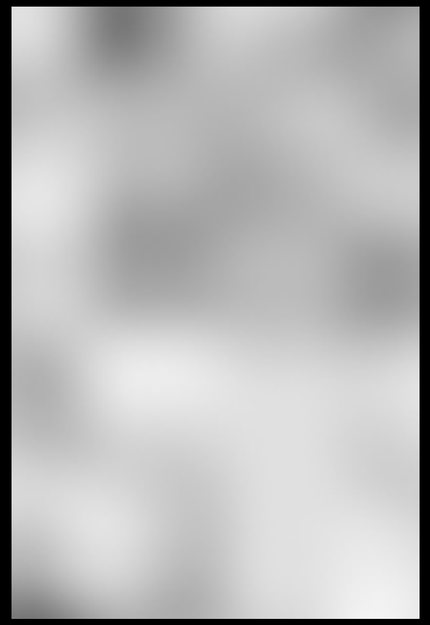
Fast Procedural Noise by Monte Carlo Sampling



Marcos Fajardo (Shiokara–Engawa Research)
Matt Pharr (NVIDIA)
EGSR 2023 Industry Track

Motivation

Rainforest, by Inigo Quilez

Motivation

Rainforest, by Inigo Quilez

Related Work

- Noise functions: [Perlin 85], [Lewis 89], [Worley 96], [Lagae et al. 09], …
- Making deterministic computation stochastic to improve performance:
  - Linear algebra: [Forsythe and Leibler 1950], [Shirley 1992], …
  - BSDF Lobes: [Széski et al. 2003]
  - Stochastic alpha: [Enderton et al. 2010], [Wyman and McGuire 2017]
  - Many lights: [Shirley et al. 1996], [Estevez and Kulla 2018]
  - Trilinear filtering: [Hofmann et al 2021]
  - Texture filtering: [Fajardo et al. 2023]
  - Gabor noise sums: [Galerne et al. 2012]

Building Blocks

Sampling Sums

$$S = \sum_i f_i \longrightarrow \langle S \rangle = \frac{f_j}{p_j} \qquad j \sim p$$

Sample Reuse [SWZ96]

$$s = \begin{cases} a & \text{if } \xi < p \\ b & \text{if } \xi \geq p \end{cases} \longrightarrow \begin{array}{l} \xi' = \xi / p \\ \xi' = (\xi - p)/(1 - p) \end{array}$$

Stochastic Linear Interpolation

$$lerp(f_0, f_1, t) = (1 - t)f_0 + tf_1$$

$$\langle lerp \rangle = \begin{cases} f_0, & \text{if } \xi > t \\ f_1 & \text{otherwise,} \end{cases}$$

Stochastic Bilinear Interpolation

$$\langle bilerp \rangle = \begin{cases} f_{00}, & \text{if } \xi_1 > x \text{ and } \xi_2 > y \\ f_{01}, & \text{if } \xi_1 > x \text{ and } \xi_2 \leq y \\ f_{10}, & \text{if } \xi_1 \leq x \text{ and } \xi_2 > y \\ f_{11}, & \text{if } \xi_1 \leq x \text{ and } \xi_2 \leq y \end{cases}$$

d-dimensional lerp: $2^d \times$ *savings*

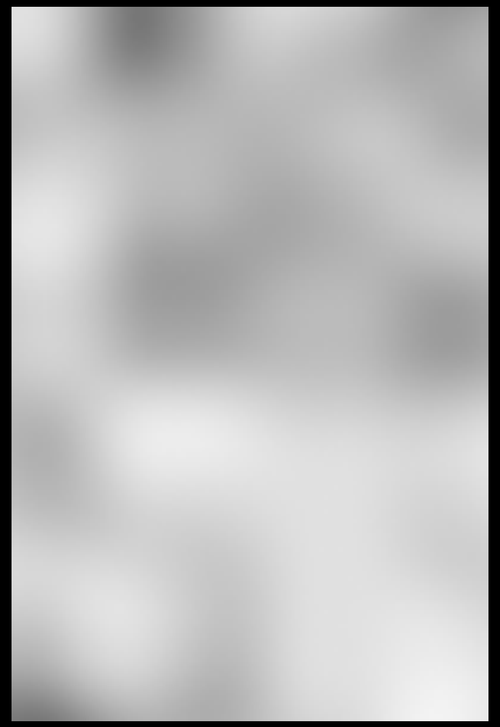
Stochastic Bilinear Interpolation

$$\langle bilerp \rangle = \begin{cases} f_{00}, & \text{if } \xi > x \text{ and } (\xi - x)/(1-x) > y \\ f_{01}, & \text{if } \xi > x \text{ and } (\xi - x)/(1-x) \leq y \\ f_{10}, & \text{if } \xi \leq x \text{ and } \xi/x > y \\ f_{11}, & \text{if } \xi \leq x \text{ and } \xi/x \leq y \end{cases}$$

d-dimensional lerp: $2^d \times$ *savings*

Perlin Noise (2D)

$$Perlin(p) = bilerp(g_{00}, g_{10}, g_{01}, g_{11}, fade(\{p_x\}), fade(\{p_y\}))$$

