Gabor Noise Implementation Report

Hao Ling - 161180076

***Abstract*—This is the report for the class assignment about implementating the Gabor Noise.**

I. Introduction

Gabor noise is a procedural noise generated by Sparse Gabor Convolution and first introduced by [1]. One is able to obtain anisotropic or isotropic Gabor noise of high quality within a short evaluating time, and control the appearance of Gabor noise by spectral control.

II. Principle

A. Gabor Kernel

The difinition of Gabor Kernel is

$$g(x, y) = Ke^{-\pi a^2(x^2+y^2)} \cos[2\pi F_0(x \cos \omega_0 + y \sin \omega_0)]$$

The Fourier transform of the Gabor Kernel is

$$G(f_x, f_y) = \frac{K}{2a^2}(e^{-\frac{\pi}{a^2}[(f_x-F_0\cos\omega_0)^2)+(f_y-F_0\sin\omega_0)^2]}+ e^{-\frac{\pi}{a^2}[(f_x+F_0\cos\omega_0)^2)+(f_y+F_0\sin\omega_0)^2]})$$

B. Isotropic Noise with anisotropic kernel

With a convolution, we can written down the isotropic Gabor noise generated by the anisotropic kernel:

$$N(x, y) = \sum_i w_i g(x - x_i, y - y_i; \omega_{0,i})$$

where $w_{0;i}$ are uniformly sampled from $[0, 2\pi)$, and x_i, y_i are governed by a Poisson distribution with mean λ.

C. Anisotropic Noise with anisotropic kernel

It is similar with the below one, the only difference is how $w_{0;i}$ are sampled. One can change $[0, 2\pi)$ to any interval $[A, B)$ to control the direction of Gabor noise.

D. Isotropic Noise with isotropic kernel

The isotropic Gabor kernel is defined as the product of a a Gaussian and a harmonic in spatial domain.

With some mathematical deduction, the n-dimention Gabor kernel is obtained.

$$g(r) = Ke^{-\pi a^2 r^2} \frac{2\pi F_0^{\frac{n}{2}}}{r^{\frac{n}{2}-1}} J_{\frac{n}{2}-1}(2\pi F_0 r)$$

The $J(x)$ is the order-n Bessel function of first kind.

I use the code from [2] to estimate the value of Bessel function.

III. Implementation

The toolchain used in the experiment is listed below.

- GCC with C++11 support to compile source code.
- opencv2.4 for GUI support.
- Mitsuba for rendering.

A. Interfaces

The base class GaborNoise is mainly consist of several important functions listing below.

- get_fourier(x, y): return the value of kernel's frequency domain at (x, y).
- get_kernel(x, y): return the value of kernel's spatial domain at (x, y).
- noise(x, y): return the value of noise at (x, y).

Listing 1. GaborNoise Interfaces

```
class GaborNoise {
private:
  virtual float_type gabor(
                    float_type K, float_type a, float_type F_0,
                    float_type omega_0, float_type x,
                    float_type y) const;
  virtual float_type fourier(float_type x, float_type y) const;
public:
  float_type get_fourier(float_type x, float_type y) const;
  float_type get_kernel(float_type x, float_type y) const;
  float_type noise(float_type x, float_type y);
}
```

There are also three derived classes overriding the virtual functions **gabor** and **fourier**.

- GaborNoise_anisotropic: producing anisotropic noise with anisotropic kernel.
- GaborNoise_isotropic: producing isotropic noise with anisotropic kernel.
- GaborNoise_isotropic_kernel: producing isotropic noise with isotropic kernel.

IV. Result

A. Speed Test

I have tested on my computer with cpu **Intel(R) Core(TM) i7-6500U @ 2.50GHZ**, using 4 threads. Since the running time of multithreads programs varies a lot, I only list the average running time below.

TABLE I
Speed Test

Impulses / cell	aniso	iso	iso_isokernel	size
64	0.4718s	0.5158s	0.4269s	$256 * 256$
128	0.7509s	0.9037s	0.6426	
256	1.5623s	1.6565s	1.3088s	
64	1.6646s	1.7880s	1.4999s	$512 * 512$
128	2.9524s	3.42392s	2.72044s	
256	5.6698s	6.9188s	5.2503s	

B. *Image result*

Use "make run" can start a interactive GUI, one can change the shape of the noise by setting the parameters, and **regenerate** the noise by pressing "s". Some amgiguous parameters are

- npc: to decide the number of impulses per cell.
- error: to decide the kernel radius, which is set to 5 by default so that the Gaussian envelope reaches 5% of its peak.
- type: 0 - anisotropic noise, 1 - isotropic noise with anisotropic kernel, 2 - isotropic noise with isotropic kernel.

Below the control panel is the generated image, the upper-left is the image of the noise, the upper-right is the image of the Gabor kernel, the lower-left is the image of the frequency domain of the noise, and the lower-right is the image of the frequency domain of the Gabor kernel.

