Journal of Mathematical Imaging and Vision (2022) 64:478–492
https://doi.org/10.1007/s10851-022-01078-y

High-Resolution Neural Texture Synthesis with Long-Range Constraints

Nicolas Gonthier[1,2] · Yann Gousseau[1] · Saïd Ladjal[1]

Received: 6 August 2020 / Accepted: 23 February 2022 / Published online: 4 April 2022

Abstract
The field of texture synthesis has witnessed important progresses over the last years, most notably through the use of convolutional neural networks. However, neural synthesis methods still struggle to reproduce large-scale structures, especially with high-resolution textures. To address this issue, we first introduce a simple multi-resolution framework that efficiently accounts for long-range dependency. Then, we show that additional statistical constraints further improve the reproduction of textures with strong regularity. This can be achieved by constraining both the Gram matrices of a neural network and the power spectrum of the image. Alternatively, one may constrain only the autocorrelation of the features of the network and drop the Gram matrices constraints. In an experimental part, the proposed methods are then extensively tested and compared to alternative approaches, both in an unsupervised way and through a user study. Experiments show the advantage of the multi-scale scheme for high-resolution textures and the advantage of combining it with additional constraints for regular textures.

Keywords Texture synthesis · Deep neural network · High resolution · Perceptual evaluation · Multi-scale

1 Introduction

Exemplar-based texture synthesis consists in automatically generating sample images from a given example texture image. These samples are required to be visually faithful to the example and as diverse as possible. For more than forty years, and despite its inherent ill-posedness, this problem has been instrumental in the development of mathematical models for images. In particular, it has enabled one to investigate, through visual experiments, the validity of various mathematical models, ranging from time series [27], Markov random fields [4] to wavelet decompositions [18,29] or nonparametric Markovian modeling [11]. More recently, convolutional neural networks have permitted impressive progresses in the field, initiated by the work by Gatys et al. [13], itself followed by numerous contributions, e.g., [25,36,38].

One challenge that has been faced by all methods since the early days of texture synthesis is the multi-scale nature of texture samples, implying that models should be able to reproduce both small and large scales, possibly over several orders of magnitude. For instance, parametric models for Markov fields are known to be intrinsically badly suited to a multi-scale modeling. Zooming such a model by a given factor implies extremely heavy computations to derive the corresponding parameters [16], impairing the design of multi-scale such models. Wavelet models are more adapted by nature to multi-scale modeling, but the faithful reproduction of structured textures requires complex interactions between scales to be accounted for. The best such modeling up to date is the second order statistical model proposed in [29], but highly structured textures still represent a challenge to such approaches. Nonparametric Markov modeling methods such as those presented in [11] or [10] indeed have the ability to deal simultaneously with several scales, albeit at a high computational cost. However, they are also well known to produce textures with very little variety, often producing verbatim copies, see [1] and the experiments in the present paper. The methods relying on convolutional neural networks, following the seminal work by Gatys et al. [13], are currently the most efficient to capture multi-scale structures. Nevertheless, they still lack efficiency when large-scale regularity is needed, as we will see in detail in this paper.

✉ Nicolas Gonthier
nicolas.gonthier@telecom-paris.fr

[1] LTCI, Télécom Paris, Institut polytechnique de Paris, 19 Place Marguerite Perey, 91120 Palaiseau, France

[2] Université Paris-Saclay, 91190 Saint-Aubin, France

Moreover, they are prone to generate artifacts that prevent a satisfactory reproduction of small scale structures.

In this work, we present several neural synthesis methods that significantly improve the ability to preserve the large-scale organization of textures. We first propose a simple multi-resolution framework that account for large-scale structures and permits the synthesis of high-resolution images. We then show that, in this multi-resolution framework, additional constraints are useful in the case of regular textures. A first approach combines the classical statistical constraints of neural approaches [13] (Gram matrices) with Fourier frequency constraints, similar to those introduced by [12]. A preliminary, mono-scale version of this idea was presented in a conference paper [25]. Alternatively, the multi-resolution framework can be combined with a statistical constraint relying on the full auto-correlation of the features of the network. This approach is closely related to the one introduced in [33], which combines correlations with Gram matrices and various additional constraints. We show that correlation terms alone yield excellent results and therefore that Gram matrices are not necessary in this case.

We then evaluate the proposed methods in an extensive experimental section. The evaluation of texture synthesis results is a challenging task. Some approaches draw on well-chosen statistics to estimate the quality of the results (the closer to the exemplar, the better), as for instance discussed in [3]. In this paper, we first evaluate results in this manner, relying on Kullback–Leibler divergence between wavelet marginals, following the texture indexing scheme from [7]. Then, we also evaluate the proposed methodology through a perceptual user study. Indeed, it is shown in [3,8,9] through extensive experiments that feature-based evaluations do not approach well human-based visual evaluation of texture similarity, especially in the case of long-range correlations, which is precisely one of the cases tackled in this paper. We therefore rely on a user study to compare our framework to both the original method from [14] and some of its improvement that focus on the respect of large-scale structures [33,36].

2 Neural Texture Synthesis

A complete state-of-the-art on the subject of texture synthesis is out of the scope of this paper. In view of the method that we propose in this work, we focus in this section on the works involving CNNs that have followed the seminal contribution of Gatys et al. [13] and particularly on works proposing new statistical constraints and focusing on long-range structure.

2.1 Accelerations and Alternative Sampling Strategy

In a first direction, several works have proposed ways to speed up the synthesis process, notably through feed-forward networks [20,34,38,39]. In [19], generative adversarial networks (GANs) are used to synthesize textures. Such methods enable fast synthesis once the networks have been trained for specific textures, but the quality of results is still inferior to the original approach [13], especially for structured textures. Zhu and other authors have proposed an evolution of the FRAME model [44] in the context of neural networks [26] under the name *DeepFrame*. Textures are synthesized from an exponential model using features from a neural network. In [6], this macrocanonical approach is pushed further and fully analyzed theoretically. It is worth noting that both approaches [6,26] rely on first-order constraints on features and therefore drop the use of the Gram matrices.

2.2 Statistical Constraints and Losses

In a different direction, a large body of works have been dedicated to add additional constraints to the synthesis, often relying on new or modified loss functions. In [15], the color of the synthesis is constrained to specified values. In [31], it is proposed to constrain the histograms of some feature maps, in order to reduce halo artifacts. In [20], a total variation term is added in the loss function for perceptual reasons. Other works such as [2,25,33] also proposed alternative losses to add further statistical constraints. Since they explicitly deal with long-range dependency and structure, they will be reviewed in the next paragraph. It should be noted that these approaches propose to combine several statistical constraints by adding them to get the final loss function. Another possibility would be to alternate different projections as it is done in the seminal work of Portilla and Simoncelli [29]. Alternative constraints have also been investigated for the closely related task of style transfer. In [24], it is shown that matching Gram matrices reduces to kernel-based comparison of features, and various kernels are investigated in this setting. Other works investigate alternatives to the original Gram matrices, such as cross-layers (rather than within-layers) Gram matrices as in [42] or [28] (both inspired by [29]). In [5], it is proposed to consider the image co-occurrences matrices, in a GAN framework, in order to capture local texture patterns. A term based on co-occurrence matrices is added to the loss function and a collection of co-occurrence matrices from the reference is added to the input of both the generator and the discriminator networks.

2.3 Multi-scale Neural Synthesis

Neural networks such as the VGG19 used in most texture synthesis methods intrinsically have a multi-scale structure by alternating convolutions, nonlinearity and subsampling. However, as we will see in the experimental section, the size of the receptive fields in these networks is not sufficient to synthesize large-scale structures, especially when the resolution increases. To the best of our knowledge, only one paper, [36], proposes to rely on a multi-scale strategy to synthesize high-resolution textures. The idea is to feed the network with a multi-scale decomposition, in this case a Gaussian pyramid, instead of a single image. In this paper we will propose an alternative approach to the multi-scale neural synthesis, where scales are considered one at a time, and compare our results with [36].

2.4 Incorporating Long-Distance Dependency

In [2], long-distance patterns are handled by adding in the loss function a cross-correlation term, made of the correlation between features maps and a shifted version of it. Different shifts over a few pixels are used depending on the layer, up to about a sixth of the image size. In [28], in the context of style transfer, a similar idea is investigated using only one-pixel shifts. In [33], the same idea is pushed further, by considering all cross-correlations at once in order to impose long-range structure for regular textures. Several other terms (smoothness, diversity) are added to the loss function. This approach, to which we will compare our results, indeed yields long-range structure, nevertheless at the price of relatively strong artifacts. Apart from these work dealing with cross-correlation of features and closely related to the present paper, [25] proposed to incorporate the power spectrum in the loss function, thereby enabling the respect of highly structured textures. In a related work, [32], it is proposed to impose the spectrum constraint by using a windowed Fourier transform, enabling non-stationary behavior to be accounted for, at the cost of the inherent stationary nature of textures.

3 Multi-scale Spectral Control for Texture Synthesis

In this section, we detail our method to synthesize high-quality texture images. After recalling in Sect. 3.1, the classical approach from Gatys et al. [13], we introduce a simple multi-scale framework in Sect. 3.2, before presenting in Sect. 3.3 the spectral constraint we propose in order to both control artifacts and preserve long-range structures. Finally, we present in Sect. 3.4, the use of the autocorrelation of the feature maps as a potential alternative to the Gram matrices.

3.1 Reminder on the Work from [13]

The seminal work [13] is based on the idea that a network trained for classification purpose, in this case a VGG network as introduced in [35], can be repurposed for a synthesis task. Roughly speaking, the synthesis is achieved by backpropagation of texture-adapted statistical constraints from the inner layers of the network up to pixels of the synthesized image.

More precisely, the method works as follows. We consider a given convolutional neural network [1] consisting of l layers. For a given color texture exemplar $I \in \mathbb{R}^{h\times w\times 3}$, where h, w are the dimensions of the image, we write f^l for the output (the activations) of layer l. This output will be called a *feature map* from now on. Each feature map f^l belongs to $\mathbb{R}^{h_l\times w_l\times m_l}$, where w_l, h_l are the spatial dimension of layer l and m_l the number of channels of the feature. We further write $N = h\times w$ and $N_l = h_l\times w_l$ for the spatial dimensions of respectively the image and the feature maps.

To synthesize a new texture, some statistics are imposed on a subset[2] $\mathscr{S}$ of the layers of the CNN. The statistics considered in [14] are strongly inspired by the work from Portilla and Simoncelli [29] and rely on the so-called Gram matrices $G^l \in \mathbb{R}^{m_l\times m_l}$, defined for each couple $p, q \in \{1, \cdots, m_l\}$ as

$$G^l_{p,q} = \frac{1}{N_l^2}\sum_{i=1}^{N_l} f^l_p(i)\cdot f^l_q(i) = \frac{1}{N_l^2}\langle f^l_q, f^l_p\rangle, \tag{3.1}$$

where $f^l_p \in \mathbb{R}^{N_l}$, for $p \in \{1,\dots,m_l\}$, is the vectorized pth channel of feature l.

