Annals of Mathematical Sciences and Applications
Volume 3, Number 1, 89–148, 2018

A survey of exemplar-based texture synthesis

Lara Raad, Axel Davy,
Agnès Desolneux, and Jean-Michel Morel

Exemplar-based texture synthesis is the process of generating, from an input sample, new texture images of arbitrary size and which are perceptually equivalent to the sample. The two main approaches are statistics-based methods and patch re-arrangement methods. In the first class, a texture is characterized by a statistical signature; then, a random sampling conditioned to this signature produces genuinely different texture images. The second class boils down to a clever "copy-paste" procedure, which stitches together large regions of the sample. Hybrid methods try to combine ideas from both approaches to avoid their hurdles. The recent approaches using convolutional neural networks fit to this classification, some being statistical and others performing patch re-arrangement in the feature space. They produce impressive synthesis on various kinds of textures. Nevertheless, we found that most real textures are organized at multiple scales, with global structures revealed at coarse scales and highly varying details at finer ones. Thus, when confronted with large natural images of textures the results of state-of-the-art methods degrade rapidly, and the problem of modeling them remains wide open.

Keywords and phrases: Texture synthesis, exemplar-based, image statistics, images patches, Julesz conjecture, convolutional neural networks.

1. Introduction

This paper proposes a review of *exemplar-based texture theory*, a topic that occupied David Mumford at the end of the last century [97, 98], and again in his book on pattern theory [62]. Textures are ubiquitous in our visual environment. In the past fifty years their definition has occupied psychophysicists, mathematicians and computer scientists who have built increasingly sophisticated models. The main progress on the elusive topic of defining

arXiv: 1707.07184

textures has come from computer graphics with the problem of reproducing other examples of the *same* texture given a sample. There is so far no complete mathematical theory that would, first, give a formal axiomatic of texture, and then prove that some texture synthesis algorithm matches this definition. Rather, each exemplar-based texture method formulates its own definition of texture and sometimes (but rarely) convergence or consistency proofs. The method to work on texture modeling still relies on a visual exploration of synthesized textures, their defects and successes being linked to some improvement or shortcoming of the mathematical model. All the more, texture modeling remains a valid challenge for mathematicians, as textures represent arguably the vaster and most common class of observable functions. They indeed cover a majority of the area of most digital images. This article accounts for the very rapid and impressive recent apparition of new texture synthesis methods with striking results. We shall retrace their theoretical roots. By performing objective experiments and not hiding the failures of each method, this paper will uncover some flaws in the current definition of *exemplar-based texture modeling*. This will lead us to propose a slightly different definition of the problem that seems to address better its challenges.

The Oxford Dictionary of English defines texture as the feel, appearance, or consistency of a surface or a substance. Focusing on visual appearance, texture is analog to color, a perceived quality of a surface, where the RGB bands are replaced by the output of a specific bank of filters [62, p.215]. Julesz defined textures as classes of pictures that cannot be discriminated in preattentive vision and advanced two statistical hypotheses to characterize them [41, 38, 40]. Grenander proposed to use the term "texture" for strictly stationary stochastic processes [31, p.398]. Giving a precise definition of textures is a slippery task; in a sense, each model implicitly proposes one and as we will see the jury is still out.

Exemplar-based texture synthesis is the process of generating, from an input texture sample, new texture images of arbitrary size and which are perceptually equivalent to the input. It is common to classify them under the two classical statistical estimation categories: *parametric methods* and *non-parametric methods*. The parametric methods aim at characterizing a given texture sample by estimating a set of statistics which will define an underlying stochastic process. The new images will then be samples of this stochastic process, i.e. they will have the same statistics as the input sample. The question here is: what would be the appropriate set of statistics to yield a correct synthesis for the wide variety of texture images? The results of these methods are satisfying but only on a small group of textures, and often fail when important structures are visible in the input. The non-parametric methods reorganize local neighbourhoods from the input sample in a consistent way to create new texture images. These methods return impressive visual results. Nevertheless, they often yield verbatim copies of large parts of the input sample. Furthermore, they can diverge, starting to reproduce iteratively one part of the input sample and neglecting the rest of it, thus growing what experts call "garbage". Because "non-parametric" methods are not completely parameter-free, and "parametric" methods can have a reduced set of parameters, in this paper we will denote by *patch re-arrangement methods* the former and by *statistics-based methods* the latter.

What constitutes a texture? The answer depends of course on human perception. But a mathematical formulation can be used to characterize patterns that are perceived as textures. The statistical characterization of texture images was initiated by Béla Julesz [38, 42]. Julesz was the first to point out that texture images could be reliably organized according to their N-th order statistics into groups of textures that are preattentively indistinguishable by humans [38]. (Focusing on pre-attentive vision helps to reduce the subjective impact of high level processing.) Julesz [42] demonstrated that many texture pairs sharing the same second-order statistics would not be discerned by human preattentive vision. This hypothesis constitutes the first Julesz axiom for texture perception. One consequence of this axiom is that two textures sharing the same Fourier modulus but with different phase should be perceptually equivalent. Indeed, the square Fourier modulus of an image corresponds to its spatial auto-correlation, thus the second-order statistics. This motivates a class of algorithms (the random phase methods) aiming at creating textures with a given second-order statistic. An example of such algorithms is [87]. In a more recent extension [23], a texture is generated by randomizing the Fourier phase while maintaining the Fourier modulus. The Random Phase Noise method in [23] correctly synthesizes textures with no salient details, namely microtexture, which adapt well to a Gaussian distribution, but it fails for more structured ones, macrotextures, as can be experimented in the executable paper [22]. Indeed, textures may share the same second and even third order statistics while being visually different [43, 11]. This led Julesz [40, 39] to propose a second theory to explain texture preattentive discrimination by introducing the notion of *textons*. Textons are local conspicuous features like bars or corners. Giving a mathematical definition for textons is far from trivial and was studied in for example [95, 17]. Julesz' second theory states that only the first order statistics of these textons are relevant for texture perception: images having the same texton densities (thus, just a first order statistic) could not be discriminated. Texton theory proposes the main axiom that texture perception is invariant to random shifts of the textons [40]. This axiom is extensively used in the stochastic dead leaves models [61, 77, 8].

Several models of the early visual processing in mammals are based on a multiscale analysis with Gabor kernels, and are used in particular for modeling the perception of texture [5, 82, 59]. Wavelet analysis provided a natural frame for these models and resulted in effective methods for texture classification and segmentation [12, 48, 84, 60]. Heeger and Bergen [33] extended Julesz' approach to multiscale statistics. They characterized a texture sample by the histograms of its wavelet coefficients. By enforcing the same histograms on a white noise image they obtained a new multiscale exemplar-based texture synthesis method. Yet this method only measures marginal statistics. It misses important correlations between pixels across scales and orientations which are crucial to generate edges and conspicuous patterns. We refer to the on-line execution of this method [9] where some successes but many failures are evident, as is also the case for RPN [22]. Within a similar range of results, the De Bonet [14] method randomizes the initial texture image and preserves only a few statistics, namely the dependencies across scales of a multi-resolution filter bench response. Other methods are also based on statistics of wavelet coefficients or more involved multiscale image representations [69, 67, 72]. The Heeger-Bergen method was extended by Portilla and Simoncelli [69] who proposed to evaluate on the sample some 700 cross-correlations, autocorrelations and statistical moments of the wavelet coefficients. Enforcing the same statistics on synthetic images, starting from white noise, achieves striking results for a wide range of texture examples. This method, which for a decade represented the state-of-the-art for psychophysically and statistically founded algorithms is nevertheless computationally heavy, and its convergence is not guaranteed. Its results, though generally indiscernible from the original samples in a pre-attentive examination, often present blur and phantoms. Earlier, Zhu, Wu and Mumford [98] proposed to model texture images by inferring a probability distribution on a set of images with the same texture appearances and then to sample from it. To infer this probability distribution, the set of images is filtered by a pre-selected set of filters (which capture the important features of a given texture image) and their histograms are extracted. These are estimates of the marginals of the probability distribution sought for. Then the maximum entropy probability distribution is constructed matching the previous marginals. To sample from this probability distribution the

Gibbs sampler is adopted, thus generating new texture images. The resulting model is a Markov random field. The limitation of this method is its practical aspect. Inferring the probability distribution and sampling from it are complex tasks. More recent work by Zhu et al. [96, 92] advanced the *Julesz ensembles* texture model based on a common set of statistics; they proved that this model is equivalent to FRAME in the limit of an infinite image grid. An efficient MCMC sampling method was also proposed. These two texture generators have been recently revisited with neural networks. Gatys' texture generator [27] and DeepFrame [58] can be seen respectively as extended versions of [69] and [98], and get significantly better results. Some new neural network methods, based on generative neural networks, also get notable results [36]. All these recent methods show that the Julesz program of seeking the right statistics to characterize a texture is still well alive.

It is worth mentioning that texture models can be used to complete missing parts of an image or texture inpainting. These methods rely on the definition of texture images as the realization of a random field. For inpainting this boils down to the estimation of a random texture model on the masked input image (a set of valid pixels of the image) from which a new image is sampled conditioned to some of the known values of input image. The method presented in [26, 25, 24] is particularly well adapted for micro-textures. A Gaussian model is estimated on the masked input image; then the result is generated by a conditional sampling from the estimated model using the kriging estimation framework.

Patch re-arrangement methods constitute a totally different category of texture synthesis algorithms. This category started by pixel re-arrangement using square patches as context. The initial Efros and Leung [19] method was inspired by Shannon's Markov random field model for the English language [78]. In analogy with Shannon's algorithm for synthesizing sentences, the texture is constructed pixel by pixel. For each new pixel in the reconstructed image, a patch centered in the pixel is compared to all the patches of the input sample. The patches in the sample that are similar help predict the pixel value in the synthetic image. Several optimizations have been proposed to accelerate this algorithm. Among them Wei and Levoy [90] managed to fix the shape and size of the learning patch, and Ashikhmin [3] proposed to extend existing patches whenever possible instead of searching in the entire sample texture. Yet, as already pointed out in the original paper [19], an iterative procedure may fail by producing "garbage" when the neighborhood's size is too small. On the other hand, it can lead to a trivial verbatim reproduction of big pieces of the sample when the neighborhood is too large. This can be experimented in the online executable paper [1]. Many extensions of [19] have been proposed that manage to accelerate the procedure and reduce the "garbage" problem by stitching entire patches instead of pixels. Among the first methods proposing to re-arrange whole patches, Xu et al. [32] proposed to synthesize a texture by picking random patches from the sample texture and placing them randomly in the output texture image. A blending step is applied across the overlapping blocks to avoid edge artifacts. In [54] the authors proposed to synthesize the image by quilting together patches that were taken from the input image among those who best match the patch under construction. A blending step was also added to overcome some edge artifacts. Efros and Freeman [20] proposed an extension of the latter introducing the quilting method (a blending step) that computes a path with minimal contrast across overlapping patches, thus mitigating the transition effect from patch to patch.

Kwatra et al. [47] extended [20] by using a graph-cut algorithm to define the edges of the patch to quilt in the synthesis image. Another extension of [19] was proposed by Kwatra et al. [46] where to synthesize a texture image they improve the quality of the synthesis image sequentially by minimizing a patch-based energy function. In the same spirit as [46], where texture optimization is performed, the authors in [50] proposed to synthesize textures in a multiscale framework using the coordinate maps of the sample texture at different scales. They introduced spatial randomness by applying a jitter function to the coordinates at each level, combined to a correction step inspired by [3]. One of the key strengths of the method is that it is a parallel synthesis algorithm which makes it extremely fast. These patch-based approaches often present satisfactory visual results. In particular they have the ability to reproduce highly structured textures (macrotextures). However, the risk remains of copying even several times verbatim large parts of the input sample. For practical applications this may result in the appearance of repeated patterns in the synthesized image. Furthermore, a fidelity to the global statistics of the initial sample is not guaranteed, in particular when the texture sample is not stationary. We refer to [89] for an extensive overview of the different patch re-arrangement methods.

Recent research tries to revisit the use of previous methods. Using neural networks has seen some success, as well as combining patch re-arrangement and statistics-based methods to overcome the drawbacks mentioned previously [66, 81]. These approaches will be called *hybrid methods*. Peyré [66] proposed to use a patch-based approach where all the patches of the synthesized image are created from a sparse dictionary learnt on the input sample. Tartavel et al. [81] extended [66] by minimizing an energy that involves a sparse dictionary of patches combined to constraints on the Fourier spectrum of the input sample in a multiscale framework. Raad et al. [70] proposed to model the self-similarities of a given input texture with conditional multivariate Gaussian distributions in the patch space in a multiscale framework. A new image is generated patch by patch, where for each given patch a multivariate Gaussian model is inferred from its nearest neighbours in the patch space of the input sample, and hereafter sampled from this model.

The academic literature shows that current methods are able to produce impressive texture synthesis on various kinds of textures. Our experiments will illustrate this, and the opposite. Indeed, this literature is still working, in a sense, on toy examples. Most textures are defined by texture samples of relatively small size and the structures are present in a small range of scales. When confronting the methods with more challenging data, the quality of the results degrades rapidly. This can be seen for most natural images of textures, which are non-stationary, due for example to the presence of illumination changes and perspective. As a matter of fact large photographs of textures are non-stationary because even homogeneous material always shows an internal variation of structure. Thus the classic exemplar-based texture synthesis problem can be seen in this light as an almost impossible Fourier spectrum extrapolation, given a very small texture example. Hence our exploration not only of the solutions, but of the problem itself will illustrate the limitations of the current question, and introduce a more general question: how to emulate the real, non-stationary textures, for which we dispose of large samples? Then the question is no longer to "extend" a small patch into a larger texture of the same kind, but rather to be able to fabricate other examples of a given large and complex texture, given only one sample of it.

This survey concentrates on the problem of texture synthesis on flat 2D domains. There are several interesting extensions and applications of the basic problem which are not discussed here. These include *surface texture synthesis*, in which a texture is to be placed onto a curved surface, *dynamic texture synthesis*, when the goal is to generate textures whose appearance evolves over time such as for videos of time-variant materials, or *solid texture synthesis*, where the aim is to generate the color content of 3D blocks of synthesized materials from which, for example, computer graphics objects can be carved out. Other related problems include *image completion* and *resolution enhancement* by texture synthesis. Also, the computational cost in real-time applications (e.g. games) or when the data volume is large (e.g. film production) impose further restrictions leading to particular algorithms.

For a discussion of these topics, we refer the reader to Wei et al. [89] and the references therein.

We now sketch our plan. We shall present the main trends in exemplar-based texture synthesis by describing in detail several methods illustrating the three main families. In each case, the strength and limitations will be commented as well as some relevant variations. Section 2 introduces the statistics-based methods which perform statistical optimization and describes several algorithms. Then Section 3 focuses on patch re-arrangement methods, presenting three works. The third main class of hybrid methods is discussed in Section 4. The experimental Section 5 first compares the main families of algorithms in a varied set of textures; then, the limitations of all current methods are revealed with high-resolution and non-stationary examples. Finally, Section 6 concludes the paper. All the results displayed were generated for this paper, with the original code published with the methods [9, 22, 68, 28, 57, 37, 1, 71, 52] or with the modifications mentioned in this paper.

2. Statistics-based methods

Statistics-based texture synthesis methods follow the general approach proposed by Julesz, illustrated in Figure 1. The synthesis is performed in two steps: first, a set of statistics is estimated from the sample texture; second, a random image is generated, subject to these statistical constraints. Methods in this class differ in the set of statistics considered and in the optimization method used to impose them on a random image. We will describe several algorithms of this class with increasing sophistication. It will appear that the number of statistics enforced plays a key role in the success.

2.1. Micro-texture synthesis by phase randomization

The Random Phase Noise (RPN) method synthesizes a new texture from a rectangular sample by simply randomizing the phase of the Fourier coefficients of the input sample. The results are very satisfying for textures that are characterized by their Fourier modulus, a class called *micro-texture* by some authors. This method is also able to create a random texture from any input image, not necessarily a texture sample. It is in spirit quite close to the noise generators from computer graphics [65, 87]. The rest of this section describes the main ideas of this method and we refer the reader to [22] for more details and a catalog of several synthesis examples.

