Constrained Sparse Texture Synthesis

Guillaume Tartavel[1], Yann Gousseau[1], and Gabriel Peyré[2]

[1] LTCI, CNRS / Telecom ParisTech
{tartavel,gousseau}@telecom-paristech.fr
[2] Ceremade, Université Paris-Dauphine
gabriel.peyre@ceremade.dauphine.fr

Abstract. This paper presents a novel texture synthesis algorithm that performs a sparse expansion of the patches of the image in a dictionary learned from an input exemplar. The synthesized texture is computed through the minimization of a non-convex energy that takes into account several constraints. Our first contribution is the computation of a sparse expansion of the patches imposing that the dictionary atoms are used in the same proportions as in the exemplar. This is crucial to enable a fair representation of the features of the input image during the synthesis process. Our second contribution is the use of additional penalty terms in the variational formulation to maintain the histogram and the low frequency content of the input. Lastly we introduce a non-linear reconstruction process that stitches together patches without introducing blur. Numerical results illustrate the importance of each of these contributions to achieve state of the art texture synthesis.

Keywords: texture synthesis, sparse decomposition, dictionary learning, variational methods.

1 Introduction

Texture synthesis aims at generating an image that is visually similar to a given input exemplar but at the same time exhibits a strong randomness. Classical methods learn a global statistical model from the exemplar, and then sample a realization from this distribution. Simplest models consider independent stationary coefficients over a Fourier [6] or a wavelet basis [7,3]. More realistic syntheses are achieved by using an adapted representation learned from the exemplar [16] or by using higher order models taking into account dependencies among the coefficients [12].

Another class of methods are based on the Markov Random Field (MRF) assumption that each pixel of the texture depends only on its neighborhood. [2] introduced a parametric MRF model to textures. [4] and [15] propose a non-parametric MRF model where the probability law of a pixel given its neighbors is sampled directly from an exemplar of the texture to be synthesized. These approaches have been improved by several author, see for instance the recent review [14].

These patch-based synthesis methods share similarities with recent sparsity-based methods developed for image restoration. These methods build a dictionary to perform a sparse expansion of the patches of the image in order to achieve state of the art denoising results, see for instance the work of Elad and Aharon [5]. Peyré shows in [11] that dictionary learning can be used for texture synthesis, the dictionary encoding in a compact manner the geometric features of the input image.

Our method builds upon the sparse texture synthesis method of Peyré [11], but extends it significantly to achieve state of the art results in terms of visual quality. We integrate several constraints to enrich the model and propose a variational energy that is minimized during the synthesis.

2 Dictionary Learning

The first step of our method trains a dictionary to approximate patches from the input exemplar. We take here the opportunity to introduce our notations while recalling the process of dictionary learning.

Matrix Notation. We denote a_i and a^j the rows and columns of a matrix $A = (a_i^j)_{i,j}$. The transposed matrix is denoted by A^*. Its ℓ^2 (Frobenius) norm is defined by $\|A\|^2 = \mathrm{tr}(A^*A) = \sum_{i,j} |a_i^j|^2$. The indicator function $\iota_{\mathcal{C}}$ of a set $\mathcal{C}$ is by definition equal to $+\infty$ outside $\mathcal{C}$ and equal to 0 inside $\mathcal{C}$. The ℓ^0 pseudo-norm of a vector w counts its non-zeros coordinates, $\|w\|_0 = \#\{i \setminus w_i \neq 0\}$.

Patches and Dictionary. We process and synthesize an image u by manipulating its patches. Given a set $(x_k)_{k=1}^K$ of K pixel locations, the patch extractor is $\Pi(u) = (p_k)_k \in \mathbb{R}^{L\times K}$ where for $t \in \{0,\dots,\tau-1\}^2$, $p_k(t) = u(x_k + t)$ defines the patch $p_k \in \mathbb{R}^L$ of $\tau \times \tau$ pixels, so that $L = d\tau^2$ where $d = 1$ for grayscale images and $d = 3$ for color images. We constrain the sampling locations on a regular grid $x_k = k\Delta$ for $k \in \mathbb{Z}^2$ where the spacing $\Delta > 0$ controls the amount of sub-sampling.