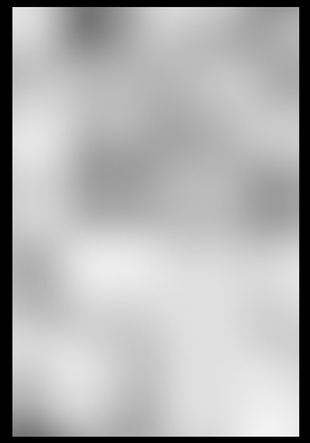


Fast Procedural Noise by Monte Carlo Sampling

Stochastic Perlin Noise (2D)

input: position (x, y) and a uniform random $\xi \in [0, 1)$
$x', y' \leftarrow x - \lfloor x \rfloor, y - \lfloor y \rfloor;$
$f_x, f_y \leftarrow fade(x'), fade(y');$
if $\xi > f_x$ **then**
  $\xi \leftarrow (\xi - f_x)/(1 - f_x);$
  if $\xi > f_y$ **then**
    return $grad(\lfloor x \rfloor, \lfloor y \rfloor, x', y');$
  else
    return $grad(\lfloor x \rfloor, \lfloor y \rfloor + 1, x', y' - 1);$
  end
else
  $\xi \leftarrow \xi / f_x;$
  if $\xi > f_y$ **then**
    return $grad(\lfloor x \rfloor + 1, \lfloor y \rfloor, x' - 1, y');$
  else
    return $grad(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1, x' - 1, y' - 1);$
  end
end

Reference



1 spp

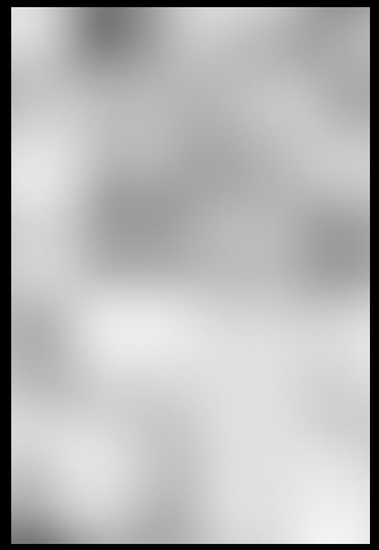
4 spp

16 spp

Fast Procedural Noise by Monte Carlo Sampling

Benefit of Blue Noise (Perlin 3D, 1spp)

Reference

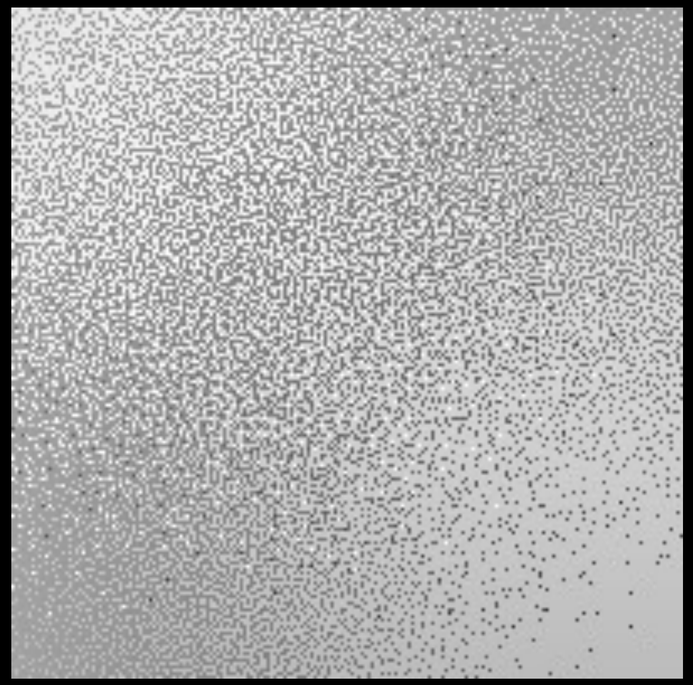


Independent

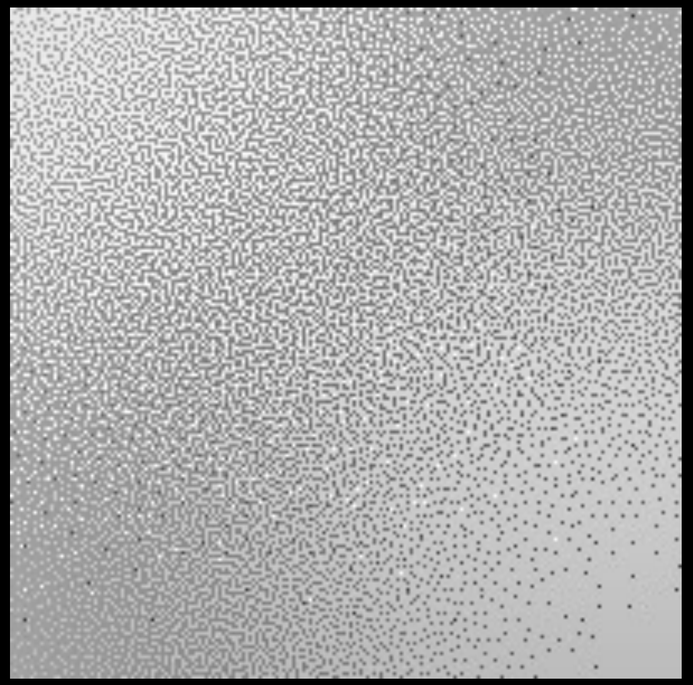
Blue Noise

Benefit of Blue Noise (Perlin 2D, 1spp)