To generate a new texture image $\tilde{I}$ on the basis of a reference one I, a gradient descent is used, starting from a white noise image, to find an image that matches the reference statistics. Usually the L-BFGS-B [43] second-order optimization method is chosen.

The corresponding loss function on the features is defined as :

$$\mathscr{L}_{Gram} = \sum_{l=1}^{L} \omega_l \|G^l - \tilde{G}^l\|_2^2, \tag{3.2}$$

with $\omega_l \in \mathbb{R}$ being the weight of the layer l.

The loss function Eq. 3.2 can be seen as multi-objective cost functions agglomerated into a single-objective cost function. Although comparing different objectives is generally difficult, choosing identical weights, i.e., $\omega_l = 1\ \forall l \in [1, L]$, yields perceptually acceptable results.

A central question of texture synthesis is to identify the best sets of statistics to incorporate in this loss function and possibly the irreducible set of those statistics ([21]). Although the method from [13] yields synthesis results of unprecedented quality, a strong limitation is its inability to

[1] In this work, as in [13], we consider the VGG19 network [35] but other choices are possible, including networks with random weights [17].

[2] For simplicity's sake, we will consider layers from 1 to L in the rest of this document but the user can choose non-consecutive layers in the network.

respect long-range dependency, particularly when large-scale structures have some regularity. This can be seen in first row of Fig. 2. Neural networks such as VGG-19 have a multiscale structure, through alternating convolution and subsampling, that allow some large-scale structures to be accounted for. Nevertheless, the size of the filters used in CNNs such as VGG-19, and therefore the size of the corresponding receptive fields, is small with respect to the size of the image especially when synthesizing high-resolution images (here 1024×1024). As we have mentioned in the introduction, several works have addressed this limitation [2,25,28,32,33], but, as we will see in the experimental section, none is fully satisfactory. In the following sections, we propose several improvement of the original neural texture synthesis method in order to address this limitation.

3.2 Multi-scale Synthesis

The first modification we introduce to the method from [13] is a straightforward multi-scale framework that will help preserve the large-scale organization of images. This strategy is relatively classical for texture synthesis methods and has been used in the past in different settings [23,37]. This approach is much simpler than the related method introduced in [36] and, as we will see in the experimental section, yields better results.

The idea is simply to first synthesize a coarse resolution image, which is then upsampled and given as initialization for a synthesis at the next scale. This process is repeated K times until the desired resolution is reached. As illustrated in Fig. 1, we first build an image pyramid from the exemplar image I, iteratively down-sampling it by factors $2^1, 2^2, \ldots, 2^K$, resulting in images $I^{(1)}, I^{(2)}, \ldots, I^{(K)}$. A first synthesis result is obtained by using the smallest image as the exemplar and white noise as initialization. Then, for step $k \in K, K-1, \ldots, 1$, we generate a new result using $I^{(k)}$ as the exemplar and the obtained synthesis result $\tilde{I}^{(k-1)}$ as the initialization instead of white noise. The upsampling of $\tilde{I}^{(k-1)}$ is performed using bilinear interpolation. The only parameter of this generic multi-scale framework is the number of scales K.

As can be seen in Fig. 18, this strategy can yield strong improvements in some cases but is not enough to allow the reproduction of highly structured textures. In the next section, we show how the result can be improved by adding a careful control of the Fourier spectrum into the multi-scale scheme.

3.3 Spectrum Constraint

We propose to include in the synthesis a new constraint based on the Fourier spectrum of the image. It is known that such a constraint alone is an efficient way to reproduce the so-called *micro-textures* [12] made of uniformly distributed small details. This constraint has also been used in combination with more structured synthesis methods in [37].

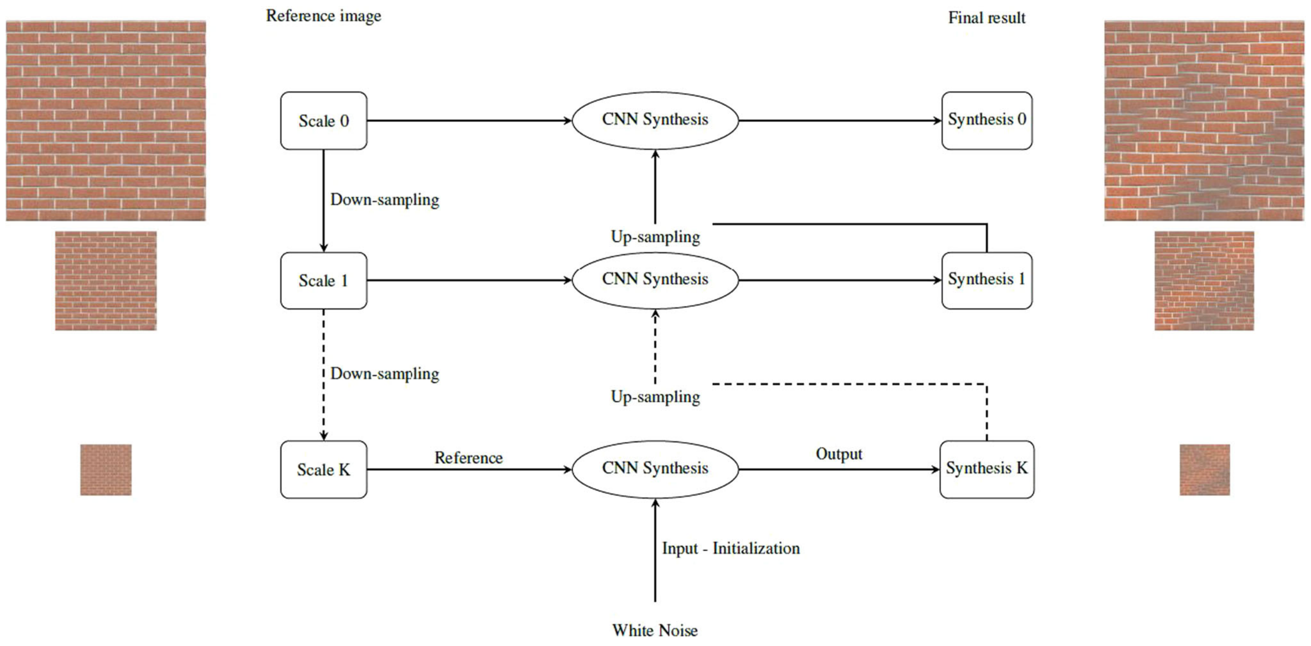


Fig. 1 Multiscale strategy. The exemplar is downsampled by factors $2^{-1}, 2^{-2}, \ldots, 2^{-k}$ to build a pyramid $I^{(1)}, I^{(2)}, \ldots, I^{(k)}$. At scale K, a new texture is synthesized by using $I^{(k)}$ as the exemplar and the upsampled result of the synthesis at scale $K-1$ as initialization (instead of white noise). We repeat this step until we reach the top of the image pyramid

Let us write $\mathscr{F}(I)$ for the discrete Fourier transform (DFT) of an image I, $\mathscr{F}^{-1}$ for the inverse DFT and $|.|$ for the complex modulus. The idea is to constrain the synthesized image $\tilde{I}$ to have a Fourier spectrum $|\mathscr{F}(\tilde{I})|$ as similar as possible to $|\mathscr{F}(I)|$, the spectrum of I. A simple way to do this is to first perform the multi-scale neural synthesis described above and then to replace the phases of the Fourier transform of the synthesized image with random phases, before applying the inverse Fourier transform to the result [12]. Now, this sequential strategy is not satisfactory, since the randomization of phases would destroy the effect of both the statistical constraints on the VGG features and the effect of the multi-scale strategy. Therefore, we propose to incorporate the Fourier constraint into the multi-scale synthesis process. A preliminary, mono-scale version of this idea was presented in [25].

In order to include the Fourier constraint into the loss function used for synthesizing images, we first introduce $\mathscr{E}_I$, the set of images having the same spectrum as I the exemplar image. In the case of color images, this is defined as

$$\mathscr{E}_I = \left\{ J \in \mathbb{R}^{h\times w\times 3} | \exists \phi \in \mathbb{R}^{h\times w} : \mathscr{F}(J) = e^{i\phi}\mathscr{F}(I) \right\}.$$

Next, we define the Fourier loss associated to the image $\tilde{I}$ as the normalized Euclidean distance between $\tilde{I}$ and $\mathscr{E}_I$,

$$\mathscr{L}_{spe} = \frac{1}{2N} d(\tilde{I}, \mathscr{E}_I)^2 = \frac{1}{2N} \|\tilde{I} - \mathscr{P}(\tilde{I})\|^2, \tag{3.3}$$

and the total loss as

$$\mathscr{L} = \mathscr{L}_{Gram} + \beta \mathscr{L}_{spe}, \tag{3.4}$$

where β is a weighting parameter. Since the Fourier loss is the distance to $\mathscr{E}_I$, its gradient is given by

$$\Delta_{spe} = N^{-1}(\tilde{I} - \mathscr{P}(\tilde{I})),$$

where $\mathscr{P}$ is the projection operator on $\mathscr{E}_I$. This projection is given by (see [37], Appendix A)

$$\mathscr{P}(\tilde{I}_c) = \mathscr{F}^{-1}\left(\frac{\mathscr{F}(\tilde{I}) \cdot \mathscr{F}(I)}{|\mathscr{F}(\tilde{I}) \cdot \mathscr{F}(I)|} \cdot \mathscr{F}(I_c) \right), c \in \{r, g, b\} \tag{3.5}$$

where $\cdot$ is the scalar product in $\mathbb{C}^3$, that is

$$\mathscr{F}(\tilde{I}) \cdot \mathscr{F}(I) = \sum_{c=r,g,b} \mathscr{F}(\tilde{I}_c)\mathscr{F}(I_c)^*,$$

I_c, for $c = r, g, b$, being the color channels of I and a^* the conjugate of complex number a. This spectrum constraint can be seen as a regularization to the ill-posed example-based synthesis problem.

3.4 Autocorrelation of the Feature Maps

In this section, we consider an alternative way to impose long-range consistency, based on the autocorrelation of the features maps. This is motivated by the fact that the autocorrelation is a proxy of repeating patterns, such as the presence of periodic elements in the signal. As explained in Sect. 2, this idea has been explored with different modality in [2,28,33].