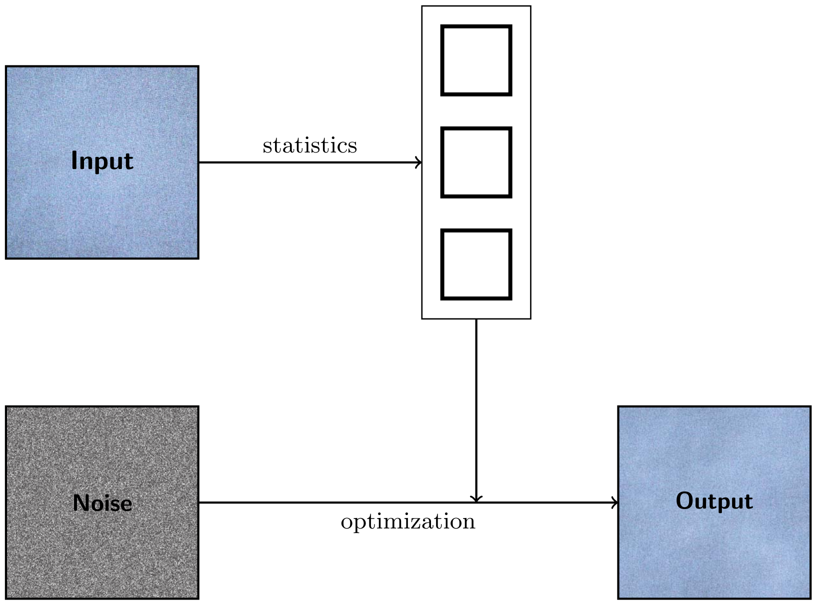

Figure 1: Statistics-based methods. A set of statistics is extracted from an input sample (analysis step). Then, starting with a noise image, an optimization procedure is applied to enforce these statistics on the output image (synthesis step).

The RPN of an image u defined on a domain Ω is obtained by adding a random phase θ to the Fourier phase of the input sample image. The random phase is a white noise image uniformly distributed over $(-\pi, \pi]$ and is constrained to be symmetric. In the case of an RGB color image $u = (u_R, u_G, u_B)$, the RPN image is obtained by adding the same random phase to the Fourier transform of each color channel. Adding the same random phase to the original phases of each color channel preserves the phase displacements between channels. This is important as it permits to create new textures without creating false colors [23].

More precisely, a *uniform random phase* is defined as a random image $\theta \in \mathbb{R}^{M \times N}$ satisfying the following conditions:

- θ is odd: $\forall m, n \in \Omega, \theta(-m, -n) = -\theta(m, n)$;
- $\theta(m, n)$ is uniform on the interval $(-\pi, \pi]$ for $(m, n) \notin \{(0, 0), (M/2, 0), (0, N/2), (M/2, N/2)\}$;
- $\theta(m, n)$ is uniform on the set $\{0, \pi\}$ for $(m, n) \in \{(0, 0), (M/2, 0), (0, N/2), (M/2, N/2)\}$;

- for every subset $\mathcal{S}$ of the Fourier domain which does not contain distinct symmetric points, the family of random variables $\{\theta(m,n)|(m,n)\in \mathcal{S}\}$ is independent.

The RPN of an image $u \in \mathbb{R}^{M\times N}$ is defined as the random image X where there exists a uniform random phase θ such that

$$\hat{X}(\xi,\eta) = \hat{u}(\xi,\eta)e^{i\theta(\xi,\eta)}, (\xi,\eta)\in\Omega, \tag{1}$$

where $\hat{u}$ denotes the Fourier transform of u. An equivalent definition is

$$\hat{X}(\xi,\eta) = |\hat{u}(\xi,\eta)|e^{i\theta(\xi,\eta)}, \tag{2}$$

where θ is a uniform random phase. Given the phase ϕ of a real-valued image and a uniform random phase θ, the random image $(\theta+\phi) \mod 2\pi$ is also a uniform random phase, which proves this equivalence. The first definition (1) leads to a natural extension of RPN to color images [23], while the second definition (2) highlights the fact that the RPN depends only on the Fourier modulus of the sample image u.

Similarly, an Asymptotic Discrete Spot Noise (ADSN) associated with an image u is defined as the convolution of a normalized zero-mean copy of u with a Gaussian white noise. A Gaussian white noise image has a uniform random phase and its Fourier modulus is a white Rayleigh noise; the phase and modulus are independent. Thus, the phase of the ADSN is a uniform random phase whereas its Fourier modulus is the pointwise multiplication of the Fourier modulus of u by a Rayleigh noise [23]. Both ADSN and RPN have uniform random phases, but the modulus distributions are different. RPN keeps the Fourier modulus of the original image, while for ADSN the Fourier modulus is degraded by a pointwise multiplication by a white Rayleigh noise. Regardless of their theoretical differences, ADSN and RPN produce results that are perceptually very similar [23].

The RPN method is the fastest method presented in this review since it basically needs the computation of two FFTs. Nevertheless, this method is limited to micro-textures and it will fail synthesizing structured textures, namely macro-textures. In Figure 2 two synthesis examples are shown. The first synthesis (left example in Figure 2) shows outstanding results. This micro-texture is indeed well represented by its Fourier modulus. However this is not at all the case for the second texture synthesis (right example in Figure 2). Clearly, the knowledge of the modulus of the Fourier coefficients of this texture is not sufficient to recover the strong contrast of the input.

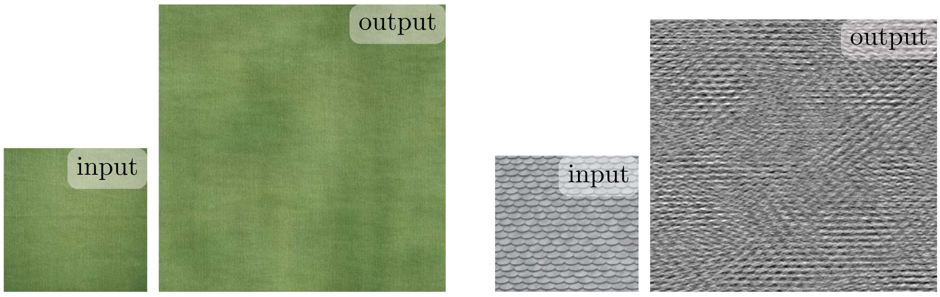

Figure 2: Synthesis results of the RPN method [87, 23]. This method works extremely well for micro-textures including tissues and granular textures with no geometric structures [22]. For more structured texture images it fails. Two examples are shown: a successful synthesis on the left and a failure case on the right.

2.2. The Heeger and Bergen pyramid based texture synthesis

Heeger and Bergen [33] proposed to characterize a texture by the first order statistics of both its color and its responses to multiscale and multi-orientation filters organized in a steerable pyramid [21]. This proposition, motivated by the study of human texture perception, focuses on the synthesis of microtextures defined as images that don't have conspicuous patterns (e.g., granite, bark, sand).

Let us describe the input texture image u and the synthesized texture v using the Heeger and Bergen method. First the image u is filtered using a steerable pyramid decomposition [21, 79] with S scales and Q orientations at each scale. The steerable pyramid is a linear multiscale and multi-orientation image decomposition. Given an input image, it is first filtered to provide a high frequency image and a low frequency image. Band-pass oriented filters are then sequentially applied to the low frequency image which is also down-sampled. These band-pass oriented filters are applied S times to the corresponding low frequency image. This decomposition yields images of different sizes corresponding to the different scales and orientations on which the gray level histograms are extracted as well as the gray level histogram of u. These histograms define the set of statistics that characterize u.

The second step consists in generating the output image v, which is initialized with a noise image. Its pixel values are iteratively modified to match the histograms of u and of its steerable decomposition. These histogram matchings are performed on v alternately in the image domain and in the multiscale transform domain, until all the output histograms match the ones of u. A third parameter is introduced here and it is the number of iterations used to achieve a stabilization of the histogram matching.

To the best of our knowledge, no theorem guarantees that this iteration will end with an image respecting all statistics; there is of course one solution to it, namely the example image. But the goal is to create an image different from the example. Hence the random initialization, which is supposed to lead always to different samples of the same texture. This remark applies to all texture synthesis methods we will consider: their success will mainly be judged visually and experimentally.

To treat RGB color images, instead of applying the method to each color channel of the input image which are highly correlated, the authors proposed to change the color space RGB to a more adapted color space. This new color space is obtained by principal component analysis of the RGB values of the input image u. In [9] a detailed explanation of the original method of Heeger and Bergen [33] is provided with a complete analysis of the steerable pyramid decomposition and the histogram matching step. The authors also provide in [9] a minor improvement in the edge handling of the convolutions as well as an experimental section illustrating the influence of the parameters, namely the number of iterations, the number of scales and the number of orientations. As we said, there is no theoretical proof of convergence of the method but an experimental study shows that the results tend to stabilize after five to ten iterations [9]. Increasing the number of orientations changes the results slightly, but four orientations are enough in general. The number of scales is very important. Taking the highest number permits to take into account all the scales of the texture. When the input texture has no evident structure then this parameter has less influence in the result.

As our experiments here will show, the results yielded by this approach are convincing for some stochastic textures, but the method fails for most complex texture images. In particular it generally fails (visually) for quasi-periodic textures, random mosaic textures, textures having more than one dominant orientation, and textures having correlations of high frequency content over large distances. This demonstrates experimentally that all the spatial information characterizing a texture is not captured by the first order statistics of a set of linear filter outputs. In Figure 3 two synthesis examples are shown: a successful synthesis and a failure case.

2.3. FRAME: a mathematical model for textures

FRAME, which stands for Filters, Random fields And Maximum Entropy, is a mathematical model of textures developed by Zhu, Wu and Mumford

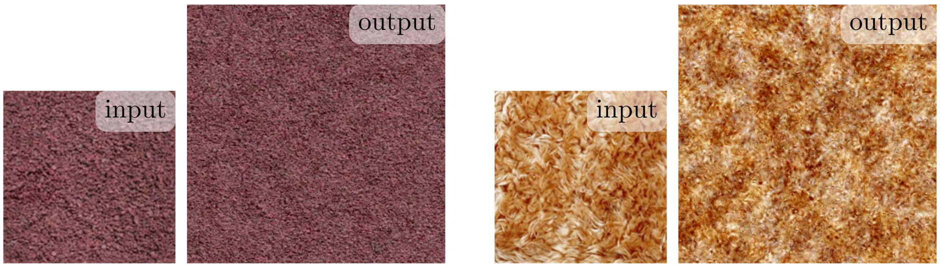

Figure 3: Synthesis results of the Heeger and Bergen method [33]. This method works for microtextures. For more structured texture images it fails. Two examples are shown: a successful synthesis on the left and a failure case on the right.

in [97] and [98]. It is the most mathematical solid work among the stream of work on texture modeling during that period. It puts the graphics method of Heeger and Bergen [33] in a mathematical sound setting, i.e., it has a formal statistical model, and it can match the marginal statistics. It also answers the Julesz quest by pursuing the minimum statistical constraint that are necessary.

The FRAME model is based on the maximum entropy principle. It starts with a set of filters that are selected from a general filter bank to capture features of the texture. These filters are applied to observed texture images, and the histograms of the filtered images are extracted. Then, the maximum entropy principle is employed to derive a distribution f, which has in expectation the same filter responses as the original image, while being of maximum entropy. More precisely, let u be an observed texture image and let F^k, $k = 1, \dots, K$ be a set of filters. Let H_u^k be the (discrete) histograms of the filter responses $F^k * u$, and for any image v, let H_v^k be the histograms of the filter responses $F^k * v$. Zhu, Wu and Mumford seek for a distribution $f(v)$ on images v such that

$$\mathbb{E}_f(H_v^k) = H_u^k, \tag{3}$$

while being of maximum entropy (i.e. while being "as random as possible"). By Lagrange Multipliers (maximization under constraints), the solution has the form

$$f(v;\lambda) = \frac{1}{Z(\lambda)} \exp\left(- \sum_{i=1}^{L} \sum_{k=1}^{K} \lambda_i^k H_u^k(i) \right),$$

where L is the number of bins of the discrete histograms. To find the value of the parameters λ satisfying Equation (3), Zhu, Wu and Mumford use a gradient descent to find the right λ and the Gibbs sampler algorithm to sample random images v from a distribution $f(\cdot;\lambda)$. The distribution $f(\cdot;\lambda)$ defines a Markov Random Field (MRF).

Finally, a stepwise algorithm is proposed to choose the filters from a general filter bank. This "filter pursuit" step is achieved thanks to the minimax entropy principle: find the set of filters such that the associated distribution f is of minimum entropy, since it is equivalent to be of minimal Kullback-Leibler divergence from the "true" underlying distribution. A detailed explanation of this fact can be found in [98].

The FRAME model was later extended, in particular with non linear filters, using the output of some layers of a neural network. It is then called DeepFrame [58], and we will talk again about it in Section 2.5.

2.4. The Portilla and Simoncelli algorithm

The key issue in FRAME and in the method of Heeger and Bergen is to choose the "right" filters and the statistics that will be matched. In [69], Portilla and Simoncelli proposed an important improvement on Heeger and Bergen's method [33]. The texture is again synthesized starting from a noise image and coercing it to have the same statistics as the input image. As we have seen, marginal statistics are not enough to capture the relations across scales and orientations. Portilla and Simoncelli proposed to match a set of joint statistics measurements of the coefficients of the steerable pyramid decomposition of the input texture. The statistics used to characterize the input texture are the autocorrelation and cross-correlation coefficients (inner and intra scales), as well as the statistical moments of order one, two, three and four of the input sample's values. To enforce these statistics on the result, the image under construction is projected iteratively into the subspace of constraints using a gradient projection approach until stabilization. The final output image may not have exactly the same statistics as the input sample. It merely represents a local minimum. Again there is no proof of a convergence of the method anyway.

Portilla and Simoncelli's technique is based on the theories of human visual perception, in particular Julesz' hypothesis stating that two images are perceptually equivalent if and only if they agree on a set of statistic measurements. The goal is to establish the minimal set of measurements in a way that all types of textures are correctly synthesized using that set of measurements. In the same way as Heeger and Bergen's method, the input texture sample is decomposed with a multiscale oriented linear basis: the steerable pyramid [21, 79]. For each pair of coefficients at nearby positions, orientations and scales, the average value of their product, of their magnitude product and their relative phase is measured. In addition to these parameters, some marginal statistics on the input image pixels distribution are kept: the mean, the variance, the skewness, the kurtosis and the range. The number of parameters will depend on the number of sub-band images and on the size of the neighbourhood considered to estimate the statistical constraints of the example texture.

The second part of the algorithm is the synthesis step coercing to a random noise image, the measurements previously computed. The synthesized image is initialized with a Gaussian white noise image and then iteratively the algorithm alternates between: 1) constructing the steerable pyramid and enforcing the sample statistics of each sub-band image matching those of the corresponding sub-bands of the target image; 2) reconstructing an image from the pyramid and then forcing it to have the same marginal statistics as the input texture.

A texture is defined as a two-dimensional stationary random field $X(m, n)$ on a finite lattice $(m, n) \in \Omega \subset \mathbb{Z}^2$. Julesz' hypothesis is the basis to connect this statistical definition to perception: there exists a set of constraint functions $\{\phi_k, k = 1, \ldots, K\}$ such as if two random fields, X and Y, are identical in expectation over this set of functions then any two samples drawn from X and Y will be perceptually equivalent under some fixed comparison conditions. The importance of human perception as a fundamental criterion of equivalence between textures can be seen through this hypothesis, as well as the existence of such a set of statistical measurements capable of capturing this equivalence. To choose the set of constraint functions Portilla and Simoncelli proceeded as follows:

1. Set an initial set of constraints and synthesize a large library of texture examples;
2. Group the synthesis failures classifying them according to visual features that distinguish them from their original texture examples and keep the group with the poorest results;
3. Add a new statistical constraint to the set capturing the missing feature of the failure group;
4. Re-synthesize the failure group and verify the wanted feature is captured; otherwise go back to the previous point;
5. Verify that the original constraints are still needed; for each constraint, find a texture example that fails when the constraint is removed from the set;

6. Delete the unnecessary constraint, re-synthesize the library and go back to the second point.

Following this strategy, the constraint set is adapted to a reference set of textures and not just to one texture, and it is driven by perceptual criteria. The set of constraints is composed of:

1. Marginal statistics formed by: skewness and kurtosis of the low-pass images of each level of the pyramid, variance of the high-pass image of the pyramid, skewness, kurtosis, variance, mean and range of the image. The marginal statistics set the general degree of pixel intensity and their distribution. This is why they cannot be discarded from the statistics set [69].
2. Autocorrelation of the low-band coefficients. This allows to capture the periodic structures of a texture as well as long-range correlation. Omitting this constraint from the original set yields unsatisfying results for textures having periodic or long-range correlation patterns [69].
3. Autocorrelation and cross-correlation of the magnitude of the sub-bands. These statistics appear to be relevant because observation reveals that oriented bands have a particular behaviour concerning certain pattern and their periodicity whatever the orientation [69]. The cross-correlations kept are of each sub-band image with others of the same scale (inner cross-correlation) and of each sub-band with sub-bands at the coarser scale (intra cross-correlation).
4. Cross-correlation of the real part of the sub-bands with the real and imaginary parts of the coefficients' phase of the coarser scale. This statistic is important to capture the strong illumination effects present in some texture images. In particular, the synthesized image looses its three-dimensional effect and the shadows structure if they are not considered [69].