2D Blue Noise

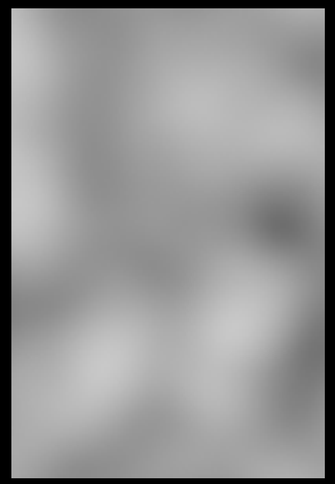


1D Blue Noise
(sample reuse)



Sparse Convolution Noise

$$SparseConv(p, n) = \frac{1}{n} \sum_{c=1}^{27} \sum_{i=1}^{n} w_{i,c}\, k(|p - p_{i,c}|)$$

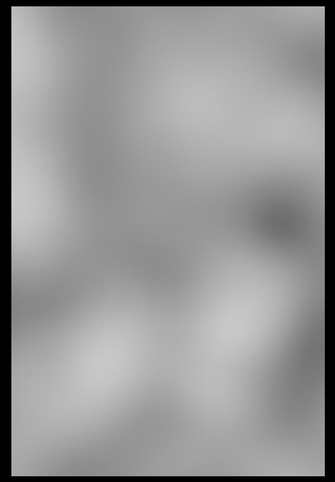


Reference
*27*n terms*

Fast Procedural Noise by Monte Carlo Sampling

Sparse Convolution Noise

$$SparseConv(p, n) = \frac{1}{n} \sum_{c=1}^{27} \sum_{i=1}^{n} w_{i,c}\, k(|p - p_{i,c}|)$$



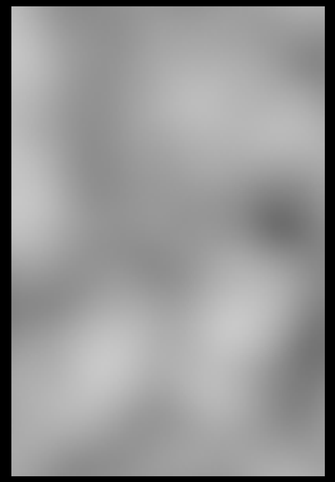
Reference
*27*n terms*

Sample both sums
1 term
(1 spp)

Fast Procedural Noise by Monte Carlo Sampling

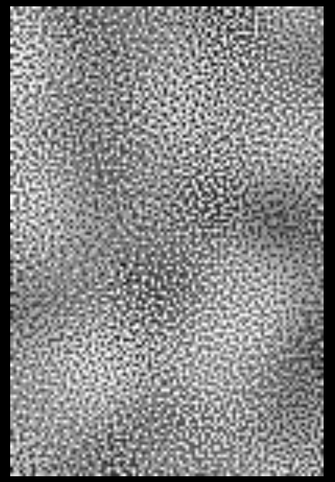
Sparse Convolution Noise

$$SparseConv(p, n) = \frac{1}{n} \sum_{c=1}^{27} \sum_{i=1}^{n} w_{i,c}\, k(|p - p_{i,c}|)$$



Reference
*27*n terms*

Sample both sums
1 term
(1 spp)



Sample inner sum
27 terms
(1 spp)

Fast Procedural Noise by Monte Carlo Sampling

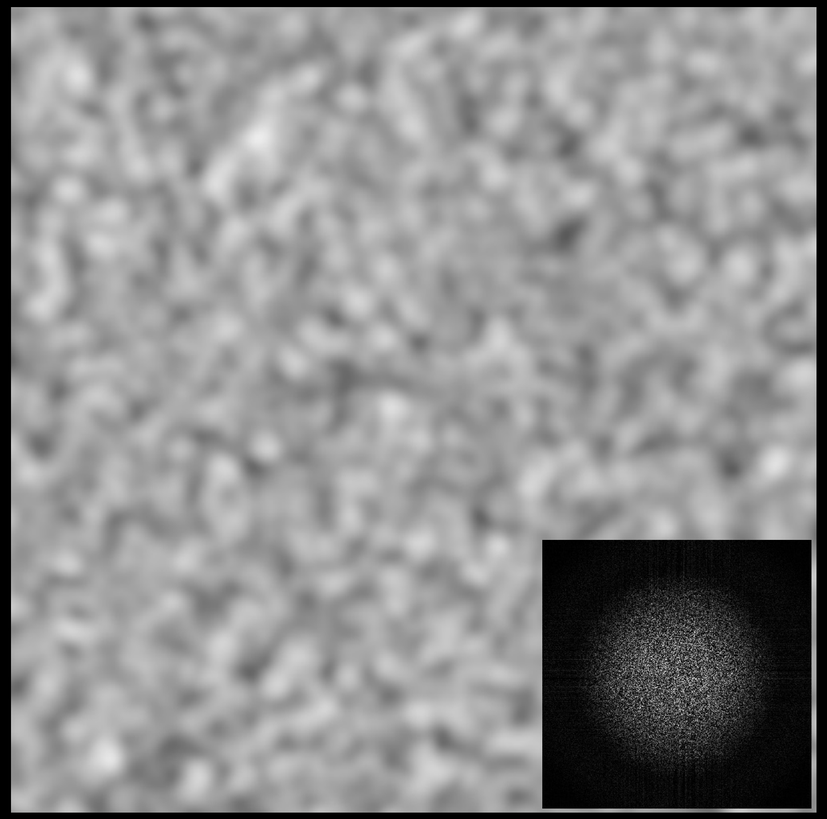
Sampling Sparse Convolution Noise

$$SparseConv(p, n) = \frac{1}{n} \sum_{c=1}^{27} \sum_{i=1}^{n} w_{i,c}\, k(|p - p_{i,c}|)$$

