Noname manuscript No.
(will be inserted by the editor)

A conditional multiscale locally Gaussian texture synthesis algorithm

Lara Raad · Agnès Desolneux · Jean-Michel Morel

Received: date / Accepted: date

Abstract Exemplar based texture synthesis is defined as the process of generating, from an input texture sample, new texture images that are perceptually equivalent to the input. In the present work, we model texture self-similarity with conditional Gaussian distributions in the patch space in order to extend the use of stitching techniques. Then, a multiscale texture synthesis algorithm is introduced, where texture patches are modeled at each scale as spatially variable Gaussian vectors in the patch space. The Gaussian distribution for each patch is inferred from the set of its nearest neighbours in the patch space obtained from the input sample. This approach is tested over several real and synthetic texture images and its results show the effectiveness of the proposed technique for a wide range of textures.

Keywords Texture Synthesis · Conditional Locally Gaussian · Patch Size · Multiscale

1 Introduction

Exemplar based texture synthesis is a well known problem that has many applications in computer graphics, computer vision and image processing, for example for fast scene generation, inpainting, and texture restoration. It is defined as the process of generating, from an input texture sample, new texture images that are perceptually equivalent to the input. Texture synthesis algorithms are roughly grouped into two categories: statistics-based [9,11,21] and non-parametric patch-based [2,5,6,13,14,18,27]. The first class of algorithms characterizes a given texture sample by estimating statistical parameters defining the underlying stochastic process. Although these methods can faithfully reproduce some of the global statistics of the sample and synthesize micro and pseudo-periodic textures, they generally do not yield high quality visual results for more structured ones, in particular when the sample is small and contains large objects. The second category rearranges local neighbourhoods of the input sample in a consistent way, and is often able to reproduce highly structured textures. Even though non-parametric methods yield visual satisfactory results, they often turn into practising verbatim copies of large parts of the input sample.

Centre de Mathématiques et de Leurs Applications
Ecole Normale Supérieure de Cachan, Université Paris-Saclay
61, avenue du Président Wilson, 94235 Cachan cedex, France

Statistics-based methods are performed in two steps: analysis and synthesis. The analysis step identifies a set of global statistics characterizing the input texture. The synthesis step creates an image satisfying the estimated set of statistics. These methods were inspired from Julesz work [12], who observed that many texture pairs having the same second-order statistics would not be preattentively discerned by humans. The success of Julesz' first hypothesis can be checked in [9] where the authors propose to synthesize textures by randomizing the Fourier phase of the sample image while maintaining its Fourier modulus, thus preserving the second order statistics of the sample. These statistics are enough to synthesize micro-textures that can be characterized by their Fourier modulus but they fail for more structured ones as can be verified in [8]. Heeger and Bergen [11] initiated more sophisticated statistics-based methods describing the input sample by the histograms of its wavelet coefficients. A new texture image is then created when enforcing these statistics on an initial white noise image. The results of this method are satisfying for a small class of textures since it only measures marginal statistics. Indeed the proposed statistics miss important correlations between pixels across scales and orientations. This experimental fact can be verified in [4]. In [21] Portilla and Simoncelli extended [11] by estimating autocorrelations, cross-correlations and statistical moments of the wavelet coefficients of the texture sample. Compared to the previous statistical attempts, convincing results are observable on a wide range of textures. This method, which represents the state of the art for psychophysically and statistically founded algorithms is nevertheless computationally demanding, 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.

Non-parametric patch-based methods constitute a different category of texture synthesis algorithms. They were initialized by Efros and Leung [5] who extended to images Shannon's Markov random field model initially devised to simulate text. In analogy with Shannon's algorithm for synthesizing sentences, the texture is constructed pixelwise. For each new pixel in the reconstructed image, a patch centered at the pixel is compared to all patches with the same size in the input sample. The nearest matches in the input help predict the pixel value in the reconstructed image. Several optimizations have been proposed to extend and accelerate this algorithm. Among them, Wei and Levoy [27], managed to fix the shape and the size of the learning patch and Ashikmin [2] proposed to extend existing patches whenever possible instead of searching in the entire sample texture. Still, these pixelwise algorithms are not always satisfactory. They are known to grow "garbage" when the compared patches are too small or when the input texture is not stationary, or may lead to verbatim copies of significant parts of the input sample for large patches as can be verified in [1]. More recent methods stitch together entire patches instead of performing a synthesis pixel by pixel. The question then is how to blend a new patch in the existing texture. In [18] this is done by a smooth transition. Efros and Freeman [6] refined this process by stitching each new patch along a minimum cost path across its overlapping zone with the texture under construction. Kwatra et al. in [14] extended the stitching procedure of [6] by a graph cut approach redefining the edges of the patches. In [13] the authors propose to synthesize a texture image by sequentially improving the quality of a synthesis by minimizing a patch-based energy function. In the same spirit as [13], where texture optimization is performed, the authors in [16] propose to synthesize textures in a multiscale framework using the coordinate maps of the sample texture at different scales. They introduce spatially randomness by applying a jitter function to the coordinates at each level combined to a correction step inspired by [2]. One of the key strengths of the method is that it is a parallel synthesis algorithm which makes it extremely fast. These non-parametric patch-based approaches often present satisfactory visual results. In particular they have the ability to reproduce highly structured textures. However, the risk remains of copying verbatim large parts of the input sample. 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 [26] for an extensive overview of the different non-parametric patch-based methods.

Recently methods such as [20,25] combine patch-based and statistics-based methods to overcome the drawbacks mentioned previously. In [20], the author proposes 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. In [25], Tartavel et al. extend the work of [20] by minimizing an energy that involves a sparse dictionary of patches combined to constraints on the Fourier spectrum of the input sample. Still more recently, Gatys et al. [10] have introduced the neural network methodology into the field. Extending the parametric approach, they characterize a texture by the cross-correlations of kernels learnt from a convolutional neural network dedicated to shape recognition. This algorithm emulates complex textures containing large objects for which the Portilla and Simoncelli algorithm is not satisfactory. Contrarily to the Portilla and Simoncelli approach, no Occam's razor was applied to reduce the number of texture parameters. The number of kernel correlations involved is perhaps exceedingly large: it appears that parts of the input are being recombined in the output. Nevertheless this learning approach is definitely promising.

In this work, an extension and detailed description of a texture synthesis framework is proposed which combines a multiscale approach to a locally Gaussian texture model in the patch space. Each texture patch of the synthesized image is sampled from its Gaussian distribution estimated on a set of similar patches taken in the input sample as proposed in [22]. Modeling patches with conditional Gaussian distributions is an approach that is also used in image processing as for instance in the denoising algorithm of Buades et al. [15]. The first question that was brought up was how to stitch together the patches. In [22], the Efros and Freeman's [6] was adopted, where every new patch overlaps the previously synthesized one, and the overlapped parts are blended. A more recent attempt considered conditional Gaussian models [23] constraining the simulated Gaussian patches to respect the values of its overlapping zone. This solution shows satisfying results for periodic or pseudo-periodic textures but fails for more complex ones. The second and central question brought up by patch-based methods is: how to handle the strong dependency of the method on the patch size in particular when dealing with macro-textures. Macro-textures show information at different scales that cannot be captured with a unique patch size. To this aim, a multiscale approach was sketched in [24] for which a full description and discussion is provided here.

The rest of this paper is structured as follows. In Section 2 the local Gaussian (LG) model is described. An analysis of the model's variance is provided as well as the description of a first synthesis method. In Section 3 two new patch models are introduced: the conditional local Gaussian model (CLG) and the regularized conditional local Gaussian model (RCLG). Section 4 presents the multiscale approach (MSLG). Section 5 shows several experiments: a comparison of the three patch models proposed, a comparison to the results of state of the art texture synthesis methods and the influence of the parameters involved in the multiscale texture synthesis algorithm. Conclusions are presented in Section 6.

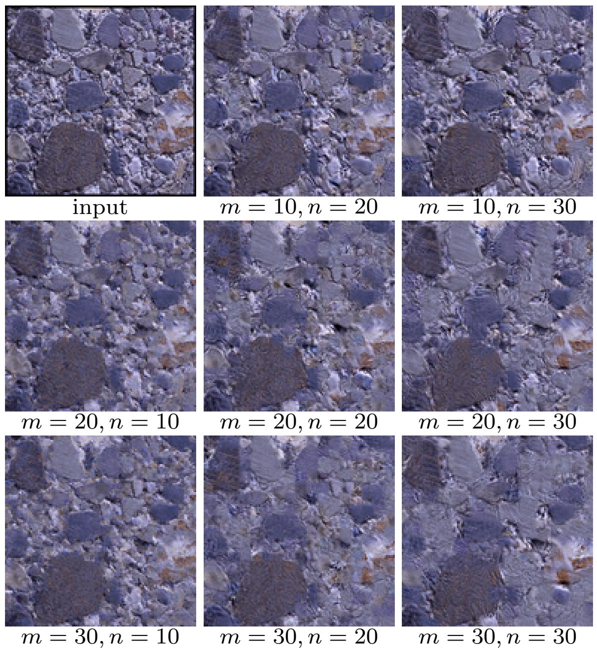

Fig. 1: Visual evaluation of the validity of a Gaussian patch model. This experiment performs a Gaussian substitution for each patch of the left top corner texture image (Algorithm 1). From left to right the patch size is $n = 10, 20, 30$. From top to bottom the number of nearest neighbours size is $m = 10, 20, 30$.

2 Gaussian patches

For all notations used in this section we refer to Table 1 for a detailed definition.

2.1 The assumption of a Gaussian patch model

For a given texture image u, the underlying distribution of every patch $p_u^{(x,y)}$ is modeled as a multiriate Gaussian distribution (LG) of mean $\mu_{(x,y)}$ and covariance matrix $\Sigma_{(x,y)}$. To validate this assumption a set of overlapping patches of u are replaced by samples of the corresponding distributions as explained in Algorithm 1. This set of patches covers the input texture u and for the overlap regions no blending technique is used. Each simulated patch overwrites the previous values along the overlap area. The resulting image $\tilde{u}$ is a plausible reconstruction of u as can be observed in Figure 1.

The parameters $\mu_{(x,y)}$ and $\Sigma_{(x,y)}$ of the Gaussian distribution of the patch $p_u^{(x,y)}$ are estimated from the set of nearest patches $\mathcal{U} = \{p_u^{(x_i,y_i)}, i = 1, \ldots, m\}$. Here $p_u^{(x_i,y_i)}$ are the m nearest patches to $p_u^{(x,y)}$ in u according to the L^2 distance. The empirical statistics $\mu_{(x,y)}$ and $\Sigma_{(x,y)}$ are then defined as

$$\begin{aligned}\mu_{(x,y)} &= \tfrac{1}{m}\textstyle\sum_{i=1}^{m} p_u^{(x_i,y_i)} \\ \Sigma_{(x,y)} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{m}\left(p_u^{(x_i,y_i)}-\mu_{(x_i,y_i)}\right)\left(p_u^{(x_j,y_j)}-\mu_{(x_j,y_j)}\right)^t}{(m-1)}.\end{aligned} \tag{1}$$

To simplify the notation, we denote by P the matrix whose columns are the normalized patches in vector form $(p_u^{(x_i,y_i)} - \mu_{(x,y)}),\ i = 1, \ldots, m$. Then we can reformulate the covariance matrix $\Sigma_{(x,y)}$

$$\Sigma_{(x,y)} = \frac{1}{(m-1)} PP^t.$$

Sampling a vector $\tilde{p}_u^{(x,y)} \sim \mathcal{N}(\mu_{(x,y)}, \Sigma_{(x,y)})$ comes down to sampling m independent normal variables as

$$\tilde{p}_u^{(x,y)} = \frac{1}{(m-1)} PWDp' + \mu_{(x,y)}. \tag{2}$$

Here $p' \sim \mathcal{N}(\mathbf{0}, I_m)$, W is a matrix whose columns are the eigenvectors of P^tP and D is a diagonal matrix with its eigenvalues.