The autocorrelation function of an image is defined as the convolution of the image with itself. Let $I \in \mathbb{R}^{h\times w}$, the autocorrelation $C(I) =\in \mathbb{R}^{h\times w}$ is defined, for $\forall k \in \{1, \dots, h\}$ and $\forall l \in \{1, \dots, w\}$, as

$$C(I)(k,l) = \frac{1}{N^2} \sum_{i=1}^{h} \sum_{j=1}^{w} I(i,j) I(|\, i+k \,|_h, |\, j+l \,|_w) \tag{3.6}$$

$$= \frac{1}{N^2} I * I \tag{3.7}$$

$| \bullet |_h$ being the modulo operation with divisor h.

And efficient way to compute the autocorrelation is to use the discrete Fourier transform (DFT). According to the Wiener–Khintchin theorem, we have:

$$C(I) = \mathscr{F}^{-1}(|\, \mathscr{F}(I) \,|^2).$$

Then, we define the autocorrelation constraint at the layer l as $A^l \in \mathbb{R}^{h_l\times w_l\times m_l}$ the tensor of the squared modulus of the Fourier transform of the features maps, i.e.,

$$A^l_p = \frac{1}{N_l^2} \,|\, \mathscr{F}(f^l_p) \,|^2 \tag{3.8}$$

with $p \in \{1, \dots, m_l\}$ being the corresponding indexes of the feature map p. Using this toric representation allows one to consider all possible shifts between pixels.

This constraint is similar to the one in [33], except that it is dealt with in the Fourier domain and there is no weighting of the elements of the autocorrelation matrix.

4 Experiments

In this section, we perform experiments to illustrate both the multi-scale framework and the additional constraints we propose for neural texture synthesis. After briefly introducing the methods we compare ourselves to, we first show some visual results. Then, we propose a method to evaluate the innovation capacity of algorithms, and more precisely their tendency to produce verbatim copy of the input. Further, we evaluate the methods quantitatively using the Kullback–Leibler divergence between wavelet statistics. Despite the interest of such quantitative evaluations, it is known that they have severe limitations, in particular to evaluate results at large scales [9]. Therefore, we have also conducted a medium scale perceptual evaluation from human observers, the results of which we analyze in Sect. 4.3.3. These different evaluations have been conducted on the 20 texture images visible in Fig. 8. These high-resolution (1024×1024) textures have been chosen to include both structured and irregular textures. Some of them display strong long-range dependency. All results can be found in Supplementary Materials. Eventually, we study the effects of various parameters and briefly illustrate the ability of our method to produce higher resolution textures.

4.1 Architecture and Parameters

We use a VGG-19 network pre-trained on ImageNet with rescaled weights[3] as in [13] and we also use the same layers, i.e., "Conv1_1," "Pooling1," "Pooling2," "Pooling3" and "Pooling4." The corresponding weights[4] are set to be $w_1 = w_2 = w_3 = w_4 = w_5 = 10^9$. When the spectrum constraint is considered, we use a weighting parameter $\beta = 10^5$ unless otherwise specified. Synthesis are performed using 2000 iterations. We use TensorFlow as a deep learning framework and Scipy as an optimization package.

4.2 Other Texture Synthesis Methods

The first method we compare ourselves to is the original synthesis method from Gatys et al. [13] that from now we refer to as "Gatys". We also consider the method "Deep Corr", introduced in [33], using the code from the authors.[5] We also consider the multi-scale texture algorithm from [36], using the code from the author,[6] using layers 3 and 8 and 5 octaves in the Gaussian pyramid as in the original paper. We use a maximum of 2000 iterations for all the CNN-based methods. From now on, we refer to this method as "Snelgrove". Those last two methods have been chosen because they explicitly address the problem of reproducing large-scale structures. We also consider the feed-forward approach proposed in [38] using a PyTorch implementation by Jorge Gutierrez.[7] We refer to this method as "Ulyanov". Finally, we consider two patch-based methods, from the works [10,11], using implementations from the online journal IPOL [1,30], with default parameters settings. We refer to these two methods, respectively, as "Efros Leung" and "Efros Freeman".

Rough execution times of the different methods are as follows. We emphasize that codes are not optimized and that these times are given to fix ideas. For our method "Gram+MSInit," synthesizing one texture of size 1024×1024 takes about 60 min[8] with a GeForce 1080 Ti . This is about twice the execution time needed by the original approach from Gatys et al. [13] (30 min) at the same scale. The "Snelgrove" method [36] takes about 40 min with the same output scale. On the other hand, "Deep Corr" [33] takes roughly a day on a similar GPU[9] and "Ulyanov" [38] needs about 6h to train a network per texture, the inference being then almost instantaneous. In comparison, the patch-based methods [10,11] are the fastest and only take about a minute.

4.3 Comparisons

In Figs. 2, 3, 4 and 5 we display synthesis results using our methods and those presented in the previous paragraph. For space reason, we only consider 4 textures, all exhibiting some kind of long-range dependency. Their resolution is 1024×1024. Some details can be seen in Fig. 6. All results can be seen in Supplementary Materials (Sect. 2).

We first notice that patch-based methods are very faithful to the reference image. However, they have the tendency to produce regions that are exact copy of the input, a verbatim effect already noticed in [1] and investigated in the next section. They also at times yield images with constant or repetitive patterns.

Among neural methods, the original "Gatys" method is still competitive, but struggles to reproduce large scales on these high-resolution textures. This is due to the size of the receptive fields, which is clearly not sufficient in this case. The method from "Ulyanov" is worse in this respect. The method "Deep Corr" improves the preservation of large-scale structures, but results are not satisfying, some structures are lacking, and artifacts are visible. In contrast, the plain use of the auto-correlation term as an additional constraint, as we propose in "Autocorr," yields better results, even though no use of the Gram matrices is made. The regularization and innovation terms present in the method from [33] may also be harmful in these cases. Next, we observe that adding the Fourier spectrum constraint alone (at a single scale) yields interesting results, but is not enough to get fully satisfying results. The multi-scale methods, be it "Snelgrove" or the one we propose, "Gram+MSInit", "Gram+Spectrum+MSInit", and "Autocorr+MSInit", all improve the original synthesis

[3] The rescaled VGG-19 network can be found at http://github.com/leongatys/DeepTextures.

[4] Due to the numerical sensitivity of the LBFGS-B optimization algorithm.

[5] The code of [33] can be found on Github : https://github.com/omrysendik/DCor.

[6] The code of [36] https://github.com/wxs/subjective-functions.

[7] https://github.com/JorgeGtz/TextureNets_implementation.

[8] We used 2000 iterations for a fair comparison between methods but excellent results are obtained with 500 iterations.

[9] The gradient computation exceeds the GPU memory size which leads to an important overhead.

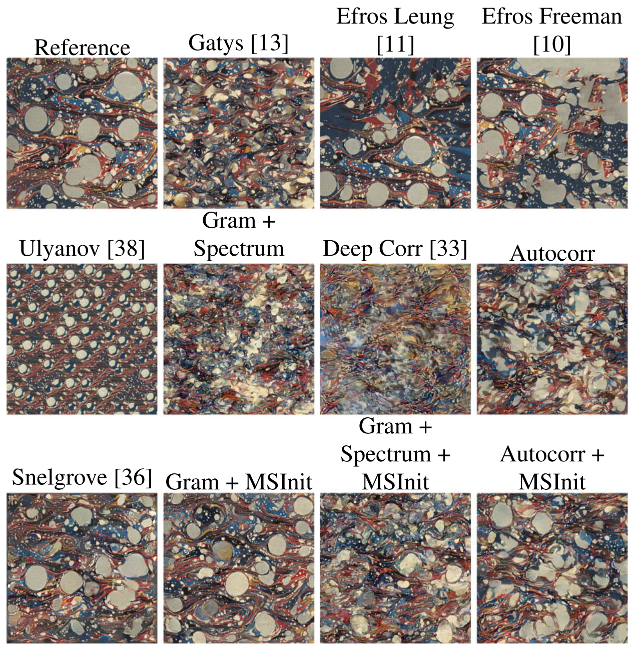


Fig. 2 Synthesis results using different methods for a given reference of size 1048 × 1048

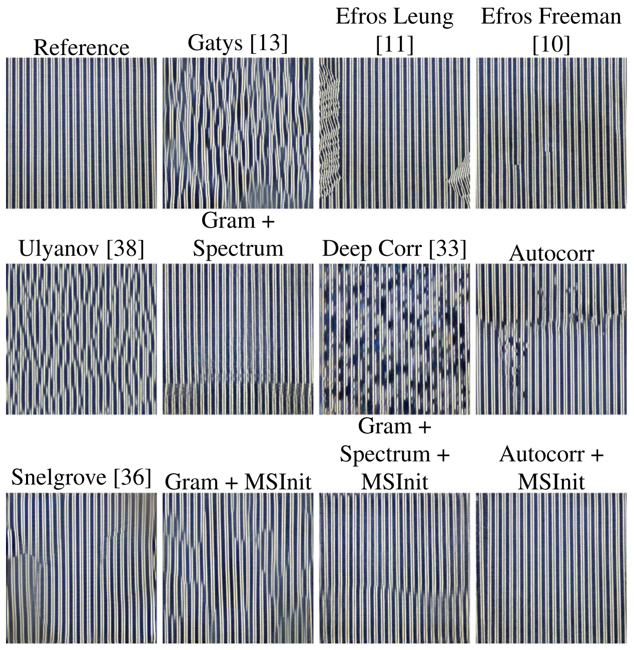


Fig. 3 Synthesis results using different methods for a given reference of size 1048 × 1048

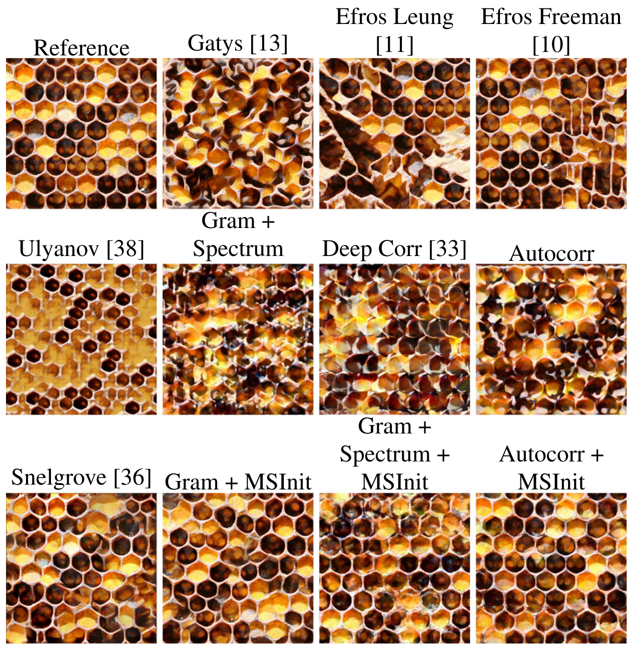


Fig. 4 Synthesis results using different methods for a given reference of size 1048 × 1048

method “Gatys”. In the case of very regular textures, as in Figs. 3, 4 and 5, our multi-scale method “XXX+MSInit” yields better results, as will be confirmed by the user study in Sect. 4.3.3. The method “Autocorr+MSInit” sometimes yields results that are clearly better than others, especially for very structured textures, as can be seen in Fig. 4 or on the last line of Fig. 6. Nevertheless, it also sometimes fails as in Fig. 2 and may produce artifacts on some examples. For this reason and for human resources constraints, we choose, among our methods, to only include “Gram+MSInit” and “Gram+Spectrum+MSInit” in the user study presented in Sect. 4.3.3.