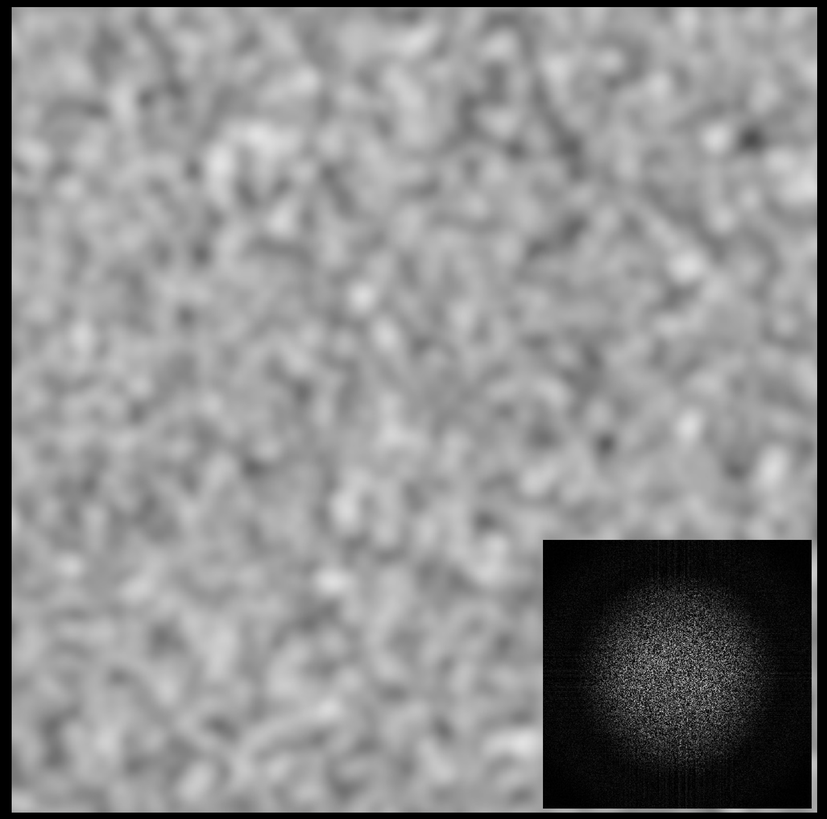
Stratified weights allow analytic importance sampling:

$$w_i = \frac{i + 1/2}{n} \longrightarrow p_i = \frac{w_i}{2} \longrightarrow i = \lceil n\sqrt{\xi} \rceil$$

Sampling Sparse Convolution Noise



Random weights



Stratified weights $w_i = \frac{i + 1/2}{n}$

Fast Procedural Noise by Monte Carlo Sampling

Multi-Octave Noise

$$Fractal(p, n, w) = \sum_{i=1}^{n} w^{i-1}\, Noise\left(\frac{p}{w^{i-1}}\right).$$

Multi-Octave Noise

$$Fractal(p, n, w) = \sum_{i=1}^{n} w^{i-1}\, Noise\left(\frac{p}{w^{i-1}}\right).$$

$$p_i = w^{i-1}\frac{1-w}{1-w^{n+1}} \longrightarrow i = \left\lfloor \frac{\log(1+\xi(w^{n+1}-1))}{\log w} \right\rfloor$$

