Locally Gaussian exemplar-based texture synthesis

Lara Raad, Agnès Desolneux, J.-M. Morel

LOCALLY GAUSSIAN EXEMPLAR BASED TEXTURE SYNTHESIS

Lara Raad, Agnès Desolneux, Jean-Michel Morel

CMLA, Ecole Normale Supérieure de Cachan, France

ABSTRACT

The main approaches to texture modeling are the statistical psychophysically inspired model and the patch-based model. In the first model the texture is characterized by a sophisticated statistical signature. The associated sampling algorithm estimates this signature from the example and produces a genuinely different texture. This texture nevertheless often loses accuracy. The second model boils down to a clever copy-paste procedure, which stitches verbatim copies of large regions of the example. We propose in this communication to involve a locally Gaussian texture model in the patch space. It permits to synthesize textures that are everywhere different from the original but with better quality than the purely statistical methods.

Index Terms— Texture Synthesis, Gaussian Modeling, Image Patches

1. INTRODUCTION

Exemplar-based texture synthesis aims at generating new texture images inspired from a texture sample. Texture synthesis algorithms can be divided into two main categories: statistics-based methods [1, 2, 3, 4, 5] and patch-based methods [6, 7, 8, 9, 10, 11, 12, 13].

Statistics-based methods estimate a set of the sample's statistics, which is then enforced in the synthesized texture. Julesz [14] discovered that many texture pairs having the same second-order statistics would not be discerned by human preattentive vision. This hypothesis is referred to as the first Julesz axiom for texture perception. It can be checked by algorithms that create new images where the texture sample's second-order statistic is emulated. This can be done for example by maintaining the Fourier modulus of the sample image and randomizing its phase [15]. The random phase method correctly synthesizes micro-texture images which adapt well to a Gaussian distribution, but it fails for more structured ones, as can be experimented in the executable paper [16].

Aknowledgment: work partially supported by the Centre National d'Etudes Spatiales (CNES, MISS Project), the European Research Council (Advanced Grant Twelve Labours), the Office of Naval Research (Grant N00014-97-1-0839), Direction Générale de l'Armement (DGA), Fondation Mathématique Jacques Hadamard and Agence Nationale de la Recherche (Stereo project).

Heeger and Bergen [1] extended the 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 an exemplar based synthesis method. Yet this method only measures marginal statistics. It misses important correlations between pixels across scales and orientations. See the online execution of this method [17] where some success but many failures are evident, like for [16]. Within a similar range of results De Bonet [3] randomizes the initial texture image and preserves only a few statistics: the dependencies across scales of a multi-resolution filter bank response. In [4] the authors propose to synthesize a texture by taking randomly 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. The results achieved are similar to [1, 3]. The Heeger-Bergen method was extended by Portilla and Simoncelli [5] 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 achieves striking results for a very wide range of texture examples. This method, which represents 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.

Patch-based methods constitute a totally different category of texture synthesis algorithms. The initial Efros and Leung [6] method is based on Shannon's Markov random field model for the English language. 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 in the pixel is compared to all the patches of the input sample. The patches in the sample that are closer help predict the pixel value in the synthetic image. Several optimizations have been proposed to accelerate this algorithm. Among them Wei and Levoy [7], who managed to fix the shape and size of the learning patch and Ashikhmin [8] who proposed to extend existing patches whenever possible instead of searching in the entire sample texture. Yet, as already pointed out in the original paper [6], an iterative procedure may fail by producing "garbage" when the neigborhood'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 [18]. Several extensions of [6] have been proposed that manage to accelerate the procedure and resolve the "garbage" problem by stitching entire patches instead of pixels. In [10] the authors propose 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 [11] proposed an extension of the latter introducing to the quilting method a blending algorithm that computes the minimum cost path across overlapping patches overcoming the transition effect from patch to patch. Kwatra et al. in [12] extend [11] by using a graphcut algorithm to define the edges of the patch to quilt in the synthesis image. Another extension of [6] is proposed by Kwatra et al. in [13] where to synthesize a texture image they improve the quality of the synthesis image sequentially by minimizing a patch-based energy function. For these methods the visual results are strikingly good, but still retain the risk of verbatim copying large parts of the input sample. For practical applications this may result in the appearance of repeated patterns in the synthesized image.