A dictionary $D = (d_n)_{n=1}^N \in \mathbb{R}^{L\times N}$ is used to approximate the patches $P = \Pi(u)$ as $P \approx DW$, where $W \in \mathbb{R}^{N\times K}$ are the coefficients of the approximation. Note that this corresponds to approximating independently each patch as $p_k \approx Dw_k$ within the dictionary. The quality of the approximation is measured using the ℓ^2 norm, $\|P - DW\|^2 = \sum_{k=1}^K \|p_k - Dw_k\|_2^2$.

Learning Stage. Given an exemplar u_0 of a texture we want to synthesize, an adapted dictionary $D_0 \in \mathbb{R}^{L\times N}$ is learned to provide an optimal sparse approximation of the patches $P_0 = \Pi(u_0) \in \mathbb{R}^{L\times K}$. Similarly to most dictionary learning methods, such as [5], we solve a non-convex optimization problem over the coefficients $W_0 \in \mathbb{R}^{N\times K}$ and the dictionary D_0

$$(W_0, D_0) \in \operatorname*{argmin}_{W,D} \|P_0 - DW\|^2 + \iota_{\mathcal{C}_{\mathrm{cols}}}(W) + \iota_{\mathcal{C}_{\mathrm{dict}}}(D) \tag{1}$$

where we enforce the coefficients to be S-sparse using

$$\mathcal{C}_{\text{cols}} = \left\{ W \in \mathbb{R}^{N \times K} \setminus \|w_k\|_0 \leqslant S \quad \forall k \right\} \tag{2}$$

and where the atoms of the dictionary are constrained to be normalized using $\mathcal{C}_{\text{dict}} = \left\{ D \in \mathbb{R}^{L \times N} \setminus \|d_n\| \leqslant 1 \quad \forall n \right\}$.

Several algorithms have been proposed to minimize approximately a non-convex energy of the form (1), see for instance the K-SVD method of [5].

3 Variational Formulation of the Synthesis Process

Once the dictionary D_0 has been learned from an input exemplar u_0, a texture u (and the associated coefficients W of $\Pi(u)$) is synthesized by minimizing a non-convex energy $E(u, W)$ equal to

$$\frac{1}{Z}\|\Pi(u) - D_0 W\|^2 + \iota_{\mathcal{C}_{\text{cols}}}(W) + \iota_{\mathcal{C}_{\text{rows}}}(W) + \alpha \mathcal{W}_2^2(\mu_u, \mu_{u_0}) + \beta \|h \star (u - u_0)\|^2. \tag{3}$$

Here $Z = \lceil \frac{\tau}{\Delta} \rceil^2$ is constant so that the ℓ^2 data fidelity is normalized with respect to the number of extracted patches. The two parameters $\alpha, \beta > 0$ are weighting the influence of their respective terms. The synthesized images are stationary points of E that are sampled at random with an iterative scheme, which is described in Sect. 4. We now give the precise definition and the rationale for each term of this energy.

Sparse Coding Constraint. The sparse coding energy $\frac{1}{Z}\|\Pi(u) - D_0 W\|^2 + \iota_{\mathcal{C}_{\text{cols}}}(W)$ is the same as the one used for the dictionary learning minimization (1). It requires that all the patches of u are well approximated by an S-sparse expansion in D_0.

Frequency Constraint. The constraint $\mathcal{C}_{\text{rows}}$ imposes that all the geometrical features of u_0 encoded in the dictionary are represented with the same respective proportions in u and u_0. It enforces that atoms of D_0 be used with the same frequencies of occurrence for the sparse expansion of both $\Pi(u_0)$ and $\Pi(u)$. It is defined as

$$\mathcal{C}_{\text{rows}} = \left\{ W \in \mathbb{R}^{N \times K} \setminus \forall n,\ \|w^n\|_0 \leqslant F_0^n \right\}.$$

The frequencies F_0^n are estimated from the input exemplar coefficients W_0 as

$$F_0^n = \frac{K}{K_0}\|w_0^n\|_0, \tag{4}$$

where K and K_0 are the number of patches extracted from u and u_0 respectively.

Histogram Constraint. Maintaining the gray-level or color histogram of a texture is perceptually important for texture synthesis. This is achieved by penalizing the deviation between the empirical gray-level or color distributions μ_u and μ_{u_0} of u and u_0.

