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 [GGM11b], 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

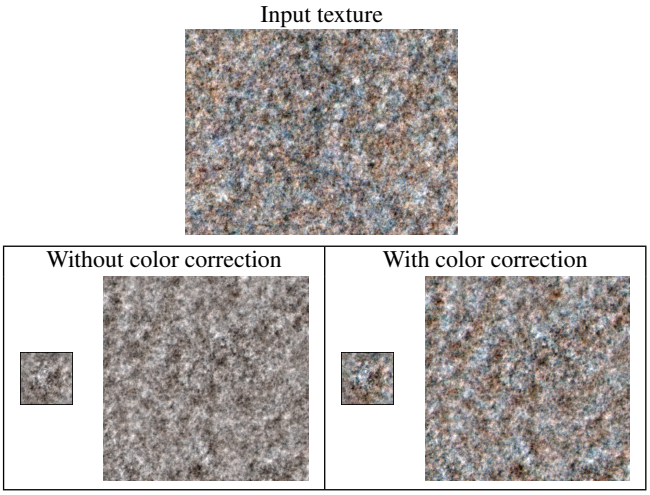

Figure 1: *Color correction of an RGB texton. Top: input Gaussian texture; bottom left: texton without color correction and associated noise; bottom right: color corrected texton and associated noise. Remark that the color content of the second noise is closer to the one of the input texture.*

variance translates visually into textures with less contrast and is not desirable. We thus perform the linear color correction proposed in [GLM14, DMR16] 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 on Texton Alternatives

The texton computation algorithm (see Algorithm 1 of the paper) summarizes the whole spectral content of the input texture into a texton having small support (and it also applies a frequency filter to avoid the low pass filtering effect of bilinear interpolation). The relative complexity of this iterative algorithm invites to look for more naive and quicker solutions to sum up the content of the texture. Figure 4 shows that taking simply a crop of either the original texture of its Gaussian version is not sufficient to reproduce faithfully the spectral content of the texture. Indeed any inhomogeneous pattern in the crop region leads to long scale distortions in the noise. Note that this will presumably be the same if one uses the output of some patch-based compact summary of the texture [WHZ*08, SCSI08] instead of the proposed texton.

5. Failure Cases with Non Gaussian Textures

As mentioned in the paper, texton noise is by construction limited to Gaussian textures. This means that if a texture is not faithfully reproduced by its Gaussian version, then the results of texton noise will not be satisfying. In fact, as shown by Figure 5, when using a large size for the texton support, texton noise enables to synthesize the Gaussian version of any input texture. We refer to the original papers on Gaussian textures [GGM11b, GGM11a] for more illustration of failure examples for the Gaussian model. In particular any texture containing sharp edges or small discernible objects will not be faithfully reproduced. Note that one can try the online demo accompanying the paper [GGM11a] to see if a texture will be well-reproduced.

References

[DMR16] DESOLNEUX A., MOISAN L., RONSIN S.: A texton for random phase and Gaussian textures. 2016. 2