4.3.1 Verbatim Copy

Texture synthesis methods should have the capacity to produce new images that are as diverse as possible. In the pioneering work FRAME [43], this is achieved by maximizing the entropy. Similar ideas have recently been explored in [6,26]. Following these ideas, texture synthesis methods could be evaluated based on their capacity to maximize the entropy under some given constraints. Such a quantitative evaluation, however, is far from being trivial and probably not tractable. In this section, we take a pragmatic and much more modest way. We propose a simple way to evaluate the tendency of methods to locally produce verbatim copy of the input. This is a known default of patch-based methods, see, e.g., [1,30].

For each pixel of a given synthesis result, we look for its nearest neighbor in the input image. The notion of proximity is defined by comparing small square neighborhoods (patches) around each pixel. In (Fig. 7), we display the corresponding displacement map. The used color scale is obtained by assigning the x coordinate of the displacement map to red and the y coordinate to blue. Verbatim copy of the input appears as constant regions in these displacement maps. As

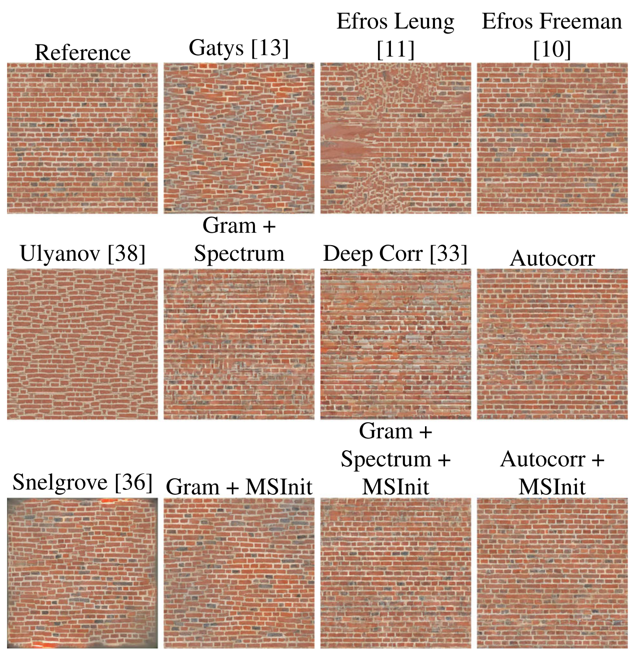


Fig. 5 Synthesis results using different methods for a given reference of size 1048 × 1048

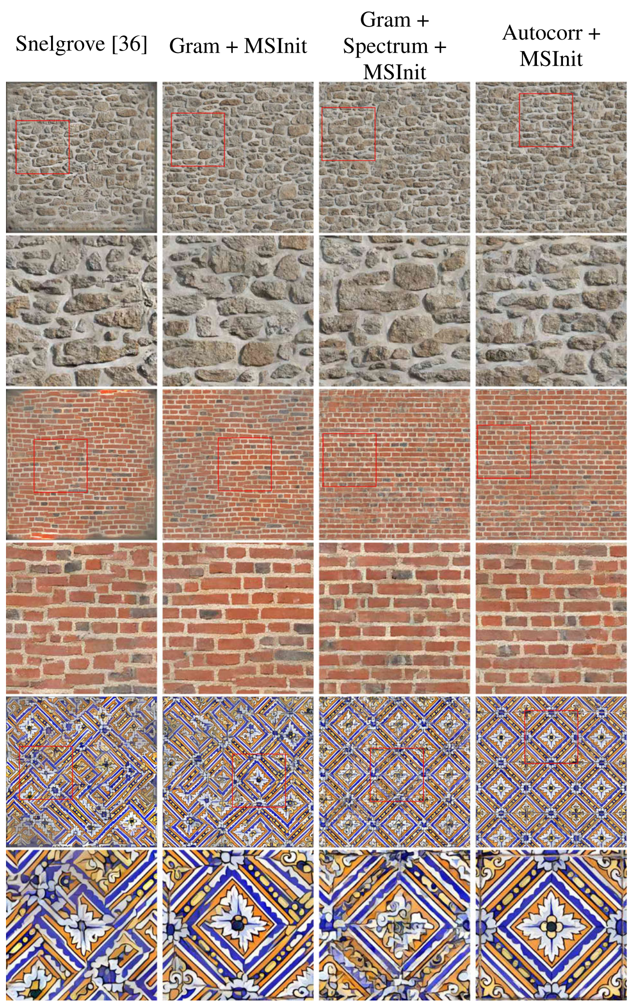


Fig. 6 Zoom-in of some of the texture synthesis results. For the MSInit cases, we use $K = 2$. The region of each image framed by a red square is shown in the row below (Color figure online)

expected, the only two methods displaying large such regions are patch-based methods. All others seem to produce a reasonable amount of innovation, even though the multi-scale method from [36] can very occasionally produce small verbatim copies, probably due to the strong constraints it puts on the Gaussian pyramid.

In order to quantify the visual effect of the displacement maps, we propose to measure the flat regions corresponding to locally constant displacements. For each pixel of the displacement map, we count how many of its neighbors (in 4-connexity) share its color value. Denoting n this number, a score is defined as $DS = 1 - n/N$, where N is the total number of investigated neighbors. The more verbatim copies there are in the synthesis, the closest the score is to 0.

The boxplots of this score for the different methods and the twenty reference images (Fig. 8) can be seen in Fig. 9. They confirm the impression given by the displacement map that the patch-based methods yield significantly more verbatim copy than neural methods. It should be noted, however, that the proposed methodology is relatively rough and does not account either for small perturbations on the pixel positions nor for noisy pixel values.

4.3.2 Feature-Based Evaluation

Feature-based evaluation of textures is not straightforward, because no existing feature is considered as the reference one. Moreover, such evaluations are inherently biased. In the most extreme case, one could even try to optimize the chosen features to synthesize new textures. In this work, we choose to rely on wavelet filters that both are classical texture features and are not used in any of the considered methods. More precisely, we rely on the texture features proposed in [7]. In this paper, two textures are compared by computing the Kullback–Leibler divergence between parametric estimation (using generalized Gaussians) of the marginal distributions of wavelet coefficients.

In order to quantify the proximity of a synthesized texture to the reference image, we propose to :

1. Compute the wavelets coefficients of the reference image and the synthesized one (in our case we choose a Daubechies 4 wavelets as in [7] with 8 scales instead of 3, in order to account for large-scale structures).
2. For each scale and orientation, estimate the parameters of a generalized Gaussian from the empirical distribution of wavelets coefficients

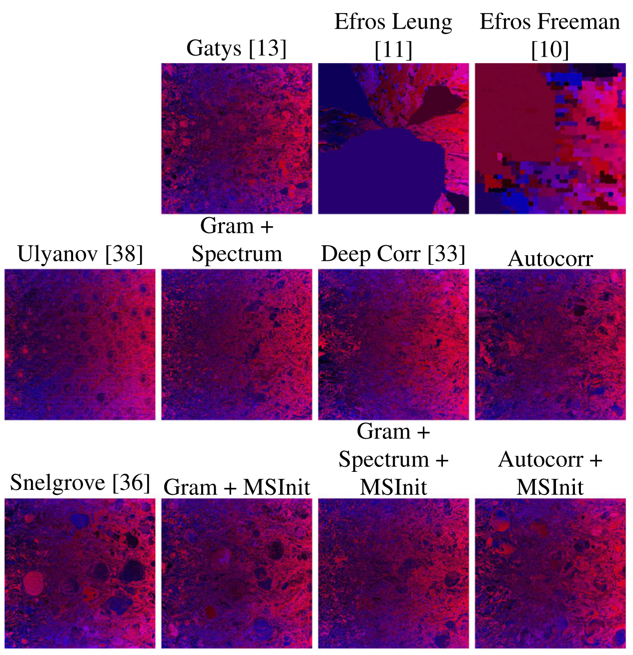


Fig. 7 Displacement map for results using different methods, for a given reference image. An area with constant color indicates a verbatim copy of the input. The synthesis can be found in 2

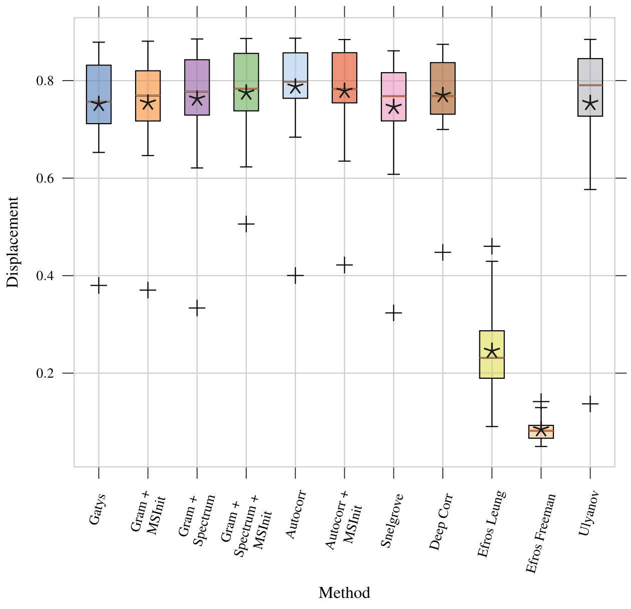


Fig. 9 Boxplots of the displacement score for the different methods on the twenty reference images of size 1024 × 1024

3. For each scale and orientations, compute the Kullback–Leibler divergence between the estimated generalized Gaussians (using a closed-form formula)

We display in Fig. 10 the boxplots of the log KL scores over the 20 considered images, for the different methods. For each box, the horizontal orange line corresponds to the average result and the star to the median. On the average, the best method for this evaluation scheme appears to be "Gram+MSInit." Then follow the two patch-based methods. This is in agreement with results from the previous paragraph, since indeed a verbatim copy will have a perfect score. The next method is "Gram+Spectrum+MSInit," followed by "Snelgrove" and "Autocorr+MSInit." This evaluation confirms the good quality of results produced by the proposed "XXX+MSInit" methods, as well as "Snelgrove," at least on

