

# Gabor Noise by Example Supplemental Material

*Bruno Galerne\**

*Ares Lagae\**

*Sylvain Lefebvre*

*George Drettakis*

*\*joint first authors*

*Report CW 621, April 2012*



Katholieke Universiteit Leuven  
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# Gabor Noise by Example Supplemental Material

*Bruno Galerne\**

*Ares Lagae\**

*Sylvain Lefebvre*

*George Drettakis*

*\*joint first authors*

*Report CW 621, April 2012*

Department of Computer Science, K.U.Leuven

## Abstract

This technical report contains supplemental material for the article GALERNE, B., LAGAE, A., LEFEBVRE, S., AND DRETTAKIS, G. 2012. Gabor Noise by Example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)* 31, 4.

**Keywords :** procedural noise, Gaussian texture, Gaussian random field, power spectrum estimation, non-negative basis pursuit denoising, decorrelated color space

**CR Subject Classification :** I.3.7.

# Gabor Noise by Example Supplemental Material

Bruno Galerne\*

MAP5, Université Paris Descartes  
and CNRS, Sorbonne Paris Cité

Ares Lagae\*

KU Leuven

Sylvain Lefebvre

ALICE/INRIA Nancy Grand-Est

George Drettakis

REVES/INRIA Sophia-Antipolis

## Abstract

This technical report contains supplemental material for the article

GALERNE, B., LAGAE, A., LEFEBVRE, S., AND DRETTAKIS, G. 2012. Gabor Noise by Example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)* 31, 4.

## 1 Efficient Procedural Evaluation

### 1.1 Impulse Generation

In this section, we give more details on the generation of impulses (Sec. 6.3)

**Sampling the Poisson distribution** The procedural evaluation of Gabor noise, as introduced by [Lagae et al. 2009], uses the algorithm of Knuth [1997, 3.4.1] for sampling the Poisson distribution, which is designed for small means. This algorithm is appropriate for Gabor noise, but breaks down for bandwidth-quantized Gabor noise, which operates at much larger impulse densities. Instead, we generate random numbers distributed according to a Poisson distribution with mean  $\lambda$  using the Gaussian approximation  $P = \lfloor \lambda + \sqrt{\lambda}X + 1/2 \rfloor$ , where  $X$  is distributed according to the standard normal distribution. Note that the  $1/2$  is a *continuity correction* which accounts for the discrete nature of the Poisson distribution. We generate random numbers distributed according to the standard normal distribution using the basic form of Box-Muller transform  $X = \sqrt{-2 \ln(U_1)} \cos(2\pi U_2)$ , where  $U_1$  and  $U_2$  are distributed according to the standard uniform distribution. Note that the polar form of the Box-Muller transform form requires rejection sampling, which would slow down the evaluation.

**Seeding Strategy** The procedural evaluation of Gabor noise, as introduced by [Lagae et al. 2009], uses sequential seeds for the random number generators of neighboring cells, which results in highly correlated sequences of random numbers. This is not an issue for Gabor noise, since the algorithm of Knuth decorrelates the sequences by consuming a variable amount of random numbers, but it leads to artifacts for bandwidth-quantized Gabor noise, since the basic form of the Box-Muller transform consumes a fixed amount of random numbers. Instead, we seed the random number generator of each cell using an approach based on random number tables,  $\text{seed}(b, x, y) = P_b[b\%L] \oplus P_x[x\%L] \oplus P_y[y\%L]$ , where  $P_b$ ,  $P_x$  and  $P_y$  are random number tables of size  $L$  (typically 256), and  $\%$  and  $\oplus$  denote modulo and XOR. This method is inspired by Kensler et al. [2008]. However, they use permutation tables for  $P_x$  and  $P_y$ , which results in only  $L$  unique seeds for the  $L^2$  cells, and leads to artifacts in the power spectrum. This is not the case for our method: We have determined experimentally that for  $L = 256$  and 32-bit random numbers the average number of duplicates is less than 1.

\*Bruno Galerne and Ares Lagae are joint first authors

e-mail: bruno.galerie@parisdescartes.fr, ares.lagae@cs.kuleuven.be, sylvain.lefebvre@inria.fr, george.drettakis@inria.fr

### 1.2 Validation

We have validated the procedural evaluation by verifying that the power spectrum estimate of the procedural noise, obtained by averaging 100 periodograms, converges to the power spectrum determined by the noise parameters.

### 1.3 CUDA Implementation

Previous implementations of Gabor noise use GLSL (OpenGL Shading Language), and pass the noise parameters to the GLSL shader in uniform variables. However, the amount of storage for uniform variables on current GPU's is not always sufficient for bandwidth-quantized Gabor noise. Therefore, our implementation uses CUDA instead of GLSL, which is a bit slower, but also more flexible. Future GPU's will most likely re-enable the use of GLSL.