An ideal exemplar-based texture synthesis algorithm should create a new texture image that is perceptually equivalent to the input sample, but this image should be as random as possible while maintaining the same statistics of the input. To the best of our knowledge, none of the methods proposed in the literature are able to combine these advantages. On one hand, statistics-based methods fail, in general, to be perceptually equivalent to the input, either because the statistics are not well enforced or because these statistics are not complete enough. On the other hand, patch-based methods are too close to a mere copy-paste. This effect can be observed in the synthesis maps in Figure 3.

In this work, we propose an algorithm that lies between statistics-based and patch-based algorithms. We gave up choosing a set of statistics from the input sample to enforce in the synthesized image. Instead, we chose to model each texture patch by a Gaussian distribution learned from its similar patches. Inspired by [11], we maintain the idea of searching for patches to stitch together in the original sample. However, instead of using the exact patch taken in the input texture, we sample the stitched patch from its Gaussian model. Local Gaussian patch models have been proved useful in image denoising [19]. Our 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. In this way, we manage to combine the positive aspects of statistics-based and patch-based methods, while overcoming some of their drawbacks.

The rest of this paper is structured as follows. In Section 2, we give an estimation method for a Gaussian model for each patch and we validate this algorithm on textural patches. In Section 3, the synthesis algorithm is described. Section 4 compares the results with state of the art texture-synthesis methods. The concluding section 5 also states the limitations of the method.

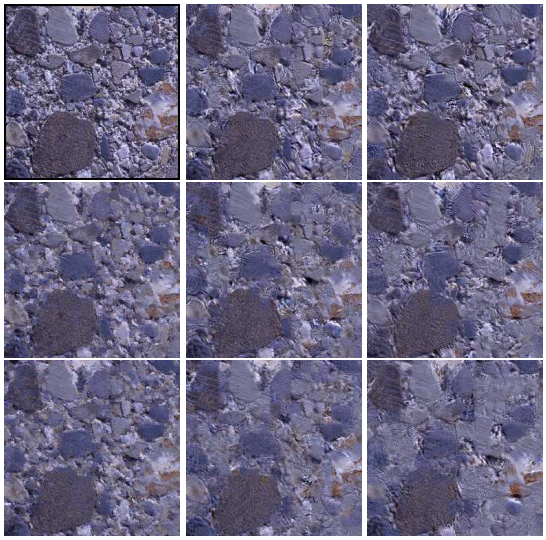

Fig. 1: Gaussian substitution for the left top corner texture image. From left to right the patch size $ps = 10, 20, 30$. From top to bottom, the neighborhood size $N = 10, 20, 30$.

2. THE PATCH GAUSSIAN MODEL

In this section we model the patches local distribution as a Gaussian vector and test what happens when the image patches are replaced by synthesized ones according to this model. We define a patch p as a square block of size $ps \times ps$ among all possible overlapping blocks of an input image I. Given the image I and fixing an overlap size o we will consider all patches from I that are taken in a raster-scan order and whose centers are separated by $(ps - o)$ pixels. Each of them will be replaced by a new patch $\tilde{p}$ sampled from a Gaussian model of parameters μ and Γ. We create in this way a simulated image $\tilde{I}$. More precisely, the steps of the procedure are the following:

1. Learn the Gaussian model of p. We fix N and define $\mathcal{S}_P = \{p_1, \ldots, p_N\}$ as the set of the N closest patches to p (among all patches of the image) according to the L^2 distance.

2. Estimate the empirical mean μ and the empirical covariance matrix Γ defined in (1) where P is a matrix whose columns are the vectorized patches $p_i \in S_p$.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} p_i, \quad \Gamma = \frac{(P-\mu)}{\sqrt{N}}\frac{(P-\mu)^t}{\sqrt{N}} = \tilde{P}\tilde{P}^t \quad (1)$$

3. Sample $\tilde{p}$ from the Gaussian distribution of parameters μ and Γ. Since $\tilde{P}^t\tilde{P}$ has the same non-zero eigenvalues as Γ and is of lower size we diagonalize and find the eigenvectors

and eigenvalues of $\tilde{P}^t\tilde{P}$ instead of Γ such as

$$\tilde{p} = \tilde{P}WDx' + \mu \tag{2}$$

where $x' \sim \mathcal{N}\left(\vec{0}, I_N\right)$, W is a matrix whose columns are the eigenvectors of $\tilde{P}^t\tilde{P}$, and D is a diagonal matrix containing its eigenvalues. We simply need to sample N independent normal variables to obtain (2).