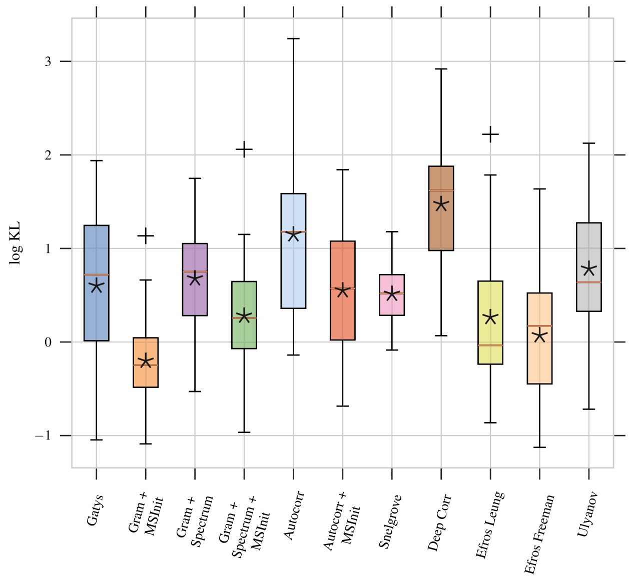


Fig. 10 Boxplots of the displacement score for the different methods on the twenty reference images of size 1024 × 1024

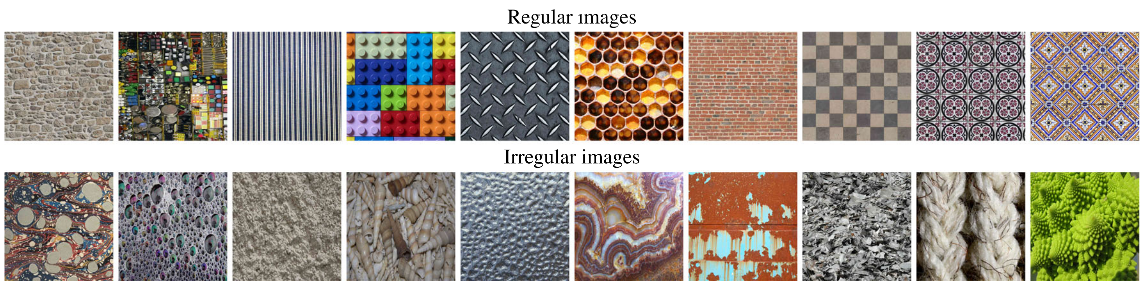


Fig. 8 Reference images used in the different evaluation methods

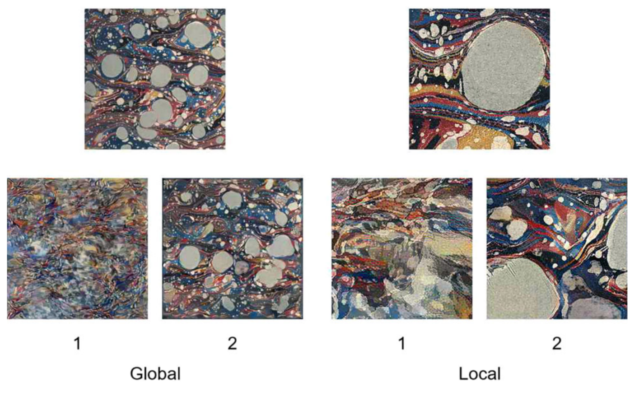


Fig. 11 Example of the layout for one question

this image dataset containing a relatively high proportion of structured textures.

4.3.3 Perceptual Evaluation of Texture Synthesis Methods

Next, we further evaluate the proposed methods by performing a medium scale user study. Indeed, as shown in [9], feature-based methods such as the one of the previous section may not correlate very well with human observations, especially for long-range structures.

For ethical reasons, we decided not to rely on micro-work platforms. Most users involved are volunteer PhD students or researchers, which certainly induces some bias. The total number of persons involved was 93, each having the possibility to answer up to 40 questions.

Methodology Each question aims at comparing two methods on a given texture. In order to evaluate results at different scales, both the complete synthesis and a detail are presented to the user, see Fig. 11. The evaluation is performed on the twenty 1024×1024 images considered in this paper. In order to get further insight on the methods, we have split the textures in two groups : regular and irregular, see Fig. 8.

Following the results of the previous sections, we chose to include in the study the five following methods : “Gatys” [13], “Gram+MSInit,” “Gram+Spectrum+MSInit,” Snelgrove [36] and “Deep Coor” [33]. The first four correspond to the best feature-based score and visual impression. The last one appears to us as the most directly related to the present work in the literature, since it explicitly aims at preserving large-scale structures through additional statistical constraints.

For each couple of methods (out of five) and each reference image, four images are presented to the user, corresponding to the two methods at two different scales (global and local). For each scale, the user has to vote for the best method (the “most similar” to the reference). We obtained 3170 answers at each scale.

More details about this methodology can be found in Annex 1.1.

In order to quantify the results of this study, we rely on the Bradley–Terry model, as used in other perceptual study, see [40]. This model gives us a performance score $\beta_i \in \mathbb{R}$ per method, computed with the votes from the users. The outcome of a duel between methods i and j is determined by $\beta_i - \beta_j$. More details about this statistical model can be found in Annex 1.2.

Duel Results First, we can consider all the duels between all pairs of methods and all reference images, either from the complete set (20 images) or from the subsets of regular and irregular images separately. Results can be averaged for the global and local scale or treated separately. The results can be found in Figs.12, 13 and 14.

Overall, the two best methods for this evaluation appear to be “Gram+MSInit” and “Gram+Spectrum+MSInit.”

For the global scale, there is a draw for the complete dataset and for the irregular images, while “Gram+Spectrum +MSInit” wins on the regular images. For the local scale, “Gram+MInit” always win.

From this, we may deduce that the spectrum constraint may be useful for preserving large-scale structure of regular textures, possibly at the price of a slight degradation at a more local scale. For more irregular textures, method “Gram+MSInit” should be preferred. When we consider all images and both scales (Fig. 12), we can extract a full ranking : “Gram+MSInit” > “Gram+Spectrum+MSInit” > Snelgrove > Gatys > Deep Cor.

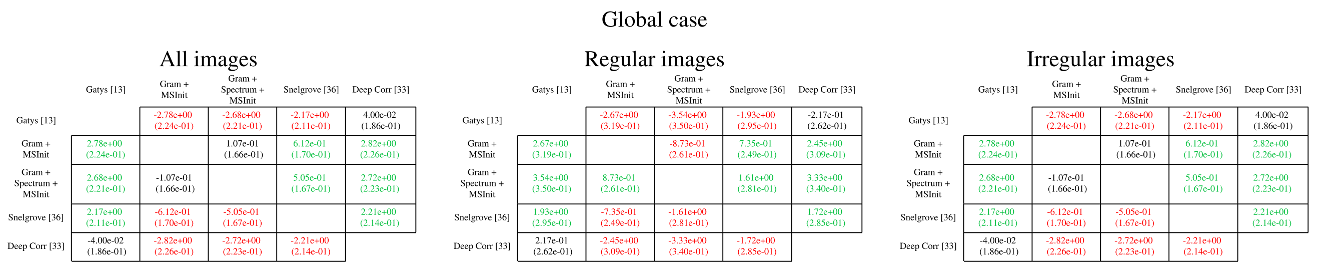
Winning Probability An alternative evaluation consists in calculating the probability that a method i is chosen among all candidates. This “winning probability” W_i is defined in Annex 1.3. These winning probabilities are displayed in Figs. 15, 16, and 17 and confirm the dual results (Figs. 12, 13, 14).

4.4 Influence of Parameters

In this section, we display experiments illustrating the effects of two parameters of the proposed method : K, the number of considered scales, and β, the weighting of the spectrum term when using the method “Gram+Spectrum+MSInit.”

4.4.1 Multi-scale Strategy

In Fig. 18, we display synthesis results with K ranging from 0 (original method from [13]) to 4. The quality of results increases up to $K = 2$. This confirms the fact that the size of the filters in the VGG19 network is too small to describe large scales. It also illustrates the fact that the VGG filters are versatile and provide good features at different scales, since the network has been trained on 224×224 input images. An interesting experiment in this respect would be to synthesize textures using the scale-invariant features from [41].



Global case

All images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-2.78e+00 (2.24e-01)	-2.68e+00 (2.21e-01)	-2.17e+00 (2.11e-01)	4.00e-02 (1.86e-01)
Gram + MSInit	2.78e+00 (2.24e-01)		1.07e-01 (1.66e-01)	6.12e-01 (1.70e-01)	2.82e+00 (2.26e-01)
Gram + Spectrum + MSInit	2.68e+00 (2.21e-01)	-1.07e-01 (1.66e-01)		5.05e-01 (1.67e-01)	2.72e+00 (2.23e-01)
Snelgrove [36]	2.17e+00 (2.11e-01)	-6.12e-01 (1.70e-01)	-5.05e-01 (1.67e-01)		2.21e+00 (2.14e-01)
Deep Corr [33]	-4.00e-02 (1.86e-01)	-2.82e+00 (2.26e-01)	-2.72e+00 (2.23e-01)	-2.21e+00 (2.14e-01)	

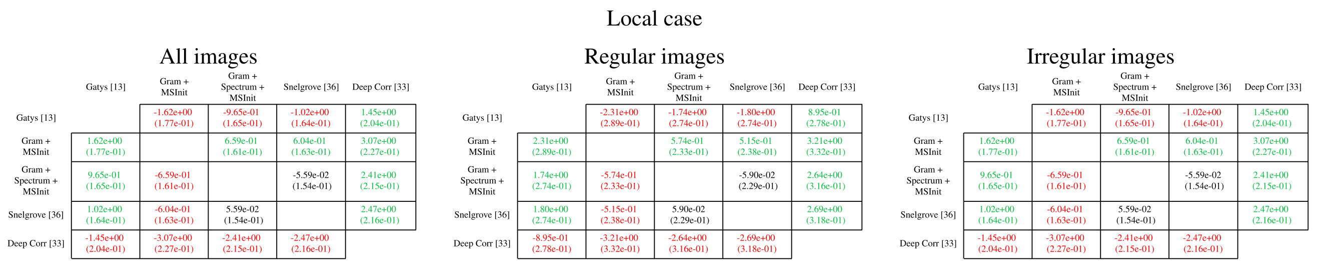
Regular images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-2.67e+00 (3.19e-01)	-3.54e+00 (3.50e-01)	-1.93e+00 (2.95e-01)	-2.17e-01 (2.62e-01)
Gram + MSInit	2.67e+00 (3.19e-01)		-8.73e-01 (2.61e-01)	7.35e-01 (2.49e-01)	2.45e+00 (3.09e-01)
Gram + Spectrum + MSInit	3.54e+00 (3.50e-01)	8.73e-01 (2.61e-01)		1.61e+00 (2.81e-01)	3.33e+00 (3.40e-01)
Snelgrove [36]	1.93e+00 (2.95e-01)	-7.35e-01 (2.49e-01)	-1.61e+00 (2.81e-01)		1.72e+00 (2.85e-01)
Deep Corr [33]	2.17e-01 (2.62e-01)	-2.45e+00 (3.09e-01)	-3.33e+00 (3.40e-01)	-1.72e+00 (2.85e-01)	

