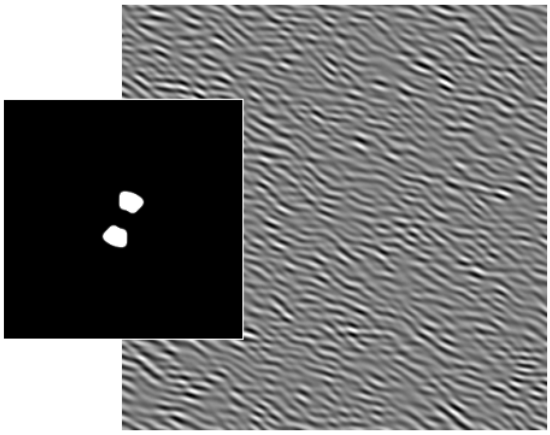
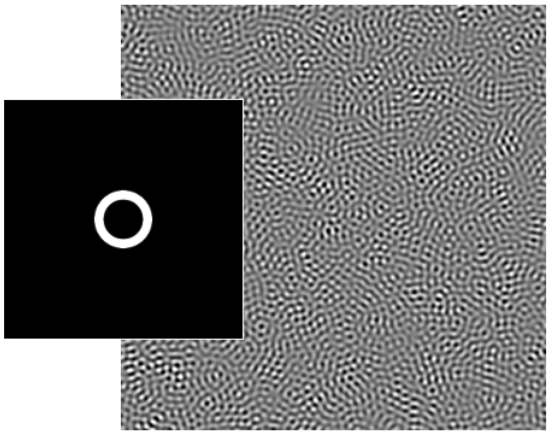
INF584 Project : Procedural Noise using Sparse Gabor Convolution

Capucine Leroux

March 2021





Contents

1 Introduction

This project consists in implementing the paper called 'Procedural Noise using Sparse Gabor Convolution' written by A. Lagae, S. Lefebvre, G. Drettakis and P. Dutre in 2009 [1]. The publication introduces a new procedural noise that is fast to evaluate, offers spectral control, anisotropic filtering and that is easy to evaluate on surfaces.

2 Gabor noise

2.1 Theory

2.1.1 Band-limited anisotropic noise

Sparse convolution

To have a good control over the noise appearance, here we want to have control over the power spectrum, which describes the contribution of each frequency band. For that, the paper uses a sparse convolution noise N, generalized here as a random pulse process :

$$N(x,y) = \sum_i w_i g(x - x_i, y - y_i) \tag{1}$$

where g is a pulse, the locations (x_i, y_i) follow a Poisson distribution of mean λ, which is the impulse density, and the weights w_i follow a random variable W. For this noise, $W \sim \mathcal{U}(-1,1)$.

Gabor kernel

The choice of the pulse is crucial. Here we choose the Gabor kernel because it is parameterized, it has a compact support in the spatial domain which implies good procedural evaluation, and it has a compact support in the frequency domain which means accurate control over the power spectrum. We have the Gabor kernel g:

$$g(x,y) = Ke^{-\pi a^2(x^2+y^2)} cos(2\pi F_0(x\cos w_0 + y \sin w_0))$$

The first part $Ke^{-\pi a^2(x^2+y^2)}$ is a gaussian enveloppe, with magnitude K and width a.
The second part $cos(2\pi F_0(x\cos w_0 + y \sin w_0))$ is a harmonic, with frequency magnitude F_0 and frequency orientation w_0.
These parameters are easy to understand and give a good control over the noise appearance. The radius r of the kernel $\simeq \frac{1}{a}$.

Variance and power spectrum

The Gabor kernel has a Fourier transform G :

$$G(f_x, f_y) = \frac{K}{2a^2}(e^{-\frac{\pi}{a^2}((f_x - F_0\cos w_0)^2 + (f_y - F_0 \sin w_0)^2)} + e^{-\frac{\pi}{a^2}((f_x + F_0\cos w_0)^2 + (f_y + F_0 \sin w_0)^2)})$$

Here, F_0 and w_0 have a fixed value for all impulses. The noise is anisotropic. In this case, the only randomized values in the noise are x_i, y_i, and w_i. Then we have the variance σ_N^2 of the noise N :

$$\sigma_N^2 = \lambda E(W^2) \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g^2(x,y)dxdy = \lambda \frac{1}{3}\frac{K^2}{4a^2}(1 + e^{\frac{2\pi F_0^2}{a^2}}) \tag{2}$$

The paper only gave the result with infinite integrals, see appendix 5.1 for the details on how I got the last formula, which I needed for implementation.
And the power spectrum S_N of our anisotropic noise N is :

$$S_N(x,y) = \lambda E(W^2)|G(x,y)|^2 = \frac{\lambda}{3}|G(x,y)|^2$$

We see that we can easily control the power spectrum with the intuitive parameters a, K, F_0 and w_0.