## 2 Interactive Noise Editing

### 2.1 Wold Decomposition

In this section, we give more details on the Wold-like decomposition for editing (Sec. 8)

We found the periodic component to be less useful for editing. Our analysis therefore starts by extracting a user chosen fixed number of evanescent components, adding each in its own group. We typically extract four components, and up to six on very structured spectra. All remaining Gaussians are added to the *random* group which is hidden from the user for clarity. The user may later decide to make it visible if he wishes to edit it as well. Evanescent components correspond to lines in the power spectrum. We extract them by performing a sparse Hough transform: We enumerate all pairs of Gaussians, each defining a line  $l$  in the power spectrum. During this process we only consider Gaussians of small bandwidth. We compute a score for each line based on the distance between the line and all other Gaussians. For a line  $l$  the score is computed as:

$$S(l) = \sum_{g_i \in \mathcal{G}} \frac{K_{g_i}}{\epsilon + d_l(\omega_{g_i})}$$

where  $\mathcal{G}$  is the set of all Gaussians below the bandwidth threshold (full width at half maximum below 0.06 in our implementation),  $K_{g_i}$  the Gaussian amplitude,  $\epsilon = 0.01$  and  $d_l(\omega_{g_i})$  measures the distance between the center of the Gaussian  $g_i$  and the line in the power spectrum. We keep the line with highest score as the first evanescent component, grouping all Gaussians in its proximity, that is all Gaussians whose center is closer to the line than a threshold (0.03 in practice). We re-iterate the process until the chosen number of components is found. This grouping approach is used throughout the accompanying video. The thresholds are fixed and are not exposed to the user.

### 3 Color

#### 3.1 Normalization of $\mathcal{J}$ -Values

We obtain a normalized value of  $\mathcal{J}(\mathbf{T})$  using

$$\mathcal{J}_n(\mathbf{T}) = \frac{3}{2} \frac{\mathcal{J}(\mathbf{T})}{\sum_{\xi} \|\mathbf{C}_{\mathbf{I}}(\xi)\|_F^2}, \quad (1)$$

where  $\|\cdot\|_F$  is the Frobenius norm. This normalization makes the  $\mathcal{J}$ -values invariant to an orthogonal color transform and linear change of contrast of  $\mathbf{I}$ . Additionally, it can be shown that the  $\mathcal{J}_n$  takes values between 0 and 1 by making use of the special structure of correlation matrices, namely that the matrix of the Fourier transform block-diagonalizes the  $3MN \times 3MN$  correlation matrix  $\Sigma(\xi_1, \xi_2) = \mathbf{C}_{\mathbf{I}}(\xi_1 - \xi_2)$  into  $3 \times 3$  rank-one blocks [Ferradans et al. 2011].

Our experiments show that for most of our exemplars  $0 \approx \mathcal{J}(\mathbf{T}_{\text{AJD}}) < \mathcal{J}(\mathbf{T}_{\text{PCA}}) \ll \mathcal{J}(\mathbf{T}_{\text{RGB}}) \approx 1$  (see `j_values.xls`), which implies that (i) the RGB color space is one of the worst possible color spaces for independent channel synthesis (as already informally observed by many authors in Computer Graphics), (ii) the difference between the PCA color space and the maximally independent color space is relatively small. Our maximally independent color space improves results over the PCA color space, but in practice this difference is barely noticeable because of (ii). However, this experiment (for the first time to our knowledge) does explain why the PCA color space has been so successful for stochastic textures in previous work.

### 4 Misc

#### 4.1 Image Resolution

We use an image resolution of  $128 \times 128$ , which we found to be sufficient for all our examples. However, note that the procedural evaluation is resolution and size independent (as illustrated in the interactive noise editor), and that the parameter estimation can be performed at a higher resolution if needed (although the  $\kappa$  parameter might have to be adjusted).

### References

- FERRADANS, S., XIA, G.-S., PEYRÉ, G., AND AUJOL, J.-F., 2011. Optimal transport mixing of gaussian texture models. <http://hal.archives-ouvertes.fr/hal-00662720/>.
- KENSLER, A., KNOLL, A., AND SHIRLEY, P. 2008. Better gradient noise. Tech. Rep. UUSCI-2008-001, SCI Institute, University of Utah.
- KNUTH, D. E. 1997. *The Art of Computer Programming*, 3rd ed., vol. 2. Addison-Wesley.
- KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2D exemplars. *ACM Trans. Graph.* 26, 3, 2:1–2:9.
- LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. 2009. Procedural noise using sparse Gabor convolution. *ACM Trans. Graph.* 28, 3, 54:1–54:10.