The set of statistics is summarized in Table 1. As mentioned previously, the number of parameters used depends on the number of scales S and orientations Q of the steerable decomposition as well as the size of the neighbourhood N_a used to compute the auto-correlations. The total number of parameters is $6 + 1 + 2(S+1) + (S+1)(N_a^2+1)/2 + SQ(N_a^2+1)/2 + SQ(Q-1)/2 + (S-1)Q^2 + 2(S-1)Q^2$, where the terms correspond (from left to right) to: the marginal statistics of u, the variance of high pass image, the skewness and kurtosis of the low band images, the auto-correlation of the low band images, the auto-correlation of the sub band images, the inner cross-correlation of the sub band images, the intra cross-correlation of the

Table 1: Summary of the set of statistical constraints for the Portilla-Simoncelli method

range of u	$\max(u)$ and $\min(u)$
mean of u	$\mu_1(u)$
variance of u	$\mu_2(u)$
skewness of u	$\mu_3(u)/(\mu_2(u))^{1.5}$
kurtosis of u	$\mu_4(u)/(\mu_2(u))^2$
lowband's skewness	$\mu_3(l_s)/(\mu_2(l_s))^{1.5},\ 1 \leq s \leq S+1$
lowband's kurtosis	$\mu_4(l_s)/(\mu_2(l_s))^2,\ 1 \leq s \leq S+1$
highband's variance	$\mu_2(h)$
$\Re\{l_s\}$ *auto-correlation*	$\Gamma_{\Re\{l_s\}}(m,n),\ 1 \leq s \leq S+1$
$\lvert u^{s,q}\rvert$ *auto-correlation*	$\Gamma_{\lvert u^{s,q}\rvert}(m,n),\ 1 \leq s \leq S,\ 0 \leq q \leq Q-1$
inner cross-correlation	$\mathcal{C}\left(\lvert u^{s,q}\rvert, \left\lvert u^{s,q'}\right\rvert\right),\ 1 \leq s \leq S,\ 0 \leq q,q' \leq Q-1$
intra cross-correlation	$\mathcal{C}\left(\lvert u^{s,q}\rvert, \left\lvert u^{s+1,q'}\right\rvert\right),\ 1 \leq s \leq S-1,\ 0 \leq q,q' \leq Q-1$
cross-correlation with the real part of the phase	$\mathcal{C}\left(\Re\{u^{s,q}\}, \frac{\Re\left\{u^{s+1,q'}\right\}}{\lvert u^{s+1,q'}\rvert}\right),\ 1 \leq s \leq S-1,\ 0 \leq q,q' \leq Q-1$
cross-correlation with the imaginary part of the phase	$\mathcal{C}\left(\Re\{u^{s,q}\}, \frac{\Im\left\{u^{s+1,q'}\right\}}{\lvert u^{s+1,q'}\rvert}\right),\ 1 \leq s \leq S-1,\ 0 \leq q,q' \leq Q-1$
Central sample moment	$\mu_n(u) = \begin{cases} \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} u(i,j) & \text{if} n=1 \\ \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} (u(i,j)-\mu_1(u))^n & \text{if} n>1 \end{cases}$
Translation operator	$\tau_{x,y}(u) : u(m,n) \mapsto u(\lfloor m-x \rfloor_M, \lfloor n-y \rfloor_N)$ $0 \leq m \leq M-1, \quad 0 \leq n \leq N-1, \quad (x,y) \in \Omega$
Correlation	$\mathcal{C}(u,v) =$ $\frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(u(i,j)-m(u))(v(i,j)-m(v))^*$
Auto-correlation	$\Gamma_u(x,y) = \mathcal{C}(u, \tau_{x,y}(u))$

low band images and the cross correlation of the real part of the sub band images with the real and imaginary part of the phase sub band images. In general $S = 4$, $Q = 4$ and $N_a = 7$ are used, leading to a total of 710 parameters.

After setting the set of statistical constraints, a sample verifying them has to be generated. Let c_k be the corresponding estimated values of the constraint functions for a particular texture image. Portilla and Simoncelli [69] "samples" an image from the set of images that yield the same estimated constraints values $A_{\vec{\phi},\vec{c}} = \{u : \phi_k(u) = c_k,\ \forall k\}$. To pick at random from this set the authors proposed to select at random a sample u_0 from $\mathbb{R}^{|\Omega|}$ and then project it sequentially onto subsets of $A_{\vec{\phi},\vec{c}}$. To emulate this the authors proposed a gradient projection. That is moving in the direction of the gradient of the constraint $\phi_k(v)$:

$$v' = v + \lambda_k \vec{\nabla}\phi_k(v)$$

choosing λ_k such that

$$\phi_k(v') = c_k. \tag{4}$$

Computing $\vec{\nabla}\phi_k(v)$ is usually simple, and it remains to find the λ_k that solves (4). When there are multiple solutions for λ_k, the one with smaller amplitude is chosen, modifying as little as possible the image. In that way, we stay as close as possible to the already projected set. When there is no solution, the λ_k is the one that comes closest to satisfying (4). Finally this method can be extended to the adjustment of a subset of constraints. Once the set of statistical measurements is defined and a method to sample from the Julesz' ensemble of textures, the synthesis can be performed as explained previously.

In a pre-attentive examination, the results are in general indistinct from the original texture samples. Nevertheless, on attentive examination the synthesis of structured textures often present blurry and jammed results. Long range structures are missed and the method tends to homogenize the output texture. Figure 4 shows two synthesis results. The first example (left in Figure 4) represents a quasi-periodic image where the method yields excellent results although it contains some global structures. In the second example (right in Figure 4), even though we recognize the nature of the input sample, one can observe that strong structures are missing. It is impossible to recover the lined up tiles.

Increasing the number of orientations Q will improve the results since more information is captured. However for $Q > 4$ the improvement is hardly noticeable. The number of levels S of the steerable pyramid is the most influential parameter. Depending on the nature of the texture, it will need to be increased to capture the details at all scales. Once again, for microtextures this parameter is less influential. Finally, the size of the neighborhood N_a used to compute the autocorrelation is important whenever the texture has periodic information.

As we will see in Section 5, even though imperfect, the results are very impressive, as they succeed modeling most textures using a moderately large

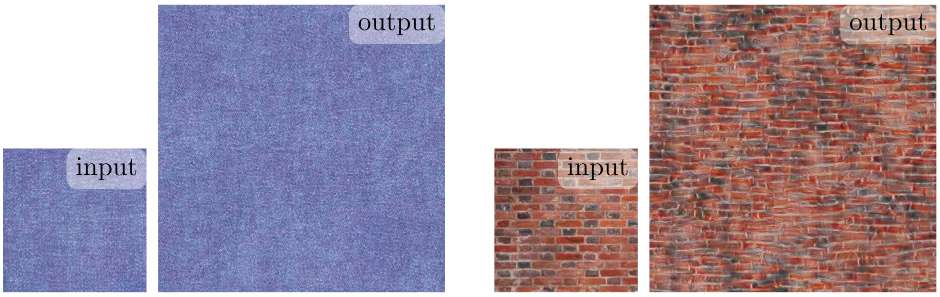

Figure 4: Synthesis results of the Portilla and Simoncelli method [69]. It is satisfactory for many small grain textures (left) but may miss the global structure (right).

set of global statistics. This brings us to the following two questions. Is the set of statistics considered enough to describe any kind of textures? Is the optimization step enough to enforce these statistics? Fifteen years later, Gatys et al. [27] proposed a texture synthesis method based on Convolutional Neural Networks (CNN) which can be seen as an extension of Portilla and Simoncelli's work, where the set of statistics used is much larger and unknown; also, the optimization is performed by the backpropagation method.

2.5. Texture synthesis using CNN

It is hard to define metrics to determine if two textures are similar or not according to human taste. Julesz' conjecture that humans cannot distinguish two textures with same second order statistics was invalidated. Yet this does not rule out a more general hypothesis, according to which there is a set of low-level filters such that if two textures respect the same statistics for these filters, they are indistinguishable. Portilla and Simoncelli's approach [69] and Zhu, Wu, and Mumford's FRAME (Filters, Random field, And Maximum Entropy) [97, 98] can be seen as fixing a set of hand-picked filters and synthesing new textures by enforcing the response to the filters to have similar statistics. The set of filters is chosen to match human expectations about textures. However determining the exact set of filters equivalent to human vision is very hard, and both approaches use only a subset of them. Portilla and Simoncelli achieve similar statistics by iterating specific projections, starting from white noise, while FRAME achieves that with a Gibbs Sampler and some simplifications (quantizing the image intensities, etc). Recently, Convolutional Neural Networks (CNNs) have given a breath of fresh air to these approaches. CNNs are compositions of layers of convolutions, non-linearities and pooling. In the past few years, CNNs have been successfully applied in a wide variety of domains, in particular in image related tasks. Arguably, the win by a large margin of CNNs [45] in the 2012 ILSVRC challenge [75], an image classification challenge, helped spark interest of the global community to these methods. We refer the reader to the corresponding literature for more details on the working of CNNs.

By taking a fully trained CNN on some visual classification task, and restricting to lower layers, one gets a set of low level filters which can directly be used for synthesizing texture, as shown in several works. The topic is quite active recently, and the question "how to best synthesize a texture with the help of neural networks" is far from being solved. In the following, we will focus on two different approaches: Gatys' texture generator [27] and DeepFrame [58]. Gatys' approach is to minimize the distance between the Gram matrices defined by the local filter responses of the network layers, while DeepFrame generates textures by sampling from an exponential model. The use of CNNs by these new approaches solves the issues of their ancestors: first the filters do not need to be handpicked anymore, they are encoded directly by the CNN. A pre-trained CNN successful on some image-related tasks can be selected for the texture generation. The choice of the CNN and whether it is pre-trained or the weights are random, affect the result. Second, the architecture of Neural Networks eases the generation process. The statistics of all the filters can be handled at the same time, via backpropagation for example. DeepFrame needs no quantization, unlike its predecessor, and synthesizes textures at a faster speed. Because the filter responses at a given Neural Network layer also encode the image content, texture transfer – also named style transfer – can be achieved by applying the statistics of the filter responses of a source image to a target image while keeping overall the filter responses similar [29]. While initially both Gatys' texture generator and DeepFrame used the VGG network [80] trained on ImageNet [15], more recent work obtained good results with networks with random weights [85] or by integrating the network training with the generation process [93]. The success of VGG for texture generation seems to stem from its training on an object classification task. This implies that its trained features are valuable "textons" able to discriminate shape and object features. One could imagine using a network trained to distinguish textures directly, instead of VGG. But, to the best of our knowledge, no network has been trained on an ImageNet equivalent to textures.

We now take a closer look at Gatys' texture generator and at DeepFrame. Gatys' texture model is a generalization of Julesz' model. It postulates that textures are described by the correlations between the neural network activations (features). Thus, by starting from random noise and imposing the correlations between the features to be the same as for a given input texture, one should get a new sample of this texture.

More precisely, Gatys's texture generator seeks to minimize the cost

$$E = \sum_l w_l ||G^l - T^l||_F^2$$

where $||.||_F$ is the Frobenius norm, w_l are weights and G^l, T^l are the Gram matrices, respectively for the image and the target texture, of the feature maps of a pretrained neural network at a layer l. In [27], a custom 19-layer VGG network was used where max pooling was replaced by average pooling and the network weights were rescaled. Let N_l be the number of feature maps at layer l (this usually corresponds to the number of "channels"), and M_l the size of each feature map at layer l ($M_l \times N_l$ is the number of outputs of layer l). If we denote by $F^l_{ij}, i \in \{1 \cdots, N_l\}, j \in \{1 \cdots, M_l\}$, the j-th output with the i-th feature map at layer l, then

$$\left(G^l\right)_{ij} = \frac{1}{M_l} \sum_{k=0}^{M_l} F^l_{ik} F^l_{jk}.$$

The texture generator minimizes the cost via backpropagation in the network, and thus falls into a local minimum. Starting from white noise, several thousand iterations can be needed to reach visual convergence. While in [27] the features were extracted from VGG [80], a Deep Convolutional Neural Network trained on image classification tasks, in [85] it is noted that taking a pre-trained network is not necessary and a network with random weights can give satisfying results. The minimization of E is done with L-BFGS-B [94] and the bounds are set to the minima and maxima of the source texture. After convergence, the histogram of the source is enforced.

To generate the results in this article, we made a few changes compared to [27]. The 19-layer VGG network used in [27] pads the outputs at every convolution layer with zeros on each layer (to have the layer outputs be the same size as the layer inputs). That, plus the fact that pixels on the border are "seen" by fewer features than the pixels in the center, means that all pixels on the image are not imposed the same distribution. If we take the same layers than in [27] (`conv1_1`, `pool1`, `pool2`, `pool3`, `pool4`) the top layer's outputs (`pool4`) depend each on a 124×124 area of the source. Thus 123 pixels should be removed on each border in order to have all remaining pixels seen by the same number of features. Removing 123 pixels on each border is not sufficient however to get the same constraints on the border and the center since the neighbouring pixels affect the features, and those neighbouring pixels are not affected by the same features. Thus to generate the results in this article, we decided to both remove the padding and generate bigger images – 256 pixels more on each border – which we then crop. The impact of this change can be seen on Figure 5. Other than that, we took the same parameters. In [2] the method solves the same problem by removing the network padding and enforcing periodicity. With the default network and parameters of Gatys' texture generator, except for the boundaries, a pixel is seen by 37504 filters. In Gatys' method, textures are only described by the Gram matrices. The number of elements in the Gram matrices totals 352256, 176640 if we remove the redundant values (the matrices are symmetric). This number of parameters doesn't depend on the image size, and once the Gram matrices of the source computed, the output texture can be any size.

To fix some of the shortcomings of Gatys' texture generator [27], several works complete the objective function. The method in [55] incorporates spectrum constraints to significantly improve the generation of textures with low frequency patterns. In [6] the proposed method considers spatial co-occurences of features to help handling long-range consistency constraints. In [91] it is noticed that the Gram matrices have several particularities that decrease the quality of the texture obtained in several cases with instabilities, particularly visible when generating a texture with a size different from the source. In our experiments we didn't notice such an instability problem, although we observed some instabilities (see for example the fourth column of Figure 20 and the first column of Figure 22). It is possible that the instabilities are affected by the parameter choice. To solve the instability problem, the authors added to the objective function a term to force the feature maps histograms to be the same as for the source. The authors of [63] also discussed some insufficiencies of Gram matrices in the case of style transfer, and in particular proposed to shift the activations to avoid sparsity. To accelerate the speed of the texture generation, the method of [83] trains for a given texture a new CNN, which outputs new samples of the texture. The CNN is trained with the same objective function as for Gatys' texture generator. Once the CNN is trained, generation is fast.

DeepFrame's texture generator samples from an exponential model. The model is defined by the probability density function

$$f(u; w) = \frac{1}{Z(w)} \exp\left[\sum_{k=1}^{K} \sum_{x\in\Omega} w_k F_k(u)(x)\right] g(u),$$

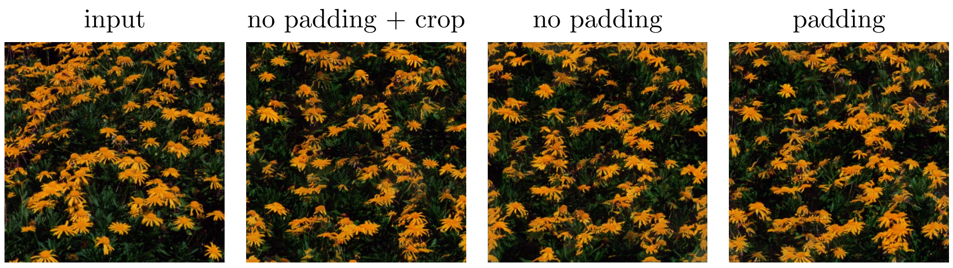

Figure 5: This figure shows the impact of the padding in the neural network. The second image shows the result of a 1024×1024 generated texture without the network padding, cropped to 512 × 512, while the figures on the right show 512 × 512 sized generated results without or with padding. The same random initialization was used for all three results (and cropped for the last two results). The differences are particularly visible on the border of the pictures, since it is where each variant imposes different statistics.

where F_k corresponds to a filter map extracted from a CNN, Ω is the image domain of u the image, $Z(w)$ is a normalizing constant and $g(u)$ is a reference distribution, like

$$g(u) = \frac{1}{(2\pi\sigma^2)^{|\Omega|/2}} \exp\left[-\frac{1}{2\sigma^2}||u||^2\right].$$

In contrast, the FRAME model defined the probability density function

$$f(u;\lambda) = \frac{1}{Z(\lambda)} \exp\left[\sum_{k=1}^{K}\sum_{x\in\Omega} \lambda_k[F_k * u(x)]\right]$$

where the $(F_k)_{k=1..K}$ were kernels, such as Gabor filters, or Difference of Gaussian filters, and λ_k was a discretization function with finite number of possible outputs.

In a first phase, the DeepFrame parameters $w = (w_k)$ are tuned for the source texture, then in a second phase new samples of the texture are generated via Langevin dynamics. While in [58] a pre-trained network is used, in the method of [93] its own network is trained on the source.