Fast Procedural Noise by Monte Carlo Sampling

Multi-Octave Noise

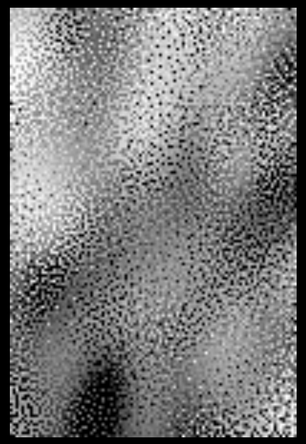
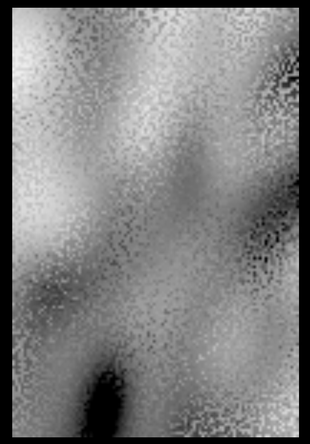
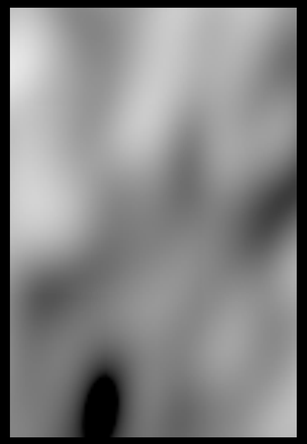
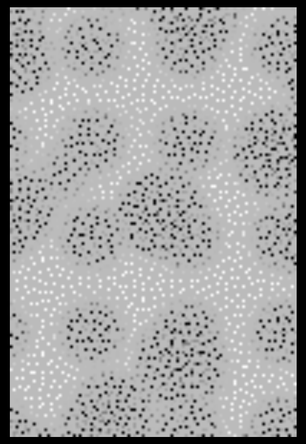
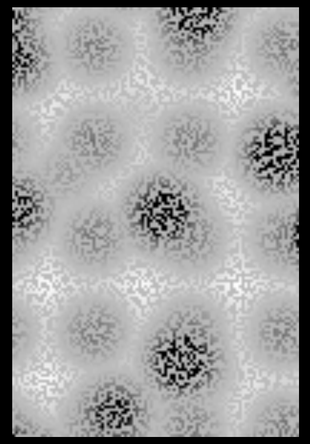
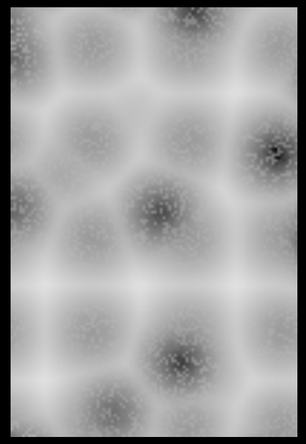
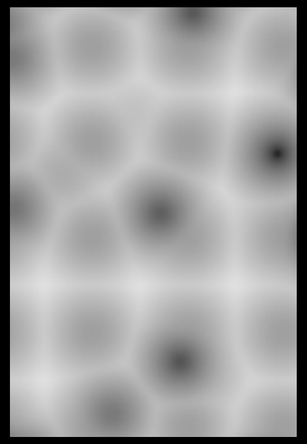
$$Fractal(p, n, w) = \sum_{i=1}^{n} w^{i-1}\, Noise\left(\frac{p}{w^{i-1}}\right).$$

$$p_i = w^{i-1}\frac{1-w}{1-w^{n+1}} \longrightarrow \cancel{i = \left\lfloor \frac{\log(1+\xi(w^{n+1}-1))}{\log w} \right\rfloor}$$

Better: linear search through the CDF of p_i

Fast Procedural Noise by Monte Carlo Sampling

Other Noise Functions

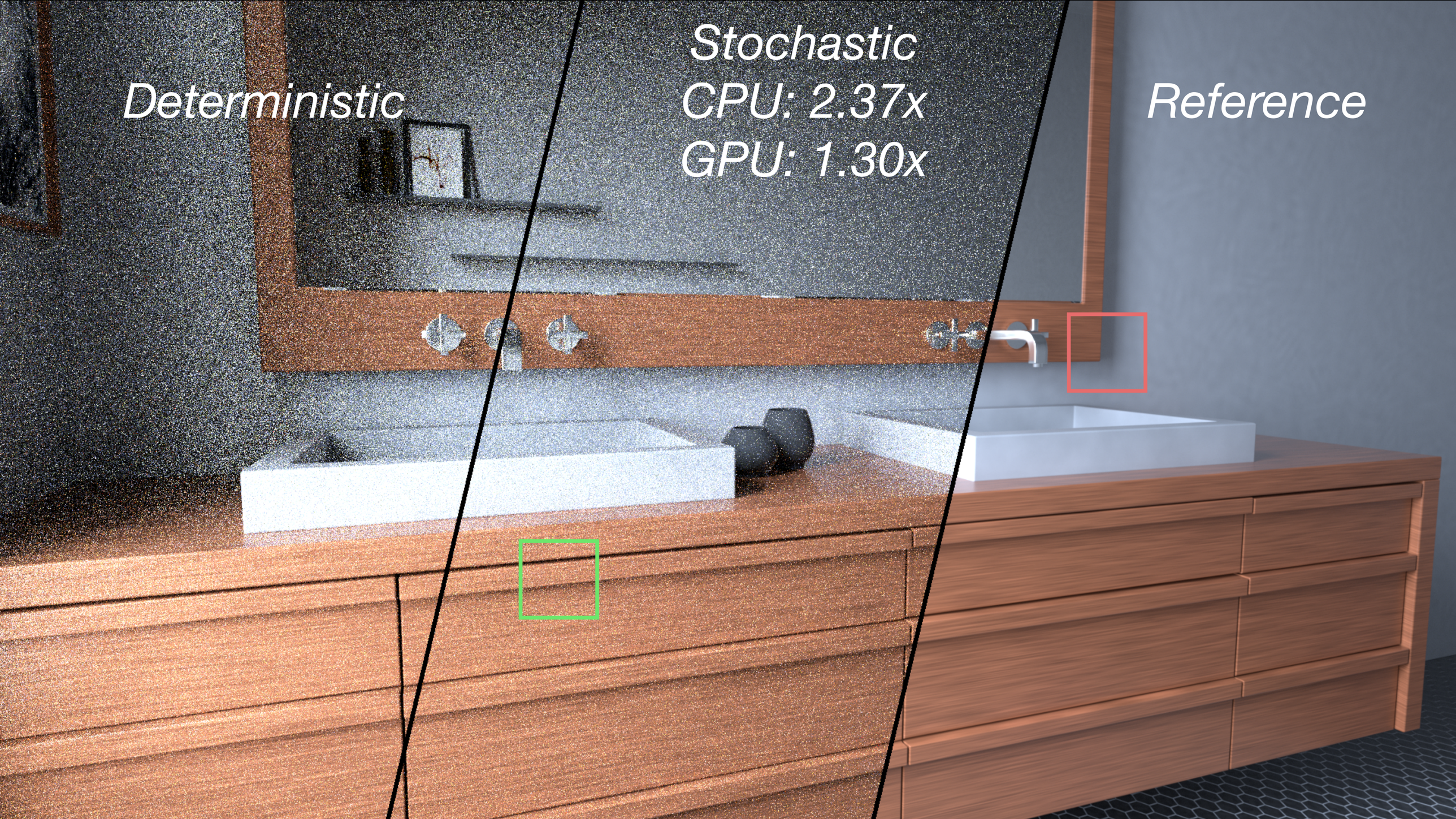
	1 spp	*4 spp*	*16 spp*	*Reference*
Gabor				
Voronoi				

