Volume xx (200y), Number z, pp. 1–6

Texton Noise: Supplementary Material

B. Galerne[1], A. Leclaire[2] and L. Moisan[1]

[1]Laboratoire MAP5, Université Paris Descartes and CNRS, Sorbonne Paris Cité
[2]CMLA, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France

1. Description of the Supplementary Material

The archive `texton_noise_sup_mat.zip` contains

- This pdf supplementary file `texton_noise_sup_mat.pdf`
- The video `texton_noise_video.mp4` that illustrates the OpenGL implementation and the on-the-fly anisotropic filtering.
- The source code folder `codes`.

1.1. Description of the Publicly Available Source Code

The source code folder `codes` contains two folders `compute_texton`, `display_texton_noise` and one `readme.txt` file. The analysis-synthesis pipeline of the texton noise can be simply followed by running the bash commands proposed in `readme.txt`.

The folder `compute_texton` contains the Matlab functions required to compute the texton associated to a given exemplar texture. In particular:

- The function `tn_compute_texton.m` computes the texton associated to one particular exemplar.
- The script `scr_compute_all_textons.m` computes all the textons for the textures given in the folder `compute_texton/input_textures`.
- All the textures shown in the paper are given in `compute_texton/input_textures/`.

The textons are written in files having the extension `.texton`.

The folder `display_texton_noise` contains the OpenGL sources required to sample and display the texton noise associated to a given texton. These sources must be compiled before execution using the provided `Makefile`. The user interface allows for changing the direction of the square, changing the mean number of impacts of the noise as well as its scale. It also enables to turn on and off the anisotropic filtering so that users can see the immediate benefit of this antialisaing procedure.

2. Details Regarding Color Texton Noise

2.1. Computation of Color Texton Interpolation Coefficients

As shown in [GGM11], the proper spectral constraint for color ADSN synthesis is not only that the Fourier spectrum of each channel is conserved but also that the relative phase shift between color channels is conserved. More precisely, given a color image $\boldsymbol{u} = (u_{\mathrm{R}}, u_{\mathrm{G}}, u_{\mathrm{B}})^T \in \mathbb{R}^{\Omega\times 3}$let us define its associated normalized version

$$\boldsymbol{t_u} = \frac{1}{\sqrt{|\Omega|}}(\boldsymbol{u} - \mathrm{mean}(\boldsymbol{u})), \tag{1}$$

(where the $\mathrm{mean}(\boldsymbol{u}) = (\mathrm{mean}(u_{\mathrm{R}}), \mathrm{mean}(u_{\mathrm{G}}), \mathrm{mean}(u_{\mathrm{B}}))^T$), and $\tilde{\boldsymbol{t}}_{\boldsymbol{u}}(k) = \boldsymbol{t_u}(-k)^T$. Another normalized image $\boldsymbol{t_v}$ with same mean corresponds to the same Gaussian texture as $\boldsymbol{u}$ if and only if $\boldsymbol{t_u} * \tilde{\boldsymbol{t}}_{\boldsymbol{u}} = \boldsymbol{t_v} * \tilde{\boldsymbol{t}}_{\boldsymbol{v}}$ where

$$\boldsymbol{t_u} * \tilde{\boldsymbol{t}}_{\boldsymbol{u}}(k) = \sum_{l\in\Omega} \boldsymbol{t_u}(l)\boldsymbol{t_u}(k+l)^T \in \mathbb{R}^{3\times 3}.$$

