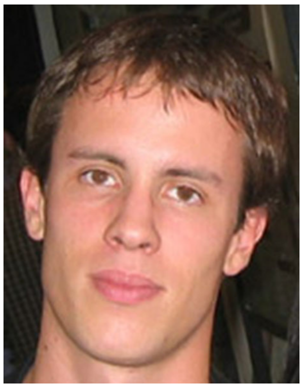
References

1. Aguerrebere, C., Gousseau, Y., Tartavel, G.: Exemplar-based Texture Synthesis: the Efros-Leung Algorithm. Image Processing On Line **2013**, 213–231 (2013)
2. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on, signal processing, **54**(11), 4311–4322 (2006)
3. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (Proceedings SIGGRAPH) 28(3) (2009)
4. Bauschke, H.H., Combettes, P.L., Luke, D.R.: Phase retrieval, error reduction algorithm, and fienup variants: a view from convex optimization. JOSA A **19**(7), 1334–1345 (2002)
5. Bonneel, N., Rabin, J., Peyré, G., Pfister, H.: Sliced and radon wasserstein barycenters of measures. to appear in Journal of Mathematical Imaging and Vision (2014). doi:10.1007/s10851-014-0506-3. http://hal.archives-ouvertes.fr/hal-00881872/
6. Briand, T., Vacher, J., Galerne, B., Rabin, J.:The Heeger and Bergen pyramid based texture synthesis algorithm. Image Processing On Line, preprint (2013)
7. Cross, G., Jain, A.: Markov random field texture models. IEEE Transactions on, Pattern Analysis and Machine Intelligence **5**(1), 25–39 (1983)
8. Desolneux, A., Moisan, L., Ronsin, S.: A compact representation of random phase and gaussian textures. In: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, pp. 1381–1384. IEEE (2012)
9. Efros, A., Freeman, W.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 341–346. ACM (2001)
10. Efros, A., Leung, T.: Texture synthesis by non-parametric sampling. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, vol. 2, pp. 1033–1038. IEEE (1999)
11. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on, Image Processing **15**(12), 3736–3745 (2006)
12. Engan, K., Aase, S., Hakon Husoy, J.: Method of optimal directions for frame design. In: Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, vol. 5, pp. 2443–2446. IEEE (1999)
13. Galerne, B., Gouseau, Y., Morel, J.M.: Micro-texture synthesis by phase randomization. Image Processing On Line (2011)
14. Galerne, B., Gousseau, Y., Morel, J.: Random phase textures: Theory and synthesis. IEEE Transactions on, Image Processing **20**(1), 257–267 (2011)
15. Galerne, B., Lagae, A., Lefebvre, S., Drettakis, G.: Gabor noise by example. ACM Trans. Graph. **31**(4), 73 (2012)
16. Heeger, D., Bergen, J.: Pyramid-based texture analysis/synthesis. In: SIGGRAPH ’95, pp. 229–238 (1995)
17. Julesz, B.: Visual pattern discrimination. IRE Trans. Inf. Theor. **8**(2), 84–92 (1962)
18. Julesz, B.: A theory of preattentive texture discrimination based on first-order statistics of textons. Biol Cybern **41**(2), 131–138 (1981)
19. Kuhn, H.: The hungarian method for the assignment problem. Nav. res. logist. q. **2**(1–2), 83–97 (1955)
20. Kwatra, V., Essa, I., Bobick, A., Kwatra, Kwatra: Texture optimization for example-based synthesis. ACM Trans. Graph. **24**, 795–802 (2005)
21. Lagae, A., Lefebvre, S., Cook, R., Derose, T., Drettakis, G., Ebert, D., Lewis, J., Perlin, K., Zwicker, M.: State of the art in procedural noise functions. EG 2010-State of the Art Reports (2010)
22. Lagae, A., Lefebvre, S., Drettakis, G., Dutré, P.: Procedural noise using sparse gabor convolution. In: ACM Transactions on Graphics (TOG), vol. 28, p. 54. ACM (2009)
23. Lefebvre, S., Hoppe, H.: Parallel controllable texture synthesis. ACM Trans. Grap. (TOG) **24**(3), 777–786 (2005)
24. Lewis, A.S., Luke, D.R., Malick, J.: Local linear convergence for alternating and averaged nonconvex projections. Found. Comput. Math. **9**(4), 485–513 (2009)
25. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online learning for matrix factorization and sparse coding. J Mach. Learn. Res. **11**, 19–60 (2010)
26. Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing **41**(12), 3397–3415 (1993)
27. Moisan, L.: Periodic plus smooth image decomposition. J. Math. Imag. Vis. **39**(2), 161–179 (2011)
28. Olshausen, B., Field, D.: Natural image statistics and efficient coding. Network **7**(2), 333–339 (1996)
29. Perlin, K.: An image synthesizer. SIGGRAPH Comput. Graph. **19**(3), 287–296 (1985)
30. Peyré, G.: Sparse modeling of textures. J. Math. Imag. Vis. **34**(1), 17–31 (2009)
31. Pickard, R., Graszyk, C., Mann, S., Wachman, J., Pickard, L., Campbell, L.: Vistex Database. Media Lab, MIT, Cambridge, Massachusetts (1995)
32. Pitie, F., Kokaram, A., Dahyot, R.: N-dimensional probability density function transfer and its application to color transfer. In: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, vol. 2, pp. 1434–1439. IEEE (2005)
33. Portilla, J., Simoncelli, E.: A parametric texture model based on joint statistics of complex wavelet coefficients. Int. J. Comput. Vis. **40**(1), 49–70 (2000)
34. Ramanarayanan, G., Bala, K.: Constrained texture synthesis via energy minimization. IEEE Transactions on Visualization and Computer Graphics **13**(1), 167–178 (2007)
35. Simoncelli, E., Freeman, W., Adelson, E., Heeger, D.: Shiftable multiscale transforms. IEEE Transactions on Information Theory **38**(2), 587–607 (1992)
36. Tartavel, G., Gousseau, Y., Peyré, G.: Constrained sparse texture synthesis. In: Proceedings of SSVM’13 (2013)
37. Tropp, J.: Greed is good: Algorithmic results for sparse approximation. IEEE Transactions on Information Theory **50**(10), 2231–2242 (2004)
38. Wei, L., Lefebvre, S., Kwatra, V., Turk, G.: State of the art in example-based texture synthesis. In: Eurographics 2009, State of the Art Report, EG-STAR. Eurographics Association (2009)

