Real-Time Ray Tracing and Denoising

Context: Temporal Anti-Aliasing

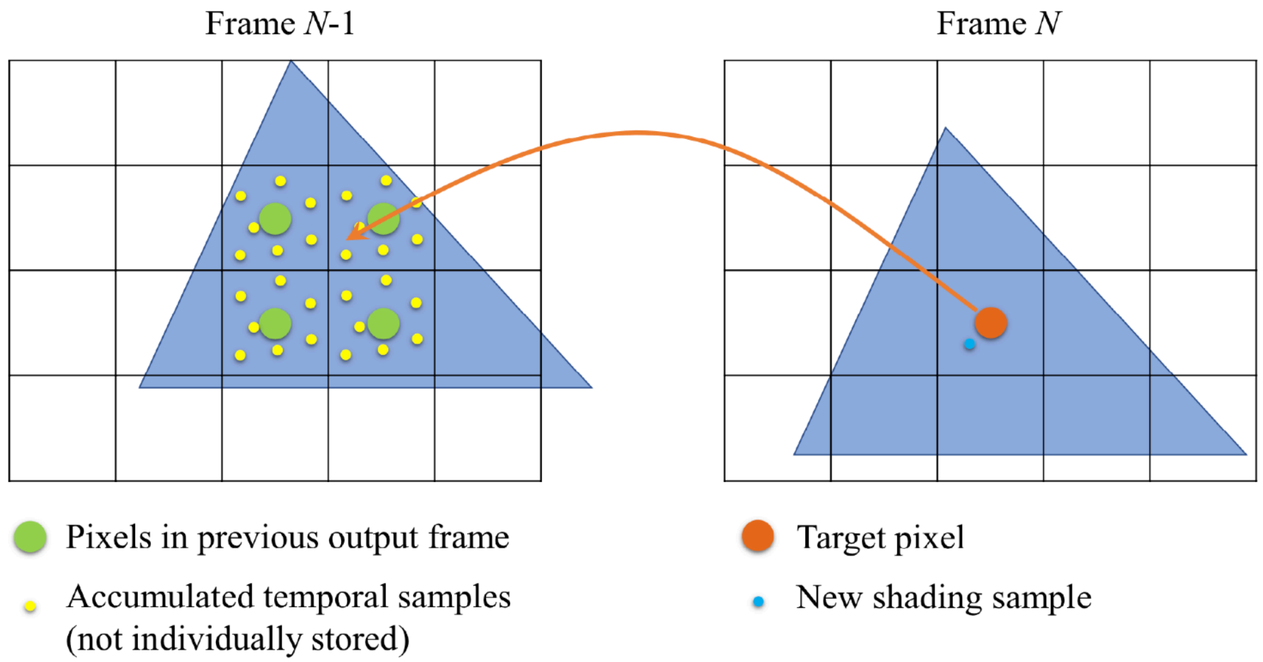
Many-Light Sampling with Temporal Feedback

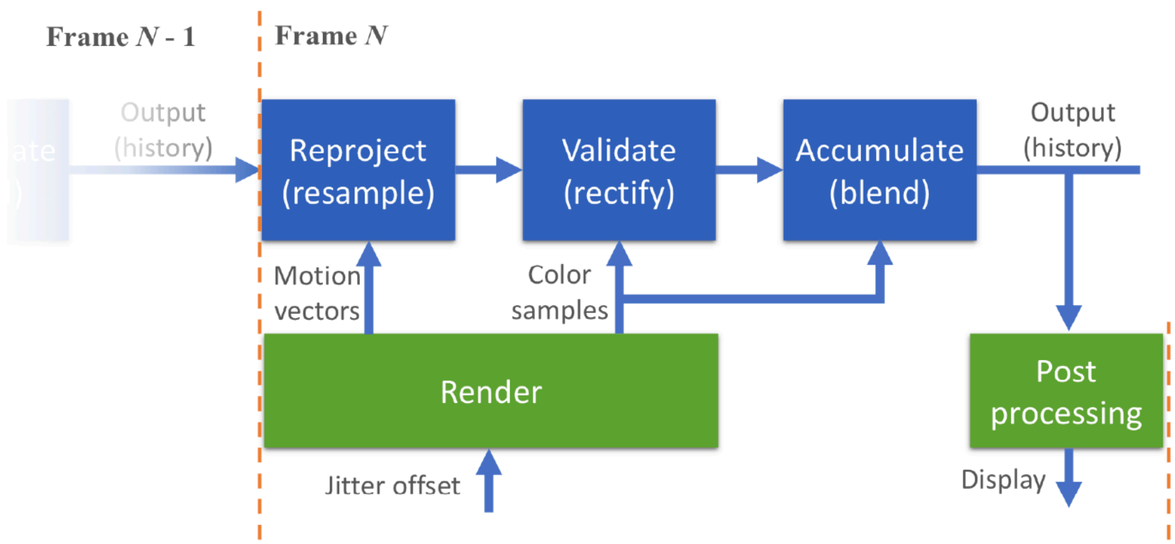
- **Resampled importance sampling**
- **Reservoir sampling**

Denoising

- **Bilateral, joint-bilateral filters**
- **CNN-based denoising**

Temporal Anti-Aliasing (TAA)





[Karis 2014, Salvi 2015, Yang et al. 2020…]

Temporal Anti-Aliasing (TAA)

[Yang et al. 2020]

Failure cases:

- New pixels are jaggy
- Smearing / temporal lag

ReSTIR

Bitterli et al. 2020. *Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting*. Proc SIGGRAPH.

ReSTIR

Chris Wyman and Alexey Panteleev. 2021. Rearchitecting Spatiotemporal Resampling for Production. (High Performance Graphics 2021).

Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. Computer Graphics Forum (High Performance Graphics 2021).

Guillaume Boissé. 2021. World-Space Spatiotemporal Reservoir Reuse for Ray-Traced Global Illumination. (SIGGRAPH Asia 2021 Technical Communications.)

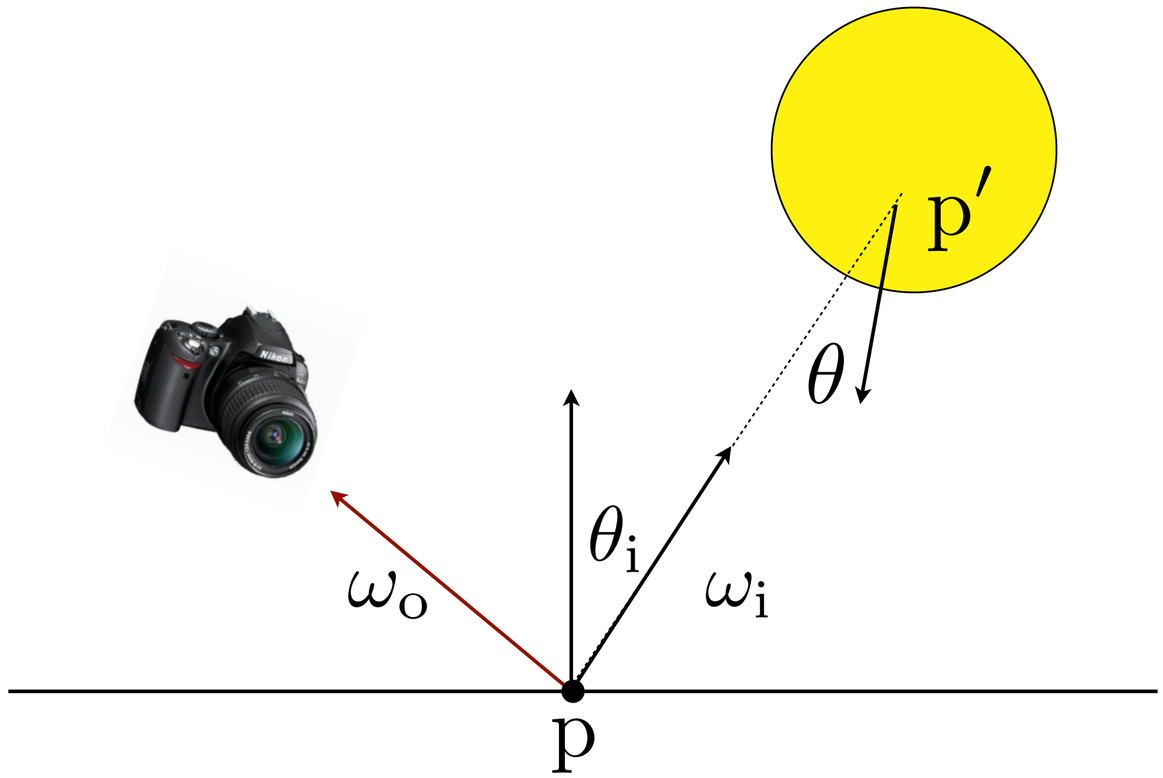
Daqi Lin, Chris Wyman, Cem Yukse. Fast Volume Rendering with Spatiotemporal Reservoir Resampling. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2021) .

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, Chris Wyman Generalized Resampled Importance Sampling: Foundations of ReSTIR. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2022).

[...]

Review: Area Sampling Estimator

$$L_{\mathrm{o}}(\mathrm{p}, \omega_{\mathrm{o}}) \approx \frac{f_r(\mathrm{p}, \omega_{\mathrm{i}} \rightarrow \omega_{\mathrm{o}})\, V(\mathrm{p}, \mathrm{p}')\, L_{\mathrm{e}}\, \cos\theta_{\mathrm{i}}\, \cos\theta}{p(light)\, p(\mathrm{p}')\, \|\mathrm{p} - \mathrm{p}'\|^2}$$



Ideal: Product Sampling

What if:

$$p(light)\, p(\mathrm{p}') \propto \frac{f_r(\mathrm{p}, \omega_\mathrm{i} \to \omega_\mathrm{o})\, L_\mathrm{e}\, \cos\theta_\mathrm{i}\, \cos\theta}{\|\mathrm{p} - \mathrm{p}'\|^2} \quad \textbf{?}$$

Estimator:

$$\begin{aligned} L_\mathrm{o}(\mathrm{p}, \omega_\mathrm{o}) &\approx \frac{f_r(\mathrm{p}, \omega_\mathrm{i} \to \omega_\mathrm{o})\, V(\mathrm{p}, \mathrm{p}')\, L_\mathrm{e}\, \cos\theta_\mathrm{i}\, \cos\theta}{p(light)\, p(\mathrm{p}')\, \|\mathrm{p} - \mathrm{p}'\|^2} \\ &= c\, V(\mathrm{p}, \mathrm{p}') \end{aligned}$$

$$c = \int_A \frac{f_r(\mathrm{p}, \omega_\mathrm{i} \to \omega_\mathrm{o})\, L_\mathrm{e}\, \cos\theta_\mathrm{i}\, \cos\theta}{\|\mathrm{p} - \mathrm{p}'\|^2} dA(\mathrm{p}')$$

Resampled Importance Sampling (RIS)

Generate M samples $x_i \sim p$

Define sample weights $w(x) = \dfrac{\hat{p}(x)}{p(x)}$

Target distribution (not necessarily normalized)

Choose a sample z **from** x_i **with probability** $\dfrac{w(z)}{\sum_{i=1}^{M} w(x_i)}$

RIS estimator: $\dfrac{f(z)}{\hat{p}(z)} \cdot \left(\dfrac{1}{M} \sum_{i=1}^{M} \dfrac{\hat{p}(x_i)}{p(x_i)} \right)$

RIS for Product Sampling

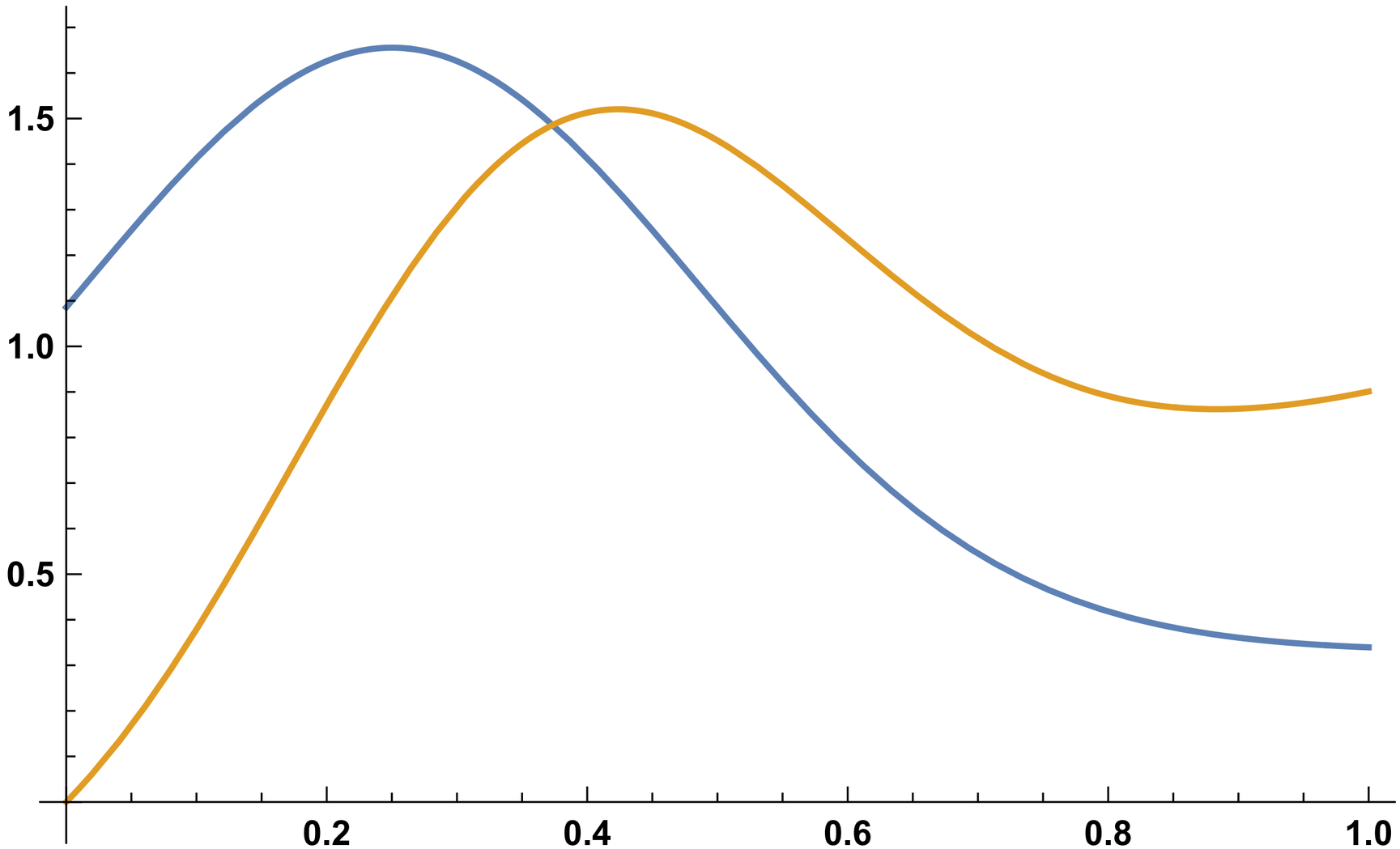
RIS generates a sample that is *approximately* from $\hat{p}$'s distribution