Fast Procedural Noise by Monte Carlo Sampling

Results: Performance

	Deterministic	Stochastic	Speedup
Perlin	7.36 ns	3.84 ns	1.92x
Perlin (fractal)	71.85 ns	5.85 ns	12.28x
Sparse Convolution	449.00 ns	41.21 ns	10.89x
Sparse Convolution (fractal)	3311.53 ns	39.00 ns	89.40x
Gabor (Poisson)	814.26 ns	135.72 ns	6.00 x
Voronoi	28.56 ns	7.02 ns	4.07x

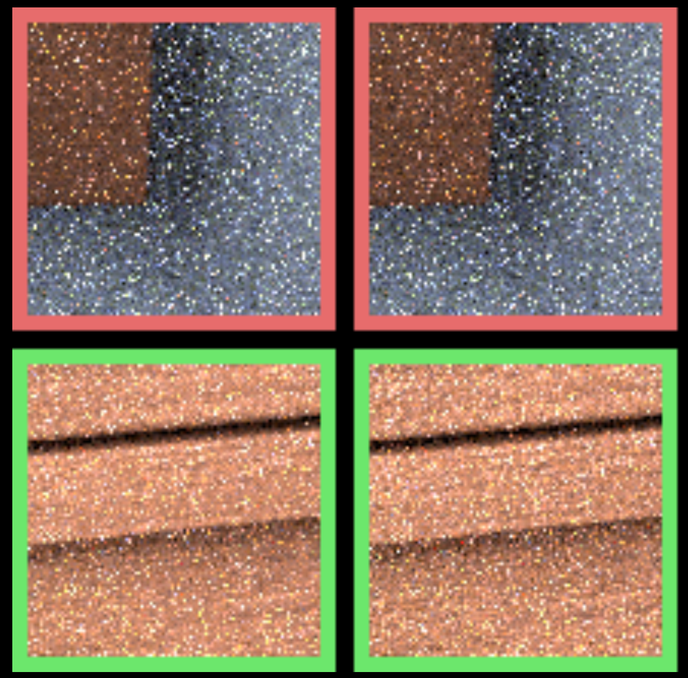
Single evaluation, 3D noise, NVIDIA 4090 RTX GPU