While both Gatys' texture generator and DeepFrame have a fixed texture model used to generate new samples, for which they learn parameters, a third successful CNN method to synthesize texture learns directly its model: in [36] a generative CNN is trained to synthesize new images from one or several samples of a source. The training is based on the adversarial model: a discriminator tries to distinguish the fake generated samples from true ones, while a generator creates new samples. Spatial invariance assumptions are encoded in the networks, but else, the texture model is in some sense learned by the two networks. This method can still be considered as a statistics-based method, because in some sense the discriminator checks the statistics of the texture are correct. To generate samples with this method ("SGAN" for Spatial Generative Adversarial Networks), we took the default network parameters, and applied the source histogram. We stopped after a few hundred epochs. The outputs suffer from a sort of noise pattern, which changes after every epoch. When the noise pattern was too important, we decided to select among the last twenty epochs the generator's result with the less noise. SGAN is a recent method, and there are certainly ways to better select the parameters and reduce this noise, but this goes beyond our goals here. Recently a new extension called PSGAN (for Periodic Spatial Generative Adversarial Networks) [7] was introduced to fix some shortcomings of SGAN, in particular to improve the generation result for textures with periodic patterns.

CNNs are also successful in the synthesis of images more general than textures [64, 86, 76, 18], in particular with methods relying on Generative Adversarial Networks (GAN) [30, 16, 73, 35, 88], but these methods are out of the scope of this study, which focuses on synthesizing new texture samples based on a single reference sample. These methods generally need a database of images.

3. Patch re-arrangement methods

In contrast to the statistics-based methods, the patch re-arrangement methods do not attempt to characterize textures by a statistical model. Spanning from the groundbreaking work by Efros and Leung [19], this family of algorithms consists of clever heuristics to re-arrange parts of the sample texture in a random way in order to create a new texture. By copying directly from the sample image, these methods often are able to keep complex structures from the input. By the same token, the process is frequently limited to copying and the results show little innovation relative to the sample. We will illustrate the family here by the original Efros and Leung [19] algorithm, a further extension by Efros and Freeman [20] which incorporate more recent techniques, and a more recent CNN based method [53].

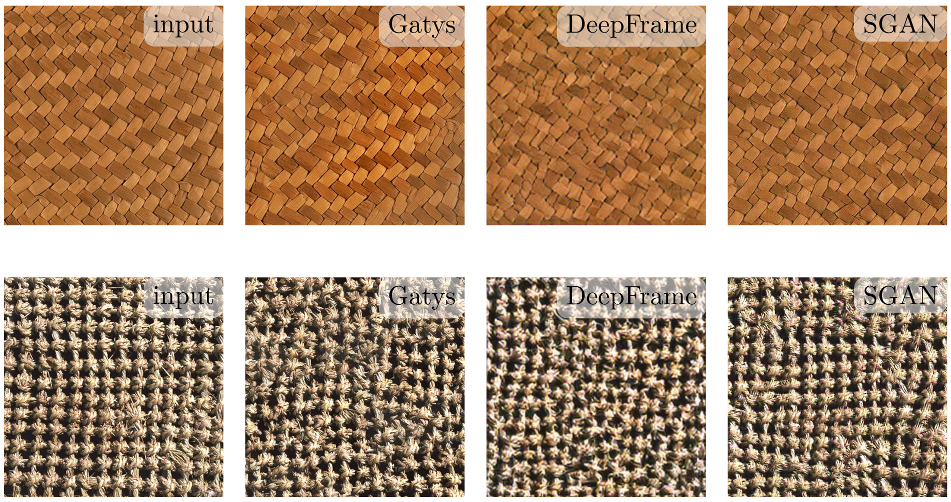

Figure 6: Comparison between Gatys' texture generator [27], DeepFrame [58] and SGAN [36]. For all three methods, we used the default parameters, except that in the case of Gatys we used the method we described above where we remove the network padding and crop the result and in case of DeepFrame and SGAN, we specified the result's histogram on the source histogram. Overall, SGAN looks the best when looking from far, but when zoomed in, Gatys seems to respect the best the local structures.

3.1. The Efros and Leung algorithm

In his foundational paper of information theory [78], Claude E. Shannon proposed to approximate the information contents of natural languages by the entropy of generative stochastic processes. He used a Markov chain to generate English text sequentially, letter by letter. Given a piece of already generated text, the next letter is sampled from the probability distribution of English text conditioned to the previous n letters. The following sequence was generated by Shannon using a third-order model:

> in no ist lat whey cratict froure birs grocid pondenome of demonstures of the reptagin is regoactiona of cre

Although very few words are real English words, this simple model produces surprisingly good English "textures". Inspired by Shannon's method, Efros and Leung [19] proposed to adapt the same ideas for image texture synthesis.

Efros and Leung in [19] synthesize a new texture image by considering that a pixel value depends on the values of its neighbouring pixels. The

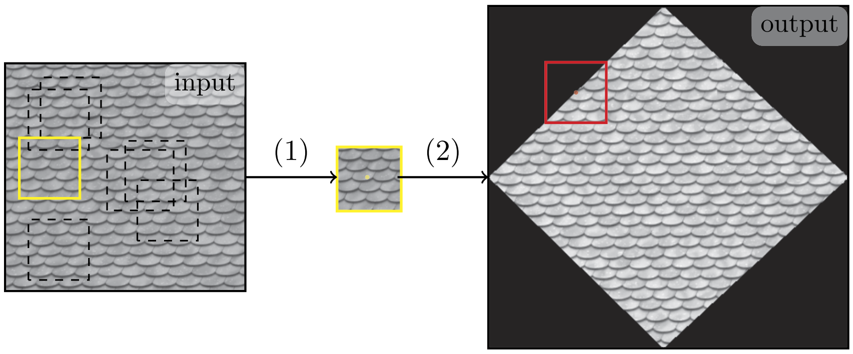

Figure 7: Overview of the Efros and Leung algorithm [19]. Given a texture image (left) a new image (right) is being synthesized a pixel at a time. For a pixel (m, n) (red point in the output) being synthesized the method finds all neighbourhoods in the left image that match the neighbourhood of (m, n) (dashed squares) and then chooses randomly one of the neighbourhoods (yellow square) and assigns its central pixel value to (m, n).

method is illustrated in Figure 7 and works as follows. For a given input texture, a new image is synthesized sequentially, pixel by pixel. For a pixel (m, n) being synthesized, the algorithm finds all the neighbourhoods in the input image that are similar to the neighbourhood of (m, n) up to a patch distance tolerance. Then one of these neighbourhoods is randomly chosen and its central pixel value is affected to the pixel (m, n). The neighbourhood of (m, n) is a square patch but only the known pixels (coming from the seed or already synthetized) of this patch are considered when comparing to the neighbourhoods of the input. Denoting p_1 and p_2 two patches of size $P \times P$, the comparison is made using a Gaussian-weighted distance defined as

$$d(p_1, p_2) = \frac{1}{\sum_{i,j} G_\sigma(i,j)} \sum_{i,j} \Big(p_1(i,j) - p_2(i,j)\Big)^2 G_\sigma(i,j), \tag{5}$$

where G_σ is a Gaussian kernel with standard deviation σ.

Levina and Bickel in [51] provided a theoretical justification of Efros and Leung's work. The Efros and Leung algorithm is based on resampling from the random field directly, without constructing an explicit model for the distribution. The authors of [51] formalized this algorithm in the framework of resampling from random fields and proved that it provides consistent estimates of the joint distribution of pixels in a window of specified size.

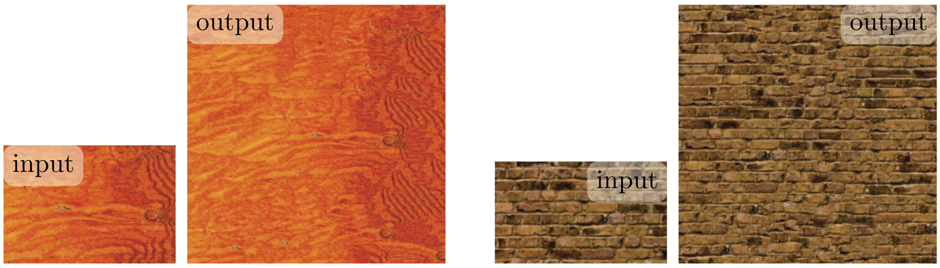

Figure 8: Synthesis results of the Efros and Leung method [19]. Left: the example shows the garbage growing effect. Right: the example shows the strength of this method to synthesize macrotextures. The patch size used for both synthesis is $P = 40$.

In general the visual results are very impressive, especially for structured textures. Nevertheless this algorithm suffers from two important drawbacks: verbatim copies of the input and garbage growing (the algorithm starts reproducing iteratively one part of the example and neglects the rest). Figure 8 shows two synthesis examples. The first synthesis result illustrates a failure case. In particular one can observe the effect of garbage growing, which reproduces incoherently the right side of the wood sample texture. The second example shows the strength of this method when it comes to synthesize textures with conspicuous patterns as in this case the brick patterns. To illustrate the verbatim-copy regions, *position and synthesis maps* are used to visualize from which regions of the input texture each synthetized pixel comes from. A synthesis and the corresponding map are shown in Figure 9 (obtained with the online demo [1]). Large continuous zones are identified in the synthesis maps which corresponds to the verbatim copies produced by the method. This representation also shows that the synthesized image is indeed a re-arrangement of pieces of the input sample.

Increasing the patch size P results in increasing the verbatim copied regions. However if the patch size is too small the local aspect of this method fails in recovering the global configuration of the input texture in particular for macrotextures. A second parameter of the method is the tolerance parameter ε which is used to select the most similar patches in the input image. Large tolerance values increase the garbage growing effect.

The Efros and Leung method also suffers from its high computational complexity. Several optimizations have been proposed to accelerate this algorithm. Among them Wei and Levoy [90] managed to fix the shape and size of the learning patch and Ashikhmin [3] proposed to extend existing

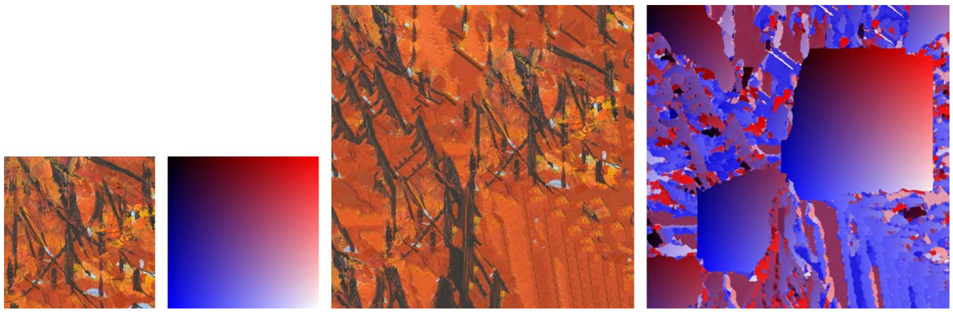
Figure 9: From left to right: texture sample, position map, synthesized image and synthesis map. The synthesis map shows for each synthesized patch its initial position in the texture sample. It allows then to identify exactly the verbatim copy regions (they correspond to continuous color areas of the map). This method reveals the verbatim copies of the input in the generated texture and the repetitions (garbage).

patches whenever possible instead of searching in the entire sample texture. The following section describes a particularly important extension of the method.

3.2. The Efros and Freeman algorithm

Efros and Freeman's method [20] is an extension of Efros and Leung's. It is based on the same principle where the pixel values are conditioned to their neighbourhood values. Efros and Freeman proposed to generate a new image sequentially, patch by patch (instead of pixel by pixel) in a raster scan order as illustrated in Figure 10. At each step a patch that is only partially defined on a region called *overlap region* is completed. This overlap region is of width w_o. This is the patch under construction. To do so a patch of the input image among those who match the patch under construction on its overlap region is randomly selected (*patch selection* step). An optimal boundary cut between the chosen patch (p_{in}) and the one under construction (p_{old}) is then computed across the overlap region (*stitching* step). This optimal boundary cut is used to construct a new patch (p_{new}) by blending the (p_{in}) and (p_{old}) along the cut. There are three possible overlap regions: vertical overlap for the first row, horizontal overlap for the first column, and L-shaped overlap everywhere else (Figure 10).

In the *patch selection* step, to select a patch p_{in} of an input image u one computes the square distance between the overlap region of the patch

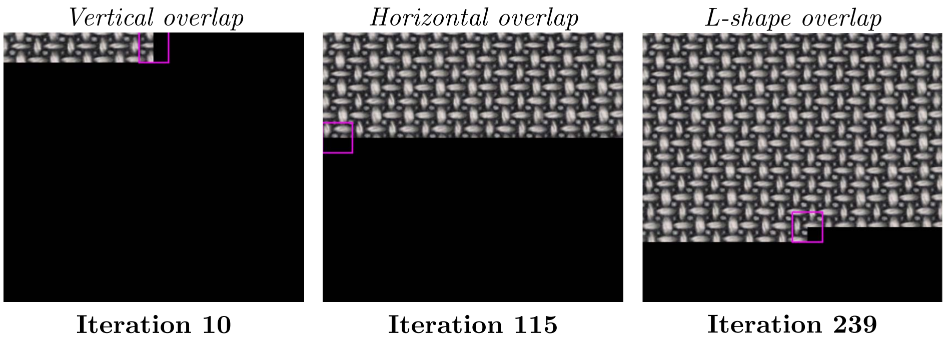

Figure 10: Three different iterations of the synthesis process are shown. At each iteration a patch is being synthesized. This patch is represented by the pink square in the three iterations shown. From left to right the three overlap cases are represented: vertical, horizontal and L-shape.

p_{old} and the corresponding regions of all the patches of u. The minimal distance $D_{\min}$ is determined and p_{in} is randomly picked among all patches whose distance to p_{old} is lower than $(1+\varepsilon)D_{\min}$ where ε is the tolerance parameter. The squared distance image d contains at each position (m,n) the distance between p_{old} and the patch from u according to some binary weight t that equals one in the overlap region and zero otherwise. More precisely, one has

$$d(m,n) = \sum_{i,j} t(i,j)(p_{\text{old}}(i,j) - u(m+i,n+j))^2. \tag{6}$$

The patch p_{in} of u having coordinates (m,n) is similar to the partially defined patch p_{old} on their overlap region. To get the final patch p_{new} one must combine the patches p_{old} and p_{in}. Denoting t the binary weight for the overlap regions as in (6), then, for any binary image r such that $0 \leq r(i,j) \leq t(i,j),\ (i,j) \in \{1,\ldots,P\}^2$, P can be defined as the combination

$$p_{\text{new}} = t\, p_{\text{old}} + (1-t)\, p_{\text{in}}.$$

The main contribution of Efros and Freeman [20] is to look for a binary shape M where the transition between p_{old} and p_{new} along the boundary of the shape is minimal. For simplicity, and to be able to use linear programming, the authors do not allow for any shape, but only for the ones whose boundaries are simple forward paths from one end to the other of the overlap region. This results in two pieces of image being sewn together along some general boundary path, hence the algorithm's name “quilting”.

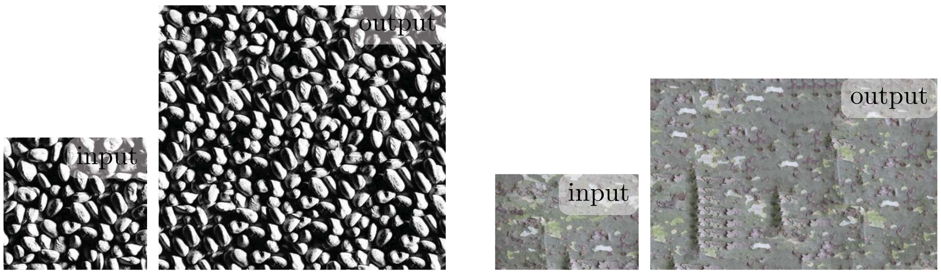

Figure 11: Synthesis results of the Efros and Freeman method [20]. It works for microtextures but risks losing the example's global statistics. It works for macrotextures too, but risks verbatim copies. Two examples are shown: a success (left) and a failure (right). The parameters used for are $P = 80$ and $O = P/4$.

This method yields very impressive visual results, in particular for highly structured textures. In terms of speed the gain is truly significant with respect to the methods which synthesize an image pixel by pixel. The patch size being larger, the risk of garbage growing is reduced compared to the Efros-Leung algorithm. Nevertheless the risk of verbatim copies remains and is even amplified. Moreover, the respect of the global statistics of the input is not guaranteed and this is quite visible when the input texture is not stationary (for example if there is a change of illumination across the image). Figure 11 shows two synthesis examples. The first one (left) shows an excellent synthesis result where the strong structures of the input are perfectly recovered. The second one (right) puts in evidence the verbatim copy of parts of the input and the garbage growing effect. To illustrate this the synthesis map of the second example is shown in Figure 12.

The parameters P and ε play the same role as in Efros and Leung's method. A third parameter, the overlap size O is used. Increasing this value tends to increase the verbatim copies of large regions. However if this value is too small then garbage growing increases. The value $O = 0.25P$ is generally satisfactory.