Consider $p \sim L_e$ and $\hat{p} = f_r\, L_e$ with $f = f_r\, L_e\, V$:

$$w(x) = \frac{\hat{p}(x)}{p(x)} \propto \frac{f_r L_e}{L_e} = f_r$$

In RIS estimator, $\dfrac{f}{\hat{p}} = V$

"Approximately"?



$$\hat{p} \propto f_r \, L_e$$
$$M = \infty$$

$$p \propto L_e$$
$$M = 1$$

Interpreting the RIS Estimator

Estimator: $$\frac{f(z)}{\hat{p}(z)} \cdot \left(\frac{1}{M} \sum_{i=1}^{M} \frac{\hat{p}(x_i)}{p(x_i)} \right) = V \left(\frac{1}{M} \sum_{i=1}^{M} \frac{\hat{p}(x_i)}{p(x_i)} \right)$$

$$f(x) = \frac{f_r L_e V \cos\theta_{\mathrm{i}} \cos\theta}{\|\mathrm{p} - \mathrm{p}'\|^2}$$

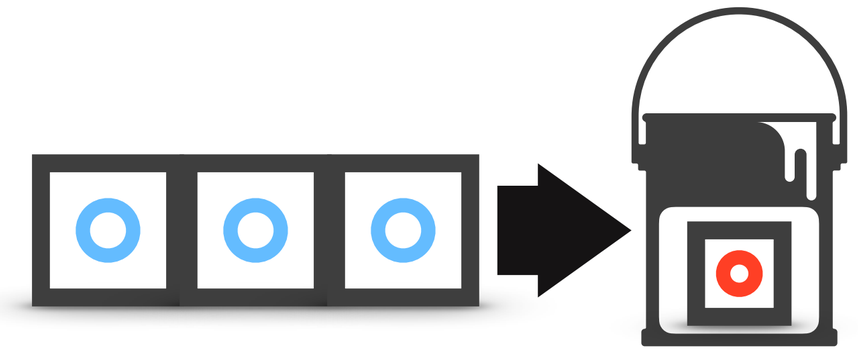
$$\hat{p}(x) = \frac{f_r L_e \cos\theta_{\mathrm{i}} \cos\theta}{\|\mathrm{p} - \mathrm{p}'\|^2}$$

Thus: $$\frac{1}{M} \sum_{i=1}^{M} \frac{\hat{p}(x_i)}{p(x_i)} \approx \int_A \frac{f_r(\mathrm{p}, \omega_{\mathrm{i}} \rightarrow \omega_{\mathrm{o}})\, L_{\mathrm{e}}\, \cos\theta_{\mathrm{i}}\, \cos\theta}{\|\mathrm{p} - \mathrm{p}'\|^2} \mathrm{d}A(\mathrm{p}')$$

Erik Pitti (CC BY 2.0)

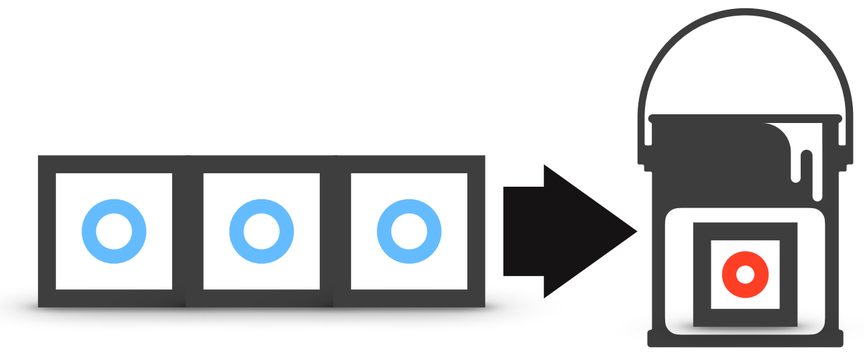
Reservoir Sampling

```
Sample reservoir;
int numConsidered = 0;


while (moreSamples) {
     sample = getNextSample();
     numConsidered += 1;
     if (randomFloat() < 1 / numConsidered)
          reservoir = sample;
}
```



Weighted Reservoir Sampling

```
Sample reservoir;
float sumWeights = 0;


while (moreSamples) {
     sample, weight = getNextSample();
     sumWeights += weight;
     if (randomFloat() < weight / sumWeights)
          reservoir = sample;
}
```



Weighted Reservoir-Based RIS

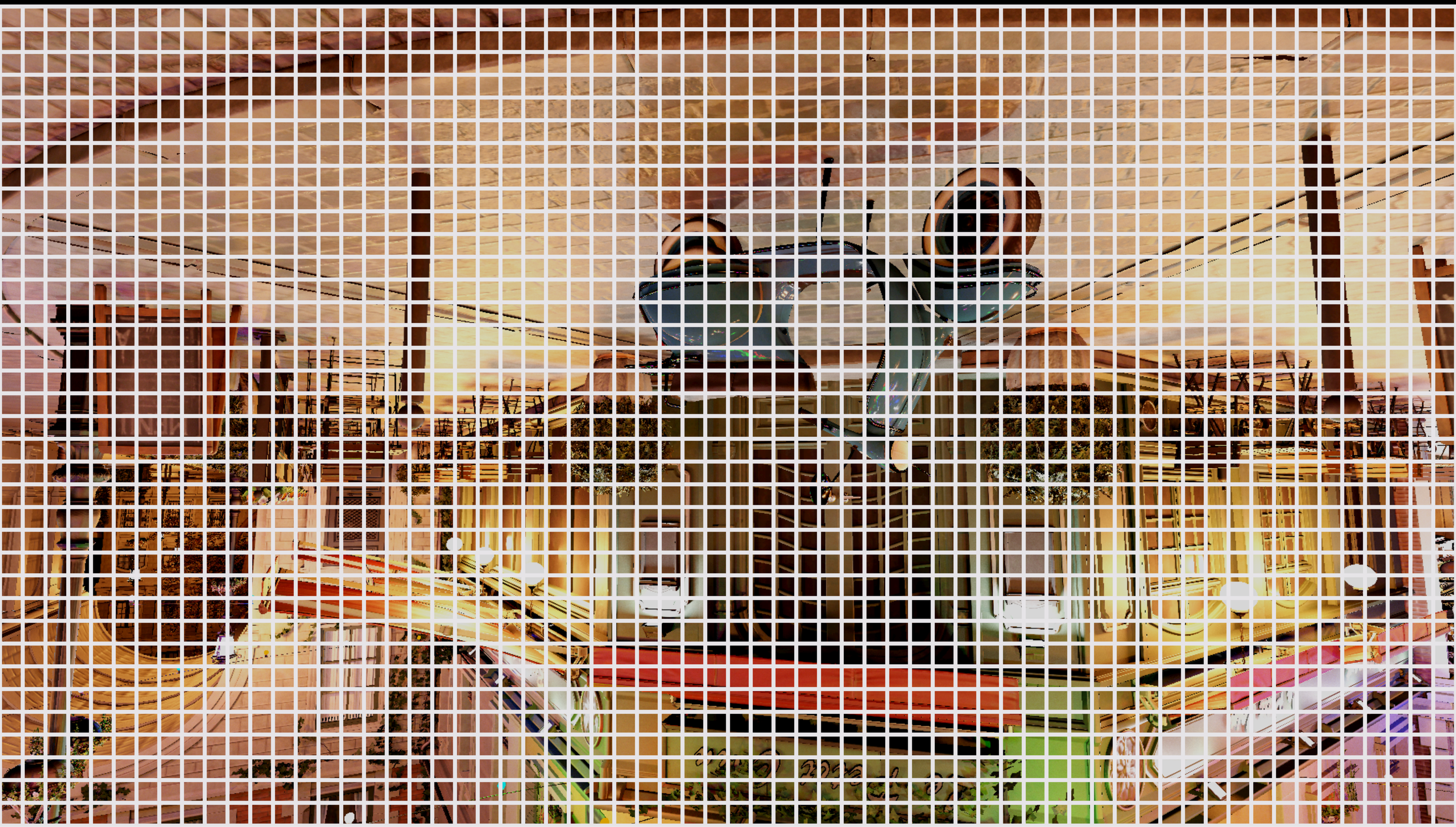
For M samples

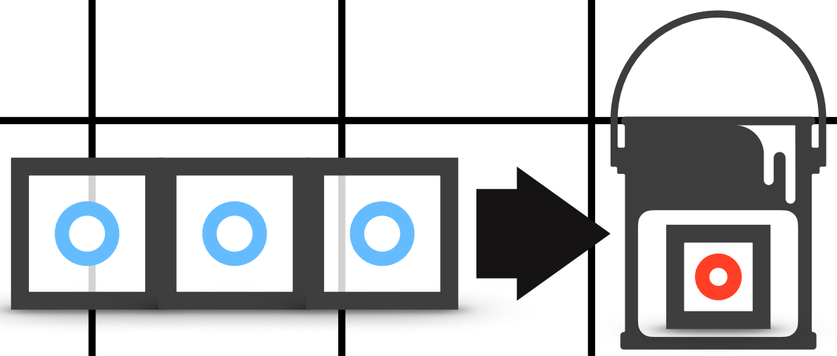
- **Generate sample** $x_i \sim p$
- **Compute weight** $w(x_i) = \hat{p}(x_i)/p(x_i)$
- **Update weight sum:** $w_s = w_s + w(x_i)$
- **Randomly keep sample with prob.** $w(x_i)/w_s$

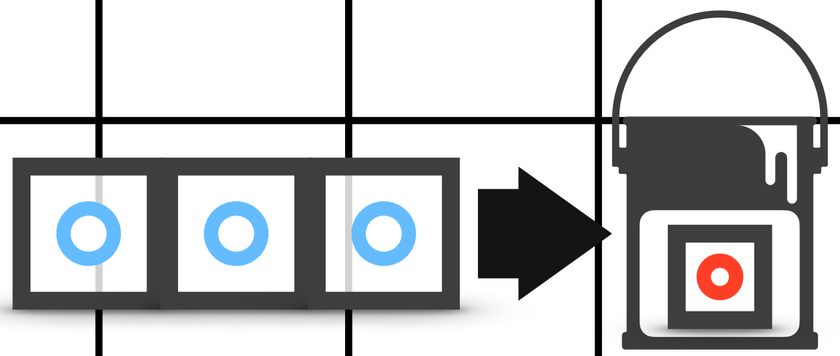
Evaluate estimator using the reservoir sample z**:**

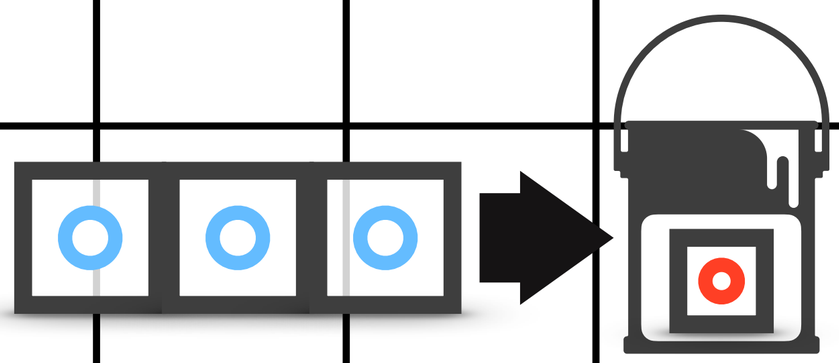
$$\frac{f(z)}{\hat{p}(z)} \cdot \left(\frac{1}{M} w_s \right)$$