The result of using local Gaussian distributions is illustrated in Figure 1. Observe that replacing the patches

u	$\Omega \to \mathbb{R}$: input texture image defined on the discrete domain $\Omega = I_M \times I_N$ of size $M \times N$ where I_c denotes the discrete interval $[0, \dots, c-1]$
w	$\Omega_r \to \mathbb{R}$: output texture image defined on the discrete domain $\Omega_r = I_{rM} \times I_{rN}$ of size $rM \times rN$
d	is equal to 1 if u is grayscale and to 3 for color images
r	ratio (size of output image w)/(size of input image u)
n	side patch size
m	number of nearest neighbours used to learn the parameters of the Gaussian distribution
o	overlap size
K	number of scales (maximum factor of zoom out is $K-1$)
u_k	$\Omega^k \to \mathbb{R}$: zoom out of u of a factor 2^k, defined on the discrete domain $\Omega^k = I_{2^{-k}M} \times I_{2^{-k}N}$ of size $2^{-k}M \times 2^{-k}N$ for $k = 1 \dots K-1$
w_k	$\Omega_r^k \to \mathbb{R}$: synthesized texture at scale k, defined on the discrete domain $\Omega_r^k = I_{r2^{-k}M} \times I_{r2^{-k}N}$ of size $r2^{-k}M \times r2^{-k}N$ for $k = 0 \dots K-1$
v_k	$\Omega_r^k \to \mathbb{R}$: zoom in of w_{k+1} of factor 2. It is the initialization of the low resolution of the synthesized image w_k for $k = 0 \dots K-2$
$p_u^{(x,y)}$	square patch of size $n \times n$ from an image u of size $M \times N$ at position (x,y), $p_u^{(x,y)} = \{u((x,y)+(i,j)),\ (i,j) \in [0,\dots,n-1]^2\}$, $(x,y) \in \mathcal{V}_u$ where $\mathcal{V}_u = I_{M-n+1} \times I_{N-n+1}$ denotes the discrete domain of the valid patches in the image u $p_u^{(x,y)}$ will be taken as a column vector of size $dn^2 \times 1$
G_σ	Gaussian kernel of mean zero and standard deviation σ, $G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$
L_{u_k}	$\Omega^k \to \mathbb{R}$: low resolution of image u_k, $L_{u_k} = u_k * G_\sigma,\ k = 0 \dots K-2$
H_{u_k}	$\Omega^k \to \mathbb{R}$: high resolution of image u_k, $H_{u_k} = u_k - u_k * G_\sigma,\ k = 0 \dots K-2$
L_{w_k}	$\Omega_r^k \to \mathbb{R}$: low resolution of image w_k, $L_{w_k} = w_k * G_\sigma,\ k = 0 \dots K-2$
H_{w_k}	$\Omega_r^k \to \mathbb{R}$: high resolution of image w_k, $H_{w_k} = w_k - w_k * G_\sigma,\ k = 0 \dots K-2$

Table 1: Summary of the notations used in this article.

Algorithm 1 Gaussian model validation

```
Input: input texture sample $u$, side patch size $n$, number of nearest neighbours $m$, overlap size $o$
Output: reconstructed texture $\tilde{u}$
1: for $x = 1 : n - o : M$ do
2:     for $y = 1 : n - o : N$ do
3:         Compute $\mathcal{U}$ the set of $m$ closest patches to $p_u^{(x,y)}$
4:         Estimate $\mu_{(x,y)}$ and $\Sigma_{(x,y)}$
5:         Sample $\tilde{p}_u^{(x,y)} \sim \mathcal{N}(\mu_{(x,y)}, \Sigma_{(x,y)})$
6:         $\tilde{u}(x+i, y+j) \leftarrow \tilde{p}_u(x,y),\ \forall (i,j) \in I_n^2$
7:     end for
8: end for
9: return $\tilde{u}$
```

of the texture by simulated ones with the corresponding Gaussian model achieves a faithful random variant of the original image for reasonable values of n and m. All pixel values of u and $\tilde{u}$ are actually different. For large values of m the model no longer represents the patch faithfully because $\mathcal{U}$ will contain outliers. Large values of n limits m for the same reason. Nevertheless, reasonable values of n and m ensure a correct reconstruction when replacing patches by others simulated from a Gaussian model.

To simplify the notations when referring to $\mu_{(x,y)}$ and $\Sigma_{(x,y)}$ they will be denoted by μ and Σ respectively.

The model's variance It is important to analyze how the variance of the Gaussian distribution varies with the patch size n and the number of nearest neighbours m used to estimate the Gaussian distributions. We expect to have a variance that is not equal to zero since this would correspond to taking the best match in u and thus having no innovation at all. On the other hand a high variance would introduce too much variability and therefore end up with a blurry reconstruction of the texture. The ranges for m and n yielding a correct estimate for the variance cannot be determined *a priori* and strongly depend on the texture sample. To measure the variability of the Gaussian model the mean standard deviation per pixel of a given patch in the texture image can be estimated by

$$\bar{\sigma} = \frac{1}{dn^2} \sum_{i=1}^{dn^2} \sqrt{\Sigma(i,i)} = \frac{1}{dn^2} \sum_{i=1}^{dn^2} \sigma_i$$

where d is the number of channels(d=1 for grayscale images and $d = 3$ fop color images).

In Figure 2, two texture examples are presented. As expected for a fixed patch size n the mean standard deviation per pixel $\bar{\sigma}$ increases when using more patches to estimate the Gaussian distribution. For a fixed value of m the same behaviour is observed when increasing n. The first example, a micro texture, shows that the variance of the patch is not negligible even for $m = 5$. Gaussian models are well suited for this kind of texture. The second example, a periodic texture, shows that the patch mean standard deviation varies between 8 and 18 for values of $m \in \{5, 10, 30, 50\}$. These are reasonable values confirming that effectively the patches simulated have an acceptable degree of innovation.

2.2 Texture synthesis algorithm

In this section a texture synthesis algorithm is presented using the Gaussian sampling described in the previous section. Given the input texture u, the output image w is synthesized sequentially patch by patch in a raster-scan order (left to right, top to bottom). Each new patch added to w overlaps part of the previously synthesized patch as can be seen in Figure 3. Depending on the stage of the synthesis three different cases of overlap can be observed: vertical (first row of raster-scan), horizontal (first column of raster-scan) and L-shape (everywhere else). This is also shown in Figure 3. Each patch is simulated following a multivariate Gaussian distribution of mean μ and covariance matrix Σ as defined in (1). When quilting the patch in w the same stitching step is applied as in Efros and Freeman's work [6] where the authors propose to compute an optimal boundary between the new patch and the previously synthesized one along their overlap region. This is done thanks to a linear programming optimization.

To define the set $\mathcal{U} = \{p_u^{(x_i,y_i)}, i = 1, \ldots, m\}$ of nearest patches to $p_w^{(x,y)}$, let us consider $p_w^{(x,y)}$ as the patch being currently synthesized and taken as a column vector of size $dn^2 \times 1$. The patch $p_w^{(x,y)}$ will overlap part of the previous synthesis. To synthesize $p_w^{(x,y)}$, we decompose it as

$$p_w^{(x,y)} = S^t S p_w^{(x,y)} + R^t R p_w^{(x,y)}. \tag{3}$$

Here, $S : \mathbb{R}^{dn^2} \to \mathbb{R}^{dn^2-k}$ and $R : \mathbb{R}^{dn^2} \to \mathbb{R}^k$ are projection operators such that $Rp_w^{(x,y)}$ is a vector of size $k \times 1$ with the values of $p_w^{(x,y)}$ on the overlap area and $Sp_w^{(x,y)}$ is a vector of size $(dn^2 - k) \times 1$ with the other components of $p_w^{(x,y)}$. This decomposition will be useful in the following section.

The patches $p_u^{(x_i,y_i)}$ used to learn the parameters of the multivariate Gaussian distribution (1) are the m nearest neighbours in u to the current patch $p_w^{(x,y)}$, for the L^2 distance restricted to the overlap area, given by $\|Rp_u^{(x_i,y_i)} - Rp_w^{(x,y)}\|_2$. Once the patch $p_w^{(x,y)}$ is synthesized from the Gaussian model (1), the values of $Rp_w^{(x,y)}$ change. The texture synthesis algorithm is summarized in Algorithm 2.

In Figure 4 the synthesis results using the Gaussian sampling are compared to those of the quilting method [6] to illustrate that the Gaussian sampling achieves visual results that are comparable to those in [6] while providing a local parametric model. With the Gaussian sampling some blur is introduced in the synthesis results but the effects of verbatim copies and garbage growing are reduced as can be observed in the second example in Figure 4. For a more extensive comparison please we refer to [22] where the Gaussian sampling is compared to several synthesis texture algorithms.

Algorithm 2 Texture Synthesis

```
Input: input texture sample $u$, side patch size $n$, overlap
    size $o$, number of nearest neighbours $m$, ratio (output
    size)/(input size) $r$
Output: synthesized texture $w$
 1: Initialize $w$ placing a seed patch in its top-left corner ($x =$
    $1, y = 1$). The image $w$ is of size $rM \times rN$, where $M \times N$
    is the size of $u$.
 2: for $x = 1 : n - o : rM$ do
 3:    for $y = 1 : n - o : rN$ do
 4:       if ($x > 1$ or $y > 1$) then
 5:          $\mathcal{U} \leftarrow$ set of $m$ nearest neighbours to $p_w^{(x,y)}$
 6:          $(\mu, \Sigma) \leftarrow$ Estimate the parameters of the multi-
             avriate Gaussian distribution on $\mathcal{U}$
 7:          Sample $\tilde{p}_w^{(x,y)} \sim G(\mu, \Sigma)$
 8:          Quilt $\tilde{p}_w^{(x,y)}$ in $w$ at position $(x, y)$
 9:       end if
10:    end for
11: end for
12: return $w$
```

2.3 A discussion on the underlying mathematical model

The underlying random field of a given input texture is assumed to be a Gaussian random field. Indeed, as illustrated with the experiment in Figure 1, a patch in the input is assumed to be a multivariate Gaussian vector. Furthermore, we assume the existence of a set of other patches following the same probability distribution, from which the parameters (mean and covariance matrix) are estimated. Thus the resulting reconstructed image is a sample of a Gaussian random field. Nevertheless, the same cannot be stated for the underlying random field of the synthesized textures when using Algorithm 2. These generated textures are not samples of a Gaussian random field. What can be affirmed is that the conditional distribution of the patches are Gaussian, meaning that if $p_w^0, p_w^1, p_w^2 \ldots$ are the overlapping patches of the image taken in a raster scan order, then the conditional probability $\mathbb{P}(p_w^n | p_w^{n-1})$ is a Gaussian multivariate distribution. However, the mean and the covariance matrix of this Gaussian distribution are computed from patches similar to p_w^{n-1} in the input sample. Therefore the joint probability distribution of the patches is not necessarily Gaussian.