3.3. High level patch re-arrangement with Convolutional Neural Networks

The CNNs texture synthesis methods presented in Section 2.5 typically generated new texture samples by enforcing similar statistics on the feature

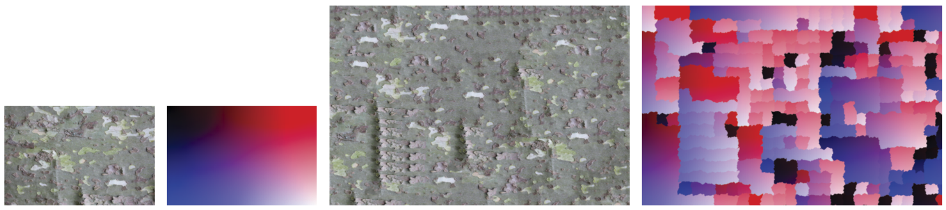
Figure 12: From left to right: texture sample, position map, synthesized image and synthesis map. The synthesis map shows for each synthesized patch its initial position in the texture sample. It puts in evidence the verbatim copy regions (they correspond to continuous color areas of the map) and the repetitions (corresponding to repeated continuous patches of the same color).

maps of a pre-trained CNN. At the crossroad of patch re-arrangement methods and CNNs, lies CNNMRF [53]. What distinguishes this method to those of Section 2.5 is that the texture samples are generated by enforcing similar patches of feature maps on selected upper layers of a pre-trained CNN. The image is then obtained with backpropagation and a smoothness constraint.

More precisely, starting from random noise, an image is generated by minimizing the energy:

$$E = \sum_{l} \sum_{i} ||\psi_i(F^l) - \psi_{NN(i)}(F_s^l)||^2 + R$$

where l goes among the selected layers (`relu3_1` and `relu4_1` of the VGG network [80]), i goes among all the positions in the layer, $\psi_i(F^l)$ represents the patch at the i-th position and $\psi_{NN(i)}(F_s^l)$ is its best matching patch in the source according to the normalized cross-correlation. The default patch size is 3×3 times the number of feature maps (often referred as the number of “channels” of the layer). R is a regularizer term to impose smoothness of the resulting image. In [53], the energy also contains a term to enforce the content of the source if doing texture transfer. This term isn't used for texture synthesis.

As noted by the author, a natural seamless patch blending is obtained by performing a patch re-arrangement on the levels of the CNN instead of doing it directly on the image, like in the other methods of this section.

Similarly to what was done in Section 2.5, we removed the padding of the VGG network to generate the results of this method. Indeed if the padding is kept, the spatial invariance assumption is violated. Moreover

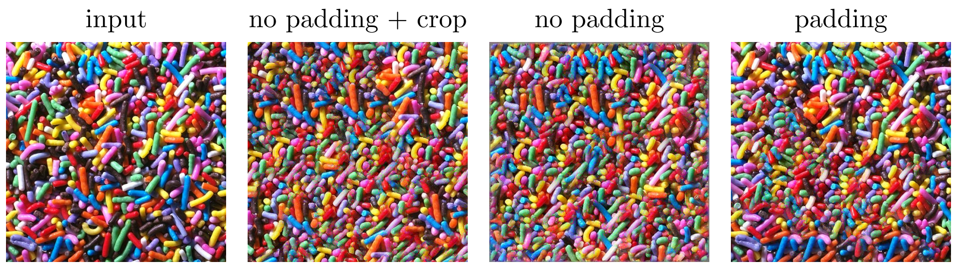

Figure 13: This figure shows the impact of the padding in the neural network when generating images with CNNMRF [53]. The second image shows the result of a 1024×1024 generated texture without the network padding, cropped to 512×512, while the figures on the right show 512×512 sized generated results without or with padding. The same random initialization was used for all three results (and cropped for the last two results).

pixels on the border of the generated images are seen by fewer features, which reinforces the violation of the spatial invariance. Thus in addition to removing the network padding, we generated bigger images and then cropped the result. On Figure 13, the generated texture with the network padding and no border crop kept tends to reproduce exactly significant parts of the input on the borders. This problem doesn't appear on the image with the padding removed and the border cropped. To generate the figures in Section 5 which features images of size 1024×1024, we couldn't add 256 pixels more on each border, as was done in Section 2.5, due to memory constraints. Instead we generated images of size 1280×1280, which were then cropped.

4. Hybrid methods

The two main approaches to texture synthesis are the statistics-based methods and the patch re-arrangement methods. In the first class, a texture is characterized by a statistical signature; then, a random sampling conditioned to this signature produces genuinely different texture images. Nevertheless, these methods often fail for macrotextures. The second class boils down to a clever "copy-paste" procedure, which stitches together verbatim copies of large regions of the example. A third kind of *hybrid methods* combines ideas from both approaches, leading to synthesized textures that are everywhere different from the original but with better quality than the purely statistics-based methods. We will describe one such method, its multiscale extension and the explicit combination of complementary algorithms.

4.1. Local Gaussian models for texture synthesis

Raad et al.'s method [70] uses locally Gaussian (LG) texture model in the patch space. Each texture patch is modeled by a multivariate Gaussian distribution learned from its similar patches. Inspired by [20], the idea of searching for patches to stitch together in the original sample is maintained. However, instead of using the exact patch taken in the input texture, the stitched patch is sampled from its Gaussian model. Locally Gaussian patch models have been proved very useful in image denoising [10]. This approach permits to maintain the coherence between patches with respect to the input sample, while creating new patches that do not exist in the sample texture but are still perceptually equivalent to it.

The multivariate Gaussian models involved are defined by their mean vector μ and their covariance matrix Σ. For a given patch p, of size $P \times P$ pixels, these parameters are estimated from the set of the R nearest patches $\mathcal{U}_p^u$ (nearest neighbours of p taken in u) as defined

$$\begin{aligned} \mu &= \tfrac{1}{R} \textstyle\sum_{p \in \mathcal{U}_p^u} p, \\ \Sigma &= \tfrac{1}{R-1} \textstyle\sum_{p \in \mathcal{U}_p^u} (p-\mu)\,(p-\mu)^t\,. \end{aligned} \tag{7}$$

The sampled vector p' is defined as

$$p' = \frac{1}{\sqrt{R-1}} \sum_{\rho \in \mathcal{U}_p^u} a_\rho(\rho - \mu) + \mu, \qquad a_\rho \sim \mathcal{N}(0,1), \tag{8}$$

where a_ρ are scalar random variables associated to each patch and following a normal distribution. Note that p' follows the distribution $\mathcal{N}(\mu, \Sigma)$. These models have reasonable variances, confirming that effectively the patches simulated have an acceptable degree of innovation [70].

The new texture image is synthesized by stitching together patches sampled from multivariate Gaussian distributions (8) in the input sample patch space. The method is iterative: the patches are synthesized in a raster-scan order (top to bottom and left to right). The goal of each iteration is to generate a new patch $p_v^{m,n}$ (patch in v placed at (m,n)) that is partially defined on a region called the overlap area (see Figure 10). The known part of the patch defines the set of patches $\mathcal{U}^u_{p_v^{m,n}}$ from which its Gaussian model is inferred. The generated patch $p_v^{m,n}$ is then sampled as defined in (8). The last step consists in stitching the patch into the output texture using the quilting method of [20].

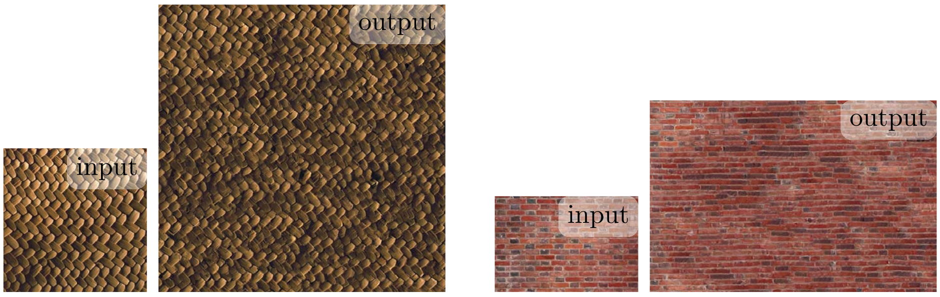

Figure 14: Synthesis results of the locally Gaussian method [70]. It works well for macrotextures. As one can observe in both examples the result is slightly blurry, a characteristics of the Gaussian model. The parameters used for are $P = 40$, $R = 30$ and $O = P/2$.

This synthesis algorithm generates a texture that is perceptually equivalent to the sample texture yet not composed of patches existing in the input texture. Thus, this method reduces some of the drawbacks of the statistics-based and the patch-based methods. Indeed the method yields satisfying results for micro- and macro-textures, and reduces the verbatim copies of the input. However, this method remains local and is (like all patch based approaches) not forced to respect the global statistics of the texture sample.

Figure 14 shows two results of the method. The algorithm remains dependent on the choice of the patch size P and of the number of nearest neighbours R as illustrated in Figure 15. These values may have to be adjusted for each texture sample. As for the overlap size a convenient value is $O = P/2$. If this value is too small then the region used to infer the Gaussian models is not enough. The patches used to infer the model can be very different on a high portion of the patch. The algorithm has a low computational complexity, compared for instance with classic patch-based denoising algorithms [49, 13]. An alternative to reduce the dependency of the method to the patch size is to work in a multiscale approach.

4.2. Multiscale texture synthesis methods

Most real textures are organized at multiple scales: the global structure is revealed at coarse scales but important detail are present at finer ones. As we have seen, the results of patch-based methods depend strongly on the patch size. Small patch sizes may capture the finer details of the input but the resulting texture will lack global coherence. On the other hand, using

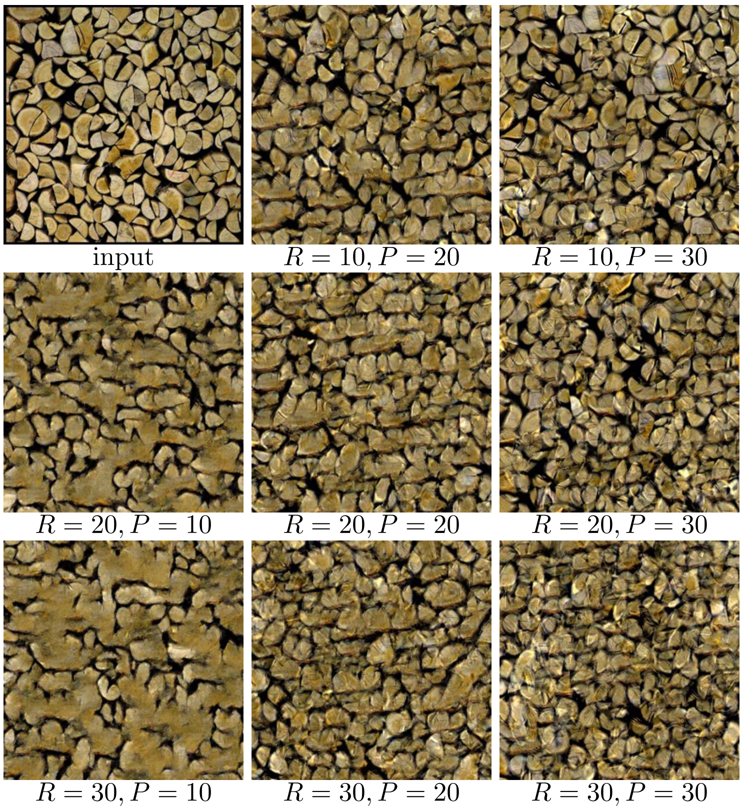

Figure 15: Texture synthesis result for the left top corner texture image. We show the results obtained for different values of R (the number of similar patches) and P (the side patch size). From left to right $P = 10, 20, 30$. From top to bottom, the number of nearest neighbours is $R = 10, 20, 30$. All the results are obtained for an overlap of a half patch size $O = P/2$.

large patches will maintain the global structures at the risk of a “copy-paste” effect. Furthermore, with large patches it becomes impossible to model the patch variability due to the lack of sufficient samples. This is apparent in the examples of Figure 14, where modeling patches as multivariate Gaussian vectors leads to a slightly blurry texture. A natural solution is to use a multiscale approach [46, 81, 50, 34, 70] using several patch sizes for a single texture synthesis, capturing different levels of details.

This section illustrates the ideas and difficulties of a multiscale extension using as example the local Gaussian models for texture synthesis presented in the previous Section [70]. The Multi-Scale Locally Gaussian (MSLG) method works at S scales and can be summarized in a few sentences. The synthesis begins at the coarsest scale ($s = S - 1$) using the local Gaussian method where the quilting step is replaced by a simple average of the overlapping patches. For the remaining scales ($s = S-2, \dots, 0$), a synthesis is performed by using the result of the previous scale ($s+1$) and the sample image at the corresponding resolution. At each scale the synthesis is done patch by patch in a raster-scan order. Each new patch, added to the synthesized image, overlaps part of the previously synthesized patch and it is the combination of a low resolution patch and a high resolution one sampled from a multivariate Gaussian distribution. The Gaussian distribution of the high frequencies of a given patch is estimated from the high frequencies of its m nearest neighbours in the corresponding scale input image. The synthesis result of the finer scale is the desired output image.

Let us denote the sample texture by u and $u_s,\ s = 1, \dots, S-1$ are the zoomed out versions by a factor $2^s,\ s = 1, \dots, S-1$. The synthesis result at each scale is denoted by $v_s,\ s = 1, \dots, S-1$ and v is the synthesis result returned by the multiscale algorithm. An additional image $\tilde{v}_s$ is needed at each scale, corresponding to a low resolution version of v_s obtained by interpolating v_{s+1}. To estimate the parameters of the Gaussian distribution of the patch $p_{v_s}^{m',n'}$ being processed, the set $\mathcal{U}^{u_s}_{p_{v_s}^{m',n'}}$ of R nearest patches in u_s is considered. The R nearest neighbours in u_s to the current patch are those minimizing the L^2 distance restricted to the overlap area:

$$\begin{aligned} d(p_{u_s}^{m,n}, p_{v_s}^{m',n'})^2 = {} & \frac{1}{|\mathcal{O}|} \sum_{(i,j)\in\mathcal{O}} (u_s(m+i, n+j) - v_s(m'+i, n'+j))^2 \\ & + \frac{1}{P^2} \sum_{i,j=0}^{P-1} (\tilde{u}_s(m+i, n+j) - \tilde{v}_s(m'+i, n'+j))^2, \end{aligned} \tag{9}$$

where $\tilde{u}_s$ denotes the low resolution of the image u_s, $\tilde{u}_s = u_s * G_\sigma$ and $\tilde{v}_{S-1} = u_{S-1} * G_\sigma$. In (9), the overlap area is denoted as $\mathcal{O}$ and the size of patch overlap is fixed to $P/2$. On the set $\mathcal{U}^{u_s}_{p_{v_s}^{m',n'}}$ only the high frequency of the patches $p_{u_s}^{i,j} - p_{\tilde{u}_s}^{i,j}$ is considered to infer the multivariate Gaussian distribution $\mathcal{N}(\mu_H, \Sigma_H)$. The patch $p_{v_s}^{m,n}$ is synthesized as the combination of a low resolution patch $p_{\tilde{v}_s}^{m,n}$ yield from the previous scale with a high resolution one $p \sim \mathcal{N}(\mu_H, \Sigma_H)$, thus $p_{v_s}^{m,n} = p_{\tilde{v}_s}^{m,n} + p$. For more details please refer to [70].

Figure 16 shows two synthesis examples. In both cases the result is satisfyingly recovering the details of the different scales for reasonable values

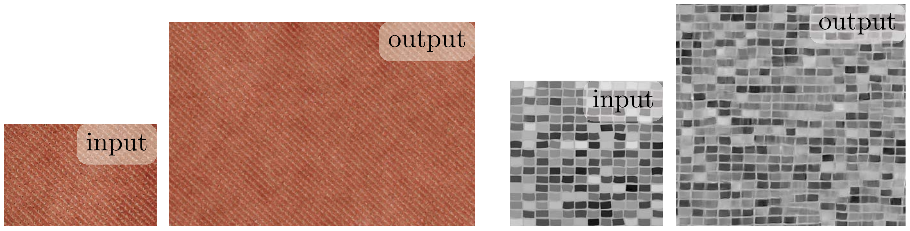

Figure 16: Synthesis results of the multiscale locally Gaussian method [70]. Both examples show that the details of different scales are correctly synthesized when using a patch size $P = 20$. However the results are slightly blurred with respect to the input. The number of scales is $S = 3$ for the first example (left) and $S = 2$ for the second example (right).

of the patch size $P = 20$. However one can notice that the results are blurry with respect to the input and this effect is increased with respect to the single scale approach.

4.3. Combination of methods

A smart combination of complementary methods may keep the advantages of each one. We will illustrate the methodology by combining a multiscale approach with three other methods:

MSLG+EF The Multi-Scale Locally Gaussian method combined with the Efros and Freeman method.