Irregular images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-2.78e+00 (2.24e-01)	-2.68e+00 (2.21e-01)	-2.17e+00 (2.11e-01)	4.00e-02 (1.86e-01)
Gram + MSInit	2.78e+00 (2.24e-01)		1.07e-01 (1.66e-01)	6.12e-01 (1.70e-01)	2.82e+00 (2.26e-01)
Gram + Spectrum + MSInit	2.68e+00 (2.21e-01)	-1.07e-01 (1.66e-01)		5.05e-01 (1.67e-01)	2.72e+00 (2.23e-01)
Snelgrove [36]	2.17e+00 (2.11e-01)	-6.12e-01 (1.70e-01)	-5.05e-01 (1.67e-01)		2.21e+00 (2.14e-01)
Deep Corr [33]	-4.00e-02 (1.86e-01)	-2.82e+00 (2.26e-01)	-2.72e+00 (2.23e-01)	-2.21e+00 (2.14e-01)	

Fig. 12 Difference between the methods strengths ($\beta_i - \beta_j$) (Eq. (1.1)) Index i corresponds to rows and index j to columns. When $|\beta_i - \beta_j| > 1.96\hat{\text{se}}_{ij}$ the method i is considered as beating the method j and the cell is displayed in green. In the opposite case, the cell is red. When the cell is white, the difference is not significant (Color figure online)



Local case

All images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-1.62e+00 (1.77e-01)	-9.65e-01 (1.65e-01)	-1.02e+00 (1.64e-01)	1.45e+00 (2.04e-01)
Gram + MSInit	1.62e+00 (1.77e-01)		6.59e-01 (1.61e-01)	6.04e-01 (1.63e-01)	3.07e+00 (2.27e-01)
Gram + Spectrum + MSInit	9.65e-01 (1.65e-01)	-6.59e-01 (1.61e-01)		-5.59e-02 (1.54e-01)	2.41e+00 (2.15e-01)
Snelgrove [36]	1.02e+00 (1.64e-01)	-6.04e-01 (1.63e-01)	5.59e-02 (1.54e-01)		2.47e+00 (2.16e-01)
Deep Corr [33]	-1.45e+00 (2.04e-01)	-3.07e+00 (2.27e-01)	-2.41e+00 (2.15e-01)	-2.47e+00 (2.16e-01)	

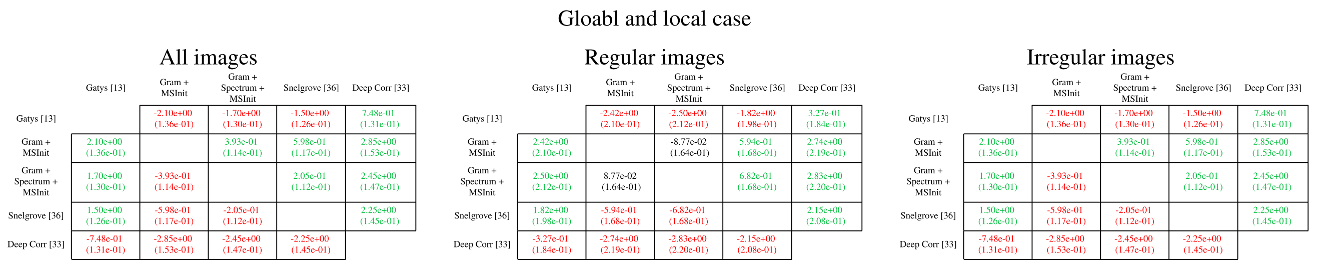
Regular images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-2.31e+00 (2.89e-01)	-1.74e+00 (2.74e-01)	-1.80e+00 (2.74e-01)	8.95e-01 (2.78e-01)
Gram + MSInit	2.31e+00 (2.89e-01)		5.74e-01 (2.33e-01)	5.15e-01 (2.38e-01)	3.21e+00 (3.32e-01)
Gram + Spectrum + MSInit	1.74e+00 (2.74e-01)	-5.74e-01 (2.33e-01)		-5.90e-02 (2.29e-01)	2.64e+00 (3.16e-01)
Snelgrove [36]	1.80e+00 (2.74e-01)	-5.15e-01 (2.38e-01)	5.90e-02 (2.29e-01)		2.69e+00 (3.18e-01)
Deep Corr [33]	-8.95e-01 (2.78e-01)	-3.21e+00 (3.32e-01)	-2.64e+00 (3.16e-01)	-2.69e+00 (3.18e-01)	

Irregular images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-1.62e+00 (1.77e-01)	-9.65e-01 (1.65e-01)	-1.02e+00 (1.64e-01)	1.45e+00 (2.04e-01)
Gram + MSInit	1.62e+00 (1.77e-01)		6.59e-01 (1.61e-01)	6.04e-01 (1.63e-01)	3.07e+00 (2.27e-01)
Gram + Spectrum + MSInit	9.65e-01 (1.65e-01)	-6.59e-01 (1.61e-01)		-5.59e-02 (1.54e-01)	2.41e+00 (2.15e-01)
Snelgrove [36]	1.02e+00 (1.64e-01)	-6.04e-01 (1.63e-01)	5.59e-02 (1.54e-01)		2.47e+00 (2.16e-01)
Deep Corr [33]	-1.45e+00 (2.04e-01)	-3.07e+00 (2.27e-01)	-2.41e+00 (2.15e-01)	-2.47e+00 (2.16e-01)	

Fig. 13 Difference between the methods strengths ($\beta_i - \beta_j$) (Eq. (1.1)) Index i corresponds to rows and index j to columns. When $|\beta_i - \beta_j| > 1.96\hat{\text{se}}_{ij}$, the method i is considered as beating the method j and the cell is displayed in green. In the opposite case, the cell is red. When the cell is white, the difference is not significant (Color figure online)



Gloabl and local case

All images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-2.10e+00 (1.36e-01)	-1.70e+00 (1.30e-01)	-1.50e+00 (1.26e-01)	7.48e-01 (1.31e-01)
Gram + MSInit	2.10e+00 (1.36e-01)		3.93e-01 (1.14e-01)	5.98e-01 (1.17e-01)	2.85e+00 (1.53e-01)
Gram + Spectrum + MSInit	1.70e+00 (1.30e-01)	-3.93e-01 (1.14e-01)		2.05e-01 (1.12e-01)	2.45e+00 (1.47e-01)
Snelgrove [36]	1.50e+00 (1.26e-01)	-5.98e-01 (1.17e-01)	-2.05e-01 (1.12e-01)		2.25e+00 (1.45e-01)
Deep Corr [33]	-7.48e-01 (1.31e-01)	-2.85e+00 (1.53e-01)	-2.45e+00 (1.47e-01)	-2.25e+00 (1.45e-01)	

Regular images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-2.42e+00 (2.10e-01)	-2.50e+00 (2.12e-01)	-1.82e+00 (1.98e-01)	3.27e-01 (1.84e-01)
Gram + MSInit	2.42e+00 (2.10e-01)		-8.77e-02 (1.64e-01)	5.94e-01 (1.68e-01)	2.74e+00 (2.19e-01)
Gram + Spectrum + MSInit	2.50e+00 (2.12e-01)	8.77e-02 (1.64e-01)		6.82e-01 (1.68e-01)	2.83e+00 (2.20e-01)
Snelgrove [36]	1.82e+00 (1.98e-01)	-5.94e-01 (1.68e-01)	-6.82e-01 (1.68e-01)		2.15e+00 (2.08e-01)
Deep Corr [33]	-3.27e-01 (1.84e-01)	-2.74e+00 (2.19e-01)	-2.83e+00 (2.20e-01)	-2.15e+00 (2.08e-01)	

Irregular images

	Gatys [13]	Gram + MSInit	Gram + Spectrum + MSInit	Snelgrove [36]	Deep Corr [33]
Gatys [13]		-2.10e+00 (1.36e-01)	-1.70e+00 (1.30e-01)	-1.50e+00 (1.26e-01)	7.48e-01 (1.31e-01)
Gram + MSInit	2.10e+00 (1.36e-01)		3.93e-01 (1.14e-01)	5.98e-01 (1.17e-01)	2.85e+00 (1.53e-01)
Gram + Spectrum + MSInit	1.70e+00 (1.30e-01)	-3.93e-01 (1.14e-01)		2.05e-01 (1.12e-01)	2.45e+00 (1.47e-01)
Snelgrove [36]	1.50e+00 (1.26e-01)	-5.98e-01 (1.17e-01)	-2.05e-01 (1.12e-01)		2.25e+00 (1.45e-01)
Deep Corr [33]	-7.48e-01 (1.31e-01)	-2.85e+00 (1.53e-01)	-2.45e+00 (1.47e-01)	-2.25e+00 (1.45e-01)	

Fig. 14 Difference between the methods strengths ($\beta_i - \beta_j$) (Eq. (1.1)) Index i corresponds to rows and index j to columns. When $|\beta_i - \beta_j| > 1.96\hat{\text{se}}_{ij}$, the method i is considered as beating the method j and the cell is displayed in green. In the opposite case, the cell is red. When the cell is white, the difference is not significant (Color figure online)

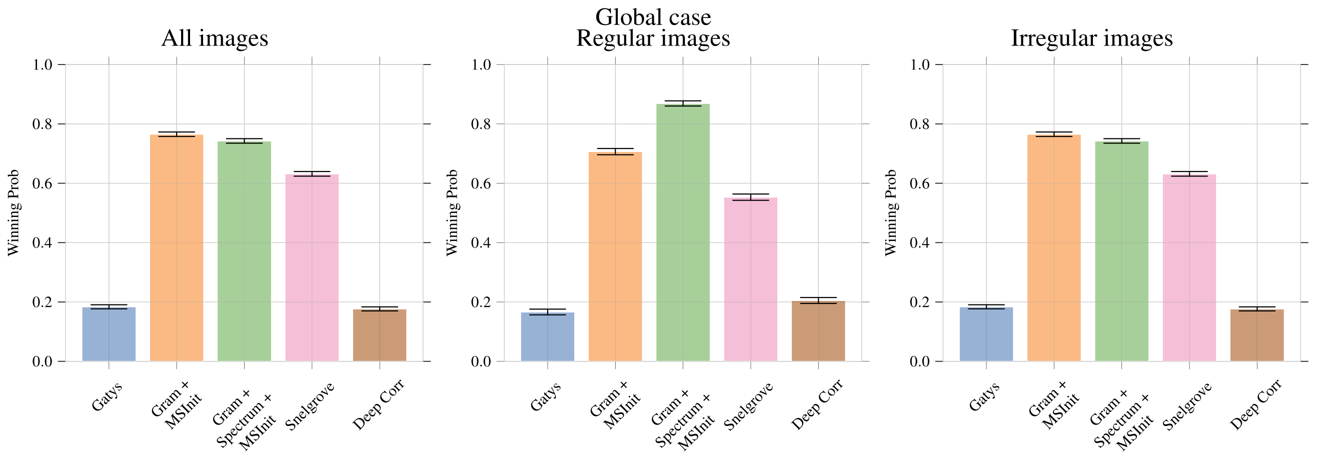


Fig. 15 Winning probabilities W_i with standard error Σ_i for the different methods for the global case