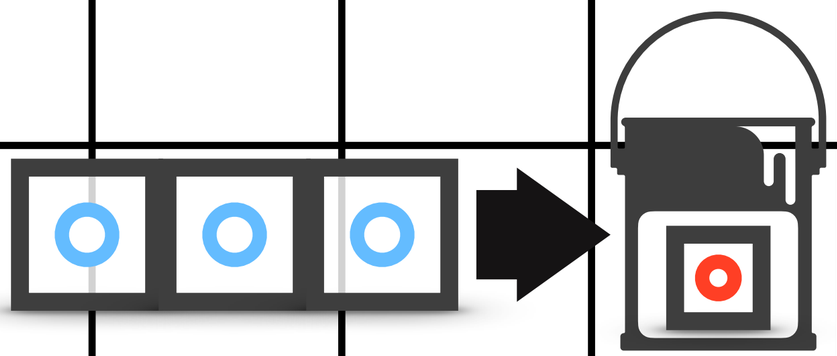

Le Petit Gim
MENU

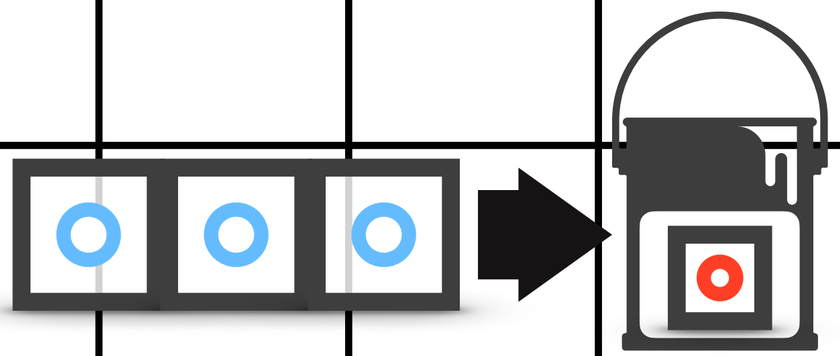






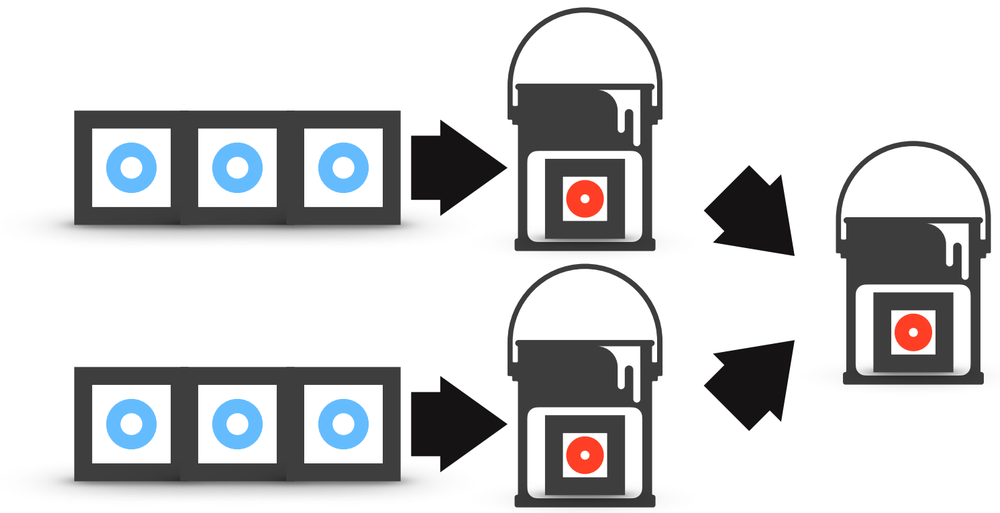


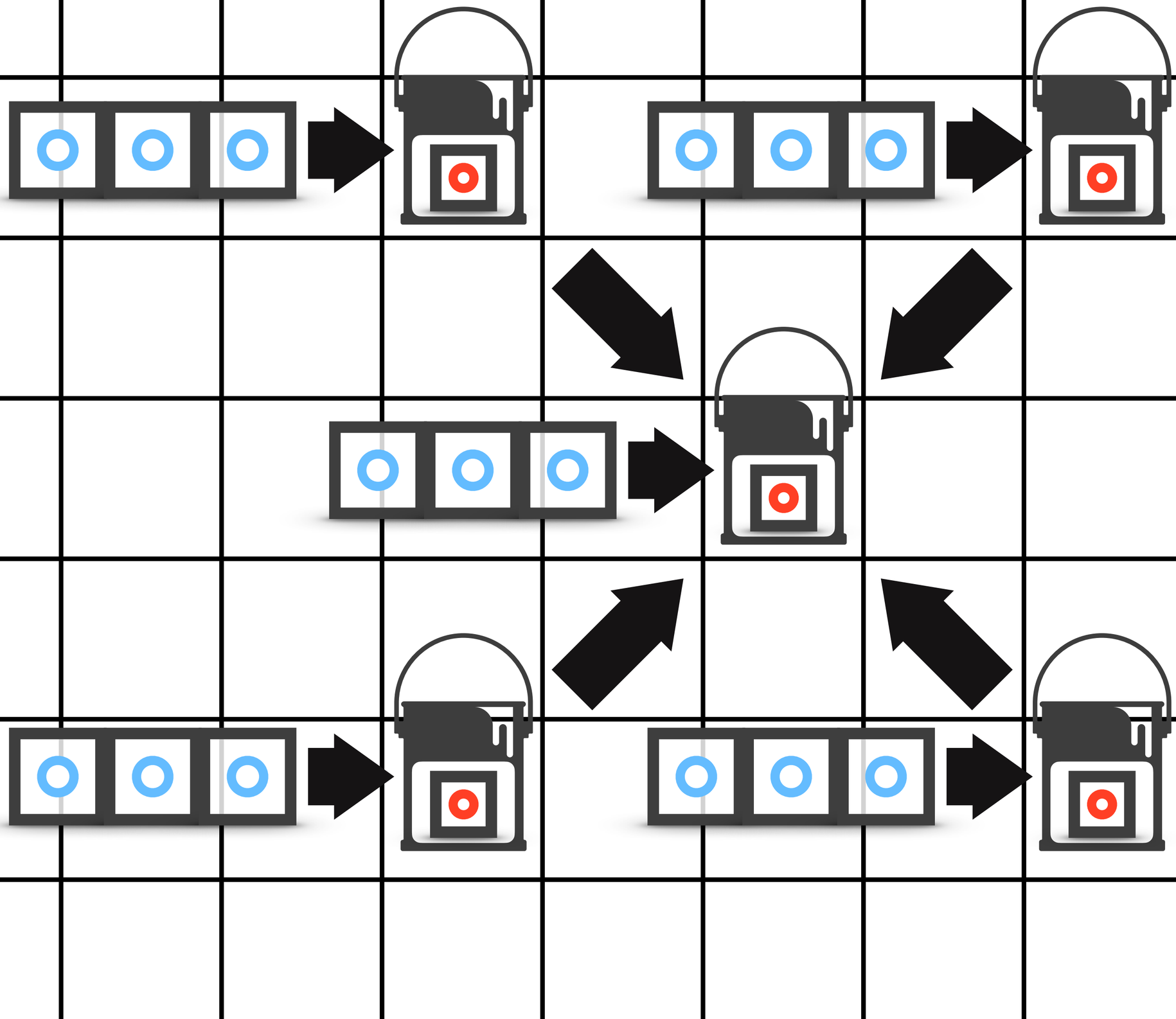




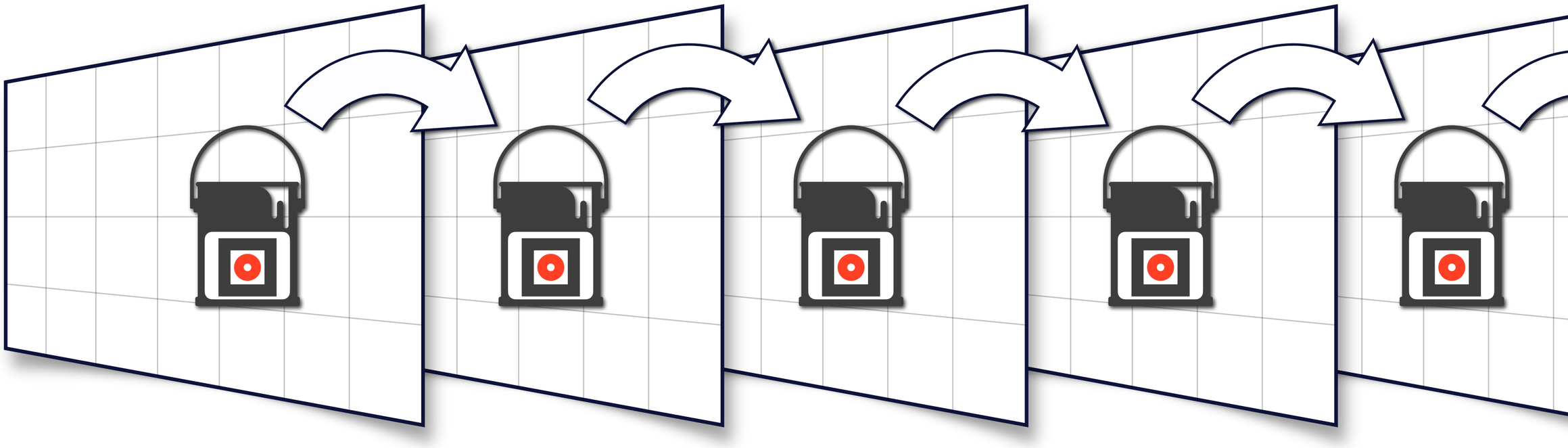
Merging Reservoirs

```
struct Reservoir {
    Sample sample;
    float sumWeights;
    int M;

    void merge(Reservoir other) {
        M += other.M;
        pKeep = sumWeights / (sumWeights + other.sumWeights);
        if (randomFloat() > pKeep)
            sample = other.sample;
        sumWeights += other.sumWeights;
    }
};
```