2.1.2 Generalization : noise with controllable band-limits

In the general case, each impulse g in the sum is different. They each have a F_{0i} and a w_{0i} randomly chosen in a limited region of frequency space : $[F_{0\min}, F_{0\max}] \times [w_{0\min}, w_{0\max}]$.
In that general case,

$$\sigma_N^2 = \frac{\lambda}{3}\frac{K^2}{4a^2}\frac{1}{F_{0\max} - F_{0\min}}\int_{F_{0\min}}^{F_{0\max}}(1 + e^{\frac{2\pi F_0^2}{a^2}})dF_0$$

$$S_N(x,y) = \frac{\lambda}{3}\frac{1}{F_{0\max} - F_{0\min}}\frac{1}{w_{0\max} - w_{0\min}}\int_{F_{0\min}}^{F_{0\max}}\int_{w_{0\min}}^{w_{0\max}}|G(x,y,w_0,F_0)|^2 \mathrm{d}w_0 \ \mathrm{d}F_0 \quad (3)$$

We talk about isotropic noise in the specific case where $F_{0\min} = F_{0\max}$ and $[w_{0\min}, w_{0\max}] = [0, 2\pi]$. We talk about anisotropic noise in the specific case where $F_{0\min} = F_{0\max}$ and $w_{0\min} = w_{0\max}$.

2.2 Implementation

A pseudo-code to evaluate the noise in 2D is given in the paper. I used this pseudo-code as a base. The method is the following :

- We use a 2D grid of cells of size $r \times r$ where $r \simeq \frac{1}{a}$.
- To evaluate the noise in (x, y), we sum over the noise value in the corresponding cell and in its neighbor cells.
- The value of the noise in a cell is found with equation 1. The number of impulses follows a Poisson law of mean λr^2 which is the number of impulses per cell.
- Uniform laws are simulated thanks to a pseudo random number generator. We seed the initial value with Morton order [2] added to a random offset, and then we use a linear congruential generator [3]. The Poisson law is implemented using the algorithm of Knuth for small means [4].

I also implemented the power spectrum to be able to see the influence of the parameters a, K, F_0 and w_0. I used the equation 3, with discretized integrals.

2.3 Results

I implemented two main functions to visualize the 2D Gabor noise. The first one evaluates the noise on the pixels of a 2D square image and find the corresponding level of grey for each pixel. Then saves both the noise black and white image and its power spectrum as ppm images. The second one opens an interactive window using OpenGL and VCL libraries, where the user can interactively visualize noise on a 2D square, changing the parameters, applying changes, and creating color noise interpolating a color scale or creating height noise.

2.3.1 Power spectrum control

The first function enables us to visualize the correlated influence of the parameters on the noise and its power spectrum.