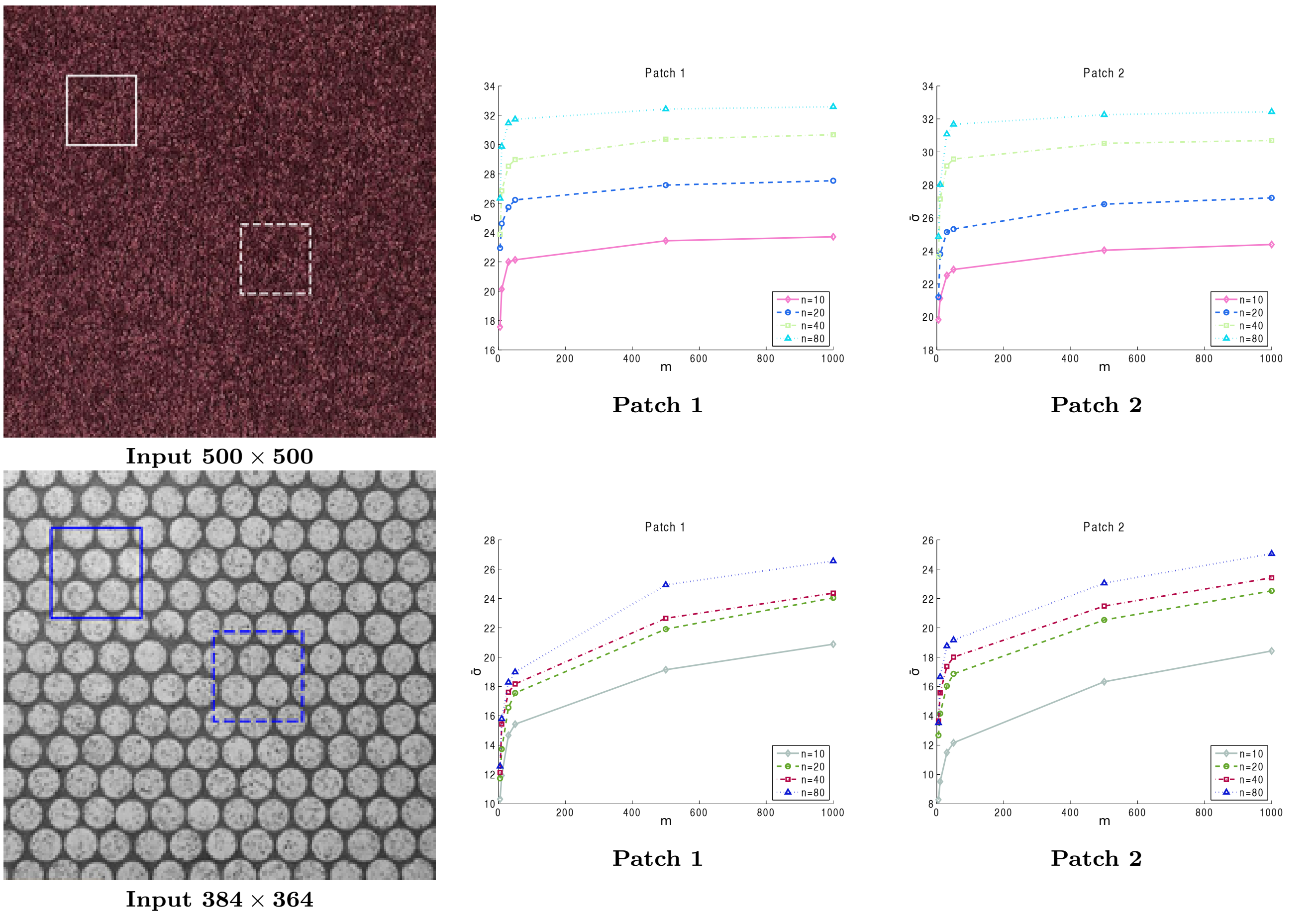

Fig. 2: Behaviour of the mean standard deviation of the patches as functions of the patch size n and the neighbourhood size m. For each texture example two patches have been selected (left column). For each of them the mean standard deviation is computed for several values of $n \in \{10, 20, 40, 80\}$ and $m \in \{5, 10, 30, 50, 500, 1000\}$. In the input image patch 1 is represented by a solid line and patch 2 by a dashed line.

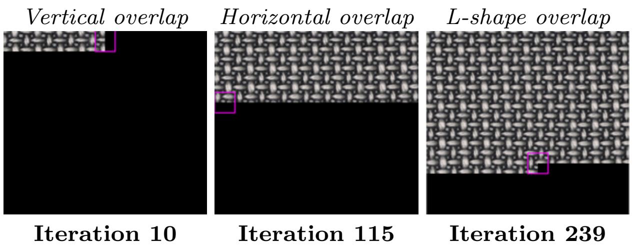

Fig. 3: Three different iterations of the synthesis process are shown. At each iteration a patch is being synthesized. This patch is represented by the pink square in the three iterations shown. From left to right the three overlap cases are represented: vertical, horizontal and L-shape.

Thus, while the algorithms are effective, they do not imply so far the existence of a complete texture model. Indeed, the input sample is characterized by a Gaussian distribution for each patch but also by *spreads*, defined as the spatial distribution of the patches belonging to the same Gaussian model. For some periodic textures the spread is deterministic (like in a chessboard for instance) but for general textures the spread itself is random. To model the whole texture as a random field a stochastic model for the spreads would therefore be necessary. For example, if the spread of the input sample is given and the patches of the generated texture are samples of conditional Gaussian distributions respecting exactly the same spread as the input's, then the underlying random field of the whole synthesized texture is Gaussian (as in the example of Figure 1). Nevertheless, to synthesize textures this is not very interesting in terms of variability among the different sampled textures. Another meaningful example where the model is complete is the case of periodic textures where the spread is deterministic. For a fixed seed patch (the one that initializes the generated texture) the underlying random field of the generated textures is a pe-

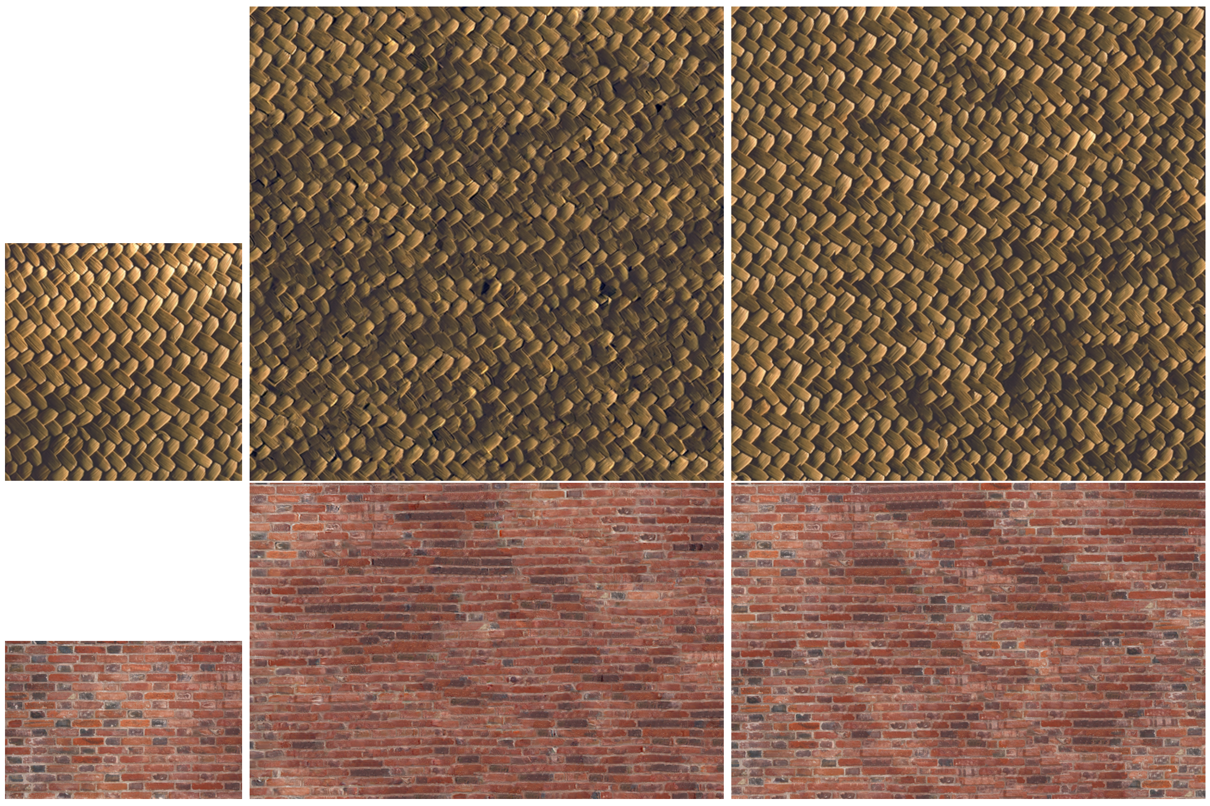

Fig. 4: Comparison of LG to the quilting method [6]. These results are done taking equal parameters and seed patch to initialize the synthesized imaged. The parameters used are $n = 40$, $o = 0.5$ and $m = 30$ (for the LG model).

riodic Gaussian random field. But this is no longer the case when the seed patch is random.

To summarize, we cannot claim that the proposed synthesis algorithm corresponds to a complete texture model, because it would require a stochastic model for the *spread*, namely the spatial interaction between patches obeying the same Gaussian model. This is left for future investigation, perhaps in the spirit of [17].

3 Conditional Gaussian patches

The purpose of this section is to analyze how to replace the stitching step by a conditional patch model on the overlapping zone. The natural idea is to condition the new Gaussian samples to the ones observed in the overlap zone, thus allowing to maintain the same or very only slightly the pixel in this overlap pixels.

This requires a new patch model to model the transition effect between patches. Each new patch will be estimated as a Gaussian vector conditioned to the pixel values of its corresponding overlap region. In this way the simulated patch would naturally "agree" with w in the overlap area, thus avoiding a stitching procedure.

Such models will be considered in the following sections.

3.1 Conditional Gaussian model

In this first pathc model the idea is to model sample patches from a Gaussian distribution that exactly match with their overlap area pixels, avoiding the unwanted discontinuities. We shall call this model Conditional Locally Gaussian (CLG).

Each patch $p_w^{(x,y)}$ is taken as a column vector of size $dn^2 \times 1$ and can be partitioned as expressed in (3).

We assume throughout that the vector $p_w^{(x,y)}$ follows a Gaussian distribution of mean μ and covariance matrix Σ. These parameters can be expressed as follows :

$$\mu = S^t S\mu + R^t R\mu = S^t\mu_1 + R^t\mu_2$$
and
$$\begin{aligned}\Sigma &= S^tS\Sigma S^tS + S^tS\Sigma R^tR + R^tR\Sigma S^tS + R^tR\Sigma R^tR\\ &= S^t\Sigma_{1,1}S + S^t\Sigma_{1,2}R + R^t\Sigma_{2,1}R + R^t\Sigma_{2,2}R.\end{aligned}$$

The problem can be formulated as finding the "best sample" $\tilde{p} = (\tilde{p}_1, \tilde{p}_2)$ conditioned to the overlap values y_0 that are known,

$$\tilde{p}_1 = \arg\max_{p_1} \mathbb{P}_{\mu,\Sigma}(P_1 = p_1 \mid P_2 = y_0). \tag{4}$$

Here P_1 corresponds to the unknown values of $p_w^{(x,y)}$ and P_2 to the values of $p_w^{(x,y)}$ on the overlap area, i.e. $P_1 = Sp_w^{(x,y)}$ and $P_2 = Rp_w^{(x,y)}$.
Using classic results on conditional multivariate Gaussian distributions [19], the distribution of P_1 conditioned to $P_2 = y_0$ is a multivariate Gaussian distribution of parameters $\bar{\mu}$ and $\bar{\Sigma}$ where

$$\bar{\mu} = \mu_1 + (S\Sigma R^t)(R\Sigma R^t)^{-1}(y_0 - \mu_2)$$
and
$$\bar{\Sigma} = (S\Sigma S^t) - S\Sigma R^t(R\Sigma R^t)^{-1}R\Sigma S^t.$$

Since the most probable sample of a multivariate Gaussian distribution is its mean, the solution to (4) is

$$\tilde{p}_w = R^t y_0 + S^t(\mu_1 + (S\Sigma R^t)(R\Sigma R^t)^{-1}(y_0 - \mu_2)). \tag{5}$$

This first result has the advantage that since the mean of the conditional distribution is taken, the risk of unwanted sharp transition between patches for some types of textures will be attenuated. Another observation is that since this new model might not have very probable samples, working with the most probable one improves the visual aspect of the sampled patch. The negative aspects of taking the most probable sample is that on the one hand the result, for a given initialization, is deterministic. On the other hand the synthesis result loses resolution by displaying the mean of the conditional distribution.