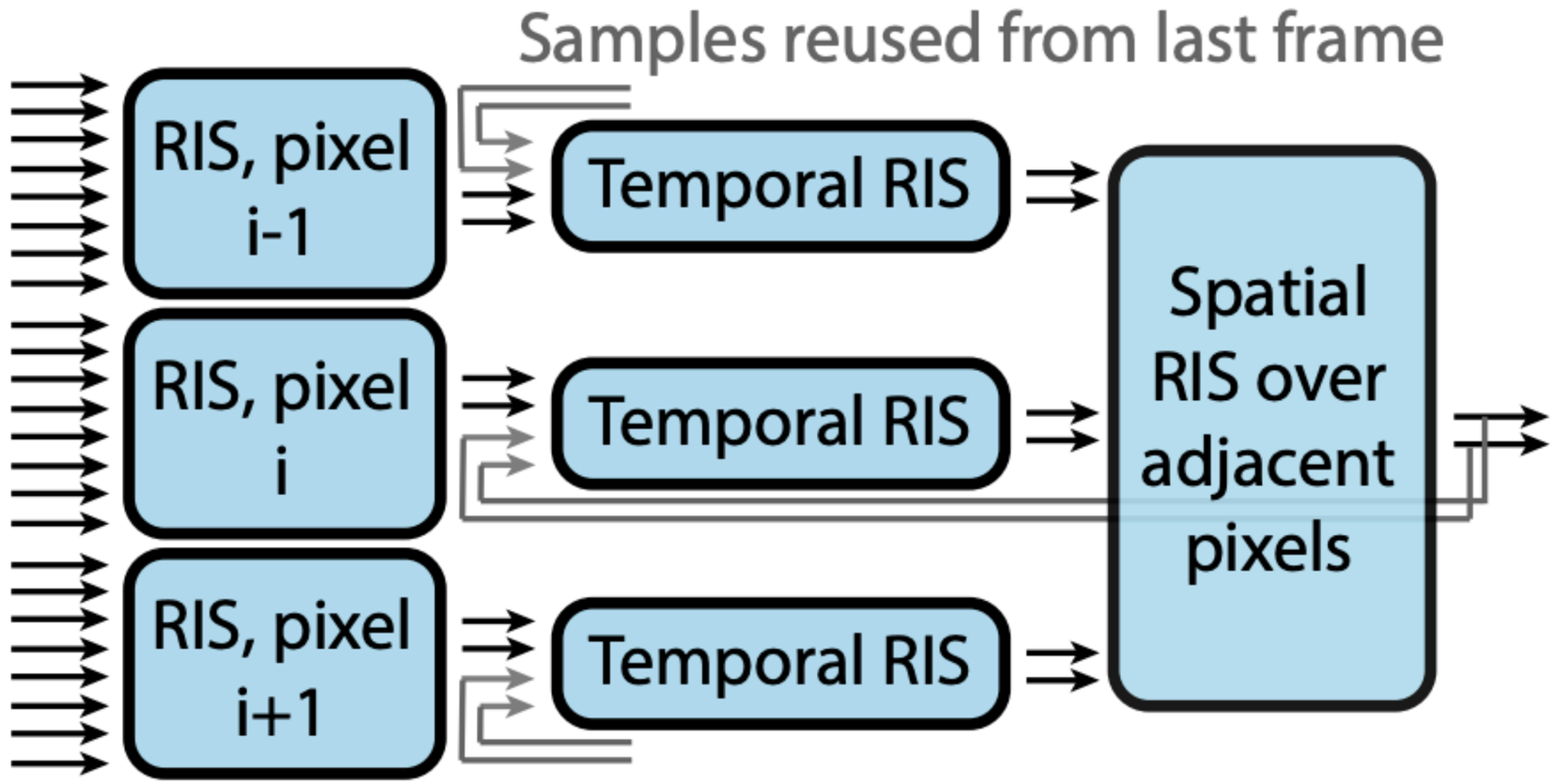




Temporal Reuse



Spatio-Temporal Sample Reuse



Sharing Samples Across Pixels

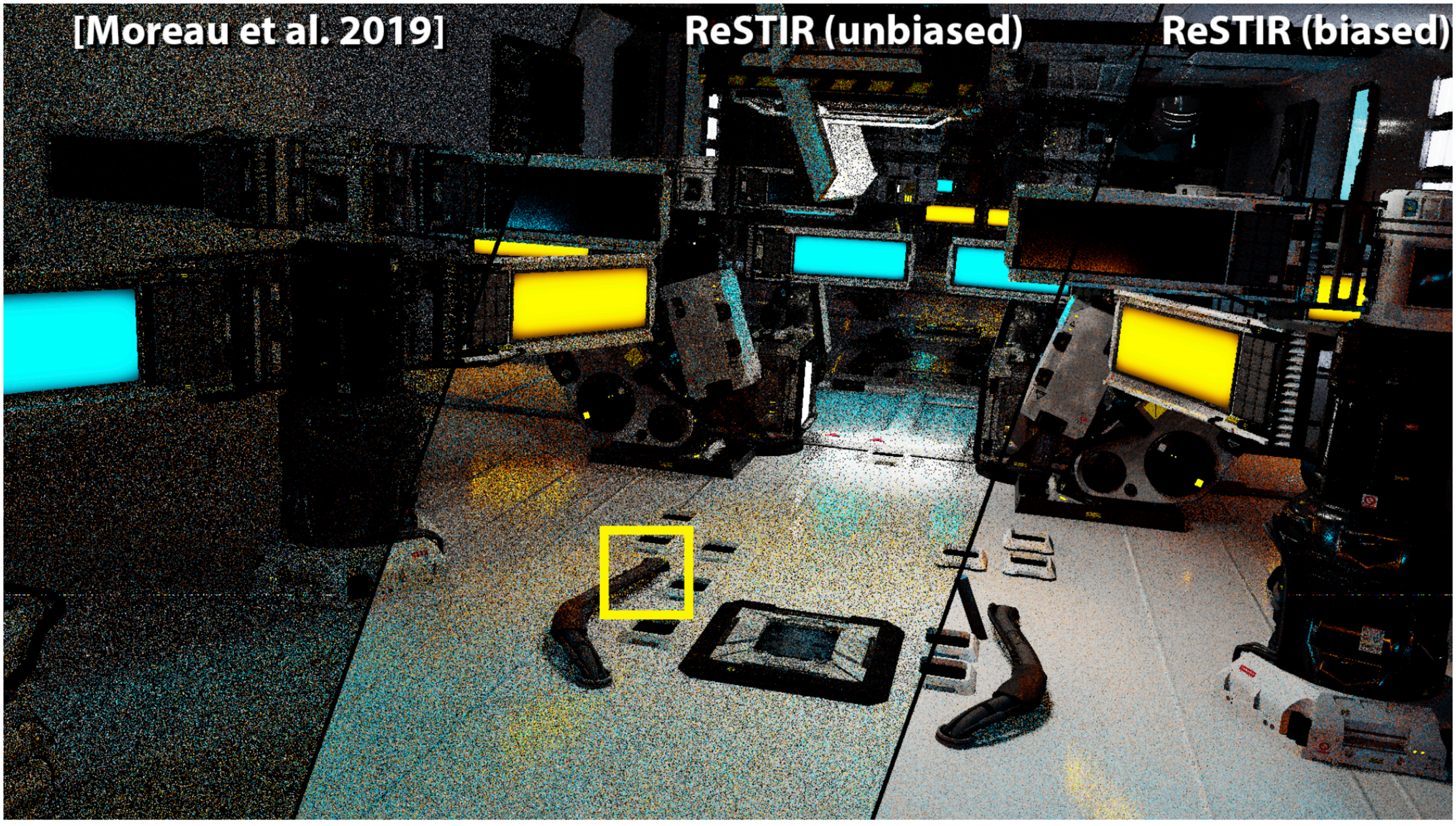
Keep a small number of independent reservoirs at each pixel

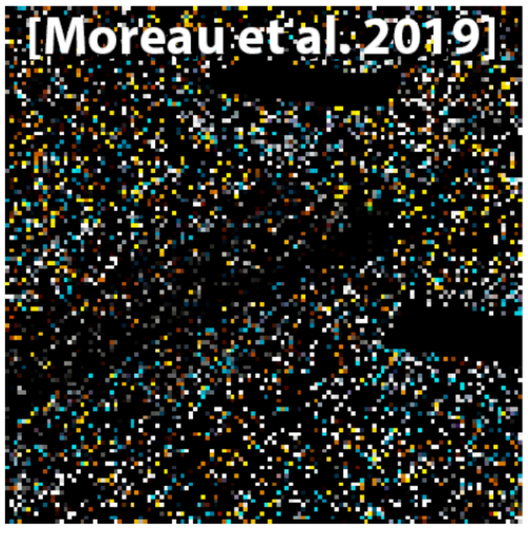
At each pixel, choose k neighbor pixels

- Merge reservoir with neighbor's reservoir
- Get equivalent of kM samples

Repeat n times

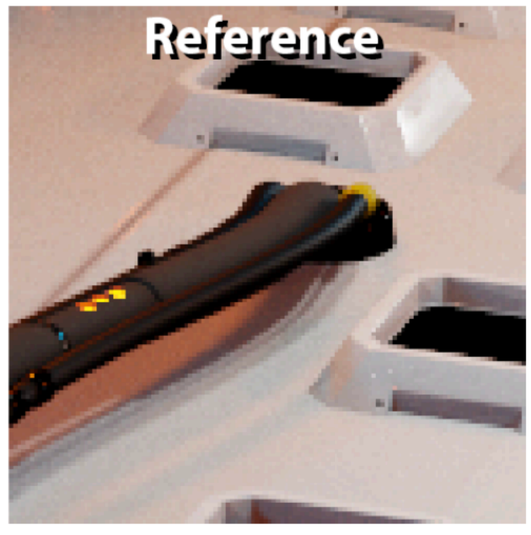
- Get equivalent of $k^n M$ samples(!)


[Moreau et al. 2019]
ReSTIR (unbiased)
ReSTIR (biased)


[Moreau et al. 2019]


ReSTIR (unbiased)


ReSTIR (biased)


Reference

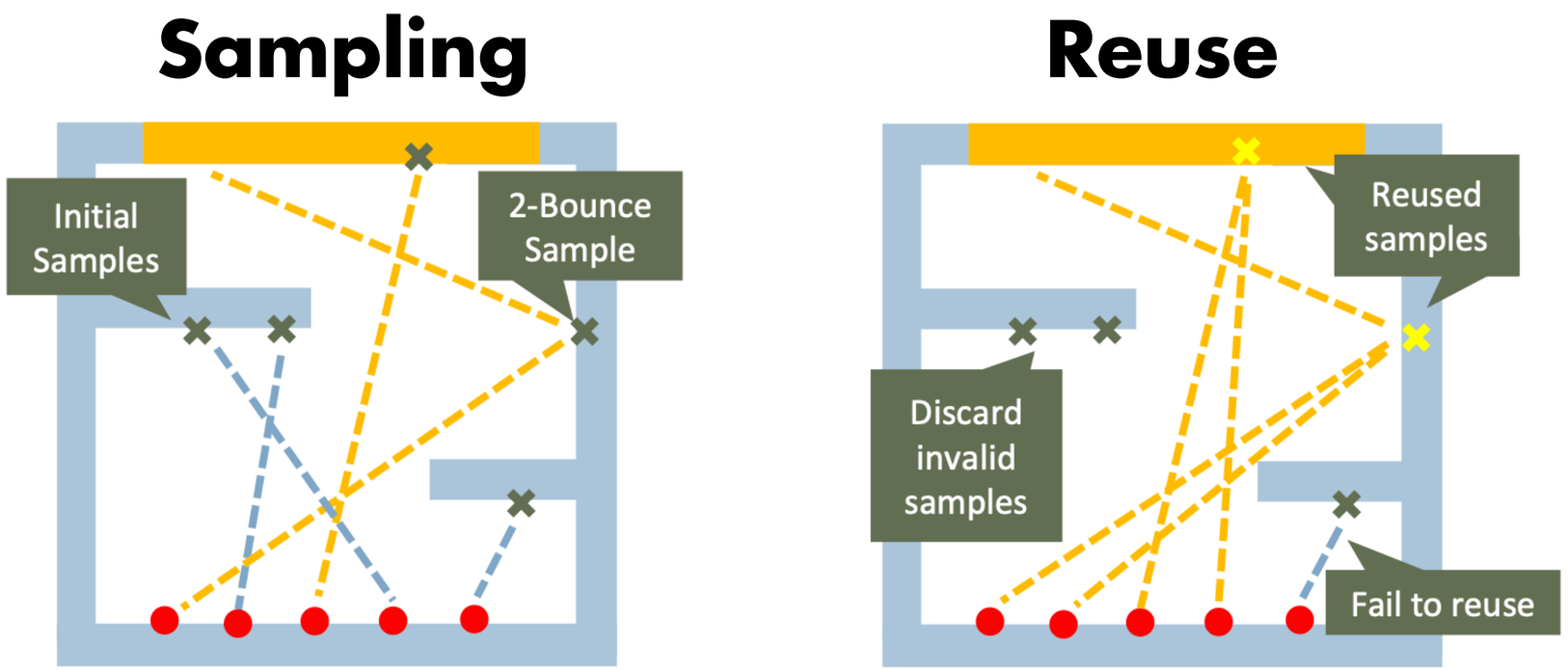
ReSTIR + Denoising

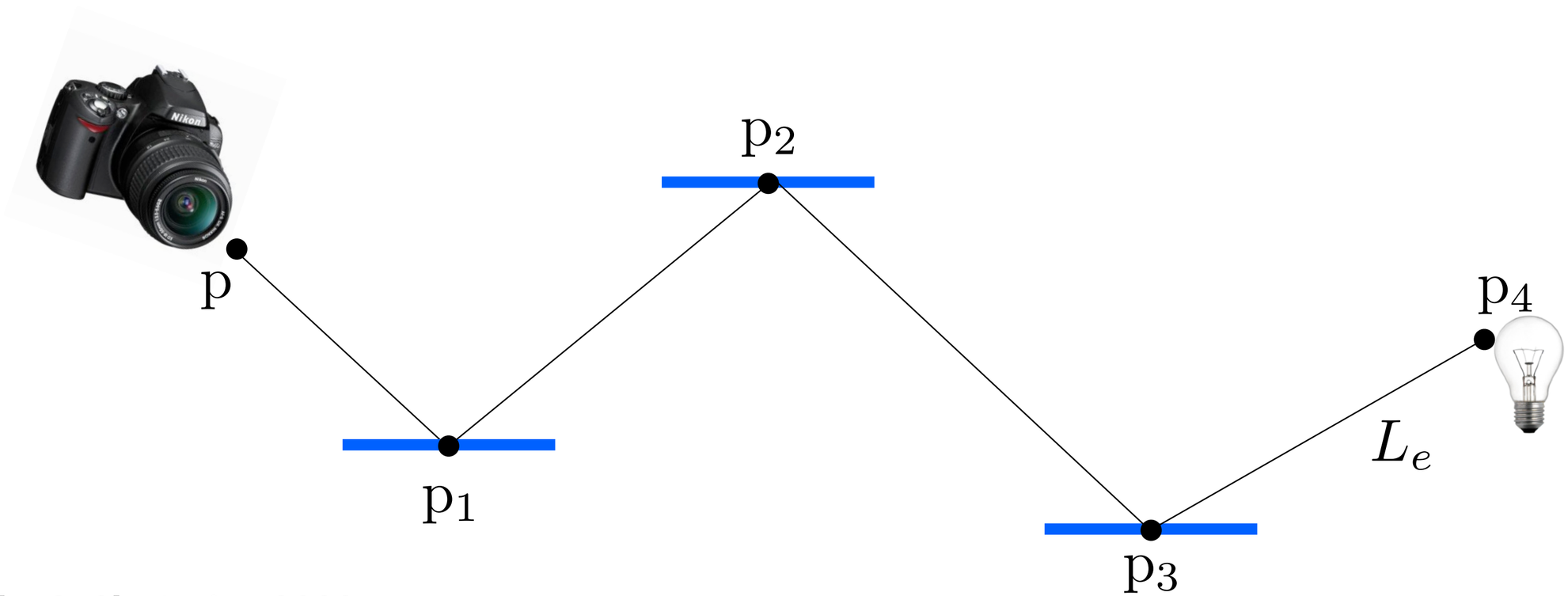
ReSTIR + Denoising

Direct Lighting

Direct + Indirect Lighting

ReSTIR GI


Sampling
Initial Samples
2-Bounce Sample
Reuse
Reused samples
Discard invalid samples
Fail to reuse


p
p1
p2
p3
p4
L_e

ReSTIR GI

Path Traced	ReSTIR GI	ReSTIR GI	
(2spp) 18.9 ms	(biased) 16.0 ms	(unbiased) 18.0 ms	Reference
3.25 MSE	0.0230 MSE (141x)	0.0195 MSE (166x)	

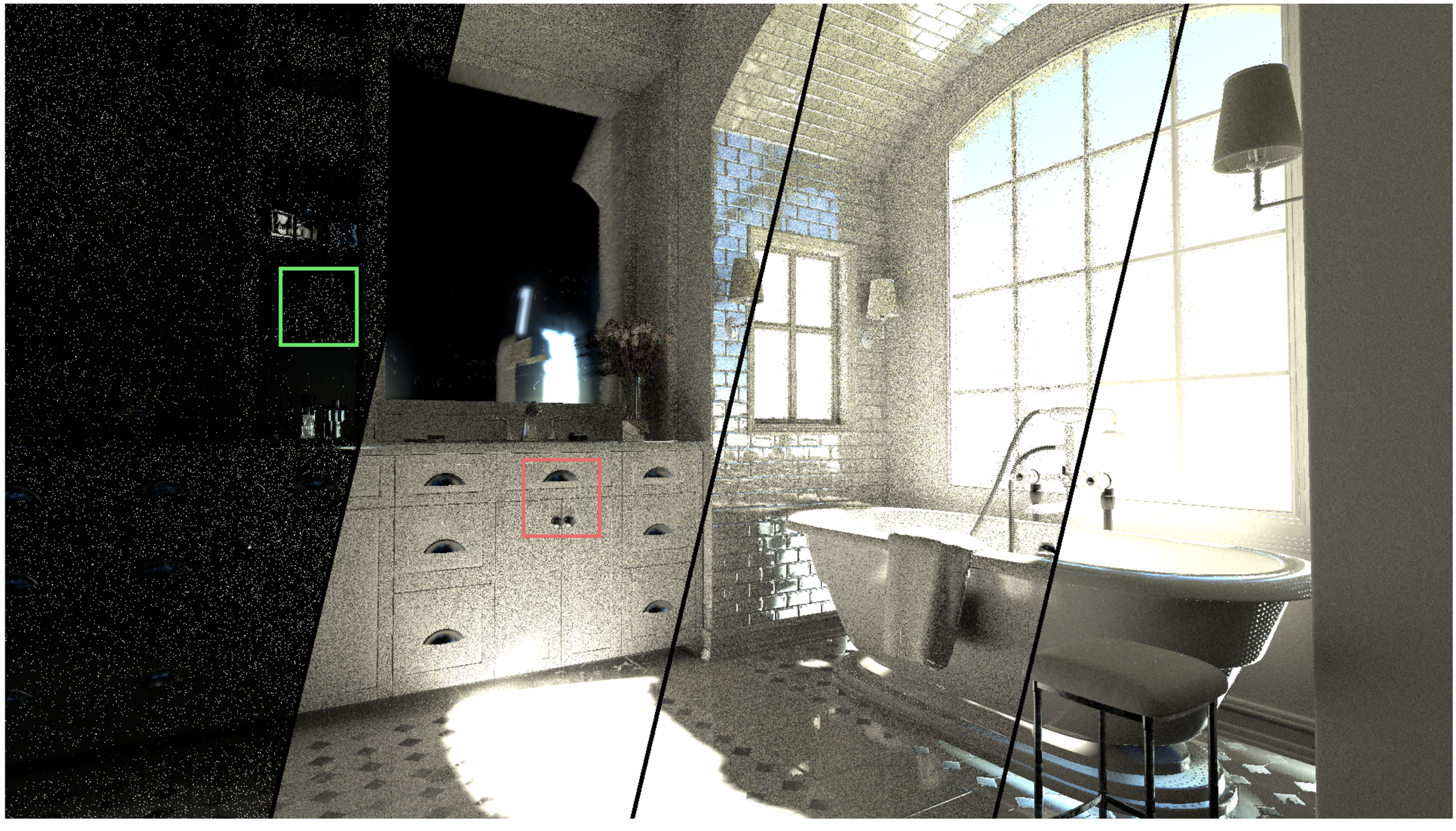
ReSTIR GI

Path Traced
(1spp) 8.2 ms
0.449 MSE

ReSTIR GI
(biased) 8.8 ms
0.0260 MSE (17.3x)