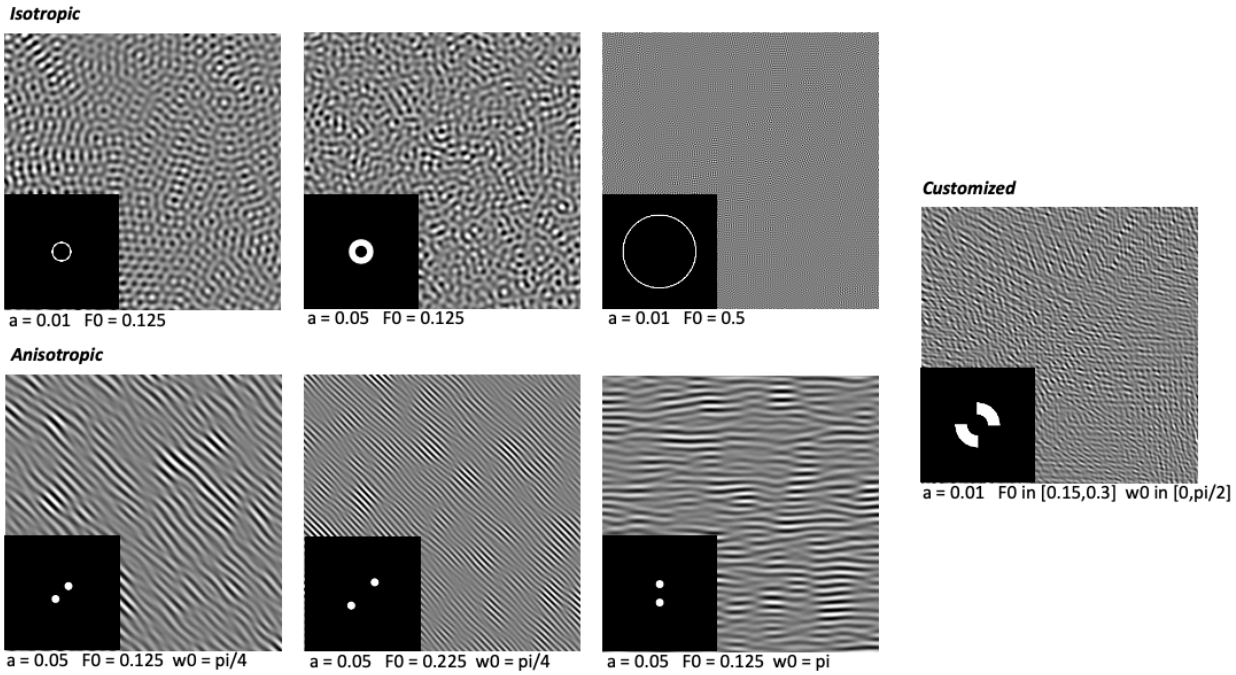


Figure 1: Influence of parameters on the noise and its power spectrum

In Figure 1, we can visualize well how a controls the gaussian width, F_0 the frequency magnitude and w_0 the frequency orientation.

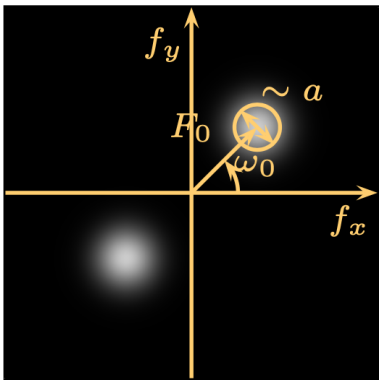


Figure 2: Parameters graphic meaning in spectral domain

2.3.2 Customizability

The second function, with its interactive window, is very useful to test how powerful this noise is. We can customize the look of the noise, either by giving it more or less height, or changing the color, or both. See examples of a variety of noises in Figure 3.