An alternative to overcome these two drawbacks is to sample a patch from the underlying Gaussian distribution, that is,

$$\tilde{p} = R^t y_0 + S^t\tilde{p}_1,\ \tilde{p}_1 \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma}). \tag{6}$$

Yet equations (5) and (6) show that these solutions do not make sense if $(R\Sigma R^t)$ is not invertible. This unfortunately is frequent, as the number of neighbours m used to build the Gaussian distribution is often very small compared to the dimension of the vectors we aim to model. Therefore the learnt Gaussian models are strongly degenerated.
The fact that Σ is not invertible implies that the Gaussian vectors $p_w^{(x,y)} \sim \mathcal{N}(\mu, \Sigma)$ live in a subspace of $\mathbb{R}^{dn^2}$. This leads to the following alternatives:

1. The Gaussian vectors subspace intersects the set of Gaussian vectors $R^t y_0 + S^t p_1$. Thus the Gaussian distribution $\mathcal{N}(\bar{\mu}, \bar{\Sigma})$ exists and there is a solution to our problem.
2. There is no intersection and in that case there is no solution.

To overcome the fact that we may have no solution a small perturbation is added to the Gaussian distribution learnt for $p_w^{(x,y)}$ as follows

$$p_w^{(x,y)} \sim \mathcal{N}(\mu, \Sigma + \sigma^2 I_{dn^2}),$$

where σ^2 is a real positive number and I_{dn^2} is the identity matrix of size $dn^2 \times dn^2$ ensuring that the problem is well conditioned. In that way the Gaussian vectors $p_w^{(x,y)}$ live in $\mathbb{R}^{dn^2}$, and this ensures the existence of the conditional multivariate Gaussian distribution sought. The new covariance matrix is denoted by $\Gamma = \Sigma + \sigma^2 I_{dn^2}$.

If we are in the case where the Gaussian vectors subspace $p_w^{(x,y)} \sim \mathcal{N}(\mu, \Sigma)$ intersects the set of Gaussian vectors $R^t y_0 + S^t p_1$, using the new distribution $\mathcal{N}(\mu, \Gamma)$ will slightly modify the solutions in (5) and (6) when using small values of σ^2. Thus it is enough to take a low value for σ^2 and the solutions obtained in both cases ($\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu, \Gamma)$) will be very close to each other.

Adding this perturbation to the multivariate Gaussian model has the drawback of increasing the computational cost of sampling a Gaussian vector. For the model in Section 2.1 a Gaussian vector was sampled computing the eigenvectors and eigenvalues of an $m \times m$ matrix. The model presented in this section requires the computation of the eigenvectors and eigenvalues of an $dn^2 \times dn^2$ matrix.

3.2 Regularized conditional Gaussian model

In the previous section the patches' statistical model was conditioned to the exact values of the synthesized pixels across the overlap area. This is too restrictive for some types of textures and is at risk of creating unlikely samples. Instead of forcing each patch $p_w^{(x,y)}$ to take the exact same values on the previously synthesized part, it is therefore natural to allow the patch to vary slightly on the overlap area. This variation is rendered necessary by the scarcity of patch samples in a small texture sample. We shall call this model Regularized Conditional Locally Gaussian (RCLG). Consider the same patch model $\mathcal{N}(\mu, \Sigma)$, but let us now allow the overlap components $P_2 = Rp_w^{(x,y)}$ to take values $P_2 = y_0 + n$, where $n \sim \mathcal{N}(0, \theta^2 I_k)$ and where θ is the degree of variation allowed for the overlap values. The joint distribution of (P_1, P_2) is defined by:

$$\begin{aligned}\mathbb{P}(p_1,p_2) =&\mathbb{P}_{\mu,\Sigma}(p_1|p_2)\mathbb{P}_{y_0,\theta^2 I_k}(p_2)\\ =&\kappa e^{-\frac{1}{2}((p_1,p_2)-(\bar{\mu},y_0))^t \Delta^{-1}((p_1,p_2)-(\bar{\mu},y_0))},\end{aligned} \tag{7}$$

where $\kappa = \frac{1}{\sqrt{2\pi|\Delta|}}$ and $\Delta = \begin{pmatrix} \bar{\Sigma} & \mathbf{0} \\ \mathbf{0} & \sigma^2 I_{k\times k}\end{pmatrix}$.

Proceeding as in the previous model (CLG) the most probable sample defined by (8) is exactly the same as in (5) and has the same drawbacks. On the other hand sampling from (7) allows to relax the overlap constraint.

$$\begin{aligned}\tilde{p} =&(\tilde{p}_1, \tilde{p}_2)\\ =&\arg\max_{(p_1,p_2)} \mathbb{P}_{\mu,\Sigma}(P_1 = p_1|P_2 = p_2)\mathbb{P}_{0,\theta^2 I_k}(p_2 - y_0)\end{aligned} \tag{8}$$

Once again, to guarantee the existence of the solution, the Gaussian distribution of $p_w^{(x,y)}$ is slightly modified as done in the CLG model. A small perturbation is added to the covariance matrix Σ and the problem is then well conditioned. Thus, as in the CLG model, the computational cost of sampling from this new Gaussian distribution is higher compared to the LG sampling.

4 A multiscale algorithm

For all notations used in this section we refer to Table 1 for a detailed definition.

Macro textures present details at different scales: a coarse one that contains the global structure of the texture and finer ones containing the details. Small patch sizes may capture the finer details of the input but the resulting texture will lack global coherence. On the other hand using large patches will maintain better the global structures on the risk of a “copy-paste” effect. Furthermore with large patches it becomes impossible to model the patch variability due to the curse of dimensionality, in other terms the lack of sufficient samples. This is for example apparent in [22] where modeling patches as multivariate Gaussian vectors leads to a slightly blurry texture.

Multiscale approaches permit to contemplate several patch sizes within one synthesis, i.e. to capture the different levels of details. If we fix the patch size to be $n \times n$ and use K scales this is similar to using K different patch sizes going from $2^{K-1}n \times 2^{K-1}n$ to $n \times n$ for the coarsest to finer details within one synthesis.

In this section the potential of a multiscale approach is illustrated by improving the method described in Algorithm 2. Let us first introduce some notations. The input texture sample is denoted by u and u_k, $k = 1, \dots, K-1$ are the zoomed out versions of u by a factor 2^k, $k = 1, \dots, K-1$. The synthesis result at each scale is denoted by w_k, $k = 1, \dots, K-1$ and w is the synthesis result returned by the multiscales algorithm. An additional image needed at each scale is the low resolution of the result w_k that is denoted by v_k and its the result of zooming in w_{k+1}. The multiscale approach can be summarized in a few sentences. The method begins by a synthesis at the coarsest scale ($k = K-1$) using the local Gaussian method in Algorithm 2 where the quilting step is replaced by a simple average of the overlapping patches. For the remaining scales ($k = K-2, \dots, 0$) a synthesis is performed by using the result of the previous scale ($k+1$) and the input of 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 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.

We shall call this method Multiscale Locally Gaussian (MSLG). In the following the different parts of the method described in Algorithm 3 are detailed.

Zoom out The zoom out operation is a Gaussian zoom out. It is performed as a smooth frequency cutoff followed by a sub-sampling of factor 2. These operations are detailed below in (9), (10) and (11).

The smooth frequency cutoff is performed with the Gaussian kernel

$$G_\sigma(x,y) = \frac{1}{\sigma^2 2\pi} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{9}$$

where we chose $\sigma = 1.4$.
The sub-sampling operation by a factor 2 applied to an image $u : I_M \times I_N \mapsto \mathbb{R}$ is defined by

$$\begin{aligned}&\mathcal{S}_2 : I_M \times I_N \mapsto I_{M/2} \times I_{N/2},\\ &\mathcal{S}_2(u)(i,j) = u(2i, 2j),\ (i,j) \in I_{M/2} \times I_{N/2}.\end{aligned} \tag{10}$$

The zoom out of an image u of factor 2 is then

$$\mathcal{Z}_2^{\text{out}}(u) = \mathcal{S}_2(u * G_\sigma). \tag{11}$$

The images obtained after applying this zoom out operation have no aliasing or ringing artifacts and they are blurry enough to avoid ringing artifacts when applying the zero padding zoom in.

Zoom in For an image $u : I_M \times I_N \to \mathbb{R}$, $v = \mathcal{Z}_2^{\text{in}}(u) : I_{2M} \times I_{2N} \to \mathbb{R}$ is the zoom in by a factor 2. This operation is performed by a zero padding of $\hat{u} : \hat{I}_{2M} \times \hat{I}_{2N} \to \mathbb{C}$ where $\hat{u}$ denotes the discrete Fourier transform of u and $\hat{I}_c$ denotes the discrete interval $[-c/2, \dots, c/2-1]$ for c even and $[-(c-1)/2, \dots, (c-1)/2]$ for c odd. This is done as

$$\hat{v}(\xi_1, \xi_2) = \begin{cases} \hat{u}(\xi_1, \xi_2) & \text{if } |\xi_1| \le \left\lfloor \frac{M-1}{2} \right\rfloor,\ |\xi_2| \le \left\lfloor \frac{N-1}{2} \right\rfloor \\ 0 & \text{else} \end{cases} \tag{12}$$

where $\hat{v}$ is the discrete Fourier transform of v and $\lfloor x \rfloor$ denotes the integer part of x.

The relation between the zoom in and zoom out operators

$$\mathcal{Z}_2^{\text{in}}(\mathcal{Z}_2^{\text{out}}(u)) = u$$

is valid when the image u verifies

$$\hat{u}(\xi_1, \xi_2) = 0,\ \forall |\xi_1| > \left\lfloor \frac{M-1}{2} \right\rfloor, |\xi_2| > \left\lfloor \frac{N-1}{2} \right\rfloor.$$

We could have chosen other interpolation techniques as for example a spline interpolation. But a zero padding is well suited due to the nature of the zoomed out images which are blurry enough to avoid any ringing artifacts.

Distance between patches To estimate the parameters of the Gaussian distribution of the patch being processed, denoted by $p_{w_k}^{(x',y')}$, the set $\mathcal{U}$ of m nearest patches in u_k to $p_{w_k}^{(x',y')}$ is considered. These patches are those minimizing the distance to $p_{w_k}^{(x',y')}$ defined in (13) for $k = K-1$ and in (14) for the remaining scales $k = K-2, \dots, 0$.