Deterministic
Stochastic
CPU: 2.37x
GPU: 1.30x
Reference

Results: Contemporary Bathroom

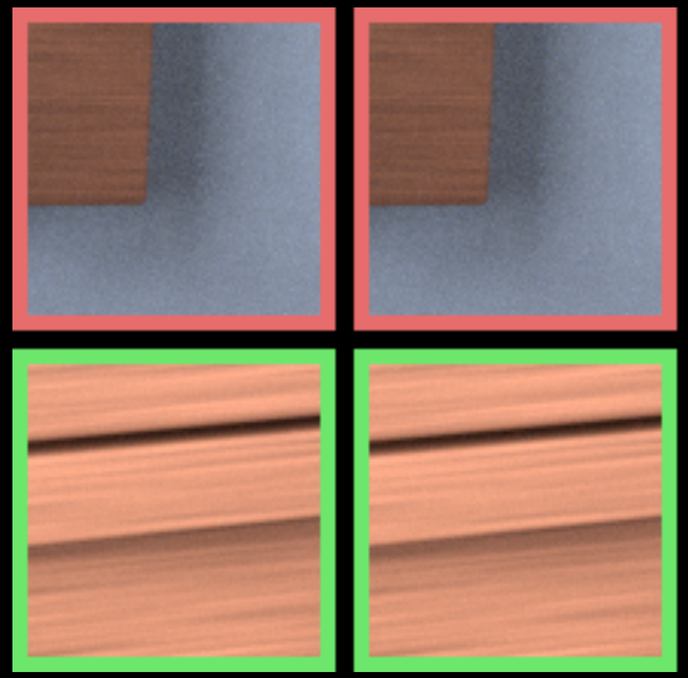
Deterministic *Stochastic*

32 spp

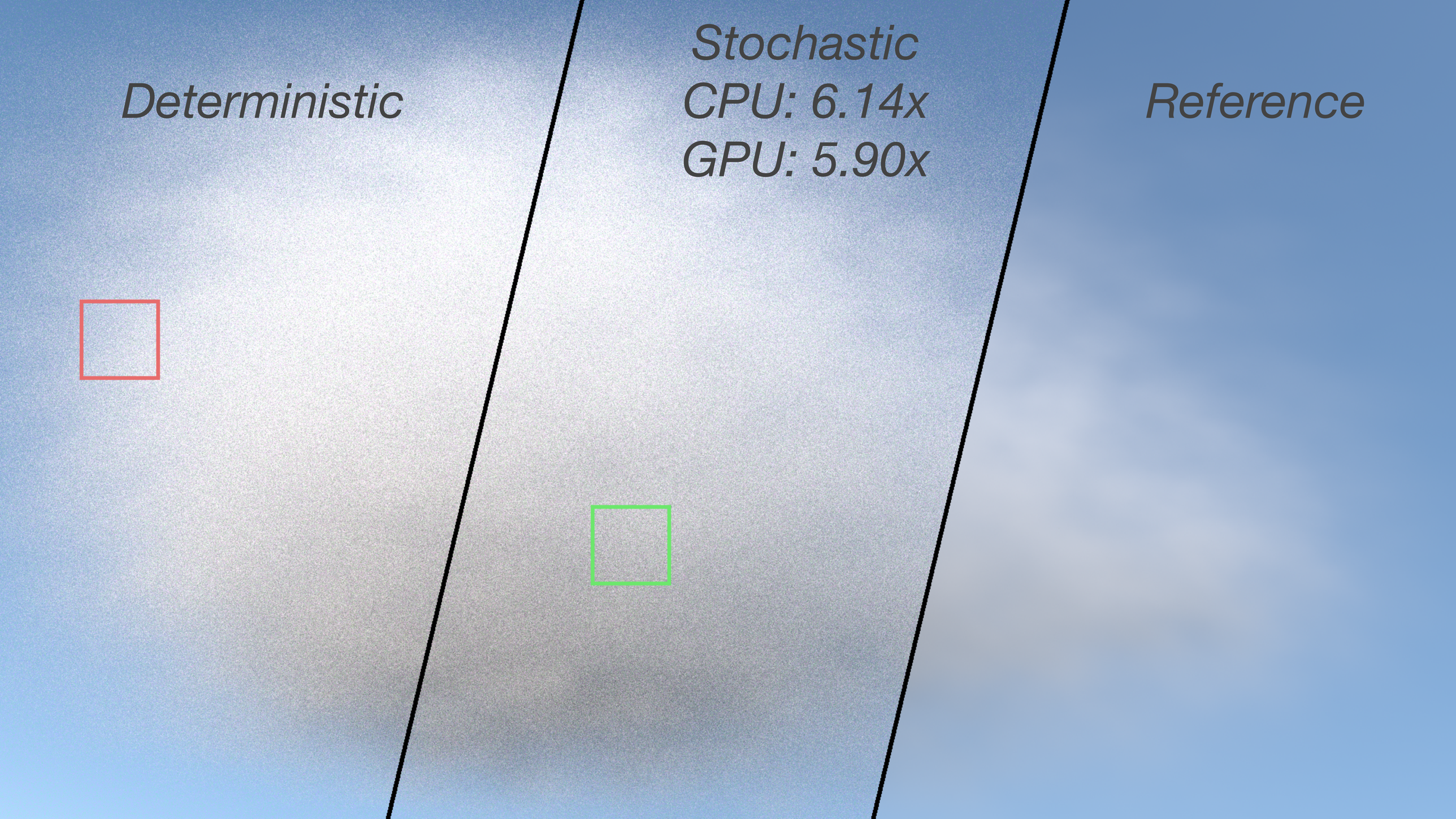


1.01x MRSE

Deterministic *Stochastic*



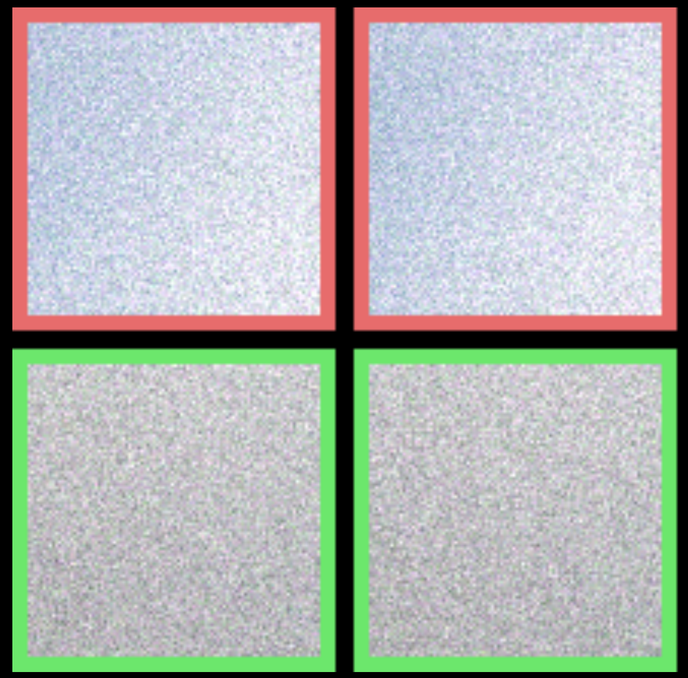
1024 spp


Deterministic
Stochastic
CPU: 6.14x
GPU: 5.90x
Reference

Sparse-Convolution Noise Cloud

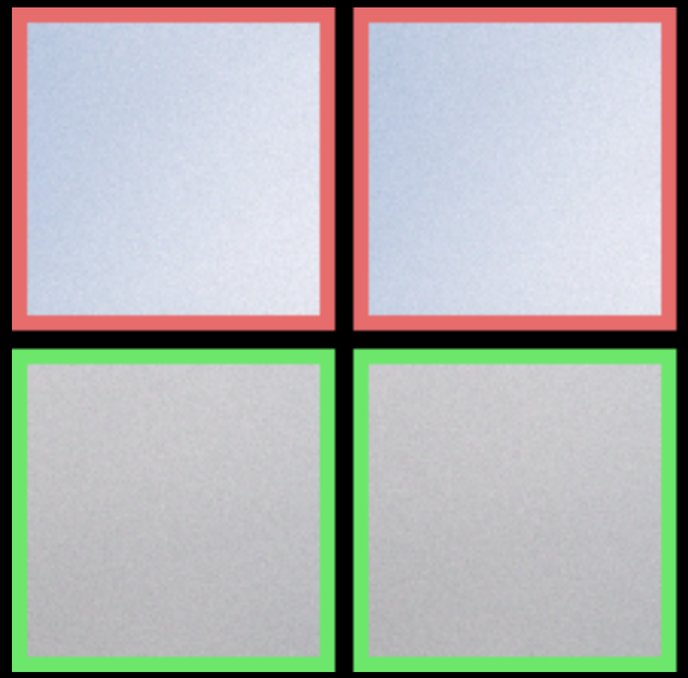
Deterministic *Stochastic*

32 spp



0.98x
MRSE

Deterministic *Stochastic*



1024 spp

Fast Procedural Noise by Monte Carlo Sampling

Results: Procedural Cloud, Sparse Conv.

Majorant	MRSE	CPU Speedup	GPU Speedup
Strict	1.02x	3.85x	3.86x
Optimized	1.03x	6.14x	5.90x