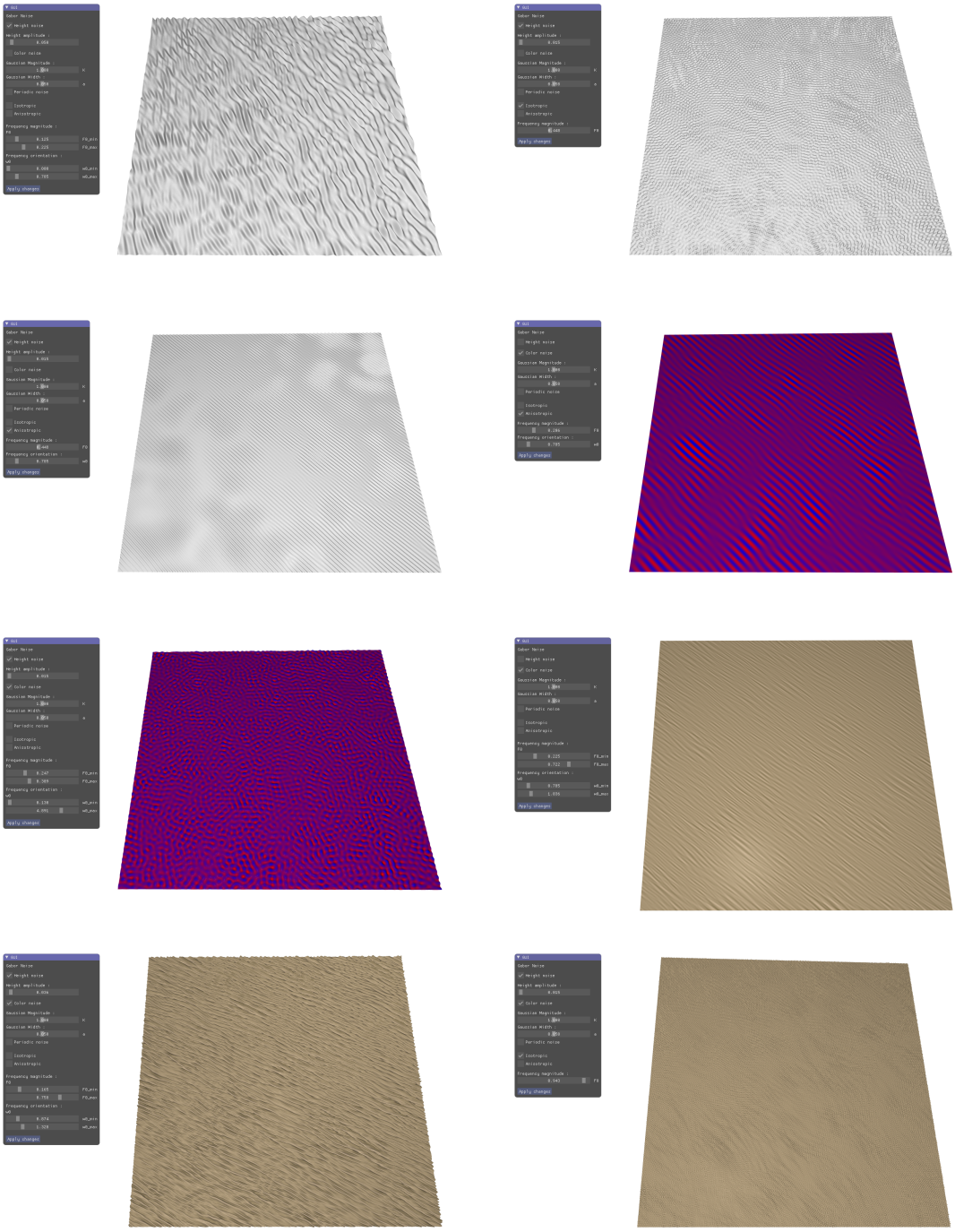


Figure 3: Interactive 2D noise window

2.3.3 Time and memory performance

One advantage of this noise is that it is computed on the fly, which means it does not require a lot of memory. It is also fast to evaluate : for a square of 500 by 500 vertices, my implementation takes about 1.5 seconds in average to compute a noise that is not necessarily isotropic or anisotropic.

3 Surface noise and anisotropic filtering

3.1 Surface noise

The publication gives an easy method to apply the 2D Gabor noise on a surface. The advantage of this is that we do not need a UV mapping system to evaluate the noise on the surface.

To evaluate on the surface the noise on the point p of normal n, the method consists in projecting the kernels on the plane defined by p and n. The system of cell grid is similar but in 3D. Inside a cell, the points are found in a 3D Poisson distribution and bounded in a cylinder. Then everything is projected on the local surface frame of the kernel, which is either randomly oriented for isotropic noise, or follows a vector field for anisotropic noise.

For more simplicity, I only tested the surface noise with isotropic noise, and kept projecting on the plane defined by (p, n), instead of computing a random orientation for each local surface frame.

Here (Figure 4) is an example with a 3D shape which has a UV mapping system. We can observe the difference between the mapping of the noise texture and the surface noise suggested by the paper.