An efficient and robust distance between distributions is the optimal transport distance, also known as the Wasserstein distance (see e.g. [13]). When u and u_0 have the same number of pixels, the L^2 Wasserstein distance is defined as

$$\mathcal{W}_2^2(\mu_u, \mu_{u_0}) = \min_{\sigma} \|u - u_0 \circ \sigma\|^2. \tag{5}$$

where σ runs over all the permutations of the pixels. This definition can be extended for images having a different number of pixels. For grayscale images, the optimal permutation is computed by simply sorting the pixel values. For color images, the Wasserstein distance is more involved to compute and to minimize. We approximate it as the sum of the grayscale distances along the three channels in a principal component orthogonal basis.

Low-Pass Constraint. Low frequency patterns, whose sizes exceed τ, are not controlled by the patch decomposition. To avoid the apparition of artifacts, we penalize the deviation of the low frequencies of u with respect to those of u_0 using the term $\|h \star (u - u_0)\|^2$, where $\star$ is the discrete convolution. We use a box filtering kernel $h = (\tau^{-2})_{1 \leqslant i,j \leqslant \tau}$ which performs an averaging over the spatial extension of a patch.

4 Synthesis Algorithm

The synthesis is obtained by randomly sampling the stationary points of $E(u, W)$ by a block-coordinate descent method that minimize E iteratively with respect to u and W. Pseudo-code 1 details the different steps of the method that are detailed in the remaining part of this section.

Algorithm 1: texture synthesis algorithm by minimization of (3).

Data: input texture u_0.
Input: parameters $\tau, \Delta, S, \alpha, \beta, N$.
Output: synthesized texture u.

1. **Dictionary learning:** compute (D_0, W_0) by minimizing (1).
2. **Frequency estimation:** compute $(F_0^n)_n$ using (4).
3. **Initialization:** set u to be a random white noise image.
4. **Block-coordinate minimization:** repeat until convergence
   - *image update:* $u \approx \underset{u}{\operatorname{argmin}}\, E(u, W)$, see Sect. 4.1.
   - *coefficient update:* $W \approx \underset{W}{\operatorname{argmin}}\, E(u, W)$, see Sect. 4.2.

4.1 Step 1: Minimization with respect to u

Given a fixed set of coefficients W, we compute the minimization of $E(u, W)$ with respect to u alone

$$\min_u \tilde{E}_W(u) = \frac{1}{Z}\|\Pi(u) - P\|^2 + \alpha \mathcal{W}_2^2(\mu_u, \mu_{u_0}) + \beta \|h \star (u - u_0)\|^2 \qquad (6)$$

where $P = D_0 W$ is fixed.

Gradient Descent. The function $\tilde{E}_W$ is smooth almost everywhere since $\mathcal{W}_2^2$ is defined in (5) as the minimum among a set of paraboloids. It has a Lipschitz gradient. We thus use a gradient descent scheme to solve approximately (6)

$$u^{(\ell+1)} = u^{(\ell)} - \eta \nabla \tilde{E}_W(u^{(\ell)})$$

where $u^{(0)}$ is initialized from the previous iteration of the synthesis process.

The gradient of $\tilde{E}_W$ reads

$$\nabla \tilde{E}_W(u) = 2\mathcal{R}(u, P) + \alpha \nabla_u \mathcal{W}_2^2(\mu_u, \mu_{u_0}) + 2\beta \bar{h} \star h \star (u - u_0) \qquad (7)$$

$$\text{where} \quad \mathcal{R}(u, P) = \frac{1}{Z}\Pi^* \left(\Pi(u) - P\right)$$

and where $\bar{h}(x) = h(-x)$. The step sizes η must be smaller than twice the inverse of the Lipshitz constant of this gradient, $0 < \eta < 4 \times (1 + \alpha + \beta)^{-1}$.

Gradient of the Wasserstein Distance. When u and u_0 are grayscale images with the same number of pixels, the gradient of $u \mapsto \mathcal{W}_2^2(\mu_u, \mu_{u_0})$ reads

$$\nabla_u \mathcal{W}_2^2(\mu_u, \mu_{u_0}) = 2\left(u - u_0 \circ \sigma_{u_0} \circ \sigma_u^{-1}\right)$$

where σ_v is a permutation that order the pixel values $(v_i)_i$ of an image v,

$$v_{\sigma_v(1)} \leqslant \ldots \leqslant v_{\sigma_v(i)} \leqslant v_{\sigma_v(i+1)} \leqslant \ldots$$

The permutation σ_u is not unique when $u \mapsto \mathcal{W}_2^2(\mu_u, \mu_{u_0})$ is not differentiable. However, a descent direction is obtained by considering any valid ordering. When u and u_0 are color images, $\nabla_u \mathcal{W}_2^2(\mu_u, \mu_{u_0})$ is computed as the sum of the gradients over the three channels of the principal components of the distribution of the pixels of u_0.