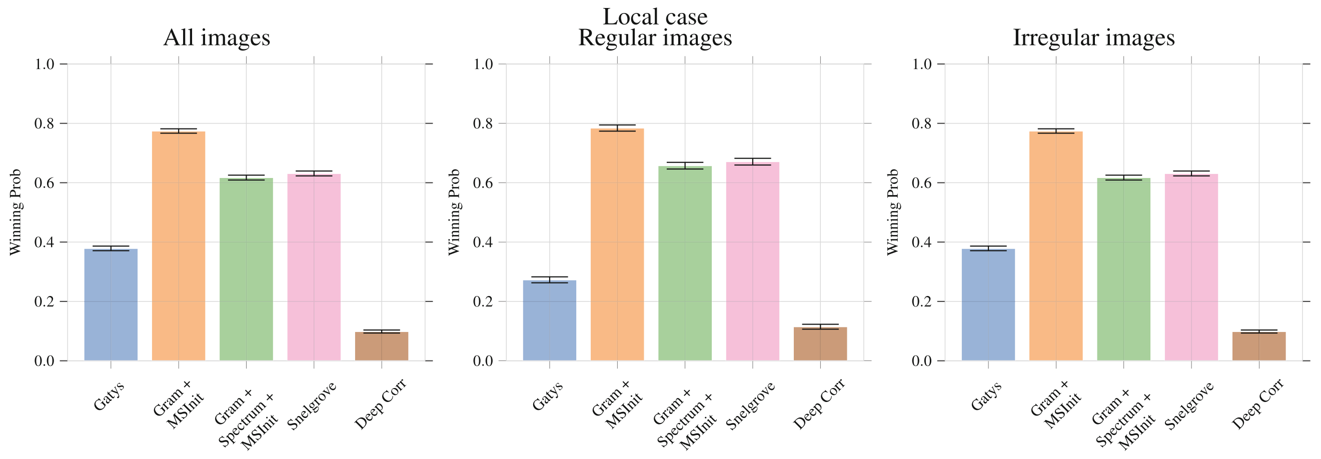


Fig. 16 Winning probabilities W_i with standard error Σ_i for the different methods for the local case

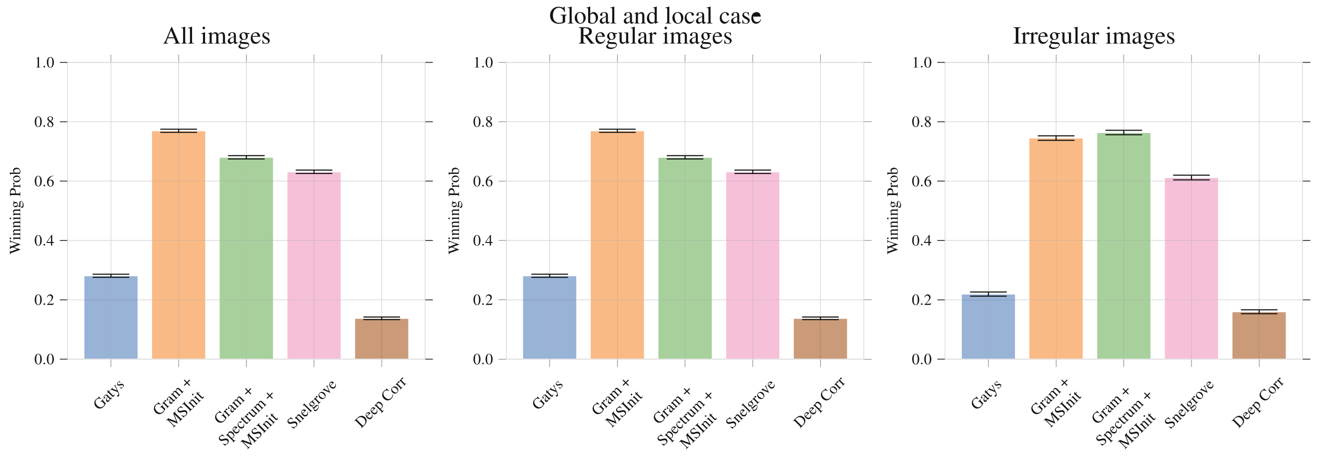


Fig. 17 Winning probabilities W_i with standard error Σ_i for the different methods for both global and local cases

From $K = 3$, the method starts to produce results that are very similar to the reference, the case $K = 4$ being almost a copy of the reference. This may be due to the fact that in these cases, the number of parameters of the synthesis model is up to two orders of magnitude larger than the number of pixels of the coarse image. In other words, the multi-scale strategy reduces too much the solution space for this optimization problem. In practice, $K = 2$ appears a good choice for synthesizing 1024 $\times$ 1024 images.

4.4.2 Weighting of the Spectrum Constraint

In Fig. 19, we display the result of the synthesis for different values of β, the parameter weighting the spectrum constraint, using the method "Gram+Spectrum+MSInit." For the structured textures for which the spectrum term is useful, the best results are obtained for a relatively large β, of the order of 10^5 for the brick image (second column). For more irregular textures, such high values may deteriorate results. This is in agreement with the results from the previous evaluations, where a value $\beta = 10^5$ was used. The problem of automatically setting this parameter is open.

4.5 High-Resolution Synthesis

We conclude this experimental section by showing synthesis results of higher resolution (2048 $\times$ 2048). We consider methods "Gatys", "Gram+MSInit", "Gram+Spectrum+MSInit" (both using $K = 3$). The results can be seen in Figs. 20 and 21. More results can be seen in Supplementary Materials (Sect. 3). Unsurprisingly the interest of the multi-scale schemes is even stronger in this case and the mono-scale method fails. Figure 21 shows the ability of the spectrum constraint to enforce large-scale regularity at this resolution.

5 Conclusion

In this paper, we have shown how a multi-resolution framework and additional statistical constraints related to long-

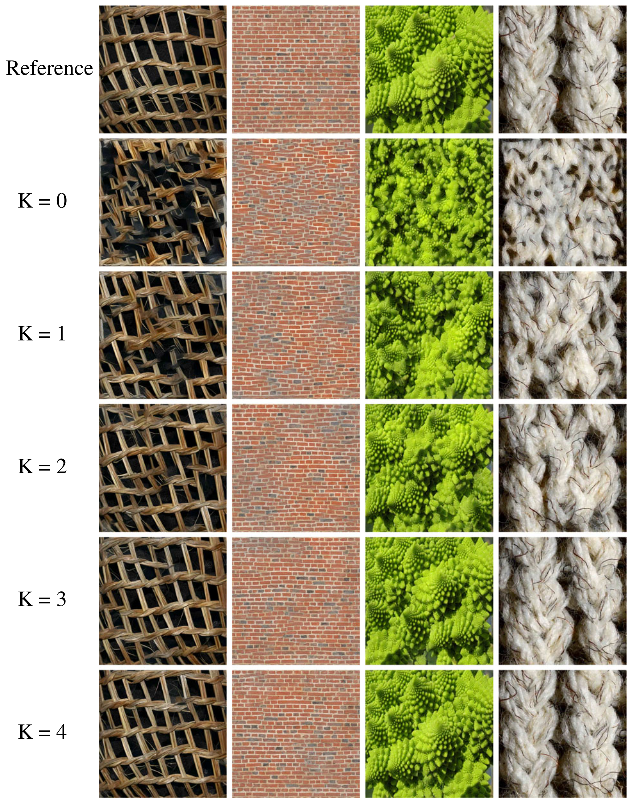


Fig. 18 Synthesis results using different numbers of scales K in the multi-scale strategy. The case $K = 0$ corresponds to the original method from [13]

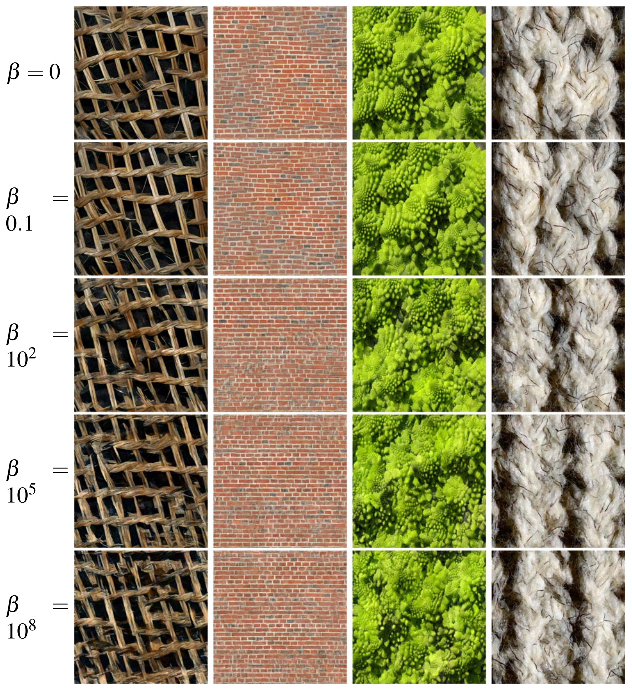


Fig. 19 Synthesis results using different β in Formula (3.4) (original can be seen in Fig. 18), $\beta = 0, 10^{-1}, 10^2, 10^5, 10^8$ with the multi-scale strategy and $K = 2$

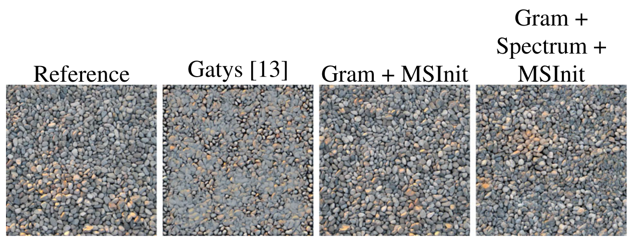


Fig. 20 Synthesis results using different methods for one given reference of size 2048×2048

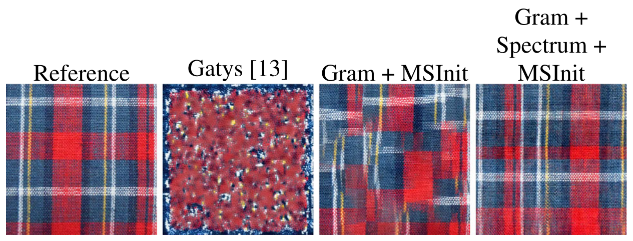


Fig. 21 Synthesis results using different methods for a given reference of size 2048×2048

range dependency enable one to significantly improve texture synthesis results in comparison to the seminal work [13], especially for high-resolution and possibly regular textures. The proposed framework is generic and could be associated with other statistical constraints such as co-occurrence matrices as proposed in [5]. A natural extension would be to investigate the use of such multi-resolution strategies for style transfer for high-resolution images, following [14]. More generally, most generative methods dealing with high-resolution images incorporate more or less explicitly some multi-resolution steps in their synthesis process. This is, for instance, the case for the very efficient StyleGan [22] approach to face synthesis. In this context, it is of great interest to investigate generic procedures to develop multi-resolution frameworks for such generative approaches.