The size of patch overlap is fixed to half the patch size $n/2$. As mentioned in Section 2 there are three overlap cases: vertical (V.O.), horizontal (H.O.) and L-shape (L.O.). Here they are denoted as

$$Op_u^{(x,y)} = \{u\left((x,y) + (i,j)\right),\ (i,j) \in \mathcal{O}\}$$

where

$$\mathcal{O} = \begin{cases} [0, \dots, n-1] \times \left[0, \dots, \frac{n}{2} - 1\right] & \text{if V.O.} \\ \left[0, \dots, \frac{n}{2} - 1\right] \times [0, \dots, n-1] & \text{if H.O.} \\ \left[0, \dots, \frac{n}{2} - 1\right] \times [0, \dots, n-1] \cup & \\ \left[\frac{n}{2}, \dots, n-1\right] \times \left[0, \dots, \frac{n}{2} - 1\right] & \text{if L.O.} \end{cases}$$

When $k = K-1$, the m nearest neighbours in u_{K-1} to the current patch $p_{w_{K-1}}^{(x',y')}$ are those minimizing the L^2 distance restricted to the overlap area:

$$d(Op_{u_{K-1}}^{(x,y)}, Op_{w_{K-1}}^{(x',y')})^2 = \frac{1}{|\mathcal{O}|} \sum_{(i,j) \in \mathcal{O}} (u_{K-1}(x+i, y+j) - w_{K-1}(x'+i, y'+j))^2. \tag{13}$$

When $k = K-2, \dots, 0$, the nearest neighbours in u_k to the patch $p_{w_k}^{(x',y')}$ are those minimizing a distance (14) similar to (13) with an additional term taking into account the low resolution v_k (the synthesis result of the previous scale $k+1$). In (14) L_{u_k} denotes the low resolution of the image u_k, $L_{u_k} = u_k * G_\sigma,\ k = 0, \dots, K-2$. Is is important to notice that when comparing $Op_{u_k}^{(x,y)}$ and $Op_{w_k}^{(x',y')}$ the low and the high resolution must be considered jointly, they are not independent.

$$d(p_{u_k}^{(x,y)}, p_{w_k}^{(x',y')})^2 = \frac{1}{|\mathcal{O}|} \sum_{(i,j) \in \mathcal{O}} (u_k(x+i, y+j) - w_k(x'+i, y'+j))^2 + \frac{1}{n^2} \sum_{i,j=0}^{n-1} (L_{u_k}(x+i, y+j) - v_k(x'+i, y'+j))^2 \tag{14}$$

Blending process The blending process consists in simply averaging the values across the overlap area as in (15). This step is applied only for the synthesis of scale $k = K-1$.

$$w_k(x+i, y+j) = \begin{cases} \frac{1}{2}(\tilde{p}_{w_{K-1}}^{(x,y)}(i,j) + p_{w_{K-1}}^{(x,y)}(i,j)) & \text{if } (i,j) \in \mathcal{O} \\ \tilde{p}_{w_{K-1}}^{(x,y)}(i,j) & \text{if } (i,j) \in I_n^2 - \mathcal{O} \end{cases} \tag{15}$$

Synthesizing patches at scales $k = K-2, \dots, 0$ At each scale k a patch $p_{w_k}^{(x,y)}$ is synthesized as the combination of a low resolution patch with a high resolution one. It can be decomposed as

$$\begin{aligned} p_{w_k}^{(x,y)} &= p_{w_k*G_\sigma}^{(x,y)} + (p_{w_k}^{(x,y)} - p_{w_k*G_\sigma}^{(x,y)}) \\ &= p_{v_k}^{(x,y)} + (p_{w_k}^{(x,y)} - p_{v_k}^{(x,y)}) \\ &= p_{L_{w_k}}^{(x,y)} + p_{H_{w_k}}^{(x,y)}. \end{aligned}$$

Here L_{w_k}denotes the low resolution image of w_k defined as $L_{w_k} = w_k * G_\sigma,\ k = 0 \dots K-2$ and H_{w_k} denotes the high resolution image of w_k defined as $H_{w_k} = w_k - w_k * G_\sigma,\ \ k = 0 \dots K-2$. The set $\mathcal{U}$ defines the Gaussian distribution of $p_{L_{w_k}}^{(x,y)} \sim \mathcal{N}(\mu_L, \Sigma_L)$ and $p_{H_{w_k}}^{(x,y)} \sim \mathcal{N}(\mu_H, \Sigma_H)$ and therefore the distribution of the patch $p_{w_k}^{(x,y)} \sim \mathcal{N}(\mu, \Sigma)$ where

$$\mu = \mu_H + \mu_L$$

and

$$\Sigma = \Sigma_L + \Sigma_H + \mathbb{E}(p_{L_{w_k}}^{(x,y)}(p_{H_{w_k}}^{(x,y)})^t) + \mathbb{E}(p_{H_{w_k}}^{(x,y)}(p_{L_{w_k}}^{(x,y)})^t).$$

Instead of sampling $p_{L_{w_k}}^{(x,y)}$ from its Gaussian distribution, $p_{v_k}^{(x,y)} \sim \mathcal{N}(\mu_L, \Sigma_L)$ is kept to conserve the low resolution synthesis from the previous scale. The high frequency patch $p_{H_{w_k}}^{(x,y)}$ is sampled form $\mathcal{N}(\mu_H, \Sigma_H)$ and then added to $p_{v_k}^{(x,y)}$. In this way the correlations between high and low resolution pixels are respected, using the low resolution synthesis v_k as initialization.

For all scales beside the coarsest one the synthesis is done by adding the high frequencies of the corresponding scale on the the low resolution basis image. It is the important to achieve a correct basis image in the coarsest scale on which the high frequencies will be added. The texture synthesis method is summarized in Algorithm 3.

Algorithm 3 Multiscale texture syntesis

```
Input: input texture sample $u$, side patch size $n$, number of
    nearest neighbours $m$, number of scales $K$, ratio (output
    size)/(input size) $r$
Output: synthesized texture $w$
 1: Define $u_k \leftarrow \mathcal{Z}_2^{\text{out}}(u_{k-1}),\ k = 1 \dots K-1$
 2: Define $L_{u_k} \leftarrow u_k * G_\sigma,\ k = 0 \dots K-2$
 3: Synthesize $w_{K-1} \leftarrow \text{Synth0}(u_{K-1}, n, m, r)$
 4: for $k = K-2 : 0$ do
 5:   $v_k \leftarrow \mathcal{Z}_2^{\text{in}}(w_{k+1})$
 6:   Initialize $w_k$ with zeros of same size as $v_k$
 7:   for $x = 1 : n/2 : (r2^{-k}M - n + 1)$ do
 8:     for $y = 1 : n/2 : (r2^{-k}N - n + 1)$ do
 9:       Compute $d(p_{u_k}^{(x',y')}, p_{w_k}^{(x,y)})$, for all $(x',y')$ in $\mathcal{V}_{u_k}$
10:       $\mathcal{U} \leftarrow \{p_{u_k}^{(x_i,y_i)},\ i = 1, \dots, m\}$ set of $m$ nearest
          patches in $u_k$ that minimize $d(p_{u_k}^{(x',y')}, p_{w_k}^{(x,y)})$
11:       $\mathcal{H} \leftarrow \{p_{u_k}^{(x,y)} - p_{u_k}^{(x,y)} * G_\sigma,\ \forall p_{u_k}^{(x,y)} \in \mathcal{U}\}$, high
          frequency of the corresponding patches in $\mathcal{U}$
12:       Learn $(\mu, \Sigma)$ the parameters of the multivariate
          Gaussian distribution on the patches of $\mathcal{H}$
13:       Sample $\tilde{p} \sim G(\mu, \Sigma)$
14:       $\tilde{p}_{w_k}^{(x,y)} \leftarrow p_{v_k}^{(x,y)} + \tilde{p}$
15:       $w_k((x,y)+(i,j)) \leftarrow \tilde{p}_{w_k}^{(x,y)}$ for $(i,j)$ in $I_n^2$
16:     end for
17:   end for
18: end for
19: return $w$
```

In general the results shown in Figure 5 are satisfying for a wide range of textures. The global structures are reproduced by the multiscale approach, while the local structures are maintained by the patch based approach. Using a Gaussian patch model allows to create new patches that do not exist in the input example while maintaining satisfying visual results. Based on the analysis of the patch's variance illustrated in Section 2 this guarantees that indeed the simulated patches are sufficiently different from the ones in the input sample. The examples of the last two rows illustrate some failure cases of the method. The main failure cause is the size of the input texture. It is obvious that the input must be large enough to provide us with a sufficient number of patch samples to estimate their distribution. If that is not the case, even though the patches are pixel-wise different of the input ones the visual aspect may remain too similar and cause a "copy-paste" effect when m is small enough. Furthermore, as in other non parametric methods,"garbage", namely the excessive use of a subset of patches in the input image is not fully avoided. This effect is nevertheless mitigated by the multiscale approach.

5 Experiments

In this section, texture synthesis results are shown using the algorithm described in Algorithm 3. In Figure 5, general results of the multiscale method are shown with success and failure cases. In Figure 8 several texture synthesis methods are compared. In Figure 7 the innovation capacity of our method is compared to [6] using coordinate maps. In Figure 6 the patch models introduced in Section 2 and 3 are compared. Finally the influence of the parameters is discussed in Figure 10. There are four of them: the patch size n, the number of neighbours m, the overlap size o and the number of scales K used in the multiscale approach. This is illustrated with two texture examples.

5.1 Model comparison

The results in Figure 6 show the effects of avoiding the use of the blending step in Algorithm 2. The three mod-

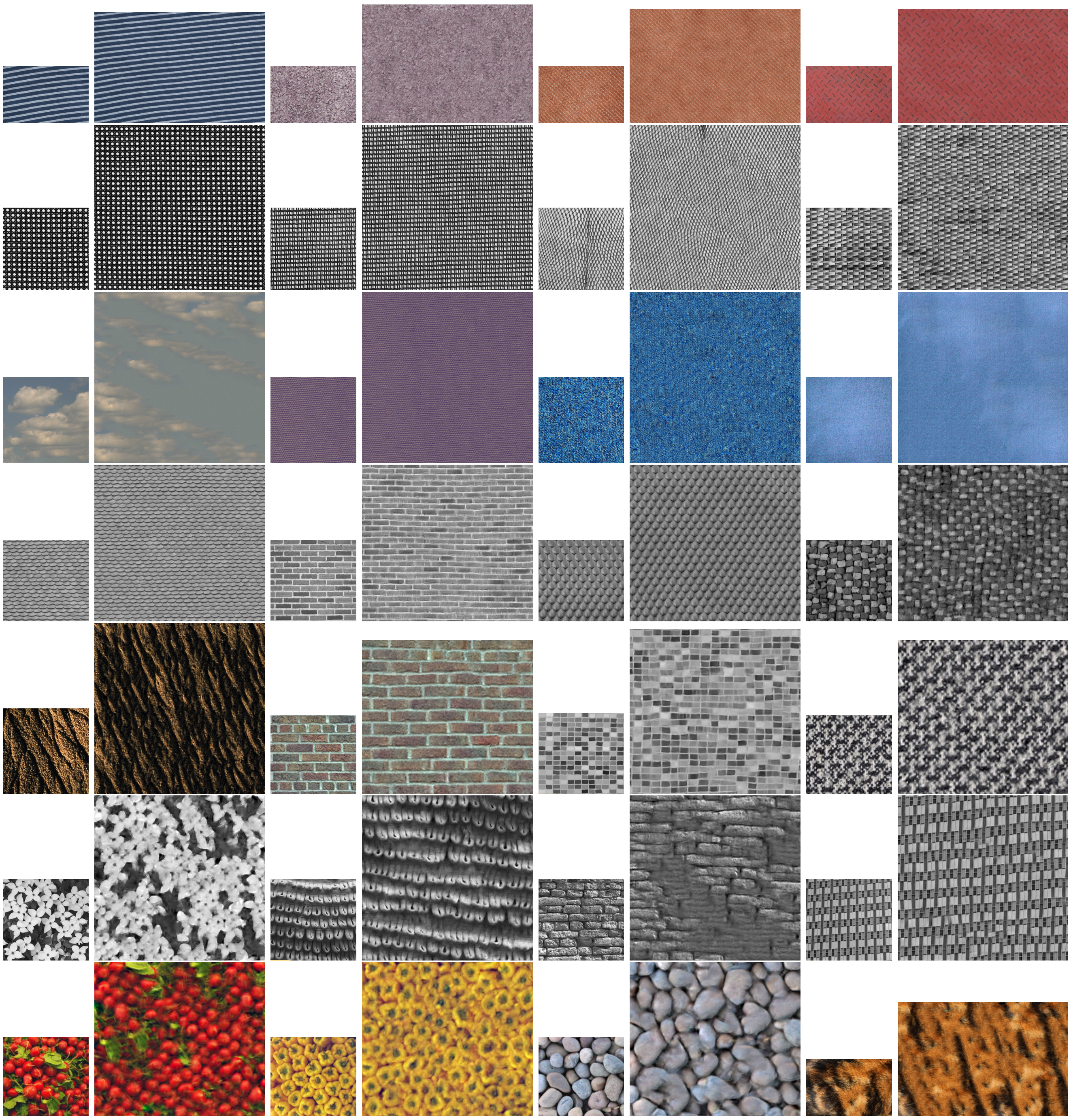

Fig. 5: *Synthesis results of the multiscale algorithm.* The small images correspond to the texture samples and the big ones are the corresponding synthesis results. The parameters used for all examples are $n = 20$, $m = 20$ and $K = 2$.

els LG, CLG and RCLG were tested on several types of textures. For the three of them the quilting step was omitted. This was done to achieve a better comparison of the capacity of respecting the overlap by modeling or not the restriction of the patches to the overlap values. The results are compared for the three models applied in the sampling mode. The results of the same models in the best sample patch mode are less interesting and are not shown here. The general conclusion of the results in Figure 6 is that the conditional patch models achieve a better transition between patches on their overlap region, as expected. Nevertheless the results for CLG and

RCLG both loose some resolution compared to the LG results. One can observe in Figure 6 that blur appears progressively in the raster-scan order of the synthesis algorithm. Indeed, when moving forward in the synthesis the m nearest patches start being too different from each other (due to the strong restriction of keeping the values of the overlap) and the result is a much blurrier patch. The first five texture examples in Figure 6 show better stitching results for CLG and RCLG than for to LG. Looking carefully at the synthesis results of the LG model the edges between patches are more noticeable than in the other two models where almost no transition effect can be seen. Nevertheless the loss of resolution caused by the Gaussian distributions is increased in both conditional models, mostly for CLG. In the last row example the results are less convincing. There are not enough reliable samples to estimate a correct conditional model. The visual results of CLG and RCLG get more and more degraded in the order of the raster scan. This is not surprising since the sample patches respecting the overlap values become increasingly unlikely.