Non-linear Improved Reconstruction. We note that $\frac{1}{Z}\Pi^*\Pi = \mathrm{diag}_i(\rho_i/Z)$ where $\rho_i \leqslant Z$ is the number of patches that overlap at a pixel location i. In the case of a perfect tiling, $\rho_i = Z$ is constant and $\frac{1}{Z}\Pi^*\Pi = \mathrm{Id}$: we can thus write

$$\mathrm{diag}_i(Z/\rho_i)\mathcal{R}(u, P) = u - \Pi^+ P$$

where $\Pi^+ = (\Pi^*\Pi)^{-1}\Pi^* = \mathrm{diag}_i(1/\rho_i)\Pi^*$ is the pseudo-inverse of Π. The term $\mathcal{R}(u, P)$ thus involves images that are reconstructed linearly by an averaging of patches. This step thus typically induces blur in the image u recovered at convergence. We improve this reconstruction by replacing the linear pseudo-inverse Π^+ by a Non-Linear (NL) reconstruction operator Π^+_{NL}, and replace, in the gradient expression (7), $\mathcal{R}(u, P)$ by $\mathcal{R}_{\mathrm{NL}}(u, P) = u - \Pi^+_{\mathrm{NL}}(P)$.

Graph-Cuts Reconstruction. As a particular example of non-linear, edge-preserving, reconstruction operator $\Pi^+_{\mathrm{NL}}(P)$, we use the graph-cut reconstruction introduced in [9] for texture synthesis. The idea is to sequentially blend each pair of adjacent patches along a cut. The patches are juxtaposed instead of being averaged. For a given patches collection, the resulting image is much sharper than the image obtained by linear reconstruction $\Pi^+(P)$.

A vertical cut γ between two consecutive patches (p_1, p_2) in P is a vertical path splitting the overlapping pixels into 2 groups. It is thus a subset of edges joining pairs of pixels (x_1, x_2). An optimal cut is computed by minimizing a functional measuring how well the two patches can be juxtaposed seamlessly along γ

$$J(\gamma, p_1, p_2) = \sum_{(x_1,x_2)\in\gamma} \frac{\|p_1(x_1) - p_2(x_1)\|^2 + \|p_1(x_2) - p_2(x_2)\|^2}{\|p_1(x_1) - p_1(x_2)\|^2 + \|p_2(x_1) - p_2(x_2)\|^2}. \tag{8}$$

The minimization of $J(\gamma, p_1, p_2)$ with respect to γ is done by linear programming. The full image reconstruction $\Pi^+_{\mathrm{NL}}(P)$ is performed in a greedy manner. Patches are first merged using vertical cuts resulting in complete rows. These rows are then merged together using large horizontal cuts.

Note that the resulting term $\mathcal{R}_{\mathrm{NL}}(u, P) = u - \Pi^+_{\mathrm{NL}}(P)$ does not correspond anymore to the true L^2 gradient. The non-linear behavior of the graph cut operator makes it difficult to analyze the convergence of the resulting process. Numerical simulations indicate that the process converges in practice, and that no blur is created by these iterations. An interesting question for future work is to understand whether the modification of the descent scheme can be re-casted as a minimization of some edge-preserving energy.

4.2 Step 2: Minimization with respect to W

The minimization of E with respect to W when u is fixed corresponds to the following combinatorial optimization problem

$$\min_W \|P - D_0 W\|^2 + \iota_{\mathcal{C}_{\mathrm{cols}}}(W) + \iota_{\mathcal{C}_{\mathrm{rows}}}(W) \tag{9}$$

where $P = \Pi(u)$ is fixed. Even in the case where $\mathcal{C}_{\mathrm{rows}}$ is dropped (usual sparse coding), this problem is known to be NP-hard. We thus extend the Matching Pursuit (MP) greedy algorithm [10] to take into account the additional constraint $\mathcal{C}_{\mathrm{rows}}$ and compute an approximate solution of (9). Pseudo-code 2 describes the steps of this Constraint Matching Pursuit (CMP) algorithm, that are detailed in the remaining part of this section.