A straightforward Fourier analysis shows that this is possible if and only if for all frequencies ξ, the DFT coefficient $\hat{\boldsymbol{t}}_{\boldsymbol{v}}(\xi)$ belongs to the circle $\left\{e^{i\theta}\hat{\boldsymbol{t}}_{\boldsymbol{u}}(\xi),\ \theta\in[0,2\pi)\right\}$. The projection onto this circle of a general Fourier coefficient $\hat{\boldsymbol{t}}_{\boldsymbol{v}}(\xi) \in \mathbb{C}^3$ is obtained for $e^{i\theta} = \dfrac{\hat{\boldsymbol{t}}_{\boldsymbol{u}}(\xi)^*\hat{\boldsymbol{t}}_{\boldsymbol{v}}(\xi)}{|\hat{\boldsymbol{t}}_{\boldsymbol{u}}(\xi)^*\hat{\boldsymbol{t}}_{\boldsymbol{v}}(\xi)|}$ where * denotes the conjugate transpose of a complex vector (see the appendix of [TGP14] or Remark 2.1.3 of [Lec15]). Hence, to compute the interpolation coefficients $\boldsymbol{\alpha}$ associated with $\boldsymbol{u}$, the spectral projection of Algorithm 1 becomes

$$\hat{\boldsymbol{\alpha}} \leftarrow \frac{1}{\sqrt{\hat{b}}}\frac{\hat{\boldsymbol{t}}_{\boldsymbol{u}}^*\hat{\boldsymbol{\alpha}}}{|\hat{\boldsymbol{t}}_{\boldsymbol{u}}^*\hat{\boldsymbol{\alpha}}|}\hat{\boldsymbol{t}}_{\boldsymbol{u}}.$$

2.2. Color Correction

The cross-correlation matrix of the input image $\boldsymbol{u}$ is the 3×3 matrix $\boldsymbol{t_u} * \tilde{\boldsymbol{t}}_{\boldsymbol{u}}(0)$, while the channel covariance matrix of the color texton noise g_λ is

$$\Gamma_{\boldsymbol{g}} = \mathbb{E}(\boldsymbol{g}_\lambda(0)\boldsymbol{g}_\lambda(0)^T) = \sum_{k\in S_b} b(k) \sum_{l\in S} \boldsymbol{\alpha}(l)\boldsymbol{\alpha}^T(l+k) \in \mathbb{R}^{3\times 3},$$

where S is the support of $\boldsymbol{\alpha}$ and $S_b = \{-1,0,1\}^2$ is the support of b. The diagonal coefficients of these two matrices are the variance of each RGB channel while the off-diagonal coefficients are the cross-correlations between channels. These correlations are important perceptually and ideally we want the two matrices $\Gamma_{\boldsymbol{u}}$ and $\Gamma_{\boldsymbol{g}}$ to be equal. However, due to the support projection that puts some coefficients to zero, the variance of each color channel is always smaller than the one of the original image. This loss of

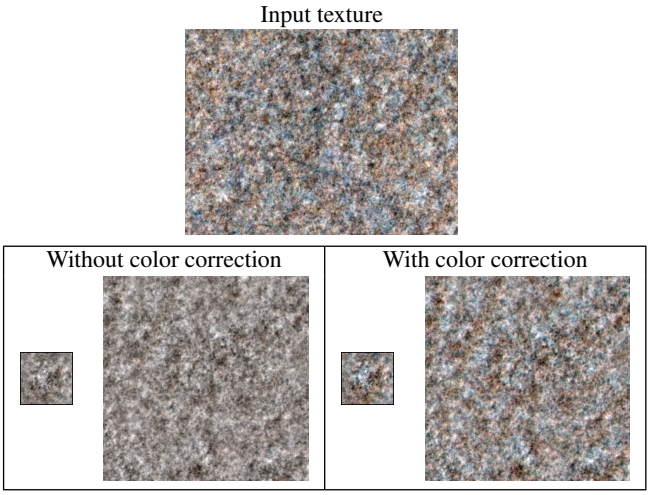

Figure 1: *Color correction of an RGB texton. Top: input Gaussian texture; bottom left: texton without color correction and associated noise; bottom right: color corrected texton and associated noise. Remark that the color content of the second noise is closer to the one of the input texture.*

variance translates visually into textures with less contrast and is not desirable. We thus perform the linear color correction proposed in [GLM14, DMR15] once the coefficients are computed:

$$\forall k \in S, \quad \boldsymbol{\alpha}(k) \leftarrow S_{\boldsymbol{u}} S_{\boldsymbol{g}}^{-1} \boldsymbol{\alpha}(k)$$

where $S_{\boldsymbol{u}} = \Gamma_{\boldsymbol{u}}^{\frac{1}{2}}$ and $S_{\boldsymbol{g}} = \Gamma_{\boldsymbol{g}}^{\frac{1}{2}}$ are the unique positive semidefinite square root matrices of $\Gamma_{\boldsymbol{u}}$ and $\Gamma_{\boldsymbol{g}}$. This linear correction ensures that the covariance of the color texton noise $\boldsymbol{g}_\lambda$ is equal to the cross-correlation of the original texture $\boldsymbol{u}$. Figure 1 illustrates the effect of this color correction. Although the visual change due to color correction is often not as noticeable as for the example of Figure 1, color correction is applied to all the results of the paper.

3. Comparison

3.1. Comparison with Noise by Example Methods

We compare texton noise results with Gabor noise by example [GLLD12] and Local Random Phase noise [GSV*14] in Figure 2. Texton noise visually reproduces the Gaussian version of any texture with a fast parameter-free analysis (see Algorithm 1 in the paper) and a faster evaluation algorithm. As can be seen with the last two rows of Figure 2, texton noise gives poor results when the input texture is not Gaussian. As said in the paper, texton noise is strictly limited to Gaussian textures, and thus cannot produce more structured textures contrary to LRP noise [GSV*14]. Hence, texton noise improves significantly the state of the art for noise by example applied to Gaussian textures, while LRP noise remains the unchallenged state of the art for the noise by example for structured textures.

3.2. Comparison with Image Quilting

We compare our results with the ones of image quilting [EF01]. Image quilting is a patch-based texture synthesis algorithm that is close to a tiling method, with the additional refinement that tile borders are merged seamlessly by computing an optimal boundary cut (using linear programming). This procedure generally gives better or similar results than the classical tiling methods discussed in Section 2.3 of the paper.

The results presented in Figure 3 have been obtained using the on-line demo associated to the preprint paper [RG16]. As one can see, image Quilting produces good results but is prone to repetitions (verbatim copy) and sometimes growing garbage (see the left side of the wood texture), as highlighted by the produced coordinate maps (see [RG16] for a more complete discussion on the limitations of the algorithm). With the Gaussian random fields approach of texton noise, such exact repetitions cannot occur and the resulting texture is by construction stationary, that is, statistically invariant by translation. Of course, in addition, texton noise produces a noise that is defined in the continuous domain $\mathbb{R}^2$ and not only on a pixel grid. However, once again, texton noise only produces Gaussian texture and cannot handle the rich variety of macro-textures that image quilting can reproduce.

4. Discussion Regarding Bilinear Interpolation

Let us now discuss the choice of the use of bilinear interpolation for texton noise.

4.1. Absence of Bilinear Interpolation Artefacts

The goal of this section is to stress that texton noise does not suffer from bilinear blocky artefacts. Blocky artefacts are present when zooming in a bilinearly interpolated image. However, by construction texton noise is obtained by summing around 30 such images that are randomly translated in the continuous domain. Consequently the blocky artefacts of each texton are not aligned and are no more visible (in the noise no individual texton is discernible). As illustrated by Figure 4, the accumulation of unaligned blocky artefacts only results in slight subpixel horizontal and vertical artefacts when zooming in. This slight artefact can be explained by the fact that the covariance of the Gaussian random field is an interpolation with the cubic spline kernel that is separable, and thus favors the horizontal and vertical directions.

4.2. Using Higher Order Interpolation

The whole analysis of Section 4 relies on very few hypotheses on the interpolation kernel ψ, namely that ψ is symmetric at the origin ($\tilde{\psi} = \psi$) and its integral is 1. This is satisfied by any $\boldsymbol{B}$-spline interpolation kernel. One can thus wonder if it would be legitimate to use $\boldsymbol{B}$-spline interpolation with an order different from 1.