From the experiments in Figure 6 one can conclude that a quilting technique is still needed for complex textures and therefore the LG model is better to model locally the texture input. It generates less blur in the results and has a significant smaller computational cost. In the rest of the experiments only the LG model in considered. Nevertheless it could be interesting to test the use of conditional models in the multiscale version at the coarsest scale.

In Figure 7 the LG model is compared to the MSLG model. One can observe the strength of using the multiscale approach in terms of reproducing global arrangements that are not kept in the LG model for a fixed patch size. This allows to obtain satisfying results for small values of n and therefore gives more freedom to choose the number of nearest patches m. One can also conclude from the experiments in Figure 7 that the multiscale approach generates blurrier textures than the LG model.

5.2 Comparison to other texture synthesis methods

In this section the results of Algorithm 3 are compared to other synthesis methods such as [21,25,6]. In general, the results obtained with the multiscale locally Gaussian method are visually comparable to the non-parametric patch based method of Efros and Freeman [6], with the advantage that now, the patches are being sampled from their Gaussian model and therefore are different. In Figure 8, the first column shows the result of the proposed method. A noticeable drawback of the method is the loss for resolution caused by the use of Gaussian distributions. The proposed method was therefore combined with the Portilla and Simoncelli's algorithm [21] as a first and simple approach to overcome this loss of resolution. The result can be seen in the second column of Figure 8. The result of combining both algorithms is very satisfying. On the one hand the local and global structures are kept due to use of the patch based and multiscale method. On the other hand [21] allows to respect the global statistics of the input or at least be quite close to them. Comparing columns one and two to the third one shows how the combination of both methods improves the results of each method used separately. Of course this solution is limited in particular when the size of the synthesized image is too large. The fourth column shows the results of the Tartavel et al. method [25]. It is interesting to compare our results to this method since both approaches are multiscale, patch based, and create systematically new patches. The results for organized highly structured textures are comparable for both cases. Nevertheless for more complex textures like the flower example and the last two rows one notices a lack of sharpness when recreating the salient objects of the input with the method in [25]. Finally the last column shows results of [6]. One can observe that for MSLG the visual results are in general comparable to those of [6] while providing a local parametric model. The results of [6] are obviously excellent. But once again, in this kind of method, the algorithm ends up copying very large parts of the input. To illustrate this we represent our synthesis results and the ones obtained with [6] using *position and synthesis maps* as can be seen in Figure 9. Each pixel position (x, y) in the sample texture u is associated to a different color from a continuous colormap. The resulting image is called the *position map*. The *synthesis map* associated to the synthesized texture w is obtained by mapping each of its positions (x', y') to the color value of the corresponding position (x, y) of u. This position (x, y) corresponds to the central pixel of the nearest patch in u to the patch centered in (x', y') in w. The synthesis map will allow to identify the tendency of an algorithm to generate verbatim copies and to visualize from which regions of the input texture are sampled the patches. To compute these synthesis maps we used the PatchMatch algorithm [3] an efficient algorithm to approximate optimal correspondences. We used the implementation provided in [7]. In Figure 9 one can observe that in general the synthesis maps associated to the results of MSLG are more "noisy" than those associated to the results of [6]. Also in the synthesis maps associated to [6] larger continuous zones are identified. This corresponds to the verbatim copies produced by the

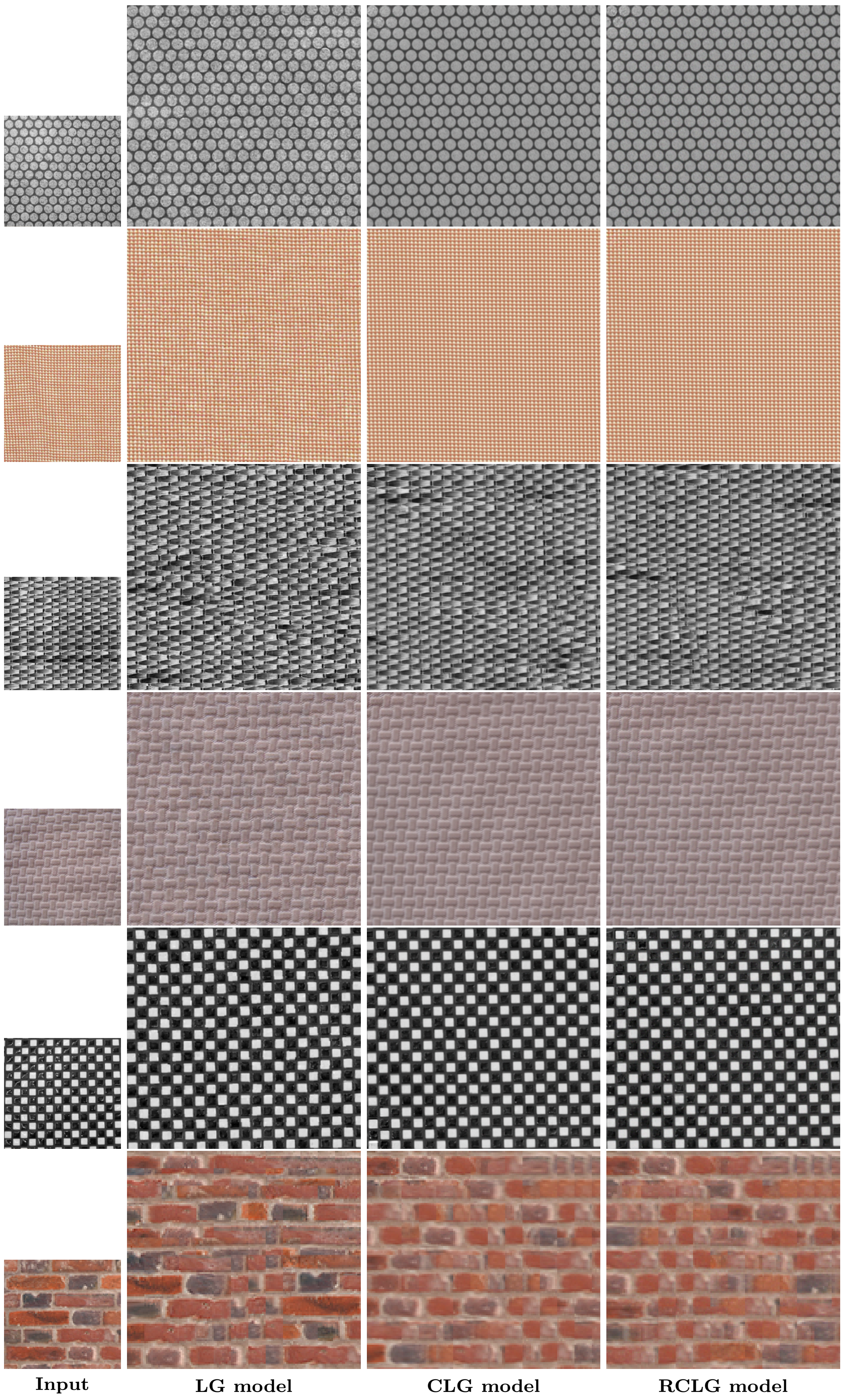

Fig. 6: *Patch model comparison.* From left to right: texture sample, synthesis result using LG model, CLG model and RCLG model. No quilting technique was applied to stitch together the simulated patches for the three presented models. The parameters used for all examples are $n = 40$, $m = 30$ and $o = 0.5$.

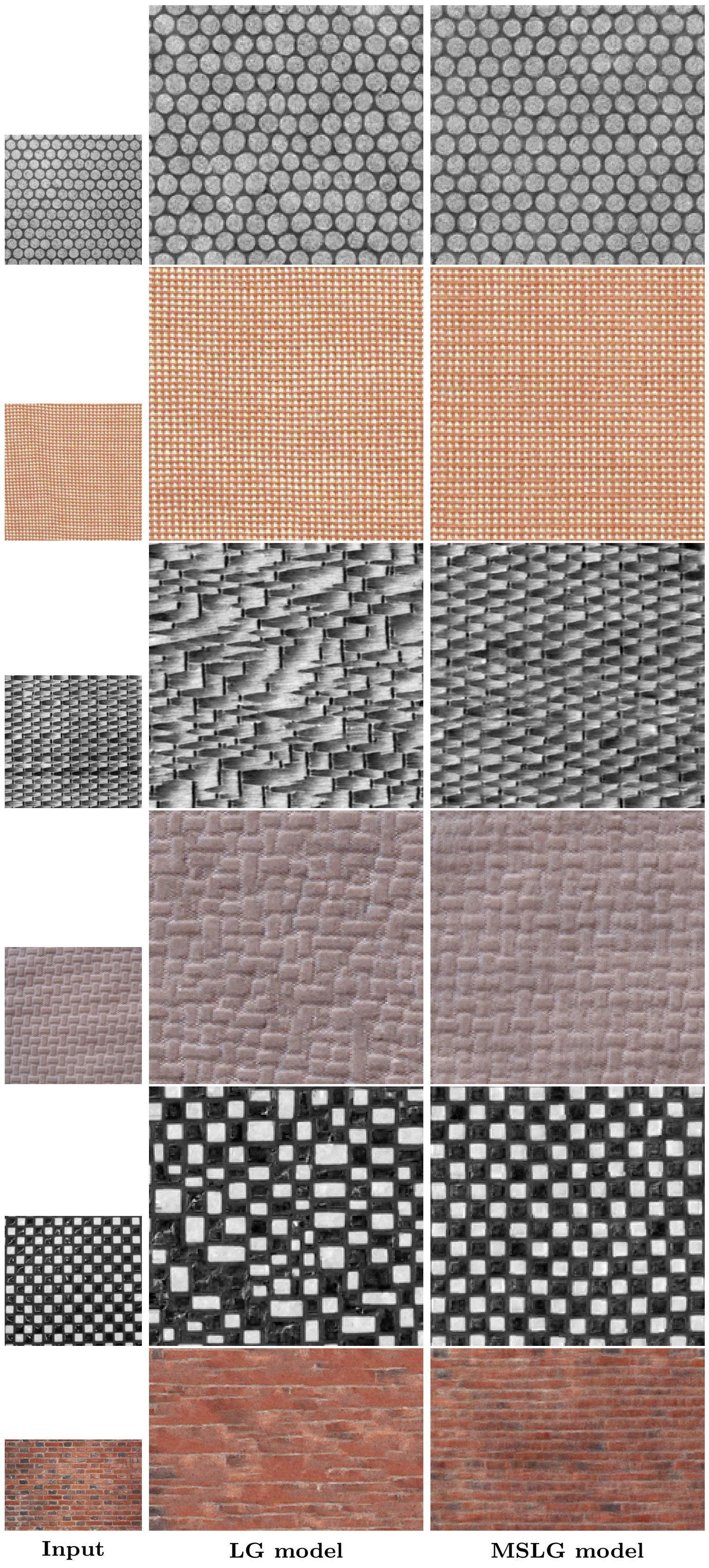

Fig. 7: *Comparison of LG to MSLG.* From left to right: texture sample, synthesis result using LG model, and MSLG model. The parameters used for these examples are $n = 20$, $m = 50$ and $o = 0.5$. For the multiscale synthesis results the number of scales K used is $K = 3$. For both methods we used the same seed patch for each texture sample.

method. It is important to notice, that in both cases, for some texture examples in the synthesis maps there are some dominant colors represented in the synthesis maps. That reflects the discussion of Section 2.3 where we said that the spread of the input sample is not be respected. For example, in row number five of Figure 9, the input sample is not stationary (change of luminosity). Both methods fail in reproducing this. They tend to stay in one region (the darker one in this particular case) and may lead to garbage growing. This is well represented with the synthesis maps where green is dominant for the example in row number five. Combining MSLG to [21] used as post-processing significantly reduces this effect, although [21] has its own limitations.