Index Selection Step. At step ℓ, the algorithm greedily updates the coefficients $W^{(\ell)}$ to reduce as much as possible the amplitude of the residual $R^{(\ell)} = P - D_0 W^{(\ell)}$ while staying within the constraint sets $\mathcal{C}_{\mathrm{rows}}$ and $\mathcal{C}_{\mathrm{cols}}$. This update only increases by at most one the number of non-zero coefficients

$$\varepsilon^\star = \operatorname*{argmin}_{\|\varepsilon\|_0=1} \|P - D_0(W^{(\ell)} + \varepsilon)\|^2 + \iota_{\mathcal{C}_{\mathrm{cols}}}(W^{(\ell)} + \varepsilon) + \iota_{\mathcal{C}_{\mathrm{rows}}}(W^{(\ell)} + \varepsilon).$$

Algorithm 2: constrained matching pursuit to approximately solve (9).

Data: patches P, dictionary D_0.
Input: sparsity S, frequencies F_0.
Output: coefficients W.
for $\ell = 0$ **to** $SK - 1$ **do**
– select the indices $(k^\star, n^\star)$ by solving (10).
– update the coefficients to obtain $W^{(\ell+1)}$ using (11).
– update the residual $R^{(\ell+1)} = P - D_0 W^{(\ell+1)}$ using (12).

Similarly as in the case of the MP algorithm, the optimal 1-sparse vector $\varepsilon^\star$ indexes an atom $d_{n^\star}$ and a patch $r_{k^\star}^{(\ell)}$ of the residual $R^{(\ell)} = (r_k^{(\ell)})_k$. These indices can also be shown to maximize the correlations

$$(k^\star, n^\star) = \underset{(k,n)\in\mathcal{I}_\ell}{\operatorname{argmax}} |\langle r_k^{(\ell)}, d_n \rangle| \tag{10}$$

where $\mathcal{I}_\ell$ is the set of indices that are still available at step ℓ

$$\mathcal{I}_\ell = \left\{ (k,n) \setminus \sum\nolimits_{n' \neq n} \|(w^{(\ell)})_k^{n'}\|_0 < S \text{ and } \sum\nolimits_{k' \neq k} \|(w^{(\ell)})_{k'}^{n}\|_0 < F_0^n \right\}$$

where $W^{(\ell)} = \left((w^{(\ell)})_k^n\right)_{k,n}$ are the coefficient at step ℓ.

Coefficient Update Step. The coefficients are then updated according the MP rule

$$(w^{(\ell+1)})_k^n = \begin{cases} (w^{(\ell)})_k^n + \langle r_k^{(\ell)}, d_n \rangle & \text{if} \quad (k,n) = (k^\star, n^\star), \\ (w^{(\ell)})_k^n & \text{otherwise;} \end{cases} \tag{11}$$

and the residual $R^{(\ell+1)} = P - D_0 W^{(\ell+1)}$ becomes

$$r_k^{(\ell+1)} = \begin{cases} r_k^{(\ell)} - \langle r_k^{(\ell)}, d_n \rangle \cdot d_n & \text{if} \quad k = k^\star, \\ r_k^{(\ell)} & \text{otherwise.} \end{cases} \tag{12}$$

Computational Complexity. Under the assumption that $S \leqslant L, N \leqslant K$, the number of operations of the CMP algorithm is $O(KN(L + \log K))$ when pre-computing the inner products and using a heap max-search. The computation of all inner products $\langle p_k, d_n \rangle$ provides a rough lower bound KNL for both our algorithm and the original version of MP [10].

4.3 Multi-scale Synthesis

The energy $E(u, W)$ is highly non-convex and the optimization process is likely to fall in bad local minima. Following several works on texture synthesis such as [11], we use a multi-scale strategy, that is particularly efficient when

synthesizing images with features having various scales, such as a quasi-periodic tiling of small scale features.