For $\boldsymbol{B}$-spline interpolation of order 0, that is, nearest neighbor interpolation, one has $\hat{b} = 1$, and thus the texton coefficients α are exactly equal to the discrete "synthesis oriented texton" recently introduced in [GLM14]. However, using nearest neighbor interpolation is not satisfying for the noise generation since the obtained textures present grid discontinuities along the texton coefficient grid

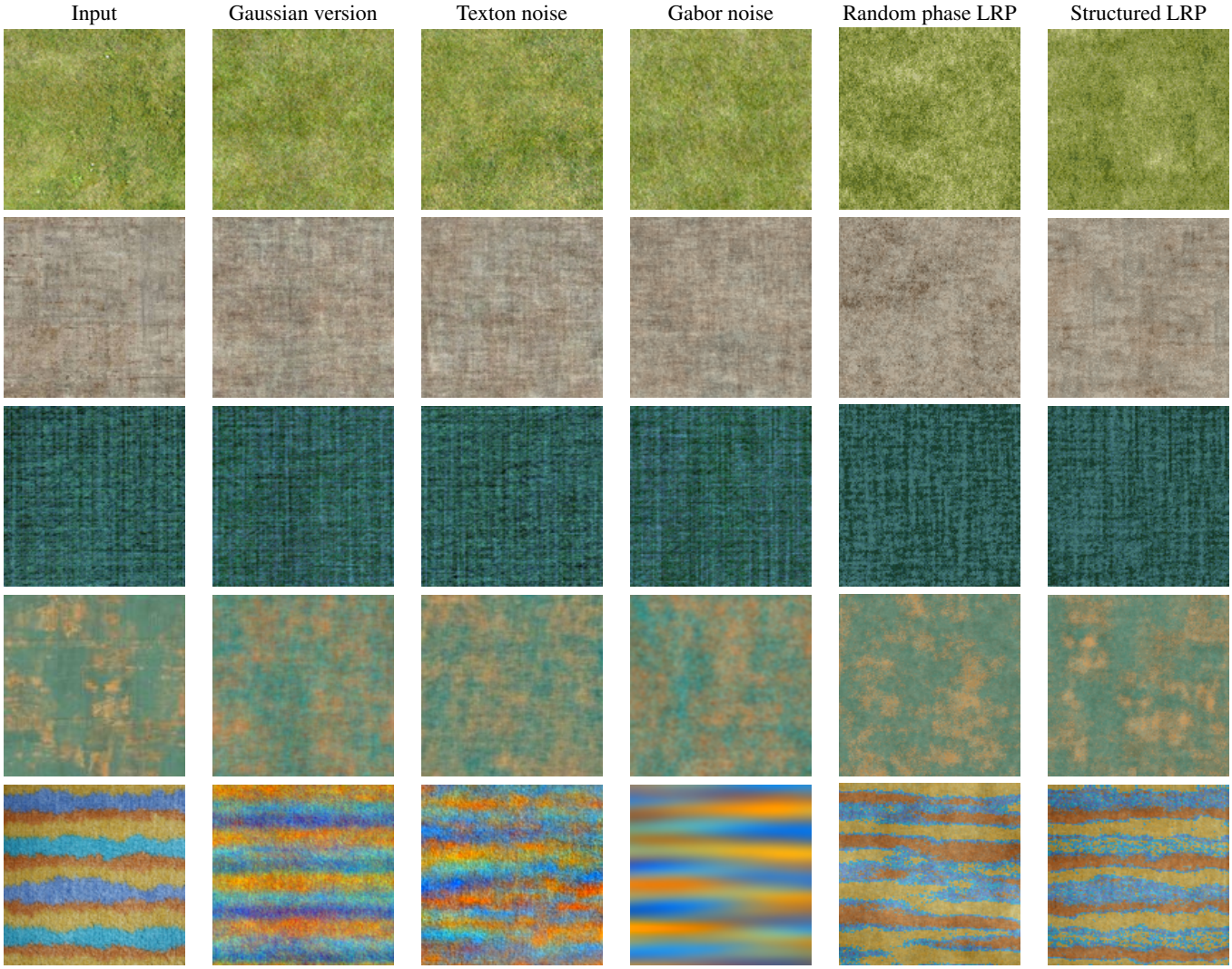

Figure 2: ***Comparison with noise by example methods:*** *Comparison of texton noise with Gabor noise by example [GLLD12] and local random phase noise [GSV*14] (with random phase and structured noise). Observe that texton noise reproduces the Gaussian version of all textures and that it gives results as good as Gabor noise by example and LRP while being respectively two and one order of magnitude faster than these methods.*

while continuity is generally a strict requirement for a procedural noise function.

Figure 5 presents texton noise simulation with different interpolation orders k. As one can observe, when k increases, the noise appears smoother since $\boldsymbol{B}$-spline interpolation kernel of order k produces images that are continuous and $k-1$ times differentiable. One can notice that for irregular textures (like the second one of Figure 5) and interpolation orders $k > 1$, ringing artifacts may appear in the output noise, whereas this is not the case for $k = 1$ (that is, bilinear interpolation, the interpolation used in the paper). This experiment thus raises a win-win situation since on one hand bilinear interpolation gives best visual results and on another hand it is natively supported by standard GPU texture fetching routines and accompanied with hardware solution for filtering. However, using texton noise with higher interpolation order may be of interest for certain applications of procedural noise where regularity is crucial, such as surfaces bumps with continuous normals.

References

[DMR15] DESOLNEUX A., MOISAN L., RONSIN S.: A texton for random phase and Gaussian textures. 2015. 2

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *SIGGRAPH '01* (New York, NY, USA, 2001), ACM, pp. 341–346. doi:10.1145/383259.383296. 2, 4

[GGM11] GALERNE B., GOUSSEAU Y., MOREL J.-M.: Random phase textures: Theory and synthesis. *IEEE Trans. Image Process. 20*, 1 (2011), 257 – 267. doi:10.1109/TIP.2010.2052822. 1

[GLLD12] GALERNE B., LAGAE A., LEFEBVRE S., DRETTAKIS G.:

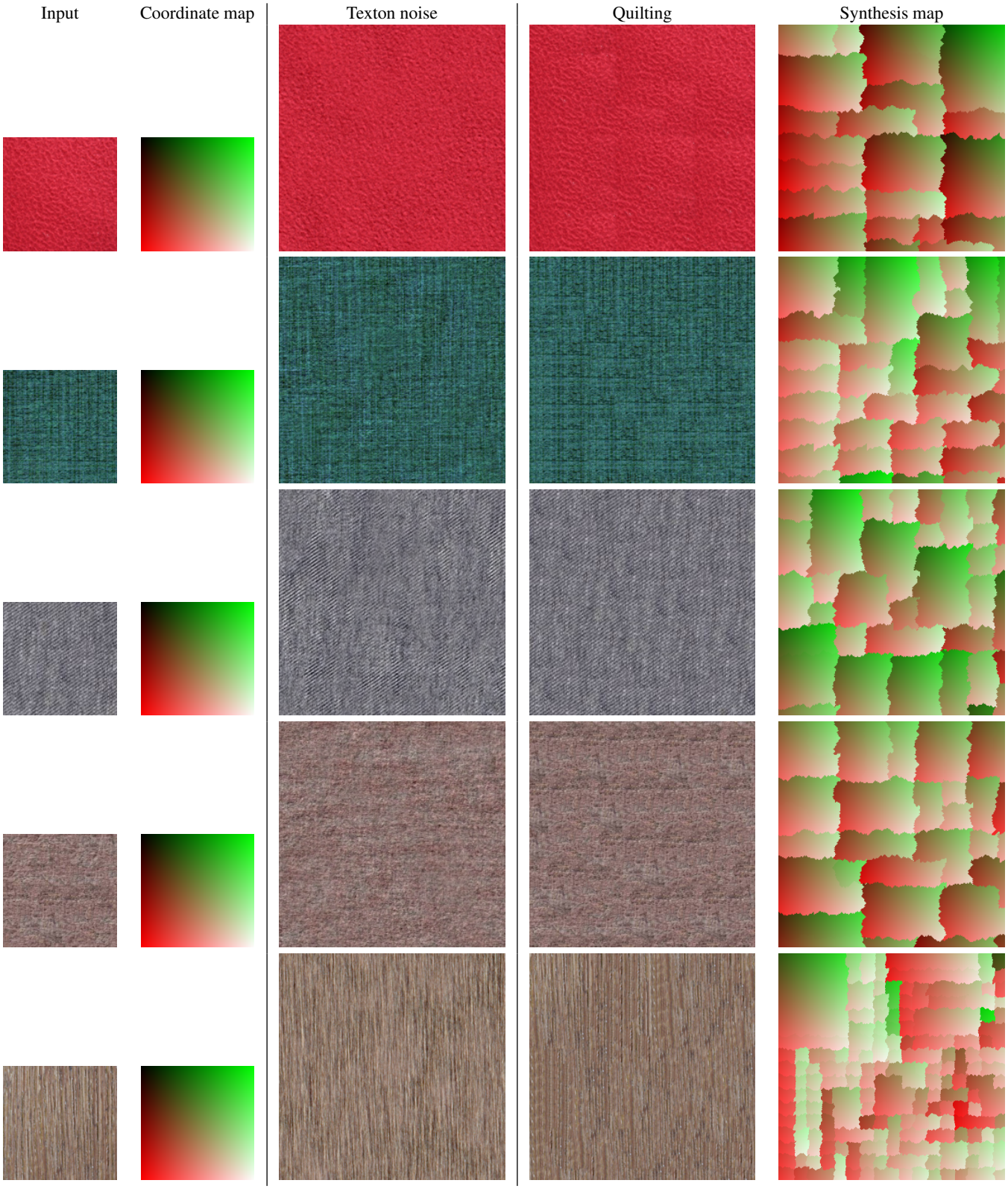

Figure 3: ***Comparison with image quilting*** *[EF01]: From left to right: Input, coordinate map of the input, texton noise on a twice larger domain, image quilting results on a twice larger pixel grid, and synthesis map to define the image quilting result. Image quilting can be used to generate various textures, and most particularly macro-textures. When used with Gaussian textures, the results often suffer from verbatim copy and sometimes growing garbage due to the raster scan order, while texton noise does not.*