MSLG+PS The Multi-Scale Locally Gaussian method combined with the Portilla and Simoncelli method.

MSLG+Gatys The Multi-Scale Locally Gaussian method combined with the Gatys et al. method.

The combination of the Multi-Scale Locally Gaussian method with the Efros and Freeman method (MSLG+EF) consists of two steps. The first step synthesizes the given input u with the Multi-Scale Locally Gaussian method generating a new texture image that we denote u_{mslg}. The second step consists in applying the Efros and Freeman algorithm to the given input sample, initializing the output image that we denote u_{ef} with the image u_{mslg}. The method is basically the same as the one described in Section 3.2. The only step of the algorithm that is modified is the *patch selection* step. In the method described in [20] at each iteration the added patch was chosen among those (in the input sample) whose overlap region was similar to the one of

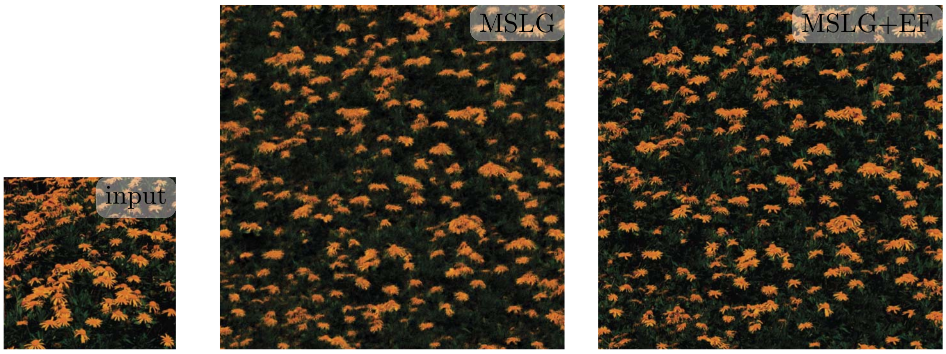

Figure 17: Synthesis results of the combination of the Multi-Scale Locally Gaussian method with the Efros and Freeman (MSLG+EF).

the patch under construction. When combining the methods, instead of only comparing the overlap areas, the entire patches are compared. Initializing the output with a first synthesis u_{mslg} enables the method to use the whole patch under construction to find a candidate in the input sample u. The candidate patch taken from u is then quilted in u_{ef} at the corresponding position with the same stitching step as in [20]. This combination allows to recover the lost resolution of the MSLG synthesis as illustrated in Figure 17. However it is not capable of masking the garbage growing effects as effectively MSLG+PS combination does.

The combination of the Multi-Scale Locally Gaussian method with the Portilla and Simoncelli method (MSLG+PS) consists of two steps. In the first step, given the input image u, a new texture u_{mslg} is generated using MSLG. The second step uses PS where the initialization "noise image" is replaced by u_{mslg} generating the output image that we denote u_{ps}. As explained in Section 2.4, the statistics to impose are learnt on the input u. What follows is a synthesis step where the output image is projected on the subspaces of constraints. There exist several local solutions to this projection step. When initializing PS with the result of MSLG, the initialization image is generally quite close to the images living in the sub-space of the whole set of constraints. Thus the result obtained is improved compared to PS images starting from a random noise image. Naturally fixing the initialization of the PS algorithm removes the randomness of the generated texture. But this is not the case since the initialization is itself random as it is generated from another random process. This combination is illustrated in Figure 18.

The combination of the Multi-Scale Locally Gaussian method with Gatys' texture generator (MSLG+Gatys) is very similar to its combination with the

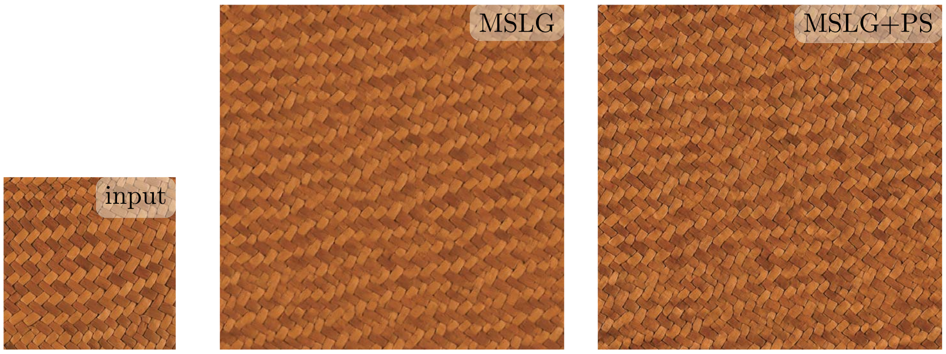

Figure 18: Synthesis results of the combination of the Multi-Scale Locally Gaussian method with the Portilla and Simoncelli methods (MSLG+PS).

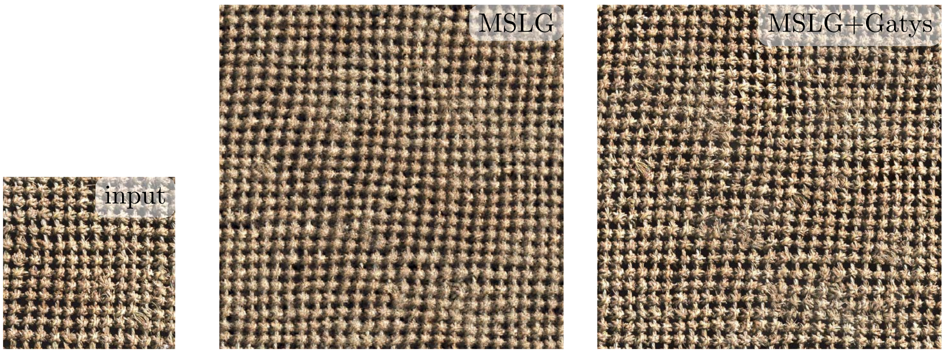

Figure 19: Synthesis results of the combination of the Multi-Scale Locally Gaussian method with Gatys' texture generator (MSLG+Gatys).

Portilla and Simoncelli method. The texture generator is initialized with the result of MSLG u_{mslg}, and the statistics of the target image are enforced via several iterations of backpropagation generating the output image denoted as $u$gatys. This combination is illustrated in Figure 19.

5. Experiments

The first part of this section compares the exemplar-based texture synthesis methods described before on a set of standard textures. These results illustrate the advantages and limitations of each one. Then, the second part attempts at the synthesis of real life and more complex textured images,

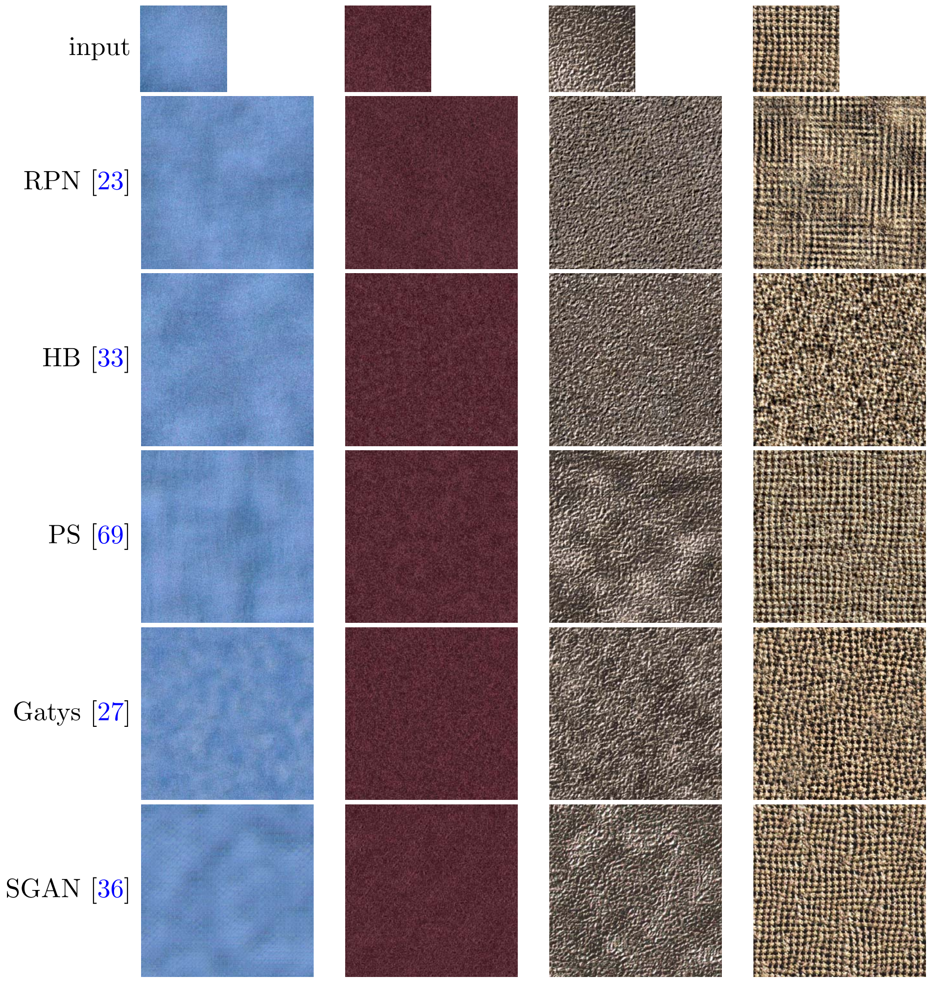

Figure 20: *Comparison of texture synthesis methods.* From top to bottom: input sample, Random Phase Noise (RPN) [23], Heeger and Bergen (HB) [33], Portilla and Simoncelli (PS) [69], Gatys (Gatys) [27] and SGAN [36].

revealing the shortcomings still present in all the methods when confronted with such a demanding task.

5.1. Comparative evaluation

We will compare the results of the following texture synthesis methods: Random Phase Noise (RPN) [87, 23], Heeger and Bergen (HB) [33], Portilla

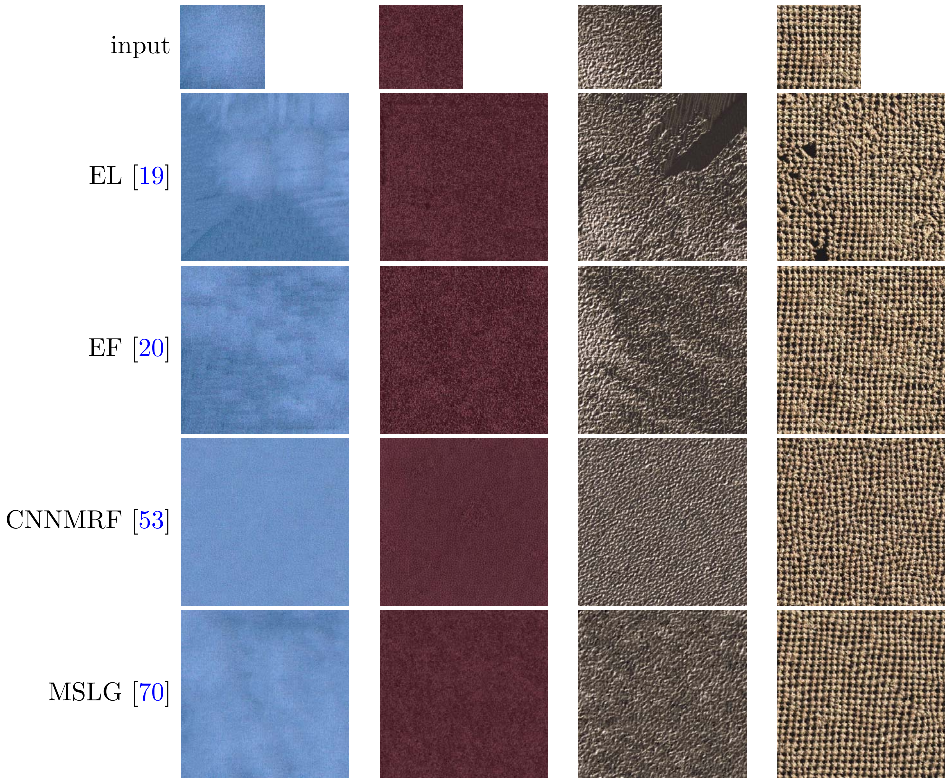

Figure 21: *Comparison of texture synthesis methods.* From top to bottom: input sample, Efros and Leung (EL) [19], Efros and Freeman (EF) [20], CNNMRF [53] and MSLG [70].

and Simoncelli (PS) [69], Gatys (Gatys) [27], SGAN [36], Efros and Leung (EL) [19], Efros and Freeman (EF) [20], CNNMRF [53] and MSLG [70]. Figures 20 to 23 show results for various texture samples, one per column; in each figure, the first row shows the sample image and the following rows correspond, as indicated, to one of the algorithms. We focus on these original texture synthesis algorithms, and do not show the numerous variants. For several of our sample textures, these variants could get better results, but we think that showing the results of the original algorithms better underlines their intrinsic strengths and weaknesses. Similarly we won't present the results of all the combinations of the different methods.

The second to sixth rows correspond to statistics-based methods described in Section 2, namely Random Phase Noise, Heeger-Bergen, Portilla-

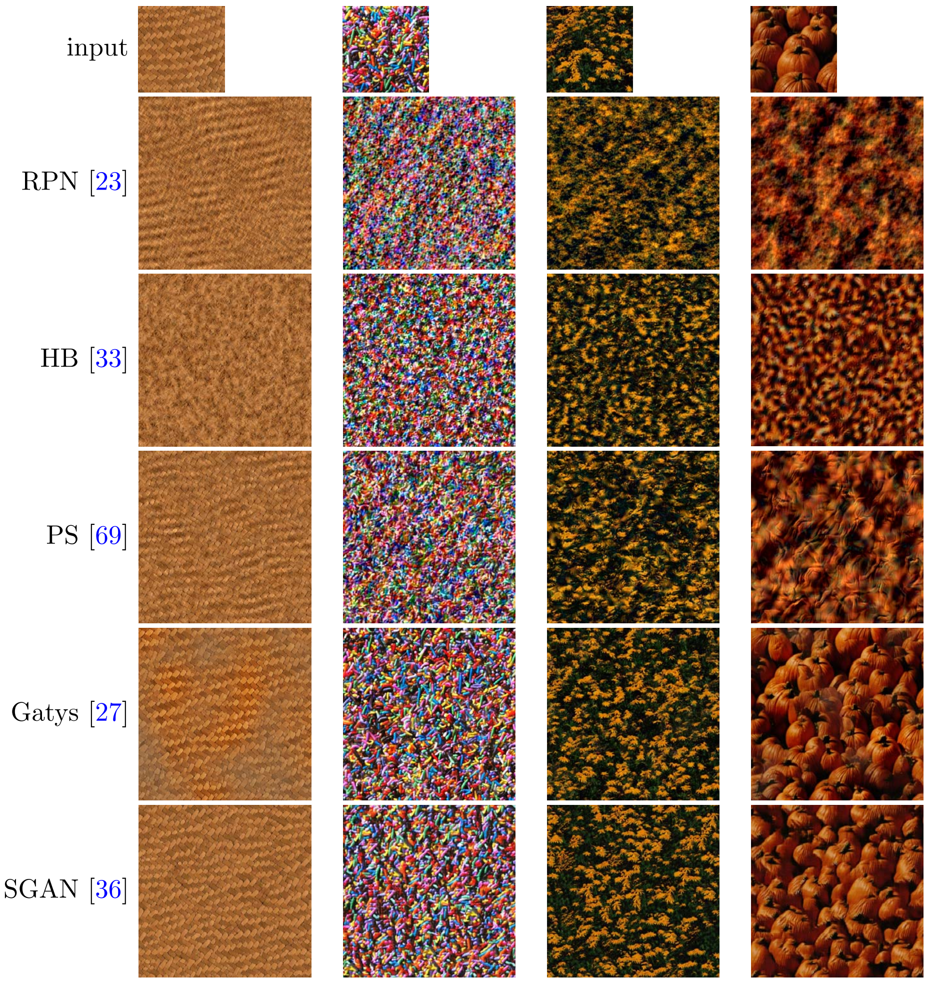

Figure 22: *Comparison of texture synthesis methods.* From top to bottom: input sample, Random Phase Noise (RPN) [23], Heeger and Bergen (HB) [33], Portilla and Simoncelli (PS) [69], Gatys (Gatys) [27] and SGAN [36].