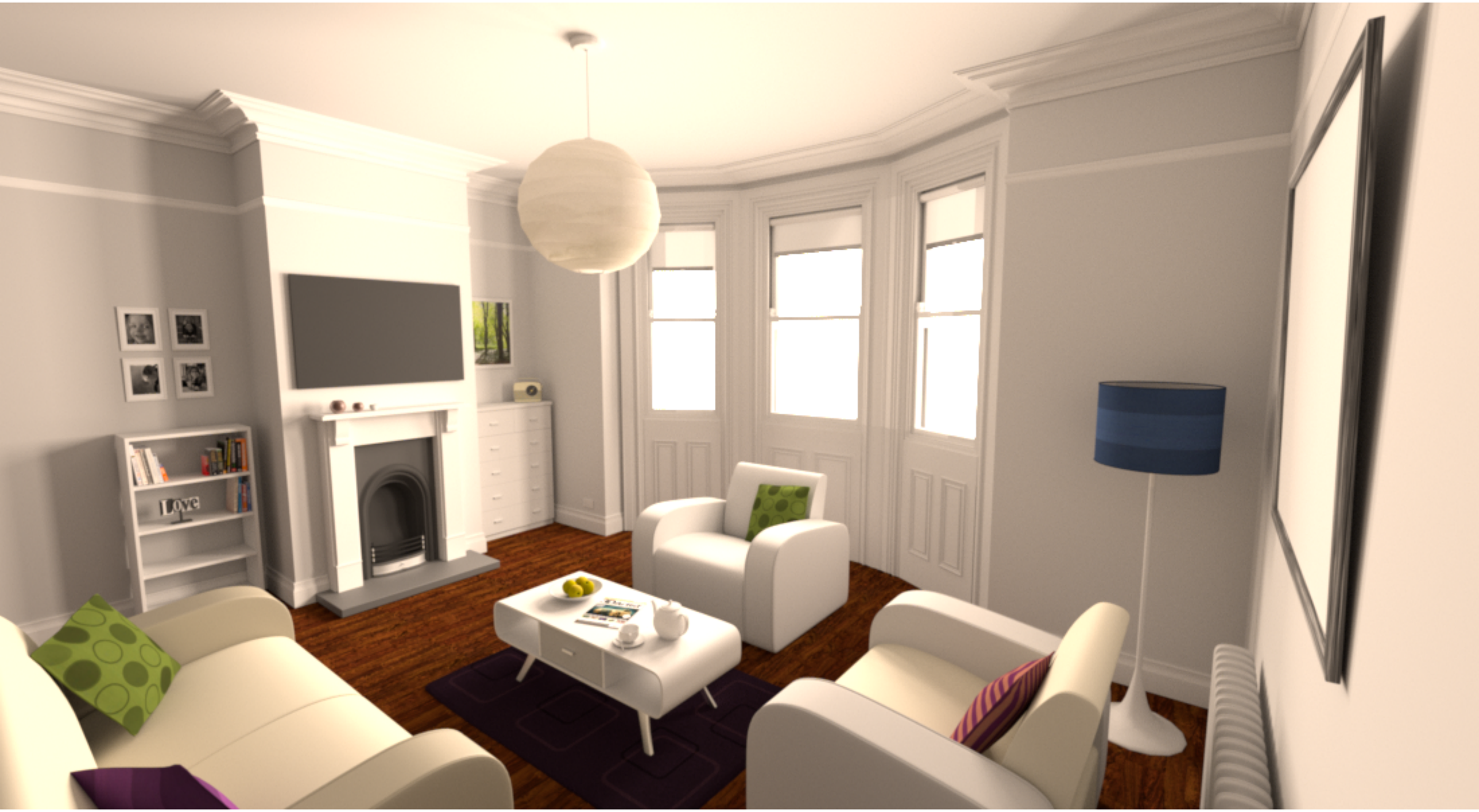
ReSTIR GI
(unbiased) 9.6 ms
0.0306 MSE (14.7x)