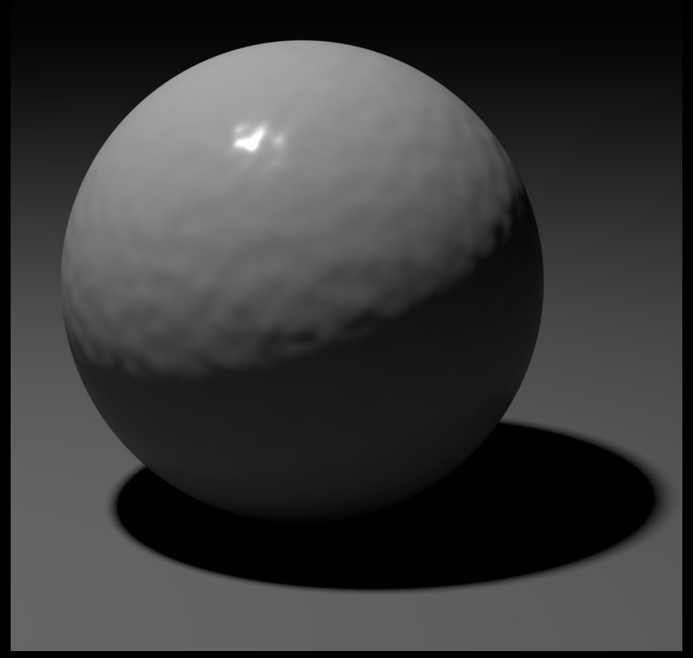
Future Work

- Debiasing: handling non-linear transformations of noise

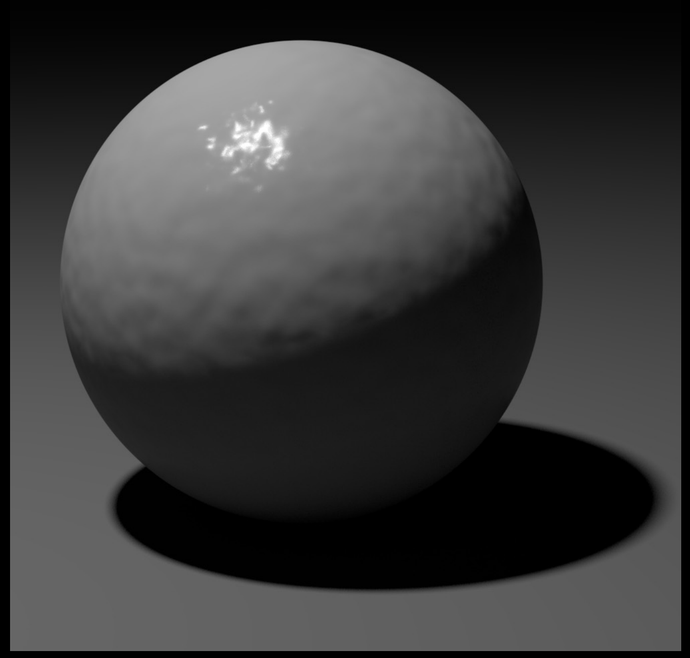
Future Work

- Debiasing: handling non-linear transformations of noise

  Failure case: bump mapping



Deterministic



Stochastic

Future Work

- Debiasing: handling non-linear transformations of noise
- Effect on denoisers: noisy albedo feature buffers are unexpected…

Future Work

- Debiasing: handling non-linear transformations of noise
- Effect on denoisers: noisy albedo feature buffers are unexpected…
- Stochastic all the (deterministic) things…
- Enable new stochastic-only noise functions?

Thanks!

marcosss@gmail.com
matt@pharr.org