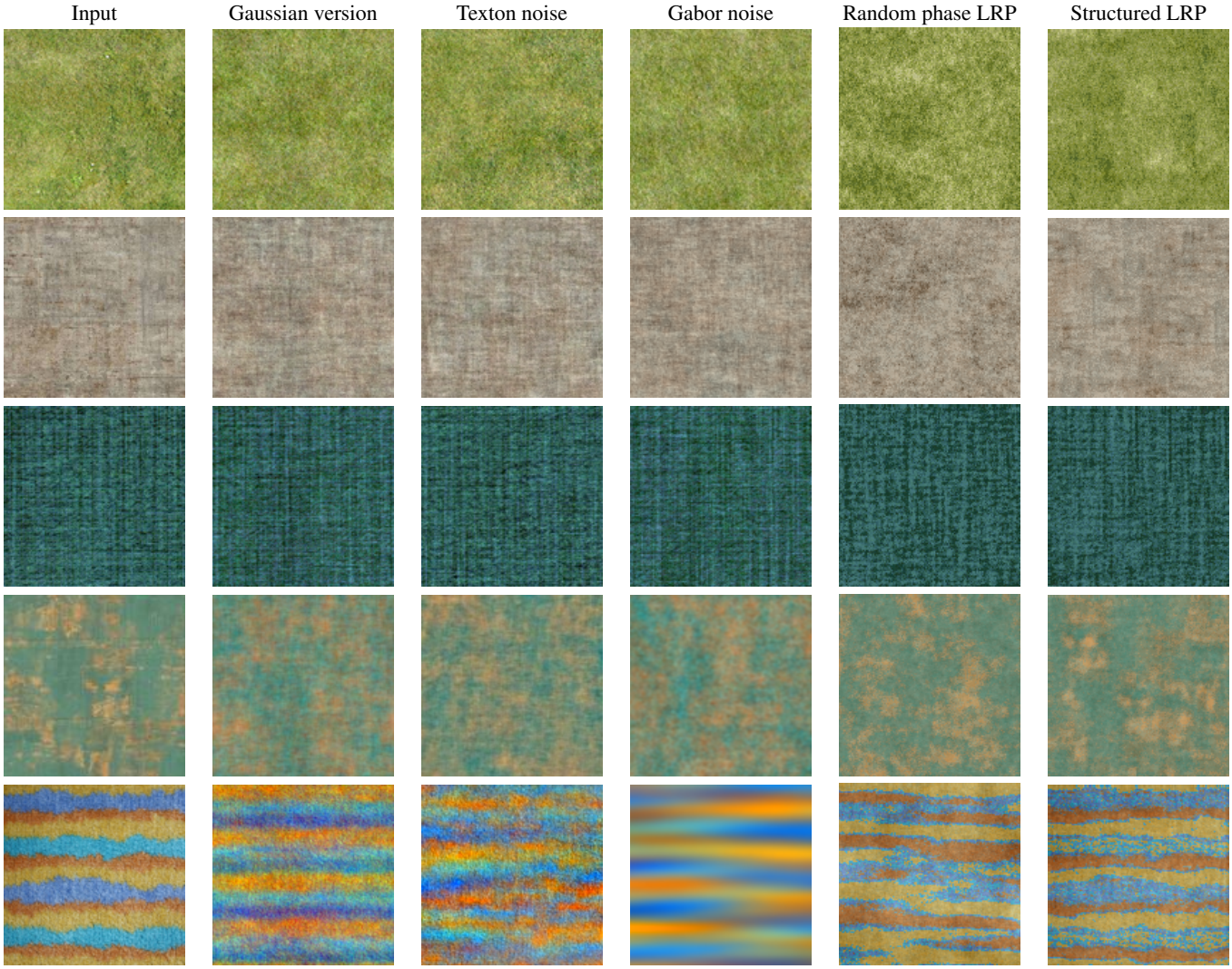

Figure 2: ***Comparison with noise by example methods:*** *Comparison of texton noise with Gabor noise by example [GLLD12] and local random phase noise [GSV*14] (with random phase and structured noise). Observe that texton noise reproduces the Gaussian version of all textures and that it gives results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

[EF01] 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* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 341–346. doi:10.1145/383259.383296. 2, 4

[GGM11a] GALERNE B., GOUSSEAU Y., MOREL J.-M.: Micro-texture synthesis by phase randomization. *Image Processing On Line 1* (2011). doi:10.5201/ipol.2011.ggm_rpn. 2

[GGM11b] 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, 2

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

[SCSI08] SIMAKOV D., CASPI Y., SHECHTMAN E., IRANI M.: Summarizing visual data using bidirectional similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008),