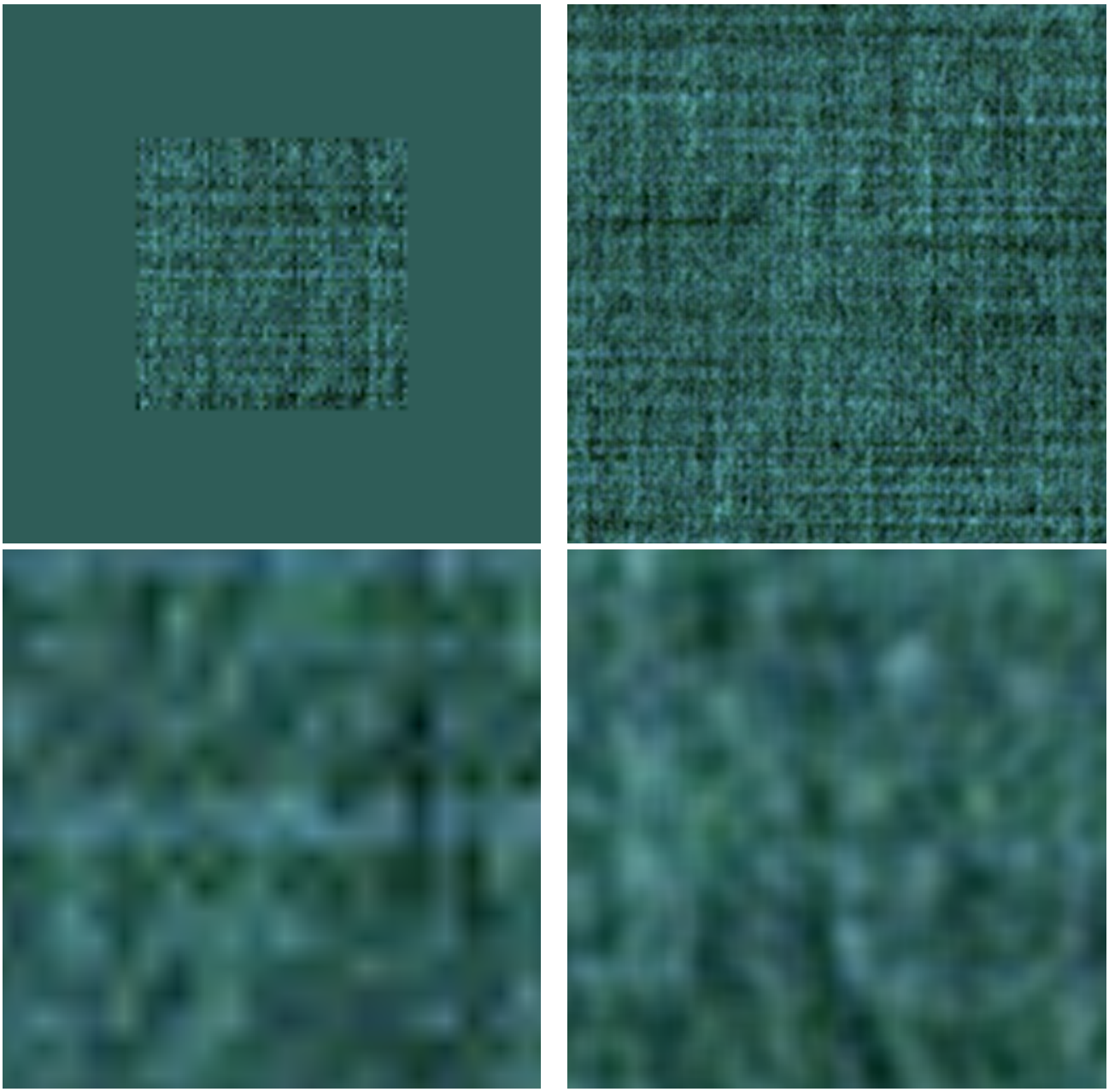

Figure 4: ***Absence of bilinear interpolation artefacts:*** *First row: A bilinearly interpolated texton with a subsampling of factor 20 (left) and corresponding texton noise at the same scale (right); Second row: close-up views of the center of both images. Although an individual texton present the usual bilinear interpolation "blocky" artefacts, texton noise does not present the same artefacts since the numerous textons summed to obtained the noise are not aligned.*