5.3 Influence of the parameters

In Figure 10 and 11 the influence of the parameters is discussed. There are four of them: the patch size $n \in \{10, 20, 40\}$, the number of neighbours $m \in \{10, 30, 50\}$, the overlap size $o \in \{0.25, 0.5\}$ and the number of scales used $K \in \{1, 2, 3\}$.

Influence of the patch size The synthesis results are very sensitive to the patch size, in particular for macro textures that have details at different scales. Figure 10 clearly shows that if the patch size is too small then the synthesis will fail. Using the multiscale approach strongly reduces the dependency of the method to the patch size. For all examples shown in this paper, the multiscale method using patches of 20×20 pixels was enough to guarantee correct synthesis results. For the one scale version the patch size used should have been much larger to produce convenient results, on the cost of reducing the variability of the Gaussian models and even creating verbatim copy effects.

Influence of the number of neighbours This parameter corresponds to the number of patches used to estimate the patches' Gaussian distribution. As has been discussed in Section 2 the value of m in general should not be too small (> 5) or too large (< 50) to avoid patches of variance null or too big. These values are not general for all textures. The choice of m is linked to the amount of self similarity in the image. Thus when m is too large the Gaussian sampling will blur up the image. In the experience of Figure 10 you can see that when $m = 150$ this leads to a texture which is too regular compared to the input sample. Otherwise since the sample texture in Figure 10 has many self similar patches the value of m can be large enough. The choice of m is a compromise between a copy-paste strategy and a risk of blurry reconstruction of the texture.

Influence of the overlap size For the nearest patches only the overlap areas are compared. This implies that the variance of the model estimated on that set of patches will be controlled only on the overlap region, thus allowing more variety in the remaining pixels of the patch. If that region is not big enough then the complementary region of the overlap will not be correctly modeled, since outliers can be considered in the set of patches. In Figure 10 two overlap cases are considered: a quarter of the patch side and half the side patch size. This influence is more noticeable in the columns for $n = 20$ and $n = 40$.

Influence of the multiscale process In Figure 11 the results show that using a single scale for a fixed patch size is not enough to reproduce faithfully the global structure of the input sample for the three texture examples. To achieve satisfying results for a single scale a larger side patch size should be considered. Still this would lead to the limitations mentioned previously. Furthermore when the number of scales is increased the global arrangements are recovered as expected. That can be checked in the three examples of Figure 11. They also put in evidence how using a simple average of the values along the overlap area of the coarsest scale is sufficient to deal with the overlapping patches. More complex quilting techniques are then avoided. The multiscale algorithm will help a better respect of the spread (but it's not a guarantee).

6 Conclusion

In this work, a local texture sampling method in the patch space using conditional Gaussian models was proposed. The motivation was to dispose of a patch stitching step by using a more robust local model for the texture. The Gaussian distribution of a patch was then conditioned to the values of its overlapping region. Two approaches were considered: CLG and RCLG models, and were compared to the local Gaussian model (LG). The results show that avoiding the stitching step is possible when dealing with periodic and pseudo-periodic textures. For more complex textures the conditional models are less performing and are fast limited by the size of the texture input sample. In general, the synthesis results using the conditional models were slightly smoother than the ones of the LG model.

The second contribution of this paper was a multiscale texture synthesis algorithm using local Gaussian models (LG). In general, the experiments showed satisfying results for a wide variety of texture samples. This is due to the patch based approach that conserves the local structures while sampling each patch from its

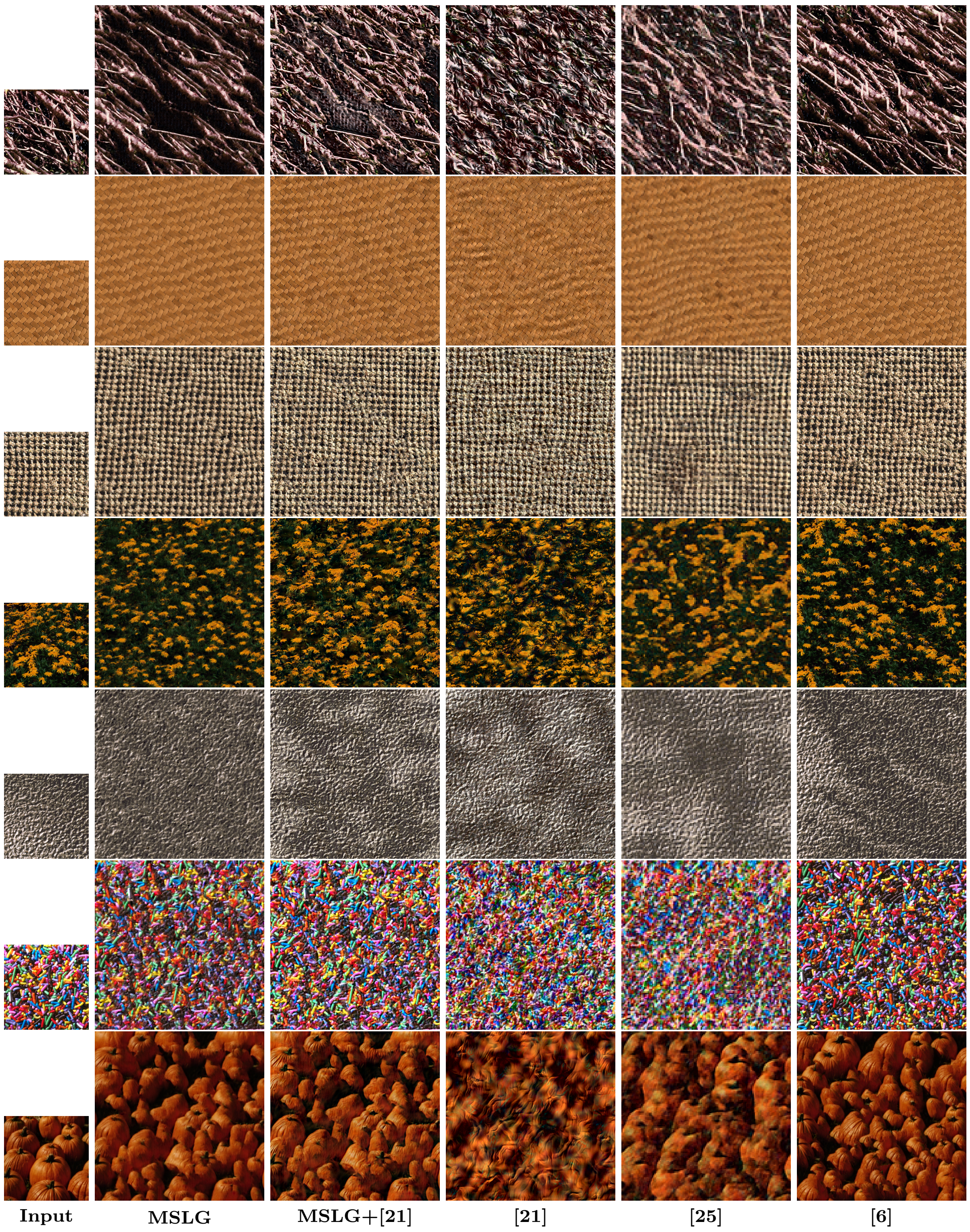

Fig. 8: *Comparison to several texture synthesis algorithms.* From left to right: input sample, MSLG, MSLG combined to Portilla and Simoncelli [21], Portilla and Simoncelli[21], Tartavel et al. [25] (source: `http://perso.telecom-paristech.fr/~tartavel/research/jmiv14.php`) and Efros and Freeman [6]. For MSLG the parameters used are $n = 20$, $m = 20$ and $K = 2$. For [21] four scales and orientations were used. For [25] the patch size is 12 and the number of scales 3. For [6] the patch size used is 20.

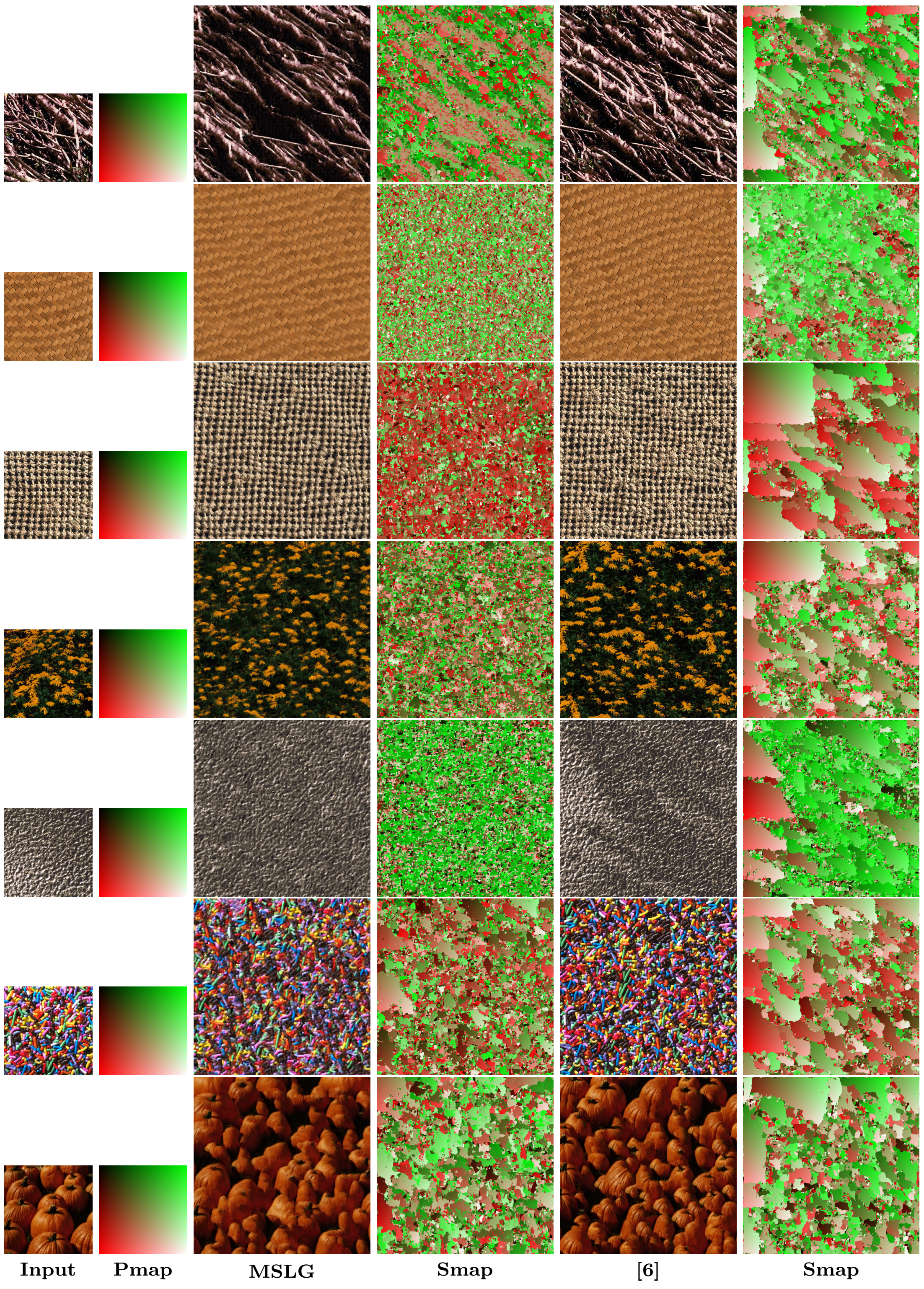

Fig. 9: *Innovation capacity.* From left to right: input sample, associated position map (Pmap), results of MSLG, associated synthesis map (Smap), results of [6] and associated synthesis map (Smap). The results of MSLG and [6] presented in this image are the same as the ones showed in Figure 8. The results are represented here with their corresponding synthesis map to illustrate the capacity of innovation of both methods. The synthesis maps show the regions from which the patch were sampled to generate the new texture image. Verbatim copy zones can be detected where continuous color zones in the synthesis maps are visible as well as unvisited zones of the input when the synthesis maps show dominant colors.