We first proceed by filtering and down-sampling the input exemplar u_0 to produce a multi-scale hierarchy of J images $(u_j)_{j=0}^{J-1}$, where u_j corresponds to a sub-sampling by a factor 2^j. Keeping a fixed patch size but a varying resolution allows the method to capture details of varying sizes. A dictionary D_j is learned for each u_j following the method described in Sect. 2. The synthesis algorithm detailed in pseudo-code 1 is then applied for $j = J-1, \dots, 1, 0$ with (u_j, D_j) in place of (u_0, D_0). Between two scales j and $j-1$, the current texture u output at scale j is up-sampled by a factor 2 using bi-cubic interpolation to serve as the initialization for the synthesis step at scale j.

5 Synthesis Experiments

In this section, we provide comparisons between the proposed method and 3 classical synthesis algorithms. We also illustrate the contribution of each term in the energy (3).

Choice of the Parameters. For all numerical experiments in this section, we use patches of width $\tau = 12$ and a spacing $\Delta = \tau/2$. The synthesis is performed through $J = 3$ scales. We choose $S = 4$ non-zero values per patch and $N = 384$ elements in the dictionary. The parameters of the energy (3) are chosen as $\alpha = \beta = 1$; we observed that changing these values within reasonable proportions has little visual influence on the results.

Comparison. Our results are compared with 3 other decomposition-based texture synthesis algorithms [7,12,11]. Peyré's approach [11] is, to the best of our knowledge, the only synthesis model using sparse dictionary decomposition; our work is based on this approach. The method from Portilla and Simoncelli [12] is a state of the art method for generic texture synthesis. Let us here emphasize that we are interested in algorithms that truly *generate* a new texture from an exemplar. Copy-paste methods such as the classical Efros-Leung algorithm [4] and numerous related approaches (see e.g. [14]) produce visually striking results on a larger class of textures than [12]. However they merely proceed, either explicitly or not, by stitching together pieces from the exemplar, as illustrated in Fig. 1. These approaches are therefore not included in the present comparison. The method from Heeger and Bergen [7] is included for methodological reasons. It relies on the prescription of both the marginals of wavelet coefficients and the gray level (or color) distribution of images. Therefore, it is closely related to our method, albeit working in a prescribed, non-adaptive dictionary.

In Figure 2 are displayed several successful synthesis examples on textures from the Brodatz database [1]. On these, the proposed method performs significantly better than the method from Heeger and Bergen [7], especially for structured textures. This is mostly due to the fact that learned dictionary are

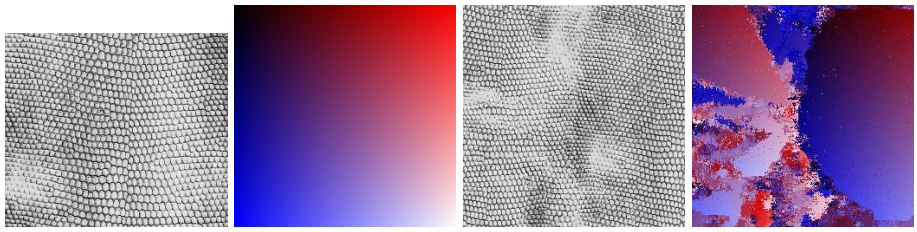

Fig. 1. A synthesis example using the method from [4]. From left to right: input, pixel coordinates visualized via a colormap, synthesis result, original position of the pixels used for the synthesis. Although pixels are synthesized one at a time, the texture is produced by stitching together pieces from the exemplar.