A strong limitation of the neural methods investigated in this work is the unreasonably large number of parameters of the models. In this respect, the next question is not "what set of statistical constraint is sufficient," but "what is the minimal set of statistical constraints" needed to produce realistic synthesis. Some works [6] have shown that second order statistics between features are not necessary to get satisfying results. This, combined with the highly redundant nature of networks such as VGG19, trained for recognition, suggests that much room is available to reduce the number of parameters in these models.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10851-022-01078-y.

Acknowledgements This work is supported by the "IDI 2017" project funded by the IDEX Paris-Saclay, ANR-11-IDEX-0003-02 and by the

ANR project MISTIC, 19-CE40-0005-01. We would like to thank Gang Liu, Kiwon Um and Gui-Song Xia for fruitful discussions. We would like to acknowledge the participation of volunteers in answering the perceptual evaluation survey.

References

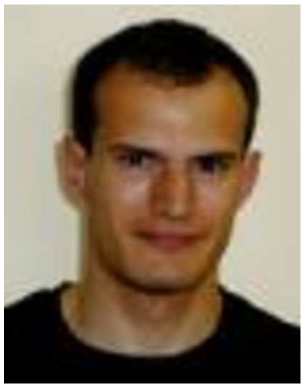
1. Aguerrebere, C., Gousseau, Y., Tartavel, G.: Exemplar-based texture synthesis: the Efros-Leung algorithm. Image Process. **3**, 223–241 (2013). https://doi.org/10.5201/ipol.2013.59
2. Berger, G., Memisevic, R.: Incorporating long-range consistency in CNN-based texture generation (2016). arXiv preprint arXiv:1606.01286
3. Clarke, A.D., Halley, F., Newell, A.J., Griffin, L.D., Chantler, M.J.: Perceptual similarity: a texture challenge. In: BMVC, pp. 1–10 (2011)
4. Cross, G.R., Jain, A.K.: Markov random field texture models. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5, pp. 25–39 (1983). https://doi.org/10.1109/TPAMI.1983.4767341
5. Darzi, A., Lang, I., Taklikar, A., Averbuch-Elor, H., Avidan, S.: Co-occurrence Based Texture Synthesis. (2020). arXiv:2005.08186 [cs]. https://arxiv.org/abs/2005.08186
6. De Bortoli, V., Desolneux, A., Galerne, B., Leclaire, A.: Macrocanonical models for texture synthesis. In: International Conference on Scale Space and Variational Methods in Computer Vision, Springer, pp. 13–24 (2019)
7. Do, M.N., Vetterli, M.: Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. IEEE Trans. Image Process. **11**, 146–158 (2002). https://doi.org/10.1109/83.982822
8. Dong, X., Chantler, M.J.: The importance of long-range interactions to texture similarity. In: International Conference on Computer Analysis of Images and Patterns, Springer, pp. 425–432 (2013)
9. Dong, X., Dong, J., Chantler, M.: Perceptual texture similarity estimation: an evaluation of computational features. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1 (2020). https://doi.org/10.1109/TPAMI.2020.2964533
10. Efros, A. A., Freeman, W. T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM, pp. 341–346 (2001)
11. Efros, A. A., Leung, T. K.: Texture synthesis by non-parametric sampling. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, IEEE, pp. 1033–1038 (1999)
12. Galerne, B., Gousseau, Y., Morel, J.-M.: Micro-texture synthesis by phase randomization. Image Process. **1**, 213–237 (2011). https://doi.org/10.5201/ipol.2011.ggm_rpn
13. Gatys, L., Ecker, A. S., Bethge, M.: Texture Synthesis Using Convolutional Neural Networks, pp. 262–270 (2015)
14. Gatys, L. A., Ecker, A. S., Bethge, M.: A Neural Algorithm of Artistic Style (2015). arXiv:1508.06576 [cs, q-bio]. https://arxiv.org/abs/1508.06576
15. Gatys, L. A., Ecker, A. S., Bethge, M.: Image Style Transfer Using Convolutional Neural Networks, pp. 2414–2423 (2016)
16. Gidas, B.: A renormalization group approach to image processing problems. IEEE Trans. Pattern Anal. Mach. Intell. **11**, 164–180 (1989)
17. He, K., Wang, Y., Hopcroft, J.: A Powerful Generative Model Using Random Weights for the Deep Image Representation, pp. 631–639 (2016)
18. Heeger, D. J., Bergen, J. R.: Pyramid-based texture analysis/synthesis. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, ACM, pp. 229–238 (1995)
19. Jetchev, N., Bergmann, U., Vollgraf, R.: Texture Synthesis with Spatial Generative Adversarial Networks (2016). arXiv:1611.08207 [cs, stat]. https://arxiv.org/abs/1611.08207
20. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual Losses For Real-time Style Transfer and Super-resolution, pp. 694–711 (2016)
21. Julesz, B.: Visual pattern discrimination. IRE Trans. Inf. Theory **8**, 84–92 (1962). https://doi.org/10.1109/TIT.1962.1057698
22. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4401–4410 (2019)
23. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. In: ACM Transactions on Graphics (ToG), vol. 24, ACM, pp. 795–802 (2005)
24. Li, Y., Wang, N., Liu, J., Hou, X.: Demystifying neural style transfer. In: IJCAI (2017). https://arxiv.org/abs/1701.01036
25. Liu, G., Gousseau, Y., Xia, G.-S.: Texture synthesis through convolutional neural networks and spectrum constraints. In: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, pp. 3234–3239 (2016)
26. Lu, Y., Zhu, S.-C., Wu, Y. N.: Learning frame models using CNN filters. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
27. McCormick, B.H., Jayaramamurthy, S.N.: Time series model for texture synthesis. Int. J. Comput. Inf. Sci. **3**, 329–343 (1974)
28. Novak, R., Nikulin, Y.: Improving the Neural Algorithm of Artistic Style (2016). arXiv:1605.04603 [cs]. https://arxiv.org/abs/1605.04603
29. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. Int. J. Comput. Vis. **40**, 49–70 (2000)
30. Raad, L., Galerne, B.: Efros and freeman image quilting algorithm for texture synthesis. Image Process. **7**, 1–22 (2017). https://doi.org/10.5201/ipol.2017.171
31. Risser, E., Wilmot, P., Barnes, C.: Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses (2017). arXiv:1701.08893 [cs]. https://arxiv.org/abs/1701.08893
32. Schreiber, S., Geldenhuys, J., de Villiers, H.: Texture synthesis using convolutional neural networks with long-range consistency and spectral constraints. In: 2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), pp. 1–6 (2016). https://doi.org/10.1109/RoboMech.2016.7813173
33. Sendik, O., Cohen-Or, D.: Deep correlations for texture synthesis. ACM Trans. Gr. (TOG) **36**, 161 (2017)
34. Shi, W., Qiao, Y.: Fast texture synthesis via pseudo optimizer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5498–5507 (2020)
35. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition (2014). arXiv:1409.1556 [cs]. https://arxiv.org/abs/1409.1556
36. Snelgrove, X.: High-resolution multi-scale neural texture synthesis. In: SIGGRAPH Asia, ACM Press, pp. 1–4 (2017). https://doi.org/10.1145/3145749.3149449
37. Tartavel, G., Gousseau, Y., Peyré, G.: Variational texture synthesis with sparsity and spectrum constraints. J. Math. Imaging Vis. **52**, 124–144 (2015). https://doi.org/10.1007/s10851-014-0547-7
38. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V. S.: Texture networks: feed-forward synthesis of textures and stylized images. In: ICML, vol. 1, p. 4 (2016)
39. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: maximizing quality and diversity in feed-forward styliza-

tion and texture synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6924–6932 (2017)

40. Um, K., Hu, X., Wang, B., Thuerey, N.: Spot the difference: accuracy of numerical simulations via the human visual system. J. Comput. Phys. (2019). https://arxiv.org/abs/1907.04179
41. van Noord, N., Postma, E.: Learning scale-variant and scale-invariant features for deep image classification. Pattern Recogn. **61**, 583–592 (2017). https://doi.org/10.1016/j.patcog.2016.06.005
42. Yeh, M.-C., Tang, S.: Improved Style Transfer by Respecting Inter-layer Correlations (2018). arXiv:1801.01933 [cs]. https://arxiv.org/abs/1801.01933
43. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw. **23**, 550–560 (1997). https://doi.org/10.1145/279232.279236
44. Zhu, S.C., Wu, Y., Mumford, D.: Filters, random fields and maximum entropy (FRAME): towards a unified theory for texture modeling. Int. J. Comput. Vis. **27**, 107–126 (1998)

Nicolas Gonthier received a Data Science M.Eng. degree from ISAE -Supaéro and an M.Sc. degree in statistics from the University of Toulouse, both in 2017 and the Ph.D. degree in image processing from the University Paris-Saclay in 2021. His PhD was funded by an interdisciplinary grant (IDI IDEX) from the University Paris-Saclay and hosted by Télécom Paris, Institut polytechnique de Paris. Currently he is a postdoctoral researcher at ENPC ParisTech. His research interests include deep learning, image processing and machine learning for cultural heritage (historical documents, artworks etc.).

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 Paris. His research interests include the mathematical modeling of natural images and textures, stochastic geometry, computational photography, computer vision, image and video processing.



Saïd Ladjal is a former student of École normale supérieure, France. He received a master degree from École Polytechnique and the Ph.D. degree from École normale supérieure de Cachan in 2005 in mathematics. He is currently an associate Professor at Télécom Paris. His research interests include image quality, computational photography and image understanding.