Simoncellli, Gatys and SGAN. Early statistics-based methods: Heeger and Bergen (1995), Portilla and Simoncelli (2000) and Random Phase Noise (1991) yield good results for microtextures, i.e. textures with no conspicuous structures, as can be seen in the first texture example of Figure 20 and to a lesser extent for the second and third example. The Heeger and Bergen's method is inspired on a model of the early visual cortex; it provides satisfying results in some cases but there is no theoretical proof of

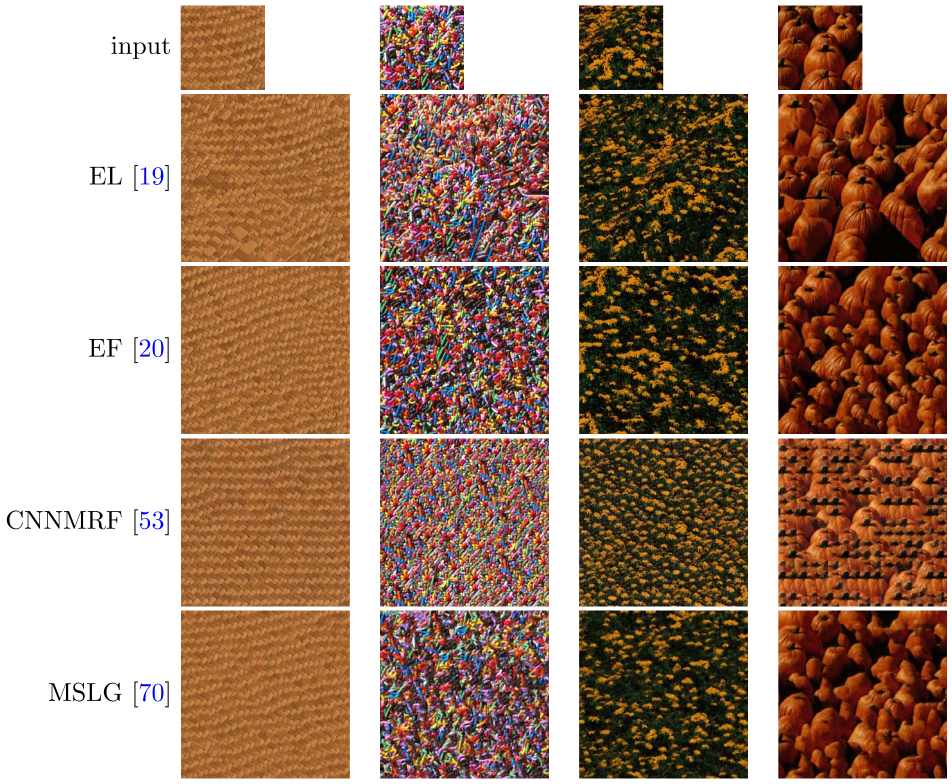

Figure 23: *Comparison of texture synthesis methods.* From top to bottom: input sample Efros and Leung (EL) [19], Efros and Freeman (EF) [20], CNNMRF [53] and MSLG [70].

the convergence of the method. On the other hand, the RPN method yields a simple and elegant theory with no convergence issue. The visual results yield by both methods are in general satisfying for microtextures. However, for textures with local structures, the results are blurry and unsatisfying. Among these three methods, the results obtained by Portilla and Simoncelli are by far the most remarkable. These results contain recognizable configurations from the sample. This can be observed for the last texture example in Figure 20 and the first two examples in Figure 22. Notice that the Heeger and Bergen and RPN methods yield unsatisfying results for these three examples. Clearly the global statistics considered by these methods are not enough to characterize these highly structured textures.

The second and third rows of the Figures 21 and 23 correspond to the patch re-arrangement methods Efros-Leung (1999) and Efros-Freeman (2001) described in Sections 3. The first three textures in Figure 21 have no conspicuous structures but are not stationary (for example, there are small changes of illumination). The Efros and Leung method, being too local, fails to recover the global characteristics of these textures. A similar and attenuated behavior is observed in Efros-Freeman's results. The methods are significantly better than their predecessors in the presence of local structure, but have their specific problems. Efros and Leung's results for the third and fourth texture in Figure 21 show two clear examples of garbage growing. The method has repeated a very small part of the input in an inconsistent way creating "garbage". In general this phenomenon is more evident in Efros and Leung's results, compared to those of Efros and Freeman. The results of Efros-Leung and Efros-Freeman for the first texture in Figure 23 show that the global organization is sometimes missed, mostly due to the fact that these methods work at a single scale. The second texture example in Figure 23 yields impressive results in the case of Efros-Freeman's method. Nevertheless, looking carefully one can notice the verbatim copies of the piece of chalks in the input image. The hybrid method MSLG (2016) described in Section 4, whose results are on the fifth row of Figures 21 and 23, faces the same issues for the three first examples in Figure 21. This is less visible though, since the Gaussian models tend to smooth slightly the result. However, the original granularity of the input sample is lost in MSLG. As mentioned in Section 3, Efros and Leung's and Efros and Freeman's results depend on the patch size, while the multiscale approach (MSLG) is more robust to that parameter. When the former two methods fail to preserve global organization, MSLG, working at multiple scales, manages to preserve this organization. In the second texture example in Figure 23, MSLG avoids the verbatim copy since the patches are being sampled from their Gaussian model and therefore are different from their original patches. Nevertheless, the Gaussian model strongly smooths the output. The synthesis of the flower texture (Figure 23 third row) is very satisfying for the three methods. Finally, the pumpkin texture shows a clear example of the verbatim copy effect in the Efros-Leung and Efros-Freeman methods. The fourth row of the Figures 21 and 23 corresponds to the CNN patch re-arrangement method CNNMRF (2016) described in Section 3. By bringing the patch re-arrangement to CNN features, the blending between the patches is improved. For example on the fourth texture of Figure 23, the separation between the patches are visible on the results of Efros-Freeman, which is not the case for CNNMRF. However CNNMRF suffers particularly from verbatim copies and fails to recover the global statistics of the image. Efros-Leung and Efros-Freeman do suffer less from these problems because the patch selection step for these methods picks randomly among a selection of patches, whereas CNNMRF takes the most likely patch.

The recent statistics-based CNN methods, discussed in Section 2.5, show significant improvement over their predecessors. Gatys (2015) is the best statistics-based method at respecting the fine details for all the textures of Figures 20 and 22, which can be well noticed with a zoom-in. However, some low frequencies or structure organizations are missed, as seen on the fourth texture of Figure 20, and some contrast instabilities can be noticed, for example on the first texture of Figure 22. As discussed in Section 2.5, some variants were proposed to fix these problems. SGAN (2016), on the other hand, better respects the low frequencies, and the results often look better than Gatys when zoomed-out. However on the fine scale, the results are incomplete and noisy, as seen on all the textures of the Figures 20 and 22. SGAN fails to generate correctly the first texture of Figure 20, possibly because this texture has no structure and is a microtexture. It is likely that better results can be obtained by tuning the parameters, but as said in Section 2.5, the default parameters were used.

Among all these methods, the CNN based methods are the most expensive in computational time. Pixel based methods, like Efros-Leung, are more expensive than patch based methods like Efros-Freeman or MSLG. The speed of statistics-based methods depends on how global the optimization is, and on the number of iterations needed. Portilla-Simoncelli's and Heeger-Bergen's speeds are comparable to patch based methods, while Random Phase Noise is the cheapest of the methods reviewed here.

These comparative evaluations show the strengths and weaknesses of the different original methods described in this survey. As said previously, some variants of these methods can get better results on some pictures. For example, a better result for the fourth texture of the Figure 20 can be seen on Figure 19. For this texture, first generating with MSLG, then refining with Gatys' texture generator, enables to combine the best of both algorithms: The fabric elements are well aligned, and look good at a fine scale. Overall, over the last three decades, tremendous progress was made to generate convincing new texture samples from a small and stationary texture sample. However one could argue that the samples used in this comparison are toy examples. Indeed, except for the third texture of Figure 20, and fourth texture of Figure 22, the samples do not suffer much of illumination changes or perspective, and are essentially stationary. Nevertheless, most textures are not stationary. Think for example of a wood texture. This leads

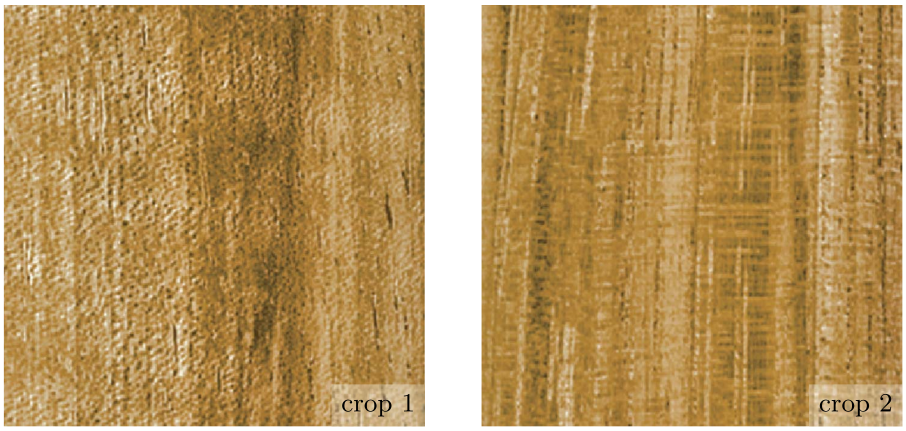

Figure 24: Two crops of different parts of a larger wood texture. The cropped images are of size 500 $\times$ 500 pixels. Each one represents a different texture belonging to a single “big texture”.

us to wonder whether the presented algorithms get acceptable results on these complex scenarios.

5.2. Getting out of toy examples

The previous examples present some quite impressive texture synthesis results by several algorithms. The texture synthesis problem seems to be almost solved for “academic” textures. Still, those results were obtained for pictures of relatively small size and taken in almost ideal conditions, in order to get almost stationary textures. In this section, we discuss the situation for more complex textures: When the same methods are applied to sample images of real and non-stationary textures, where long-range structure is present as well as varying detail at every scale. Figures 24 and 25 show some realistic examples of real world images that nobody would hesitate calling textures. Nevertheless on second thoughts they do have a complex, non-stationary structures, because every large enough image has it. But these are precisely the examples that need being emulated! In this endeavor, we can relax the requirement that the synthesis must make a larger image. Let’s just ask if a method is able to reproduce a perceptually similar texture at the very same size.

Each of textures in Figures 24 and 25 show different salient sub-textures within the same image. Since the methods in Section 5.1 usually assume that

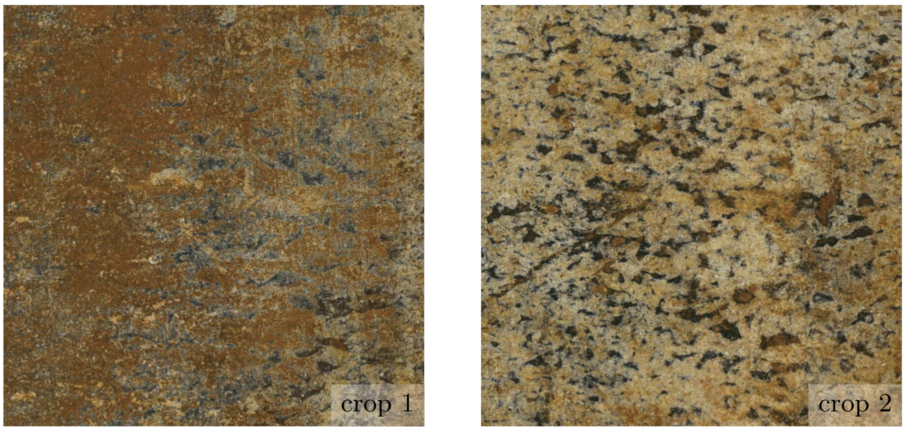

Figure 25: Two crops of different parts of a larger stone texture. The cropped images are of size 512 × 512 pixels. Each one represents a different texture belonging to a single "big texture".

the texture is stationary, it is not completely fair to use these methods on these samples. Several works have investigated ways to handle these complex cases [74, 4, 44, 56]. In this section we show the results of the state of the art algorithms presented in this paper, and will show that they are still far from emulating to real world textures, even without the requirement of building a larger texture patch from the sample.

Figures 26 and 27 show the results of the presented statistics-based, patch re-arrangement and hybrid methods on some of these more complex examples. The best results are Gatys' texture generator on the second texture and MSLG+PS and MSLG+Gatys on the first texture and fourth texture. When applying RPN or PS to them the results obtained are often too blurry. Gatys' texture generator fails to catch the low frequency structures for the last two textures. EF, CNNMRF and MSLG suffer from garbage growing and verbatim copies on the first three textures. This is true especially when the input is not stationary. SGAN fails to generate properly on the first two textures, and while the global organization of the third and fourth pictures is good, it suffers from the noise at small scale mentioned previously. As noticed in the previous section, the MSLG results are slightly blurry.

These results show that while some methods can get good results on some of these challenging texture samples, no method manages to get satisfying results for all four textures.

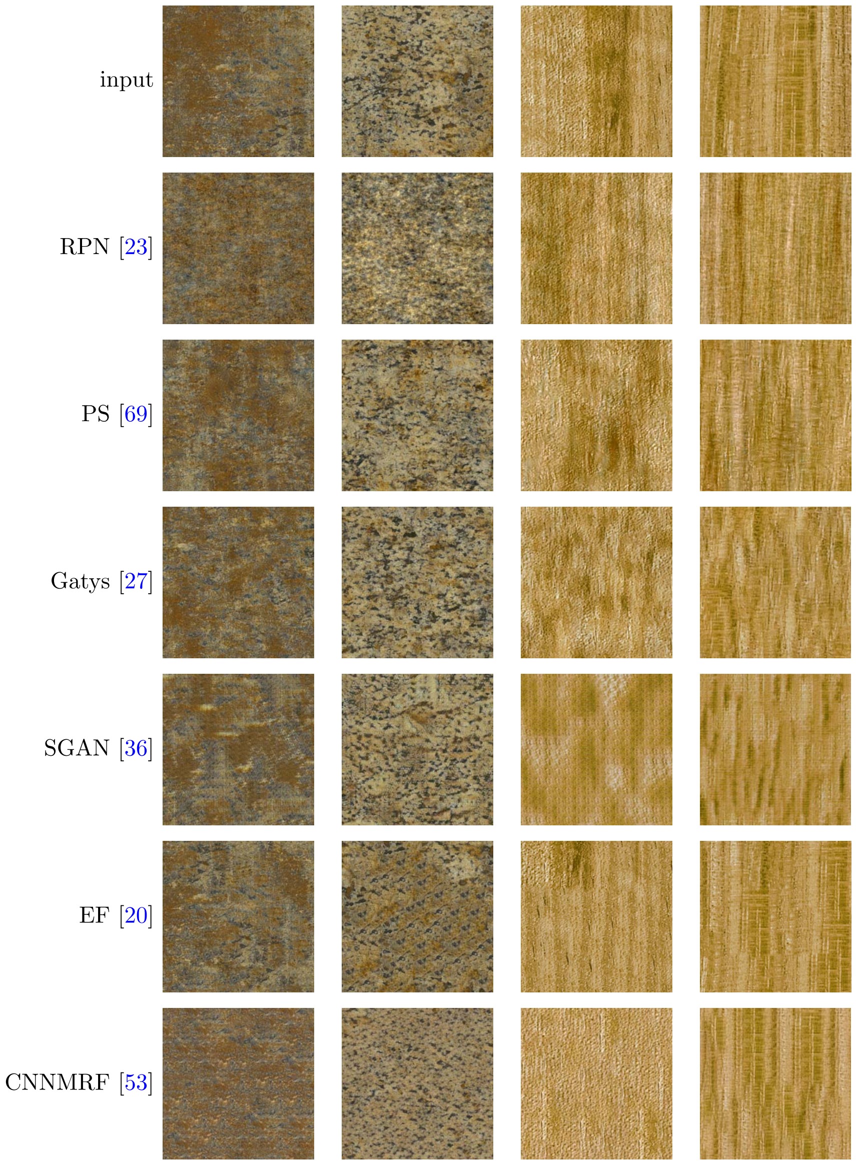

Figure 26: Synthesis results for statistical based and patch re-arrangement methods on complex texture. They show the current limitations of all best methods. RPN scrambles the textures. PS loses long range coherence of the wood veins. EF and CNNMRF's copy paste is quite visible for all textures and incurs in garbage growing. PS and Gatys have satisfying results on the left hand two textures, but miss to emulate long range interactions on the wood textures. SGAN grows periodic noise patterns.