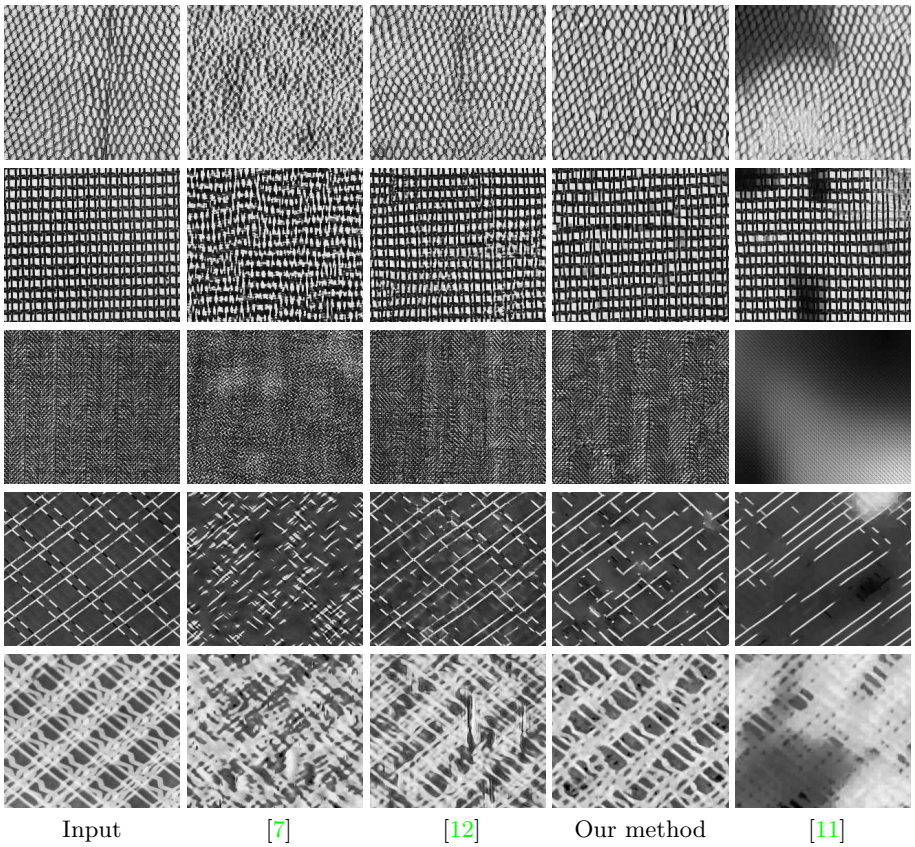

Fig. 2. From left to right: input texture, result using [7], result using [12], result from the proposed method, and result using the original framework [11]. The latter is often too smooth because of the multi-scale processing. All textures are from the Brodatz album [1].

more efficient than wavelet dictionary at capturing edges, corners or other geometric characteristics of these textures. Second, results on these examples are comparable to those from [12]. Observe that this last method relies on second order statistics (correlations) between wavelet coefficients, while our approach only controls the proportion in which each dictionary atom is used. This indicates that the learned atoms could provide an interesting mathematical model of textons, as defined in [8]. Third, the importance of the penalty terms we introduced in energy (3) is evident through the comparison with the original method [11].

In Figure 3 are displayed two failure examples, a very large scale texture in the first row and a micro-texture in the second row. While the synthesis of very large scale textures without copy-pasting is still an open problem, micro-textures are successfully captured by relatively simple models such as the random phase model from [6]. A rough explanation of the inability of our method to synthesize such textures is that the sparse decomposition model is not adapted for noise-like patches.

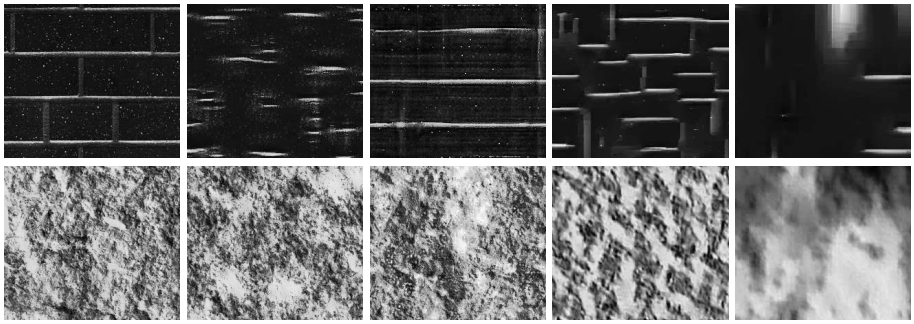

Fig. 3. Failure examples. Top: a large scale texture, bottom: a micro-texture for which the sparse hypothesis is not adapted.

Step-by-Step Analysis. In this second set of experiments, we illustrate the contributions of both the different components of energy (3) and the chosen minimization strategy. For each tested texture, we compare the following synthesis procedures:

- *basic*: only keep the first two terms (sparse coding constraint) and the fourth term (histogram constraint) of energy (3), which gives a method very similar to the initial framework of [11],
- *atom frequency*: add the atom frequency constraint $\mathcal{C}_{\text{rows}}$,
- *graph cut*: add the graph-cut reconstruction described in Sect. 4.1,
- *multi-scale*: add the multi-scale strategy described in Sect. 4.3,
- *low frequency*: add the low frequency constraint (last term of (3)), yielding the complete proposed procedure.