4. Place the sample $\tilde{p}$ in $\tilde{I}$ overlapping the part which has been already synthesized. Across this overlap area we blend the pixel values with a bilinear interpolation.

The size ps of the patches and the size of the neighborhood N are user-specified values. The results of the Gaussian modeling are shown in Figure 1. Observe that replacing the patches of each texture by simulated ones with the corresponding Gaussian model achieves a faithful random variant of the original image, for reasonable values of ps and N. All pixel values of $\tilde{I}$ and I are actually different, but both images are visually very similar for the adequate values of N and ps. For large values of N the model no longer represents the patch faithfully because $\mathcal{S}_p$ will contain outliers. Nevertheless, reasonable values for ps and N ensure a correct reconstruction when replacing patches by others simulated from a Gaussian model. We found that when using 30×30 patches, it is reasonable to consider a neighborhood of 10 to 20 patches. These results justify the use of Gaussian sampling as a step of our synthesis algorithm.

3. THE LOCALLY GAUSSIAN TEXTURE SYNTHESIS ALGORITHM

Like in the Efros-Freeman [11] method, the synthesized texture is constructed sequentially patch-wise in a raster-scan order. Each new patch is placed in the output image overlapping part of the previous one. The overlap size o is fixed and for all the presented results o is half the size ps of the patch.

We initialize the new texture image placing a seed patch in its top left corner. For this we pick a random patch from the texture sample, learn its Gaussian model and sample a new patch from it (the seed patch). In continuation, each synthesized patch is sampled from a Gaussian model learnt from the N patches with their left half most similar to the right half of the last synthesized texture patch. This new patch is then quilted on the current texture.

Patch stitching follows the procedure presented in [11]. It assigns new edges to the stitched patch to adjust it to the synthesized image along the overlap area. These edges are selected as curves minimizing the error between the new patch and part of the synthesized image across the overlap region. The synthesis procedure can be summarized as follows.

1. Define parameters values N and ps
2. Initialize the output image with the seed patch
3. For each patch taken in a raster-scan order
   (a) Find the N nearest neighbors in the input sample that best agree with the patch under construction along the overlap area (the left half of the patch)
   (b) Learn the Gaussian model estimating μ and Γ
   (c) Sample a patch from this model
   (d) Quilt the patch in the synthesis image

Fig. 2: Texture synthesis result for the left top corner texture image. We show the results obtained for different values of N and ps. From left to right $ps = 10, 20, 0$. From top to bottom, the neighborhood size $N = 10, 20, 30$. All the results are obtained for an overlap of half patch size.

In this way, a new texture image, perceptually similar to the original sample and whose patches have been randomized, is obtained. The next section shows some results.

4. EXPERIMENTS

The results on the synthesis algorithm can be seen in Figure 2. For this example the efficacy of the algorithm varies with the parameters N, ps and o. A first observation is that the preferable patch size has a sample-dependent lower bound. If the patch size is too small the algorithm does not capture the variability of the sample. On the other hand the larger the patch size, the less neighbors we can use to build a faithful model for the patches. The second observation relates to the size of the overlap between patches. If the proportion of the overlap area with respect to the patch size is too small, the obtained model may no longer be a correct representation of the patch to simulate. Indeed, the patches are compared on the overlap area but the model is learnt on the whole patch. We observed that for an overlap of half the patch size the model is faithful enough to achieve good results.