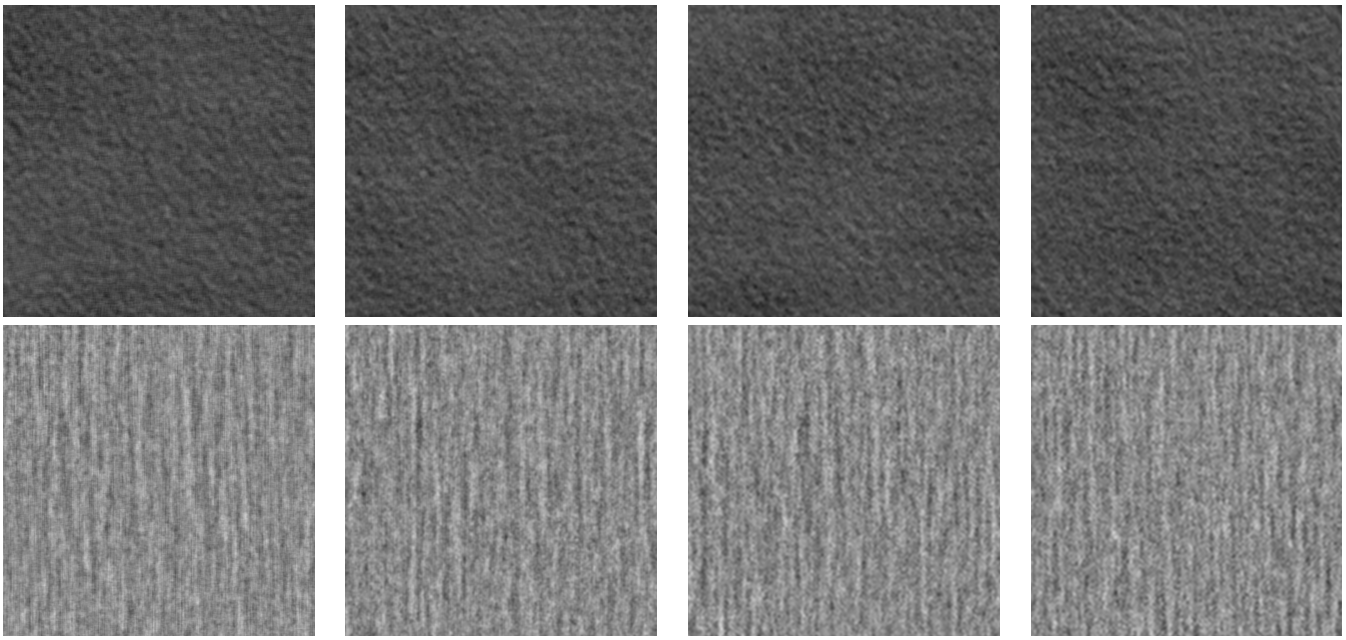

Figure 5: ***Influence of interpolation order:*** *In this figure we compare texton noise with interpolation order $k = 0, 1, 3, 5$. All images have the same scale than the input 128×128 images with a subsampling of a factor 20. As expected, directional artifacts appear with $k = 0$ because of the texton grid discontinuity. Also, for the example of the second row, ringing artifacts are slightly noticeable for $k = 3$ or $k = 5$ (please zoom in to observe the artefacts).*

Gabor noise by example. *ACM Trans. Graph. 31*, 4 (jul 2012), 73:1–73:9. doi:10.1145/2185520.2185569. 2, 3

[GLM14] GALERNE B., LECLAIRE A., MOISAN L.: A texton for fast and flexible Gaussian texture synthesis. In *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)* (2014), pp. 1686–1690. 2

[GSV*14] GILET G., SAUVAGE B., VANHOEY K., DISCHLER J.-M., GHAZANFARPOUR D.: Local random-phase noise for procedural texturing. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA 2014) 33*, 6 (nov 2014), 195:1–195:11. doi:10.1145/2661229.2661249. 2, 3

[Lec15] LECLAIRE A.: *Random Phase Fields and Gaussian Fields for Image Sharpness Assessment and Fast Texture Synthesis*. PhD thesis, Université Paris Descartes, June 2015. URL: https://tel.archives-ouvertes.fr/tel-01196693/en. 1

[RG16] RAAD L., GALERNE B.: Efros and Freeman image quilting algorithm for texture synthesis. *Submitted to Image Processing On Line* (2016). 2

[TGP14] TARTAVEL G., GOUSSEAU Y., PEYRÉ G.: Variational texture synthesis with sparsity and spectrum constraints. *Journal of Mathematical Imaging and Vision 52*, 1 (2014), 124–144. doi:10.1007/s10851-014-0547-7. 1