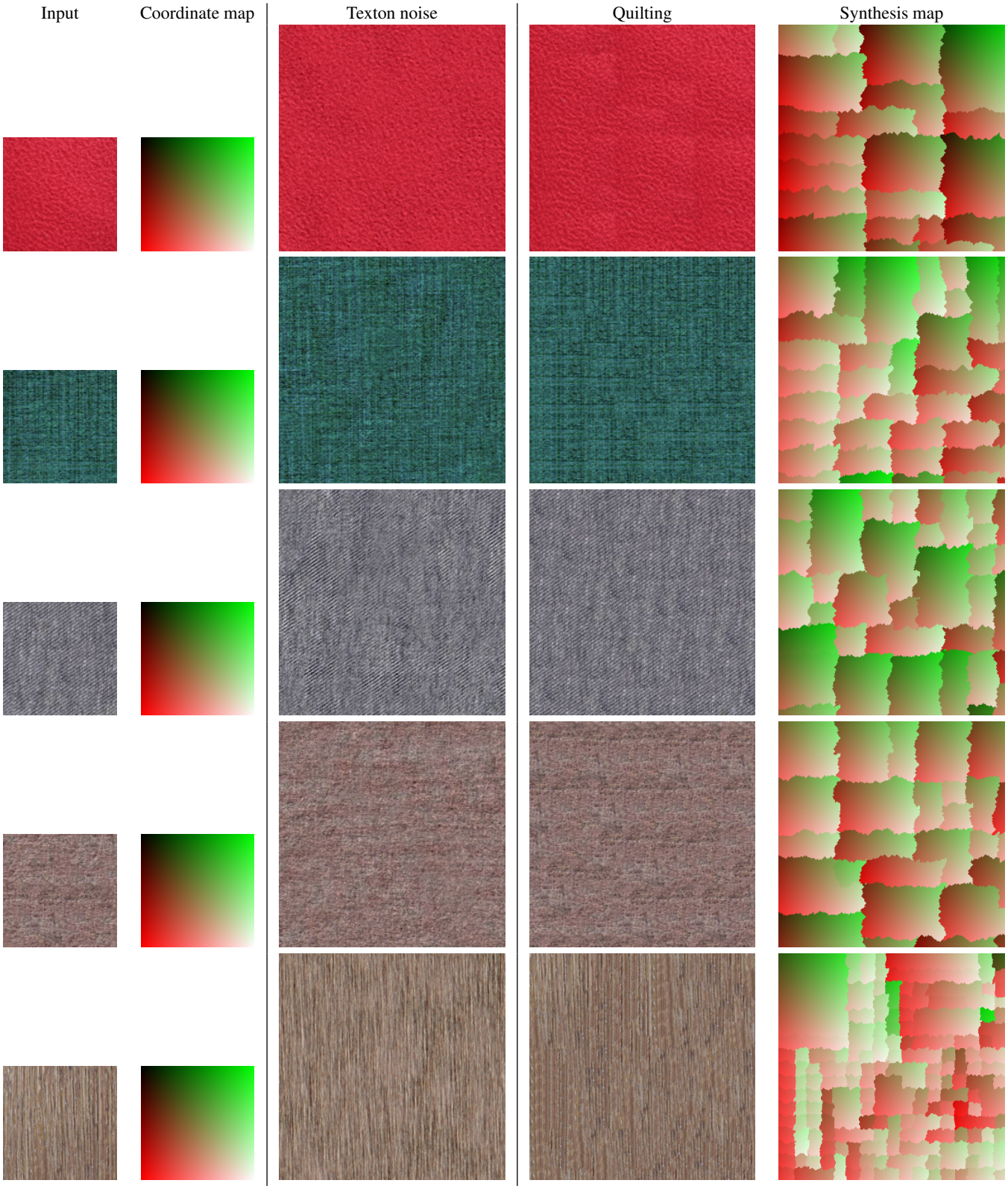

Figure 3: ***Comparison with image quilting*** *[EF01]: From left to right: Input, coordinate map of the input, texton noise on a twice larger domain, image quilting results on a twice larger pixel grid, and synthesis map to define the image quilting result. Image quilting can be used to generate various 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.*