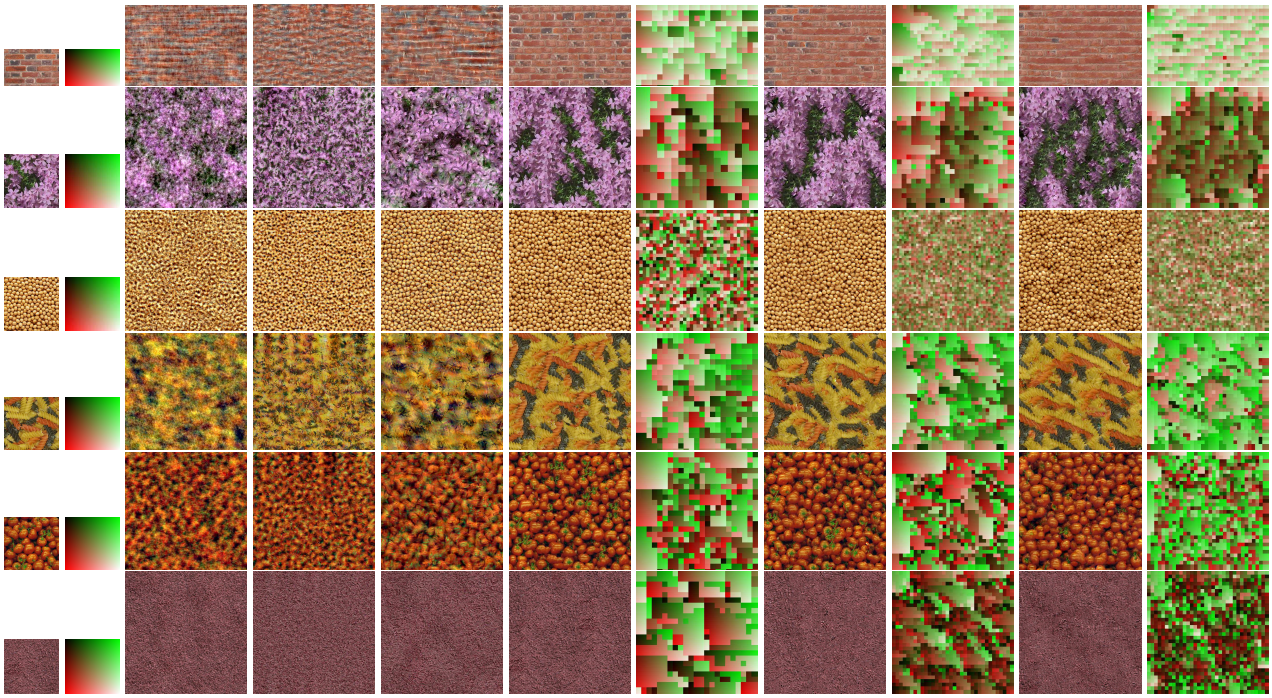

Fig. 3: Comparison with various other texture synthesis methods. From left to right: texture sample, position map (each pixel q in the texture sample is associated to a different color from a continuous colormap), synthesis results of [16], [17], [5], [11], synthesis map of [11], our algorithm and its synthesis map, our algorithm and its synthesis map forcing the nearest neighbors to be at least r pixels away from the best neighbor ($r = 3$). The synthesis map shows for each synthesized patch its position in the original image. It allows then to identify exactly the verbatim copy regions (continuous color areas of the map). For our algorithm the synthesis maps are computed using the barycenter of the N best neighbors of each simulated patch. For all the examples $N = 10$, $ps = 30$ except for the third row where $ps = 10$. For a correct visual comparison we recommend a $4\times$ zoom in.

Figure 3 compares several state of the art texture synthesis methods. We can observe that for statistics-based methods (three first columns) the quality of our algorithm's visual results is considerably superior. However, when the size of the patch ps is too big compared to the texture's structure (last row in Figure 3) blurring artifacts can appear in our results. On the other hand, like for the Efros-Freeman algorithm [11] our method maintains a spatial coherence between patches. We obtain similar visual results and most important avoid verbatim copies from the input sample, as can be seen with the synthesis maps in Figure 3.