Reference



Denoising

White Room, 4096 pixel samples

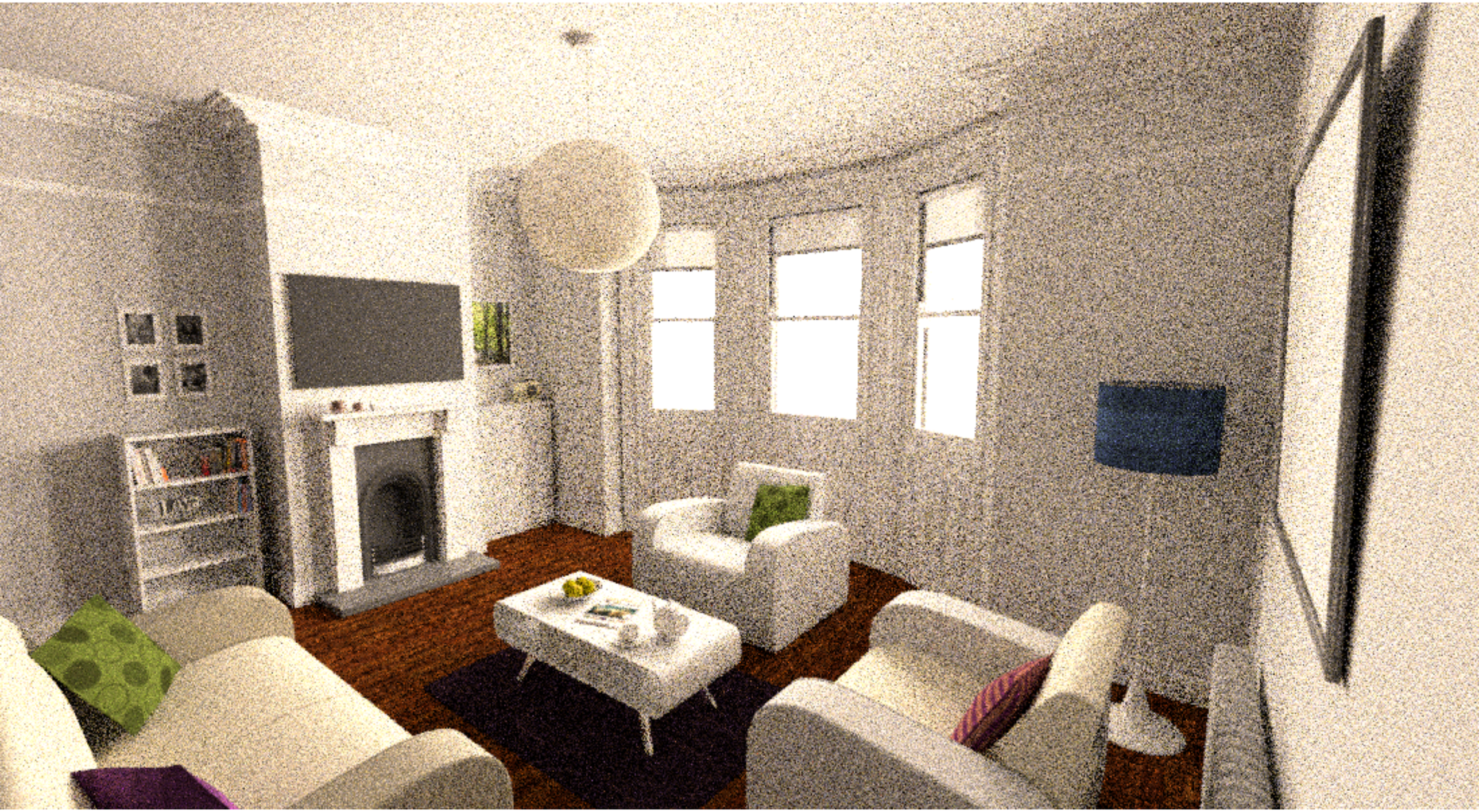


White Room, 1 pixel sample

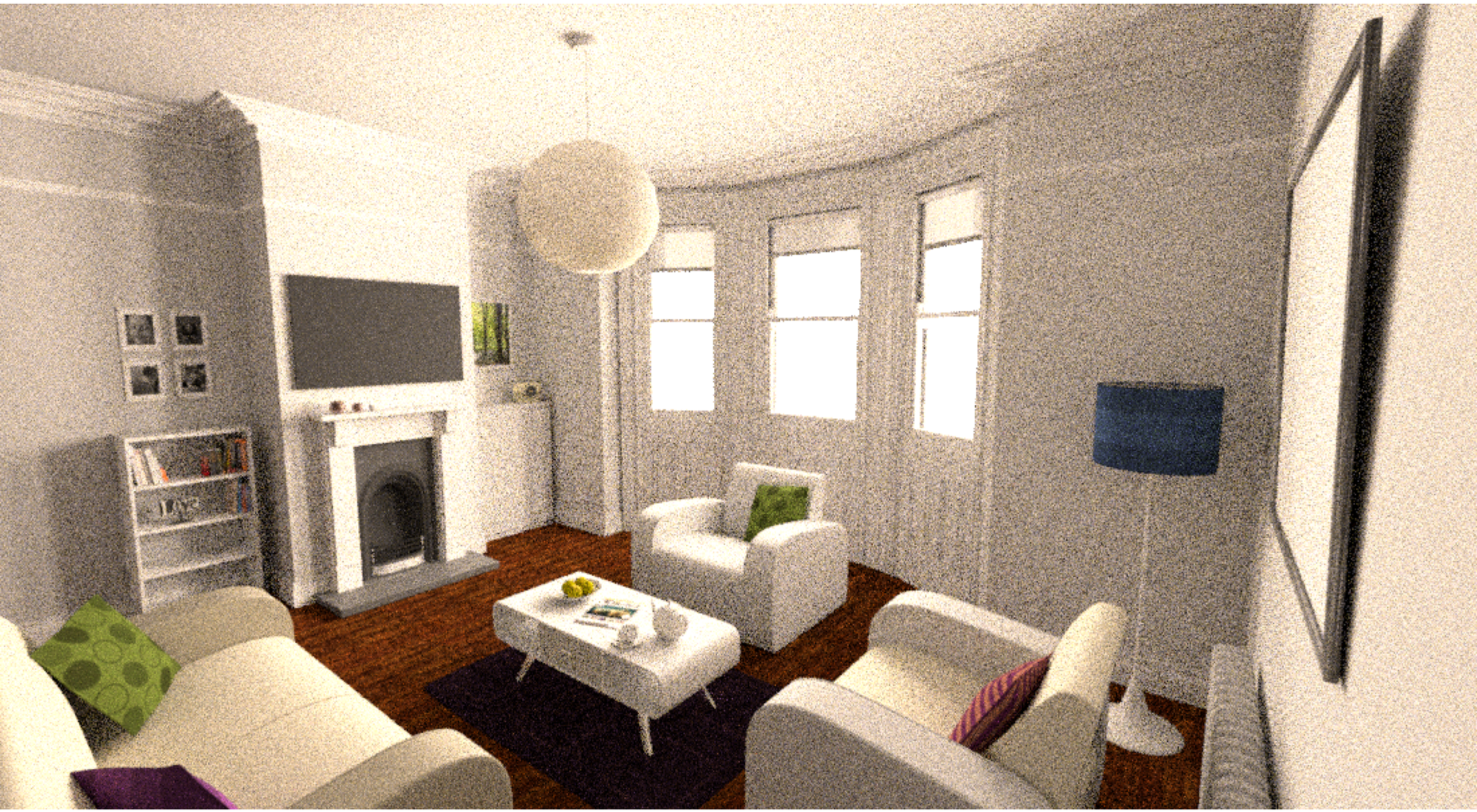
White Room, 2 pixel samples

White Room, 4 pixel samples

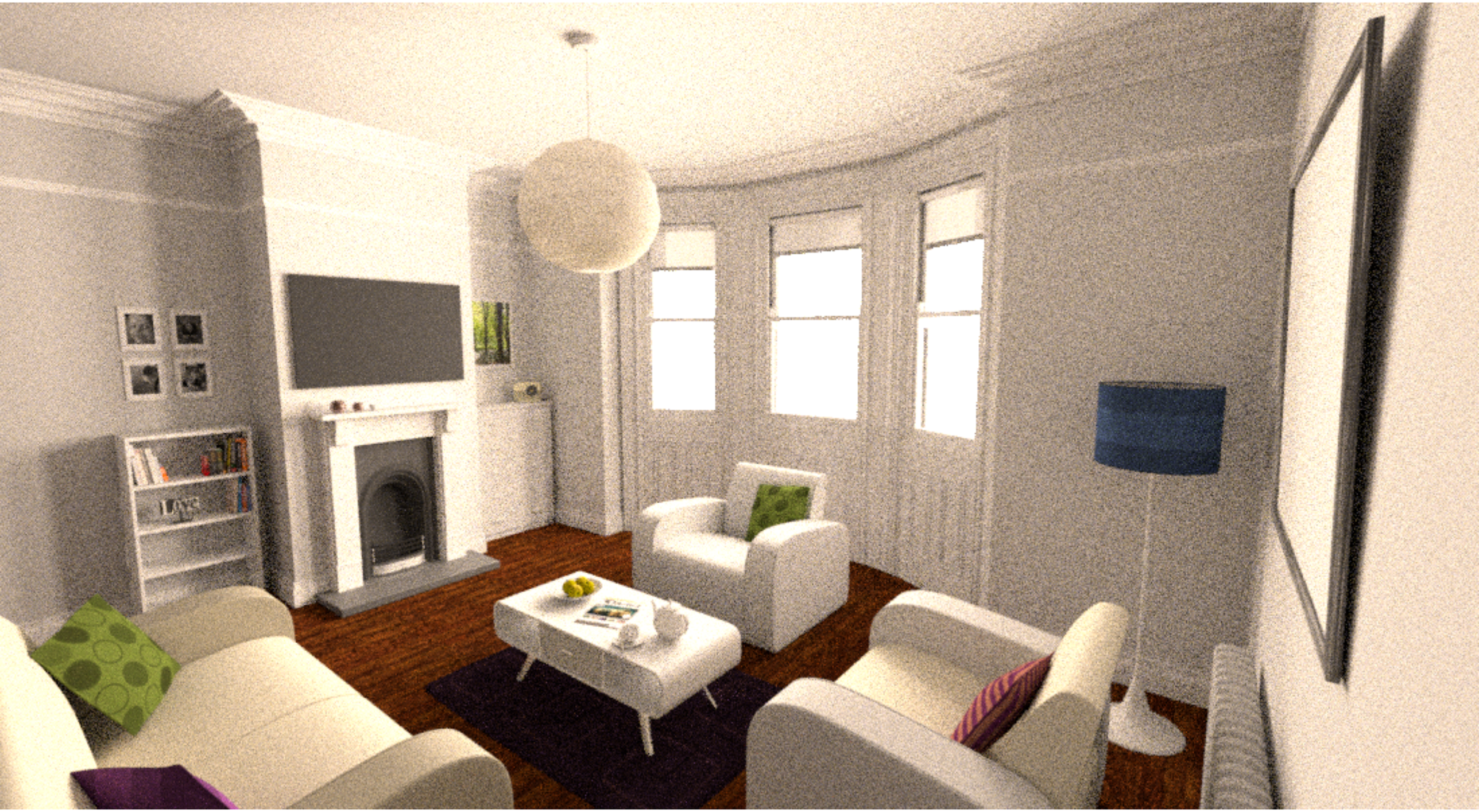
White Room, 8 pixel samples



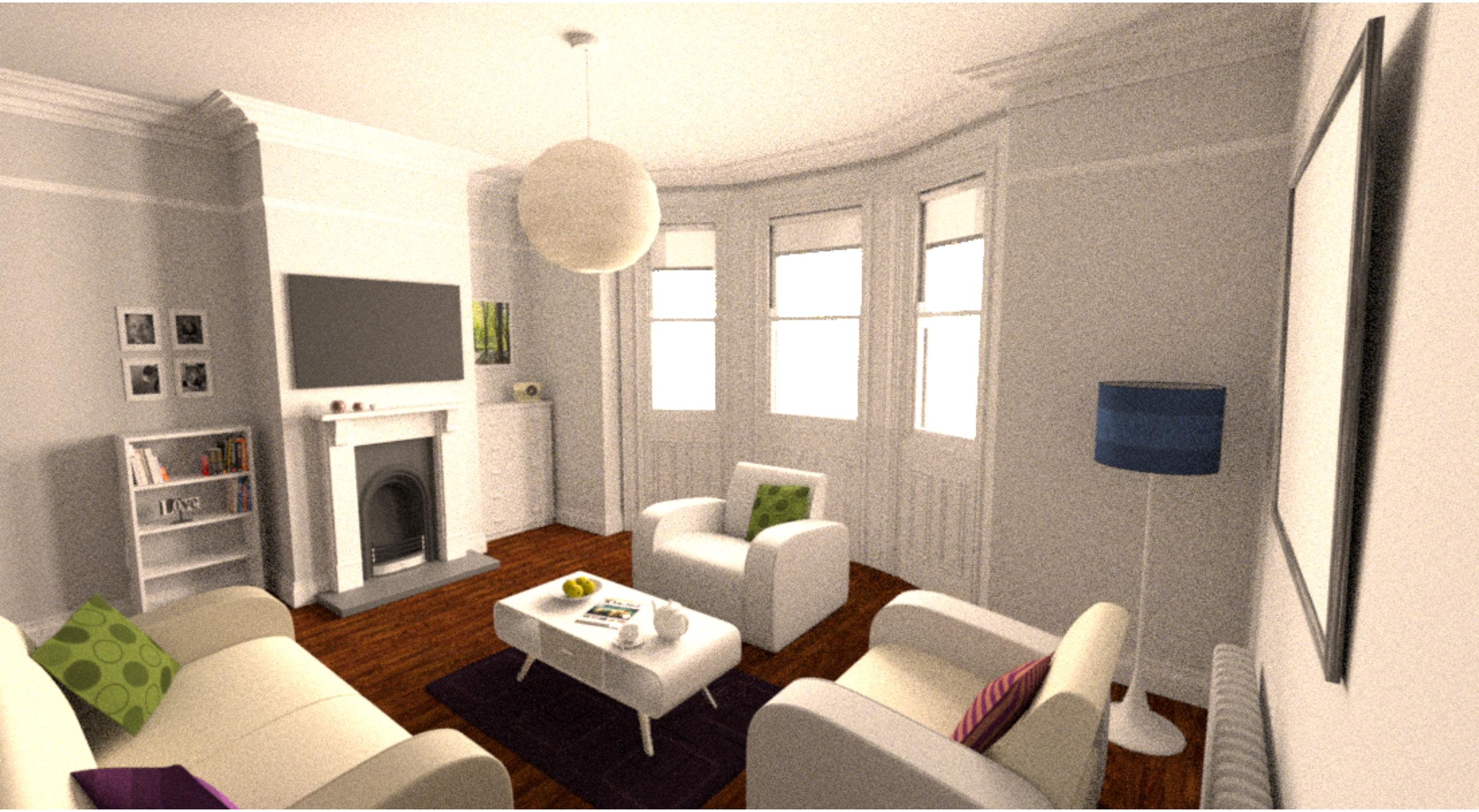
White Room, 16 pixel samples



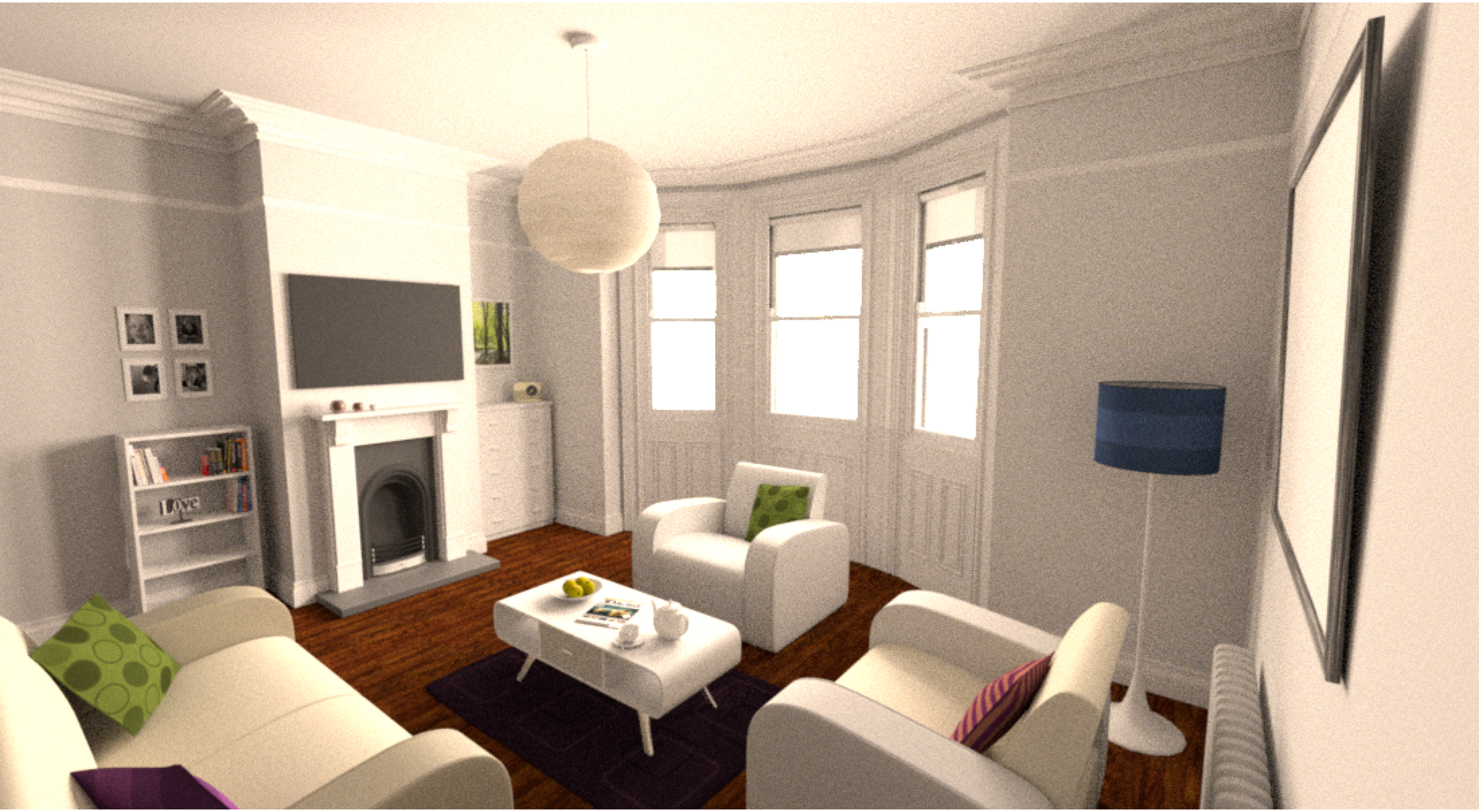
White Room, 32 pixel samples



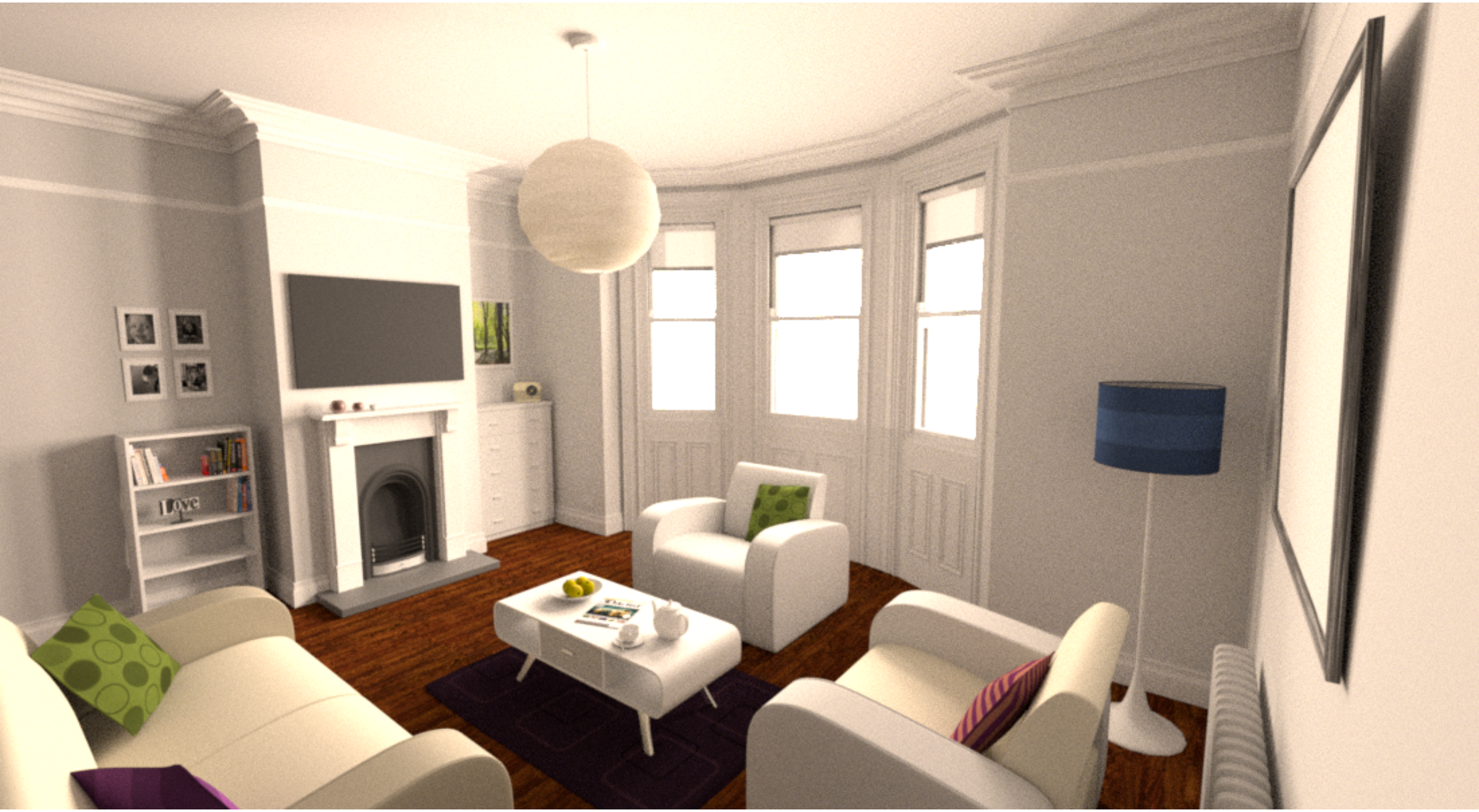
White Room, 64 pixel samples



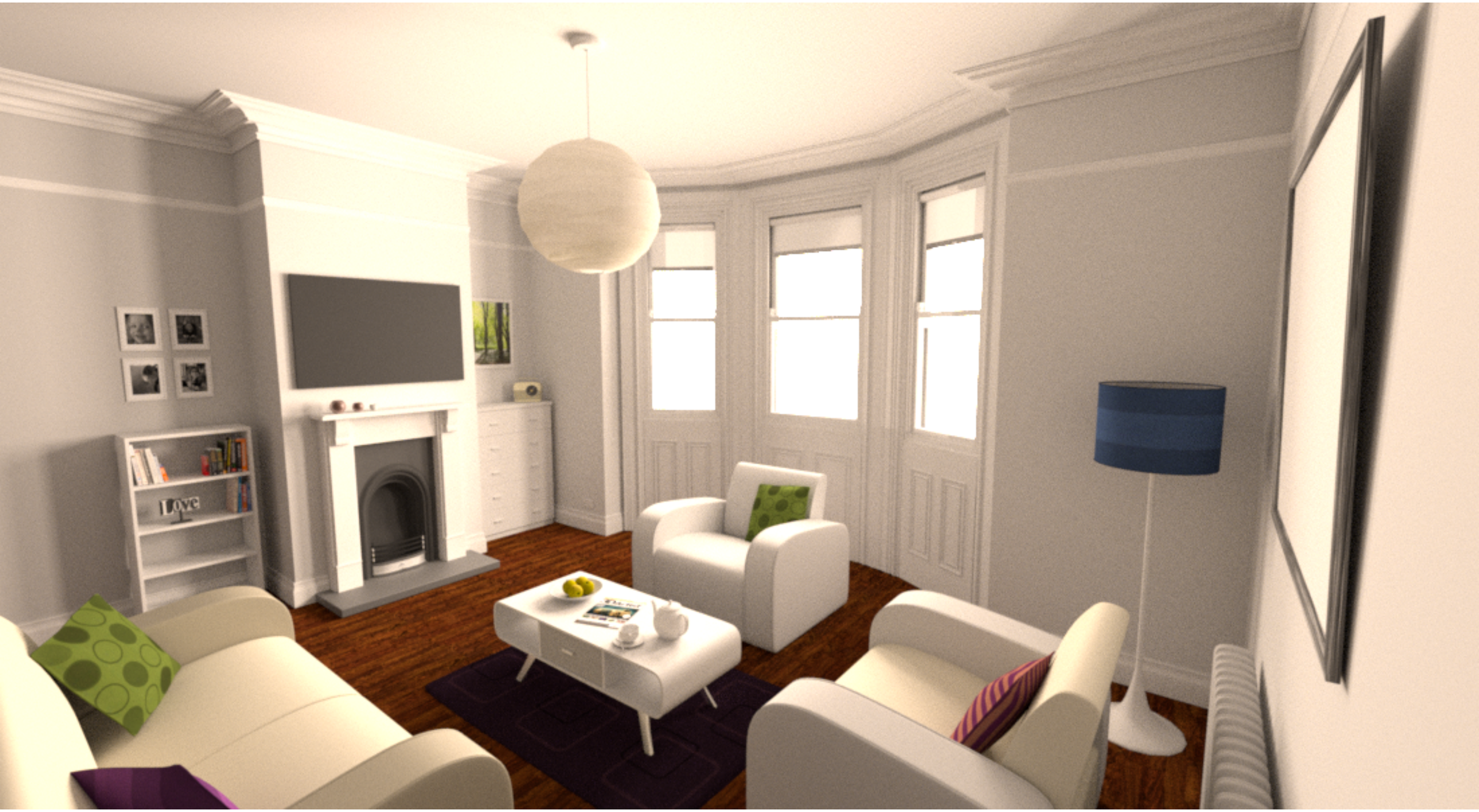