39. Wei, L., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH '00, pp. 479–488. ACM Press/Addison-Wesley Publishing Co. (2000)
40. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, vol. 1, pp. I-120. IEEE (2004)
41. Xia, G.S., Ferradans, S., Peyré, G., Aujol, J.F.: Synthesizing and mixing stationary gaussian texture models. SIAM J. Imag. Sci. 7(1), 476–508 (2014). doi:10.1137/130918010. http://hal.archives-ouvertes.fr/hal-00816342/

Guillaume Tartavel received the M.Sc. degree from École Normale Supérieure de Cachan, France, and the Engineer diploma from Telecom ParisTech, Paris, France, specializing in image processing and computer science. He is now preparing the Ph.D. degree in the LTCI lab of Telecom-ParisTech, Paris, France, with Yann Gousseau and Gabriel Peyré. His research interests include texture modeling, image/texture restoration, sparse representations, and variational approaches.

Yann Gousseau received the engineering degree from the École Centrale de Paris, France, in 1995, and the Ph.D. degree in applied mathematics from the University of Paris-Dauphine in 2000. He is currently a professor at Télécom ParisTech. His research interests include the mathematical modeling of natural images and textures, stochastic geometry, image analysis, computer vision, and image processing.


Gabriel Peyré graduated from Ecole Normale Supérieure de Cachan, France, in 2003 and received his Ph.D. in applied mathematics from École Polytechnique, Paris, France, in 2005. Since 2006, he has been a researcher at the Centre Nationale de Recherche Scientifique (CNRS), working at Ceremade, University Paris-Dauphine. He is head of the research group SIGMA-Vision, which is funded by the European Research Council (ERC). SIGMA-Vision activity is focused on sparse and adaptive representations with applications in computer vision, computer graphics and neurosciences. Since 2005 Gabriel Peyré has co-authored 35 papers in international journals, 40 conference proceedings and two books. He was the main contributor of the 3rd edition of the book "Wavelet tour of signal processing" of S. Mallat. He is the creator of the "Numerical tour of signal processing" (www.numerical-tours.com), a popular online repository of Matlab/Scilab resources to teach modern signal and image processing.