Several observations can be drawn from the results shown in Fig. 4. First, the atom frequency constraint is important for the generation of geometric structures and avoids an excessive use of smooth patches. Second, the non-linear image reconstruction procedure yields sharper results than the averaging of patches by the operator Π^*. Third, the multi-scale strategy introduces large scale coherence, without the computational cost of using larger patch sizes. Last, the low frequency constraint prevents from large scale variations due to the independence of patches.

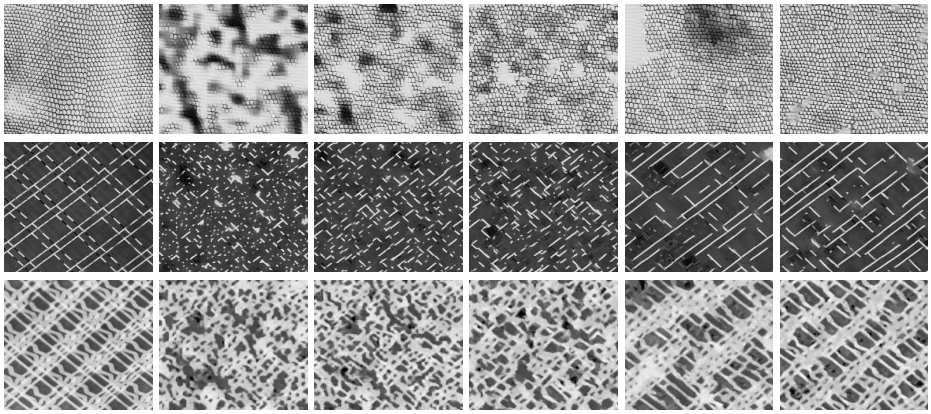

Fig. 4. Step-by-step examples. For each example, the result in the second column is obtained using a basic sparse synthesis scheme. Each column then shows the effect of adding a new constraint or of changing the minimization strategy. The last column is the complete proposed synthesis procedure.

Conclusions and Future Work

In this article we presented a variational approach to the texture synthesis problem. It extends significantly the initial sparsity-based framework of [11].

We identified a set of constraints to make the sparse approach suitable for texture synthesis. The first constraint controls the frequency of occurrence of each atom of the dictionary. The second constraint compensates the lack of coherence between adjacent patches. The third refinement is a cut-based reconstruction in the patch-based framework. The last and common refinement is the multi-scale processing.

The resulting model is well adapted to textures with sharp edges and small quasi-periodic patterns as shown in Fig. 2. It is less suitable for textures with high frequencies or structures at very large scale. Interesting perspectives include a better modeling of noisy textures, possibly through constraints on the power spectrum of images as in [6], as well as the use of a multi-scale learned dictionary.

Another perspective is to explore the variational approach which formulates the synthesis problem as a (highly non-convex) minimization problem. This paper uses a basic gradient descent but more efficient approaches may be used. The solutions given by the minimization algorithm are (at most) local minima of the energy. Do they all look similar? If not, how to get a "good" solution?

Acknowledgement: this work has been partly supported by the ANR project MATAIM and by the ERC project SIGMA-Vision.

References

1. P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover, New York, 1966.
2. G.R. Cross and A.K. Jain. Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39, 1983.
3. J.S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368, 1997.
4. A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.
5. M. Elad and M. Aharon. Image denoising via learned dictionaries and sparse representation. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 895–900. IEEE, 2006.
6. B. Galerne, Y. Gousseau, and J.-M. Morel. Random phase textures: Theory and synthesis. *Image Processing, IEEE Transactions on*, 20(1):257–267, 2011.
7. D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95*, pages 229–238, 1995.
8. B. Julesz. A theory of preattentive texture discrimination based on first-order statistics of textons. *Biological Cybernetics*, 41(2):131–138, August 1981.
9. V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003.
10. S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415, 1993.
11. G. Peyré. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34(1):17–31, 2009.
12. J. Portilla and E.P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.
13. Cédric Villani. *Topics in optimal transportation*, volume 58. Amer Mathematical Society, 2003.
14. L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 2009.
15. L.Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.
16. S.C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.