White Room, 128 pixel samples



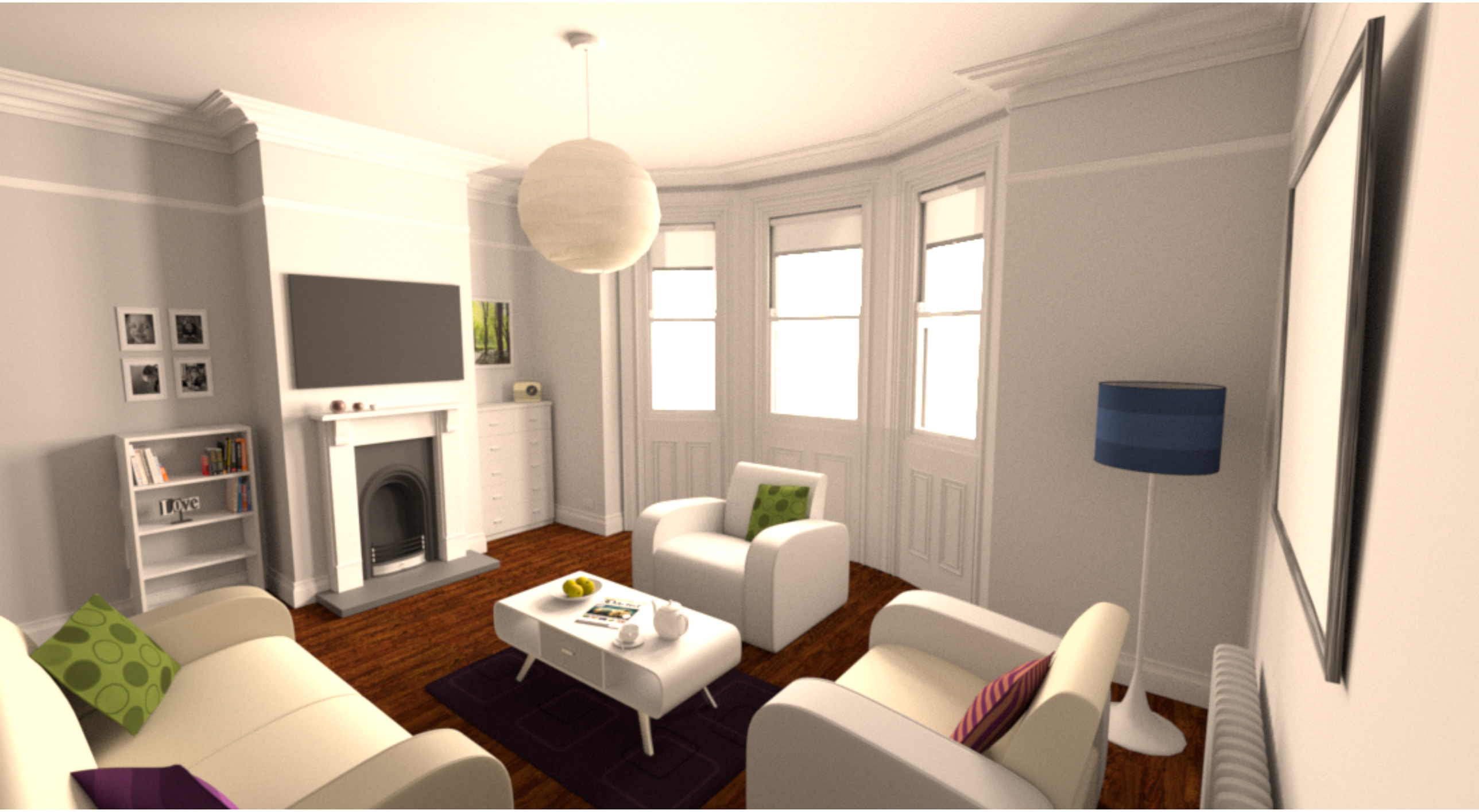
White Room, 256 pixel samples



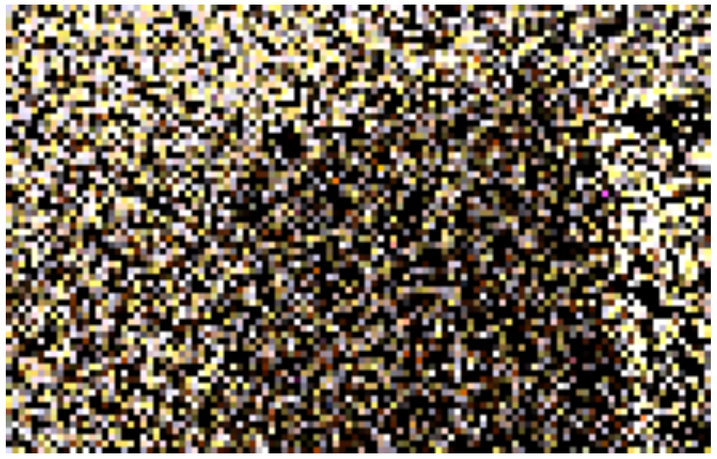
White Room, 512 pixel samples



White Room, 1024 pixel samples



Zoom-in by Bookshelf

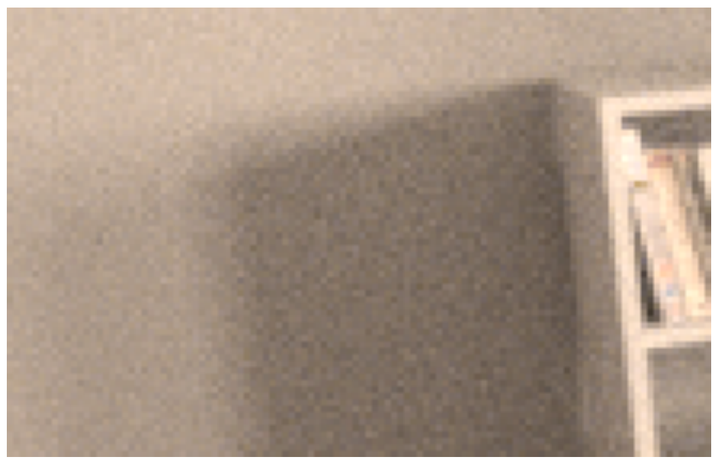


1 sample

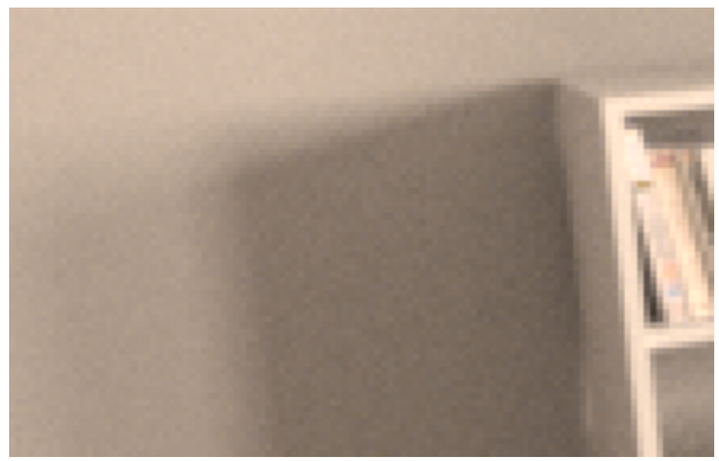
4 samples

16 samples

64 samples



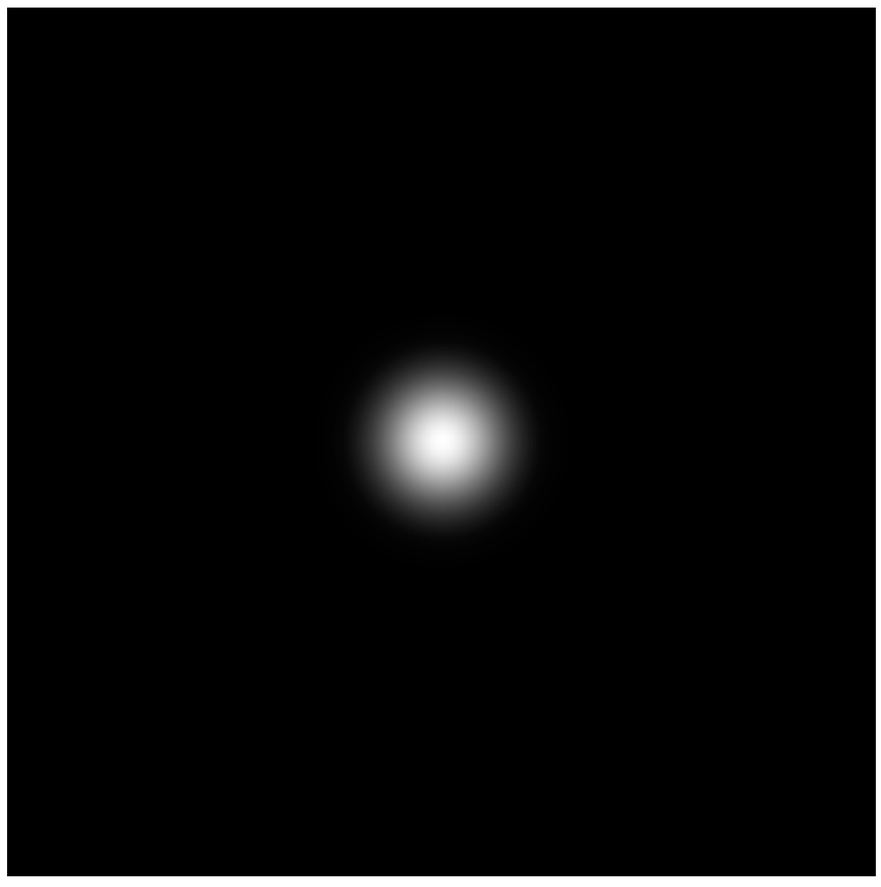
256 samples



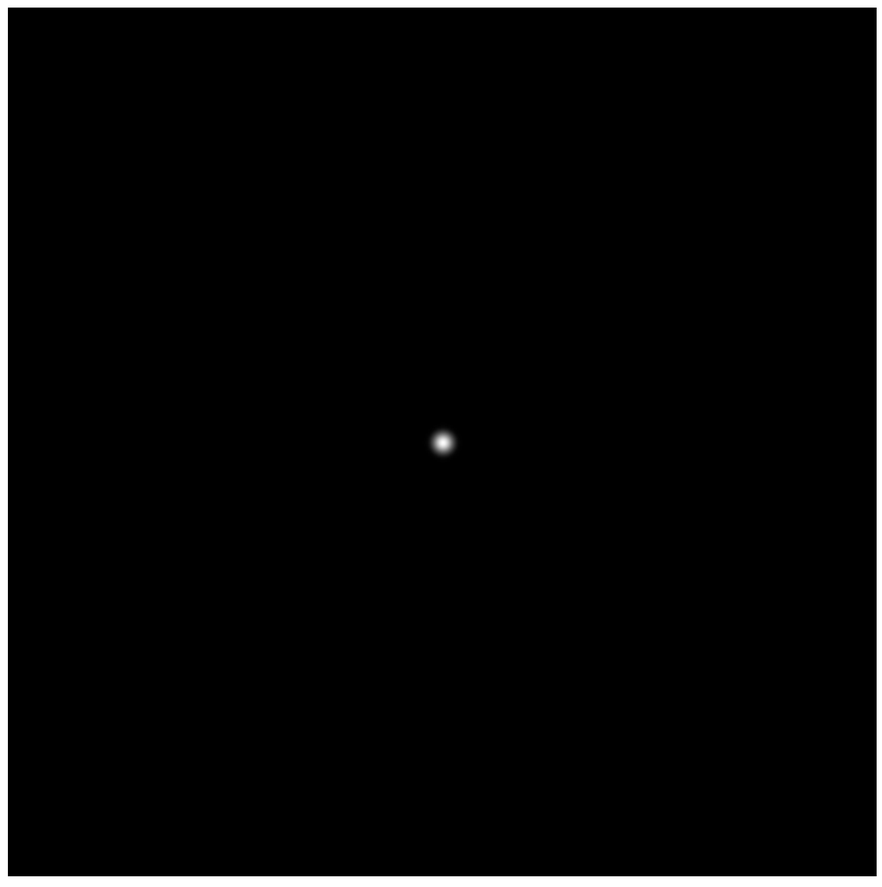
1024 samples

Gaussian Filter

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