5. CONCLUSION

In this paper we have presented a method that synthesizes textures by stitching patches that are samples of a local Gaussian model in the patch space. This model is learnt for each patch from a set of similar patches in the textures. The algorithm synthesizes a texture that is perceptually equivalent to the sample image, but not composed of patches existing in the input texture. Our method overcomes some of the drawbacks of the statistics-based and the patch-based methods. The stitching procedure is a bit complex and could be replaced by Poisson editing [23]. Like the Efros-Leung or the Efros-Freeman methods, the algorithm remains dependent on the choice of ps and N, that may have to be adjusted for each texture sample. In our opinion the texture samples used in the literature are too small, particularly for macro-textures like the ones presented here and in most papers. Thus, our local Gaussian model is forced to use only 10 to 20 degrees of freedom because only some 10 to 20 patches are similar enough. This estimate should improve with larger texture samples. Our algorithm is complex, as we estimate a Gaussian model for each patch. This complexity is nevertheless comparable to classic patch-based denoising algorithms [21], [24]. A multiscale version of the algorithm should also be considered. Unlike the statistics-based algorithms, but like the other patch-based methods, our algorithm is not forced to respect the global statistics of the texture sample. This can be observed in the second row of Figure 3 where our result correctly reproduces an image of petals but fails to insert a correct proportion of the leaves in the synthetic image. Future work should focus on using larger texture samples to better catch the variability of large patches, and on estimating automatically ps and N.

6. REFERENCES

[1] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *SIGGRAPH*, New York, NY, USA, 1995, pp. 229–238.

[2] C. S. Zhu, N. Y. Wu, and D. Mumford, "Minimax Entropy Principle and Its Application to Texture Modeling," *Neural Computation*, vol. 9, no. 8, Nov. 1997.

[3] J. S. De Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images," SIGGRAPH '97, pp. 361–368.

[4] Y.-Q. Xu, B. Guo, and H. Shum, "Chaos Mosaic: Fast and Memory Efficient Texture Synthesis," Tech. Rep., Microsoft Research, 2000.

[5] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients.," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 49–70, 2000.

[6] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *IEEE International Conference on Computer Vision*, 1999, pp. 1033–1038.

[7] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization.," in *SIGGRAPH*, 2000, pp. 479–488.

[8] M. Ashikhmin, "Synthesizing natural textures," in *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 2001, pp. 217–226.

[9] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 327–340.

[10] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Transactions on Graphics (ToG)*, vol. 20, no. 3, pp. 127–150, 2001.

[11] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer.," in *SIGGRAPH*, 2001, pp. 341–346.

[12] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," in *ACM Transactions on Graphics (TOG)*. ACM, 2003, vol. 22, pp. 277–286.

[13] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," in *ACM Transactions on Graphics (TOG)*. ACM, 2005, vol. 24, pp. 795–802.

[14] B. Julesz, "Visual pattern discrimination.," *IEEE Trans. Inf. Theory*, vol. 8, no. 2, pp. 84–92, 1962.

[15] B. Galerne, Y. Gousseau, and J.-M. Morel, "Random phase textures: Theory and synthesis.," *IEEE Transactions in Image Processing*, 2010.

[16] B. Galerne, Y. Gousseau, and J.-M. Morel, "Micro-Texture Synthesis by Phase Randomization," *Image Processing On Line*, vol. 2011, 2011, `http://dx.doi.org/10.5201/ipol.2011.ggm_rpn`.

[17] T. Briand, J. Vacher, B. Galerne, and J. Rabin, "The Heeger and Bergen Pyramid Based Texture Synthesis Algorithm," *Image Processing On Line, preprint*, vol. 2014, 2014.

[18] C. Aguerrebere, Y. Gousseau, and G. Tartavel, "Exemplar-based Texture Synthesis: the Efros-Leung Algorithm," *Image Processing On Line*, vol. 2013, pp. 213–231, 2013.

[19] M. Lebrun, A. Buades, and J.-M. Morel, "Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm," *Image Processing On Line*, vol. 2013, pp. 1–42, 2013.

[20] A. Buades, M. Lebrun, and J.-M. Morel, "Implementation of the non-local bayes image denoising algorithm,," *Image Processing On Line*, 2012.

[21] M. Lebrun, A. Buades, and J.-M. Morel, "A nonlocal bayesian image denoising algorithm," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1665–1688, 2013.

[22] E. P. Simoncelli and B. A. Olshausen, "Natural image statistics and neural representation," *Annual review of neuroscience*, vol. 24, no. 1, pp. 1193–1216, 2001.

[23] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 313–318, 2003.

[24] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *Image Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 2080–2095, 2007.