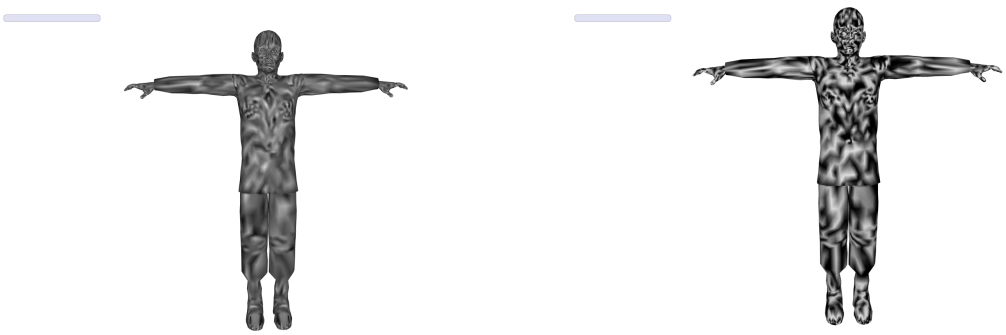


Figure 4: Difference between mapping (Left) and the paper surface noise (Right)

3.2 Anisotropic filtering

Last, the paper gives a way to filter anisotropically the kernels in order to get high quality rendering and avoid aliasing. Filtering the kernel means applying a convolution between the kernel and the filter in spatial domain, or multiplying the kernel with the filter in the frequency domain.

The anisotropic filter used is the one introduced by Heckbert in 1989 [5] :

$$f(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2\sigma^2}(x^2+y^2)}$$

in the spatial domain, or :

$$F(J^T[f_u, fv]^T) = e^{-2\pi^2\sigma^2[f_u,f_v]JJ^T[f_u,f_v]^T}$$

in the frequency domain. J is the Jacobian of the mapping from image to texture coordinates, and σ is the Gaussian width.
It turns out that applying the filter to a Gabor kernel is equivalent to modifying its parameters a, K, F_0 and w_0. For this, we only need to know J and σ.

I did not implement this part of the paper.

4 Conclusion

This noise has many good features that are usually very important when creating textures and using noises. It is procedural, fast to evaluate, does not take a lot of memory space, it can create isotropic, anisotropic or any noise in a very customizable way. The control the user have over the noise is very powerful, thanks to the spectral control Gabor kernels and sparse convolution offer. With very few intuitive parameters, the noise can be controlled easily. And last, it is easy to evaluate on surface, and to filter anisotropically to give a better quality of rendering.

One limit to my implementation is that I did not fully implement the surface noise and did not add the filtering feature. However, thanks to the interactive user interface, we can test the noise very easily, and we can also observe the power spectrum. I applied the noise of height and color, one other idea could be to test the noise on the roughness of the material for example.

5 Appendix

5.1 Details equation (2)

Calculation of σ_N^2 :

First, $E(W^2)$:
$W \sim \mathcal{U}[-1,1]$
$E(W^2) = \frac{1}{2}\int_{-1}^{1} w^2 \mathrm{d}w = \frac{1}{3}$

Second, $I = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g^2(x,y)dxdy$:
$I = K^2 \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{-A^2(x^2+y^2)}\cos^2(Bx+Cy)dxdy$
where $A^2 = 2\pi a^2$, $B = 2\pi F_0 \cos w_0$, $C = 2\pi F_0 \sin w_0$.
$I = \frac{K^2}{2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{-A^2(x^2+y2)}(\cos(2Bx+2Cy)+1)dxdy$
$I = \frac{K^2}{2}(\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{-A^2(x^2+y2)}dxdy + Re(\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{-A^2(x^2+y2)+2i(Bx+Cy)}dxdy))$
$I = \frac{K^2}{2}(\frac{\pi}{A^2} + \frac{\pi}{A^2}e^{-\frac{B^2+C^2}{A^2}}) = \frac{K^2}{4a^2}(1+e^{\frac{2\pi F_0^2}{a^2}})$

References

[1] *Procedural Noise using Sparse Gabor Convolution*
LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., DUTRE, P.
ACM Transactions on Graphics, 2009

[2] *A computer oriented geodetic data base and a new technique in file sequencing*
MORTON, G.
Tech. rep., IBM, 1966

[3] *Optimal multipliers for pseudo-random number generation by the linear congruential method*
BOROSH, I., NIEDERREITER, H.
BIT Numerical Mathematics, 1983

[4] *The Art of Computer Programming*
KNUTH, D. E.
1997

[5] *Fundamentals of Texture Mapping and Image Warping*
HECKBERT, P. S.
1989