Some images using the noise as texture are also shown.

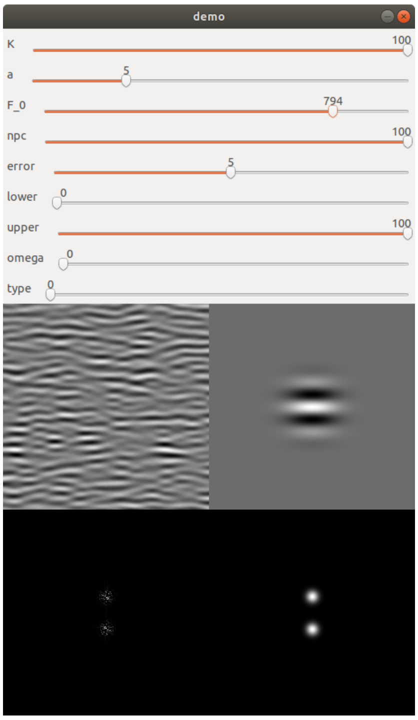

Fig. 1. anisotropic noise

References

[1] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. Procedural noise using sparse gabor convolution. In *ACM Transactions on Graphics (TOG)*, volume 28, page 54. ACM, 2009.

[2] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes in C*, volume 2. Cambridge university press Cambridge, 1996.

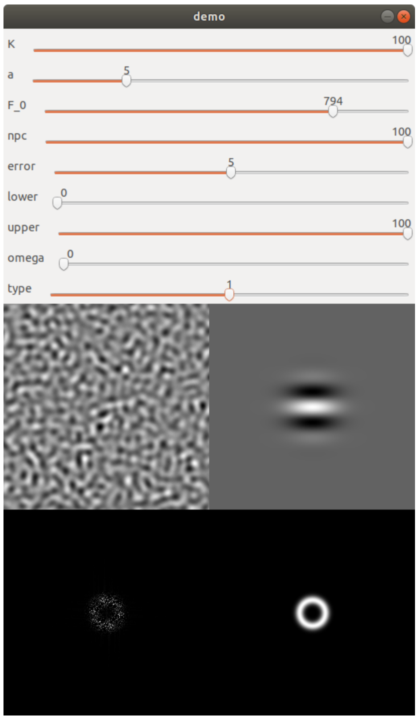

Fig. 2. isotropic noise with anisotropic kernel

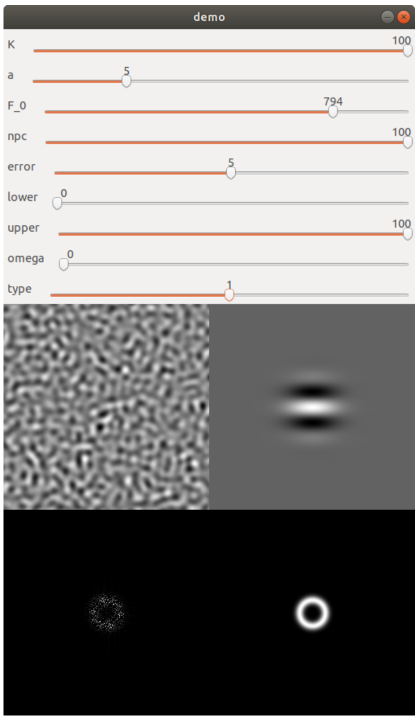

Fig. 3. isotropic noise with isotropic kernel

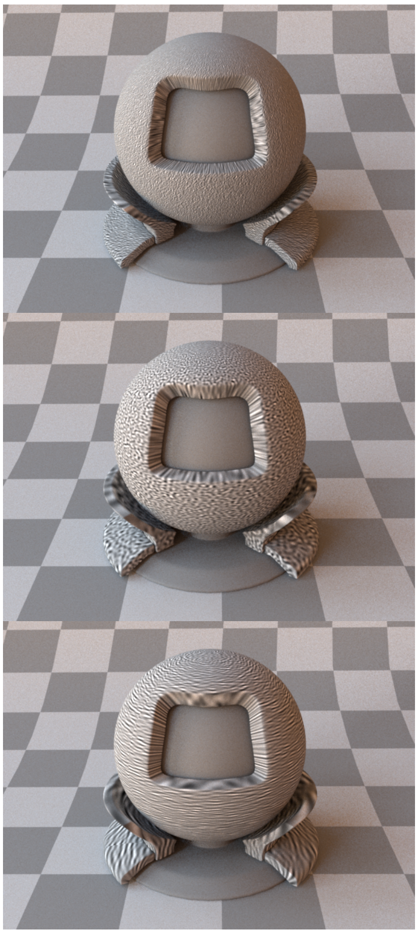

Fig. 4. Using the noise as texture