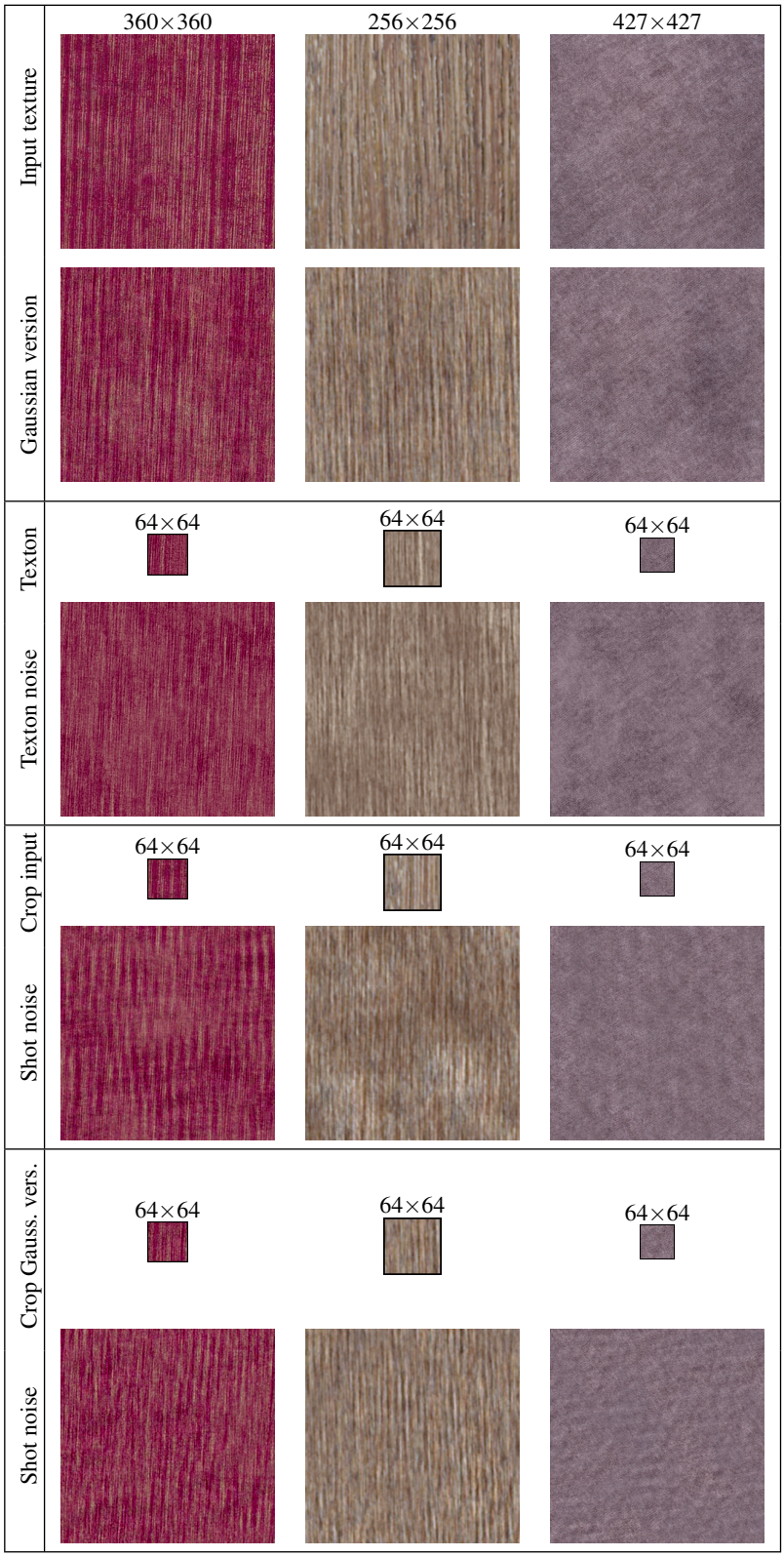

Figure 4: ***Naive alternatives to texton noise:*** *This figure illustrates two naive alternatives to texton noise by replacing the texton by a crop of the input texture (third row) and by replacing the texton by a crop of the Gaussian version of the texture (fourth row). One can observe that these naive crops do not produce satisfying results since they do not sum up faithfully the full spectrum of the input texture.*