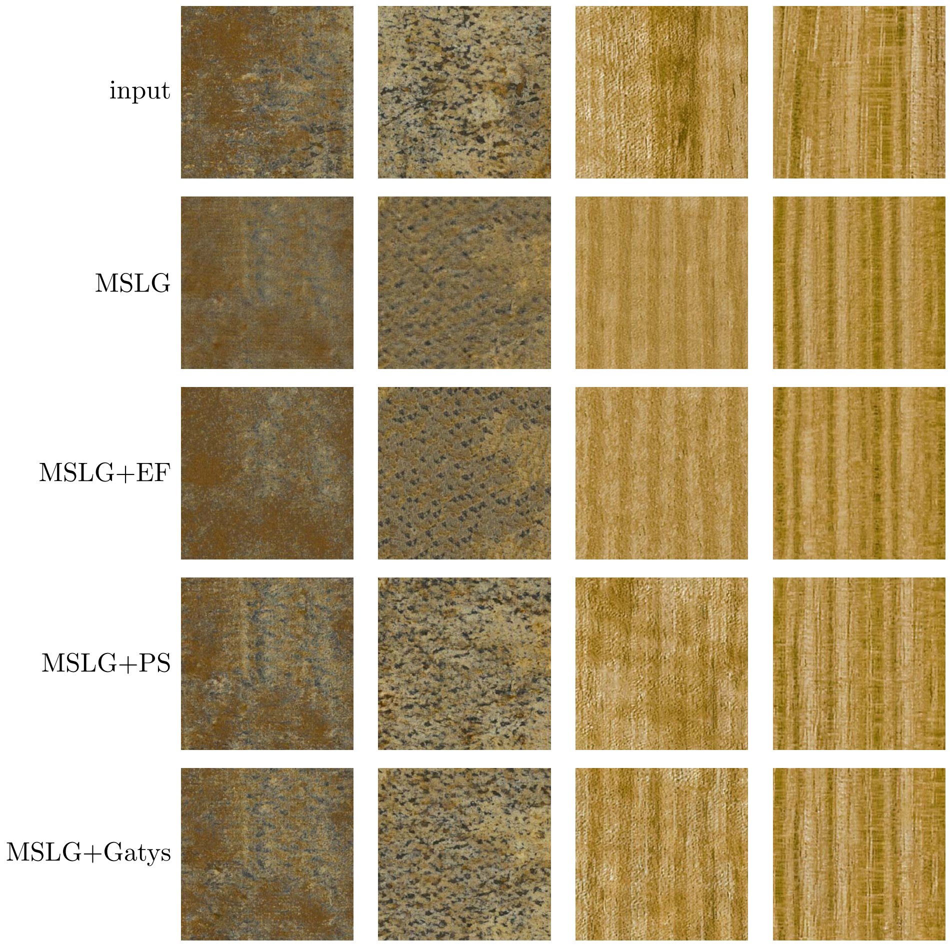

Figure 27: Synthesis results for the hybrid methods. In columns 2) and 3), MSLG has repetitions and garbage growing; thus all the generated results based on the MSLG outputs keep this defect. In columns 1) and 4), MSLG respects well the global statistics of the textures, and the combination with other methods indeed improves the result. MSLG+PS and MSLG+Gatys perform better on these examples than MSLG+EF.

6. Conclusion

With the multiplication of applications in computer graphics to the entertainment industry, the interest in the generation of synthetic objects with realistic texture has grown rapidly. High budget film sets, computer games, and in some cases digital art, spend intensive human efforts to imitate the appearance and feel of real world items. For this reason, exemplar-based texture synthesis has been the focus of intensive work for three decades. And as the available computational power increased, so has the sophistication of these methods.

In the end of the last millennium, statistics-based methods, such as RPN, Heeger-Bergen and Portilla-Simoncelli focused on a reduced set of statistics. The results were quite satisfactory on micro-textures, but could be blurry and far from the originals for more complex structures. Patch re-arrangement methods, such as Efros-Leung and Efros-Freeman, managed to respect significantly better the feel and the low level structures of these textures, but could have issues, such as discontinuities, verbatim copy, garbage growing or simply not respecting some essential statistics of the textures, such as the average intensity. Hybrid methods, such as MSLG, fix some of the issues of patch re-arrangement methods, but still share some of their issues. Very recently, statistics-based methods have been revisited with Convolutional Neural Networks (CNNs). CNN based methods significantly increase the number of texture statistics involved in their model, for example by a factor of 25 approximately in the case of Gatys' texture generator compared to Portilla-Simoncelli. The results show a spectacular progress over their predecessors, but no method is perfect yet. Patch re-arrangement methods were revisited as well by CNNs. CNNMRF improves the blending between the image patches, but the results still suffer from the problems mentioned above for patch re-arrangement methods. In this review, we presented three statistics-based neural methods with different models: Gatys' texture generator, DeepFrame and SGAN. When zoomed-in, the outputs of Gatys' texture generator are the best among the statistics-based methods, but miss some important low frequency constraints of the texture when zoomed-out. Some variants aim at fixing this shortcoming. SGAN succeeds better on several examples to respect the global structure of the texture, but the details of the texture are poor. While all the other statistics-based methods have an explicit texture model, the SGAN model is more implicit.

Our experimental results look no doubt sometimes worse than in the original papers, but precisely we did not select the best examples. Our examination of the history of the method leads to the following conclusions.

- The *exemplar-based texture synthesis* problem is implicitly ill-posed, as it requires to extrapolate a Fourier spectrum by enlarging the image given a very small sample of it. Having very small samples may have been historically interesting in computer graphics, but is no longer a technical issue,

given the available memories and computational power in all computers.
- By working on small texture examples the literature has somehow unrealistically restricted the problem. Indeed it is simply not true that textures are as stationary as those examples suggest.
- When trying to work on larger examples, we have seen that no texture sample is really stationary. A realistically large texture sample in fact contains smaller patches of very different textures.
- This explains first why patch based copy-paste methods are doomed in spite of some apparent success in some quasi-periodic texture with no conspicuous detail. On more involved samples, they cannot but reproduce recognizable details.
- This also explains why progress in this topic is linked to the design of methods enforcing more and more statistical parameters. The number of statistics enforced by statistical models is growing fast: 710 for Portilla-Simoncelli, 176640 for the default model in Gatys' texture generator. With some results showing that the filters can be chosen with random weights [85], one can wonder if the solution is not to just use the highest number of statistics possible to emulate a texture. One may also wonder where to draw a reasonable limit between synthesizing complex textures and rendering scenes containing textured objects.
- Thus, the Portilla-Simoncelli method, of enforcing a high number of statistics, wins, but it is somewhat a Pyrrhic victory. Indeed, the more random statistics we pile up, the better the exemplar-based results. But it remains to find numerical tools applying automatically an Occam's razor as Portilla and Simoncelli did manually. This is still needed to realize the goal of Julesz' program, which was to find the minimal sufficient set of statistics rendering two textures indistinguishable.

Acknowledgements

We thank Rafael Grompone von Gioi for valuable corrections and suggestions, Arthur Leclaire and Yang Lu for their feedback and for helping produce some of the experiments, and the anonymous referees for valuable advice.

References

[1] Cecilia Aguerrebere, Yann Gousseau, and Guillaume Tartavel. Exemplar-based texture synthesis: the Efros-Leung algorithm. *Image Processing On Line*, 2013:213–231, 2013.

[2] Miika Aittala, Timo Aila, and Jaakko Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics*, 35(4), 2016.

[3] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226. ACM, 2001.

[4] Abdourrahmane M. Atto, Zhangyun Tan, Olivier Alata, and Maxime Moreaud. Non-stationary texture synthesis from random field modeling. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4266–4270. IEEE, 2014.

[5] James R. Bergen and Edward H. Adelson. Visual texture segmentation based on energy measures (a). *Journal of the Optical Society of America A*, 3, 1986.

[6] Guillaume Berger and Roland Memisevic. Incorporating long-range consistency in cnn-based texture generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[7] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. *arXiv preprint arXiv:1705.06566*, 2017.

[8] Charles Bordenave, Yann Gousseau, and François Roueff. The dead leaves model: a general tessellation modeling occlusion. *Advances in Applied Probability*, pages 31–46, 2006. MR2213962

[9] Thibaud Briand, Jonathan Vacher, Bruno Galerne, and Julien Rabin. The Heeger & Bergen pyramid based texture synthesis algorithm. *Image Processing On Line*, 4:276–299, 2014.

[10] Antoni Buades, Marc Lebrun, and Jean-Michel Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. *Image Processing On Line*, 2013:1–42, 2013. MR3097031

[11] Terry Caelli and Béla Julesz. Experiments in the visual perception of texture. *Biol. Cybernetics*, 28:167–175, 1978.

[12] Tianhorng Chang and C. C. Jay Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Transactions on Image Processing*, 2(4):429–441, 1993.

[13] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095, 2007. MR2460626

[14] Jeremy 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. ACM Press/Addison-Wesley Publishing Co., 1997.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[16] Emily L. Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[17] Agnès Desolneux, Lionel Moisan, and Samuel Ronsin. A compact representation of random phase and Gaussian textures. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012. MR2664526

[18] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[19] Alexei Efros, Thomas K. Leung, et al. 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.

[20] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[21] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[22] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. Micro-texture synthesis by phase randomization. *Image Processing On Line*, 2011, 2011.

[23] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. Random phase textures: Theory and synthesis. *Image Processing, IEEE Transactions on*, 20(1):257–267, 2011. MR2789729

[24] Bruno Galerne and Arthur Leclaire. An Algorithm for Gaussian Texture Inpainting. *submitted to Image Processing Online*, (7):1–16, 2017. https://doi.org/10.5201/ipol.2017.198.

[25] Bruno Galerne and Arthur Leclaire. Texture Inpainting using Efficient Gaussian Conditional Simulation. *SIAM Journal on Imaging Sciences*, 10:1446–1474, 2017. https://doi.org/10.1137/16M1109047. MR3693601

[26] Bruno Galerne, Arthur Leclaire, and Lionel Moisan. Microtexture inpainting through Gaussian conditional simulation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1204–1208. IEEE, 2016.

[27] Leon Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[28] Leon Gatys, Alexander S. Ecker, and Matthias Bethge. Code to synthesise textures using convolutional neural networks as described in gatys et al. 2015. https://github.com/leongatys/DeepTextures, 2015–2016.

[29] Leon Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[31] Ulf Grenander. *General Pattern Theory*. Oxford University Press, 1993. MR1270904

[32] Baining Guo, Harry Shum, and Ying-Qing Xu. Chaos mosaic: Fast and memory efficient texture synthesis. *Microsoft research paper MSR-TR-2000-32*, 2000.

[33] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.

[34] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[35] Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*, 2016.

[36] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016.

[37] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Spatial generative adversarial networks. https://github.com/zalandoresearch/spatial_gan, 2016–2017.

[38] Béla Julesz. Visual pattern discrimination. *Information Theory, IRE Transactions on*, 8(2):84–92, 1962.

[39] Béla Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[40] Béla Julesz. A theory of preattentive texture discrimination based on first-order statistics of textons. *Biological cybernetics*, 41(2):131–138, 1981. MR0637489

[41] Béla Julesz. *Dialogues on Perception*. The MIT Press, 1995.

[42] Béla Julesz, Edgar N. Gilbert, Larry A. Shepp, and Harry L. Frisch. Inability of humans to discriminate between visual textures that agree in second-order statistics—revisited. *Perception*, 2(4):391–405, 1973.

[43] Béla Julesz, Edgar N. Gilbert, and Jonathan D. Victor. Visual discrimination of textures with identical third-order statistics. *Biological Cybernetics*, 31(3):137–140, 1978.

[44] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[46] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 795–802. ACM, 2005.

[47] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (ToG)*, volume 22, pages 277–286. ACM, 2003.

[48] Andrew Laine and Jian Fan. Texture classification by wavelet packet signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191, 1993.

[49] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. A nonlocal Bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013.

[50] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Transactions on Graphics (TOG)*, 24(3):777–786, 2005.

[51] Elizaveta Levina and Peter J. Bickel. Texture synthesis and nonparametric resampling of random fields. *The Annals of Statistics*, 34(4):1751–1773, 2006.

[52] Chuan Li and Michael Wand. Code for paper "combining markov random fields and convolutional neural networks for image synthesis". https://github.com/chuanli11/CNNMRF, 2015–2016.

[53] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2479–2486, 2016.

[54] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001.

[55] Gang Liu, Yann Gousseau, and Gui-Song Xia. Texture synthesis through convolutional neural networks and spectrum constraints. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4–8, 2016*, pages 3234–3239. IEEE, 2016.

[56] Yitzchak David Lockerman, Basile Sauvage, Rémi Allègre, Jean-Michel Dischler, Julie Dorsey, and Holly E Rushmeier. Multi-scale label-map extraction for texture synthesis. *ACM Trans. Graph.*, 35(4):140–1, 2016.

[57] Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Code for Learning FRAME Models Using CNN Filters. http://www.stat.ucla.edu/~yang.lu/project/deepFrame/main.html.

[58] Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Learning FRAME Models Using CNN Filters. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[59] Jitendra Malik and Pietro Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A*, 7(5):923–932, 1990.

[60] B.S. Manjunathi and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[61] Georges Matheron. Schéma booléen séquentiel de partition aléatoire. *Rapport technique N-83, Centre de Morphologie Mathématique, École des Mines de Paris*, 214, 1968.

[62] David Mumford and Agnès Desolneux. *Pattern Theory: The Stochastic Analysis of Real-World Signals*. A K Peters, Ltd., 2010. MR2723182

[63] Roman Novak and Yaroslav Nikulin. Improving the neural algorithm of artistic style. *arXiv preprint arXiv:1605.04603*, 2016.

[64] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[65] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[66] Gabriel Peyré. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34(1):17–31, 2009. MR2505403

[67] Gabriel Peyré. Texture synthesis with grouplets. *IEEE Trans. Pattern. Anal. Mach. Intell.*, 4(32):733–746, 2010.

[68] Javier Portilla and Eero P. Simoncelli. Representation and synthesis of visual texture. http://www.cns.nyu.edu/~lcv/texture/.

[69] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

[70] Lara Raad, Agnès Desolneux, and Jean-Michel Morel. A conditional multiscale locally Gaussian texture synthesis algorithm. *Journal of Mathematical Imaging and Vision*, 56(2):260–279, 2016. MR3535021

[71] Lara Raad and Bruno Galerne. Efros and freeman image quilting algorithm for texture synthesis. *IPOL Journal*, 7:1–22, 2017. MR3608122

[72] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 435–446. Springer Berlin / Heidelberg, 2012.

[73] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[74] Amir Rosenberger, Daniel Cohen-Or, and Dani Lischinski. Layered shape synthesis: automatic generation of control maps for non-stationary textures. In *ACM Transactions on Graphics (TOG)*, volume 28, page 107. ACM, 2009.

[75] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[76] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[77] Jean Serra. *Image analysis and mathematical morphology, v. 1.* Academic press, 1982. MR0753649

[78] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. MR0026286

[79] Eero P. Simoncelli and William T. Freeman. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *ICIP (3)*, pages 444–447, 1995.

[80] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[81] Guillaume Tartavel, Yann Gousseau, and Gabriel Peyré. Variational texture synthesis with sparsity and spectrum constraints. *Journal of Mathematical Imaging and Vision*, 52(1):124–144, 2014. MR3337519

[82] M. R. Turner. Texture discrimination by Gabor functions. *Biological Cybernetics*, 55:71–82, 1986.

[83] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*, 2016.

[84] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, 1995.

[85] Ivan Ustyuzhaninov, Wieland Brendel, Leon Gatys, and Matthias Bethge. What does it take to generate natural textures? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[86] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016.

[87] Jarke J. van Wijk. Spot noise texture synthesis for data visualization. In *SIGGRAPH '91*, pages 309–318, New York, NY, USA, 1991. ACM.

[88] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pages 318–335. Springer, 2016.

[89] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117. Eurographics Association, 2009.

[90] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[91] Pierre Wilmot, Eric Risser, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[92] Ying Nian Wu, Song-Chun Zhu, and Xiuwen Liu. Equivalence of Julesz ensembles and FRAME models. *International Journal of Computer Vision*, 38(3):247–265, 2000.

[93] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644, 2016.

[94] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Lbfgs-b: Fortran subroutines for large-scale bound constrained optimization. *Report NAM-11, EECS Department, Northwestern University*, 1994.

[95] Song-Chun Zhu, Cheng-en Guo, Yizhou Wang, and Zijian Xu. What are Textons? *International Journal of Computer Vision*, 62(1–2):121–143, 2005.

[96] Song-Chun Zhu, Xiu Wen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient Markov Chain Monte Carlo – toward a "trichromacy" theory of texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):554–569, 2000.

[97] Song-Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural computation*, 9(8):1627–1660, 1997.

[98] Song-Chun Zhu, Ying Nian Wu, and David 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.

LARA RAAD
DEPT. OF INFORMATION & COMMUNICATION TECHNOLOGIES
UNIVERSITAT POMPEU FABRA
C/ ROC BORONAT 138
08018 BARCELONA
SPAIN
E-mail address: lara.raad@upf.edu

AXEL DAVY
CMLA, ENS PARIS-SACLAY
61 AVENUE DU PRESIDENT WILSON
94235 CACHAN CEDEX
FRANCE
E-mail address: axel.davy@cmla.ens-cachan.fr

AGNÈS DESOLNEUX
CMLA, ENS PARIS-SACLAY
61 AVENUE DU PRESIDENT WILSON
94235 CACHAN CEDEX
FRANCE
E-mail address: desolneux@cmla.ens-cachan.fr

JEAN-MICHEL MOREL
CMLA, ENS PARIS-SACLAY
61 AVENUE DU PRESIDENT WILSON
94235 CACHAN CEDEX
FRANCE
E-mail address: morel@cmla.ens-cachan.fr

RECEIVED JUNE 18, 2017