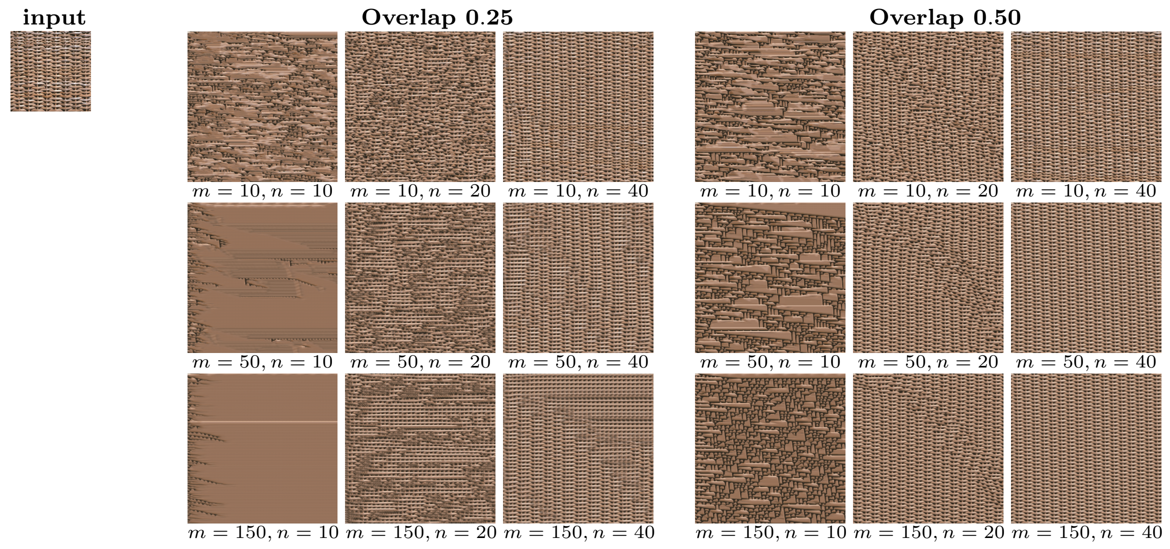

Fig. 10: *Influence of the choice of the patch size, number of nearest neighbours and overlap size.* For a given texture example several synthesis results are shown for different sets of parameters. Two columns of 3×3 results are presented. The left column corresponds to an overlap size $o = 0.25 \times n$ and the right column to an overlap size $o = 0.5 \times n$. For each column of 3×3, from top to bottom the number of nearest neighbours m takes the values $10, 50, 150$. From left to right the patch size n takes the values $10, 20, 40$. These results clearly show that for a fixed value of n and m using an overlap size $o = 0.25 \times n$ is not enough to determine a correct set of nearest neighbours. For a fixed patch size, for example ($n = 40, o = 0.5 \times n$) one can observe that increasing the value of m smoothes the synthesis result.

Gaussian distribution, thus creating new patches that do not exist in the input sample. On the other hand the multiscale approach permits to synthesize the global arrangement of the salient structures of the input sample. The experiments also put in evidence that the use of Gaussian distributions, for some texture examples, have a tendency to slightly smooth the result compared to the initial resolution of the sample. To overcome this effect the method was combined to Portilla and Simoncelli's algorithm [21] as a first solution.

Several aspects of this method are still open to elucidation, such as the way patches are compared, the adaptation of the number of neighbours to the patch being modeled and the conservation of the texture's global statistics.

Acknowledgements Work partly founded by the European Research Council (advanced grant Twelve Labours) and the Office of Naval research (ONR grant N00014-14-1-0023).

References

1. Aguerrebere, C., Gousseau, Y., Tartavel, G.: Exemplar-based texture synthesis: the efros-leung algorithm. Image Processing On Line **2013**, 213–231 (2013)
2. Ashikhmin, M.: Synthesizing natural textures. In: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 217–226. ACM (2001)
3. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics-TOG **28**(3), 24 (2009)
4. Briand, T., Vacher, J., Galerne, B., Rabin, J.: The heeger & bergen pyramid based texture synthesis algorithm. Image Processing On Line **4**, 276–299 (2014)
5. Efros, A., Leung, T.K., et al.: Texture synthesis by non-parametric sampling. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, vol. 2, pp. 1033–1038. IEEE (1999)
6. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 341–346. ACM (2001)
7. Fedorov, V., Facciolo, G., Arias, P.: Variational Framework for Non-Local Inpainting. Image Processing On Line **5**, 362–386 (2015). DOI 10.5201/ipol.2015.136
8. Galerne, B., Gousseau, Y., Morel, J.M.: Micro-texture synthesis by phase randomization. Image Processing On Line **2011** (2011)
9. Galerne, B., Gousseau, Y., Morel, J.M.: Random phase textures: Theory and synthesis. Image Processing, IEEE Transactions on **20**(1), 257–267 (2011)
10. Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. arXiv preprint arXiv:1505.07376 (2015)
11. Heeger, D.J., Bergen, J.R.: Pyramid-based texture analysis/synthesis. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 229–238. ACM (1995)
12. Julesz, B.: Visual pattern discrimination. Information Theory, IRE Transactions on **8**(2), 84–92 (1962)

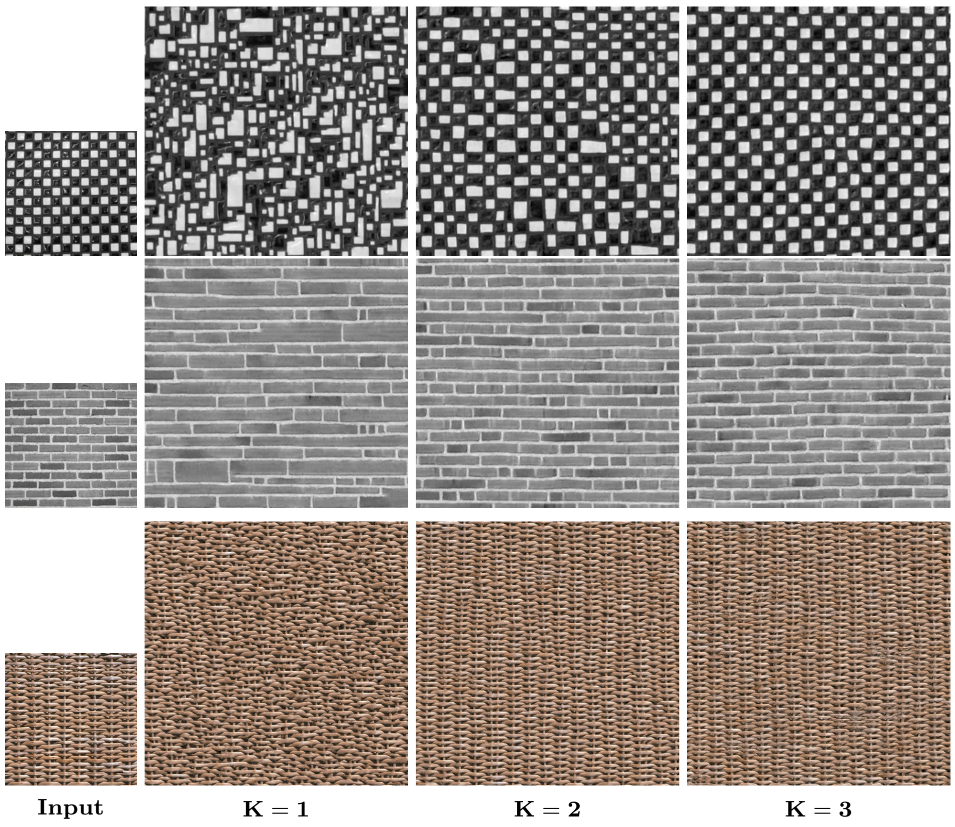

Fig. 11: *Multiscale influence.* For each row from left to right: input sample, three synthesis results for $K = 1, 2, 3$ scales. For the three texture examples from top to bottom the parameters used were $(n = 10, m = 20), (n = 20, m = 20), (n = 20, m = 30)$. In the three examples the global arrangement for $K = 1$ is not respected while for $K = 2$ and $K = 3$ are correctly synthesized.

13. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. In: ACM Transactions on Graphics (TOG), vol. 24, pp. 795–802. ACM (2005)
14. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. In: ACM Transactions on Graphics (ToG), vol. 22, pp. 277–286. ACM (2003)
15. Lebrun, M., Buades, A., Morel, J.M.: Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. Image Processing On Line **3**, 1–42 (2013). DOI 10.5201/ipol.2013.16
16. Lefebvre, S., Hoppe, H.: Parallel controllable texture synthesis. ACM Transactions on Graphics (TOG) **24**(3), 777–786 (2005)
17. Levina, E., Bickel, P.J.: Texture synthesis and nonparametric resampling of random fields. The Annals of Statistics pp. 1751–1773 (2006)
18. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Transactions on Graphics (ToG) **20**(3), 127–150 (2001)
19. Morrison, D.F.: Multivariate statistical methods. 3. New York, NY. Mc (1990)
20. Peyré, G.: Sparse modeling of textures. Journal of Mathematical Imaging and Vision **34**(1), 17–31 (2009)
21. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. International Journal of Computer Vision **40**(1), 49–70 (2000)
22. Raad, L., Desolneux, A., Morel, J.M.: Locally gaussian exemplar based texture synthesis. In: Image Processing (ICIP), 2014 IEEE International Conference on, pp. 4667–4671. IEEE (2014)
23. Raad, L., Desolneux, A., Morel, J.M.: Conditional gaussian models for texture synthesis. In: Scale Space and Variational Methods in Computer Vision, pp. 474–485. Springer (2015)
24. Raad, L., Desolneux, A., Morel, J.M.: Multiscale exemplar based texture synthesis by locally gaussian models. In: Iberoamerican Congress on Pattern Recognition (CIARP) (2015)
25. Tartavel, G., Gousseau, Y., Peyré, G.: Variational texture synthesis with sparsity and spectrum constraints. Journal of Mathematical Imaging and Vision **52**(1), 124–144 (2014)
26. Wei, L.Y., Lefebvre, S., Kwatra, V., Turk, G.: State of the art in example-based texture synthesis. In: Eurographics 2009, State of the Art Report, EG-STAR, pp. 93–117. Eurographics Association (2009)
27. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 479–488. ACM Press/Addison-Wesley Publishing Co. (2000)