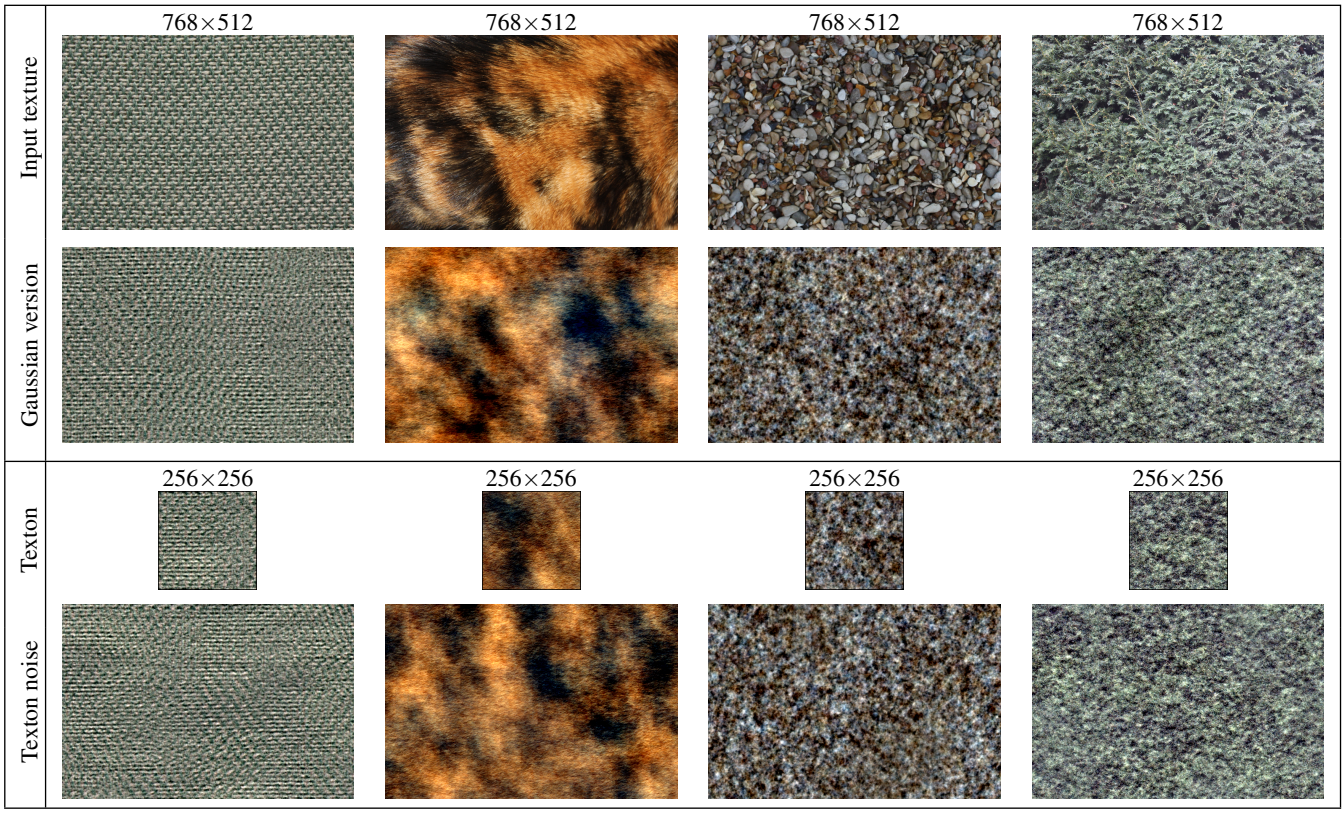

Figure 5: ***Failure cases for non Gaussian textures:*** *Several synthesis results with non Gaussian texture input. First row: Input texture (with size); Second row: Gaussian version of the texture; Third row: Computed texton (with size); Fourth row: Texton noise. One can observe that texton noise enables to synthesize the Gaussian version of any texture.*

IEEE, pp. 1–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587842. 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

[WHZ*08] WEI L.-Y., HAN J., ZHOU K., BAO H., GUO B., SHUM H.-Y.: Inverse texture synthesis. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 52:1–52:9. doi:10.1145/1399504.1360651. 2