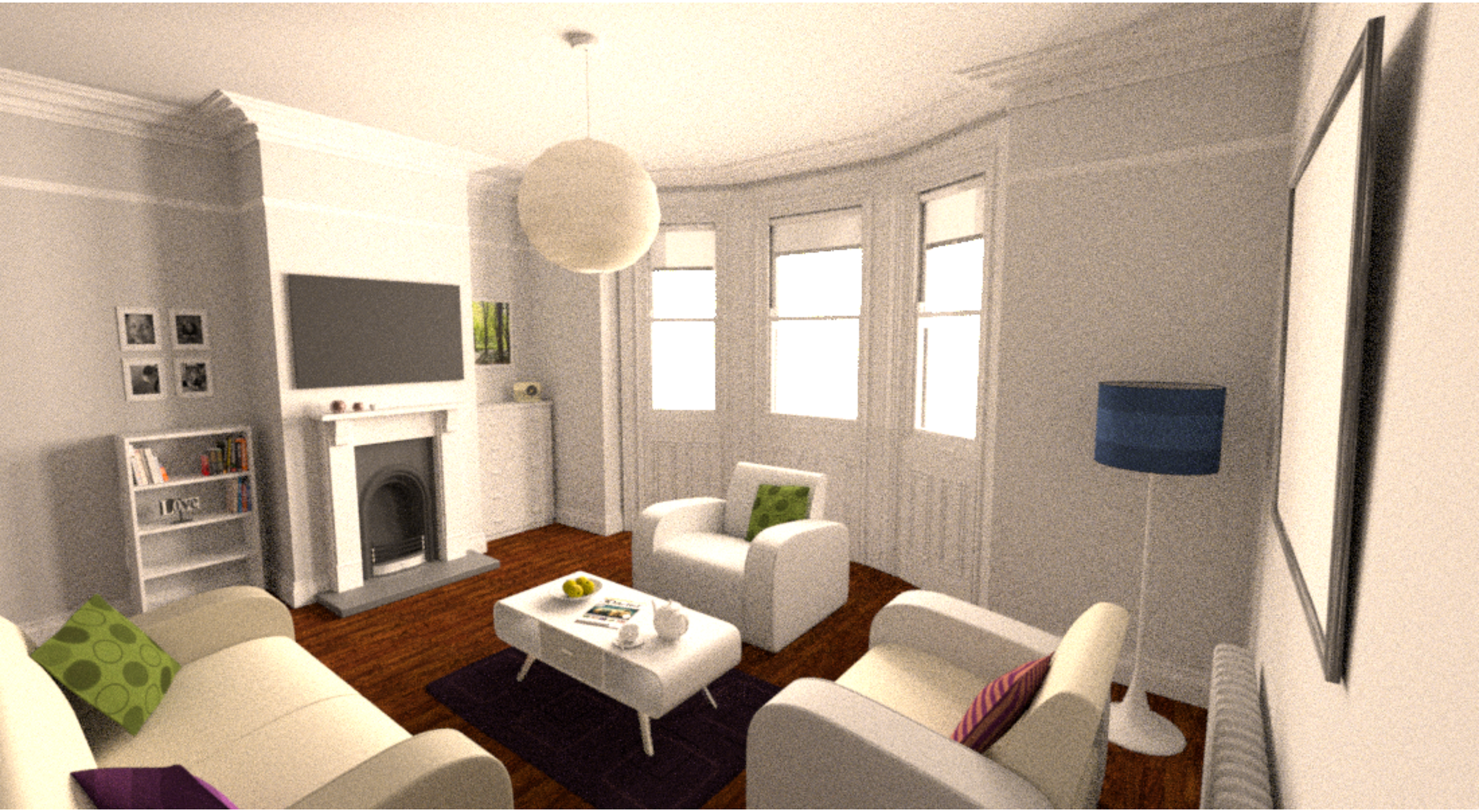


Spatial Domain

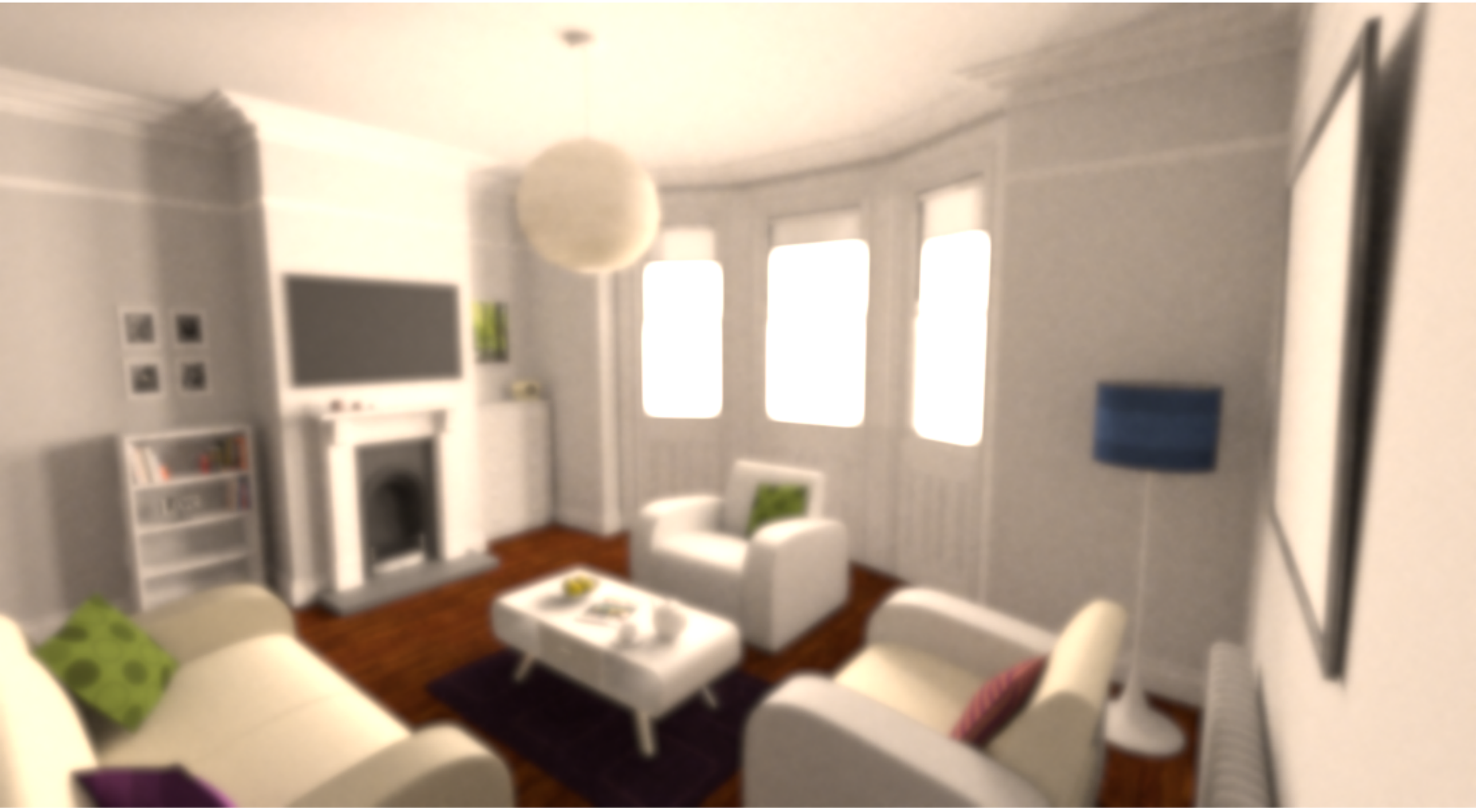


Frequency Domain

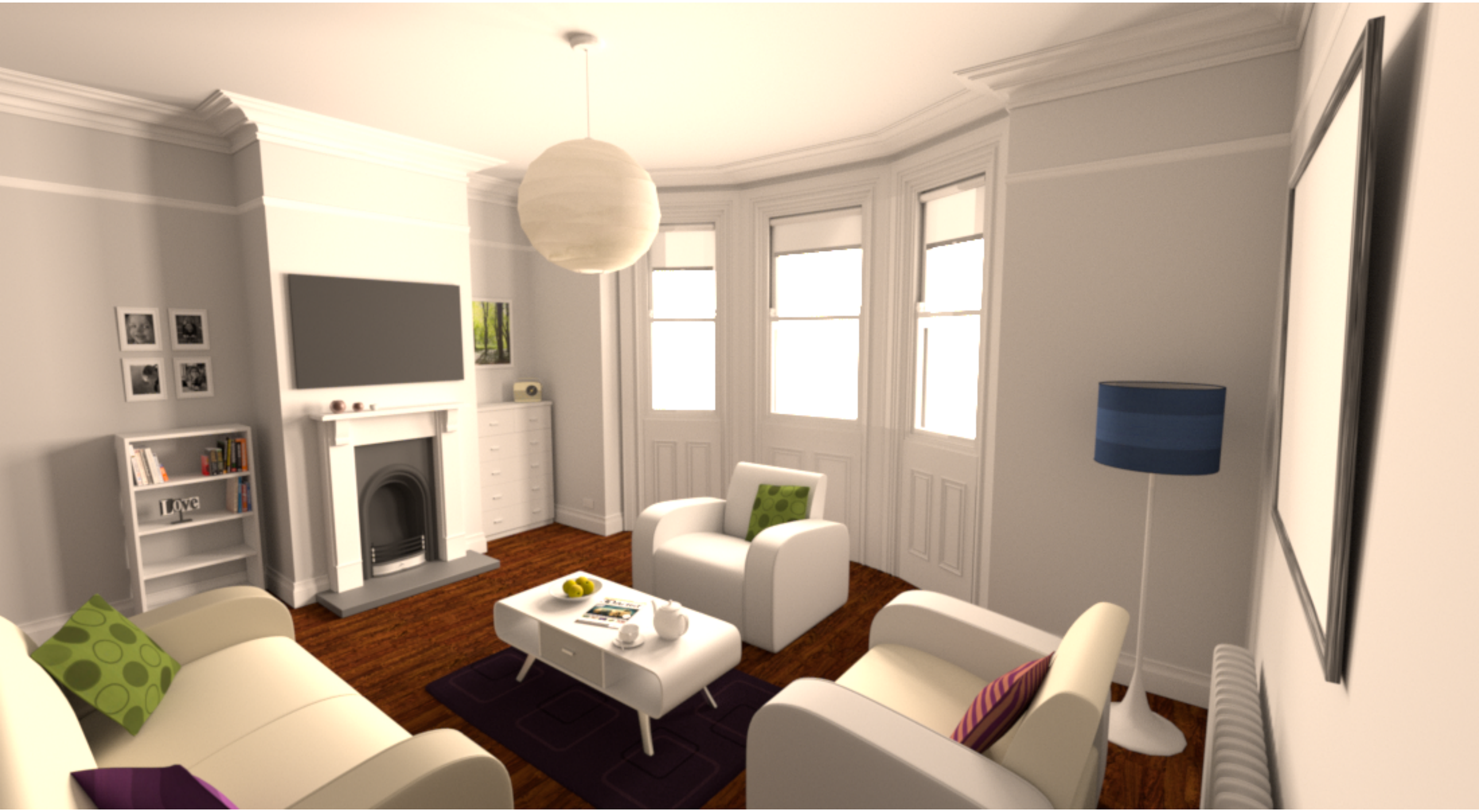
64 pixel samples



19x19 Gaussian Blur



White Room, 4096 pixel samples



Separate Illumination & Reflection

Hemispherical directional reflectance:

$$\rho_{\mathrm{hd}}(\omega_{\mathrm{o}}) = \int_{H^2} f_r(\omega_{\mathrm{i}} \to \omega_{\mathrm{o}}) \, \cos\theta_{\mathrm{i}} \, \mathrm{d}\omega_{\mathrm{i}}$$

Recall the reflection equation:

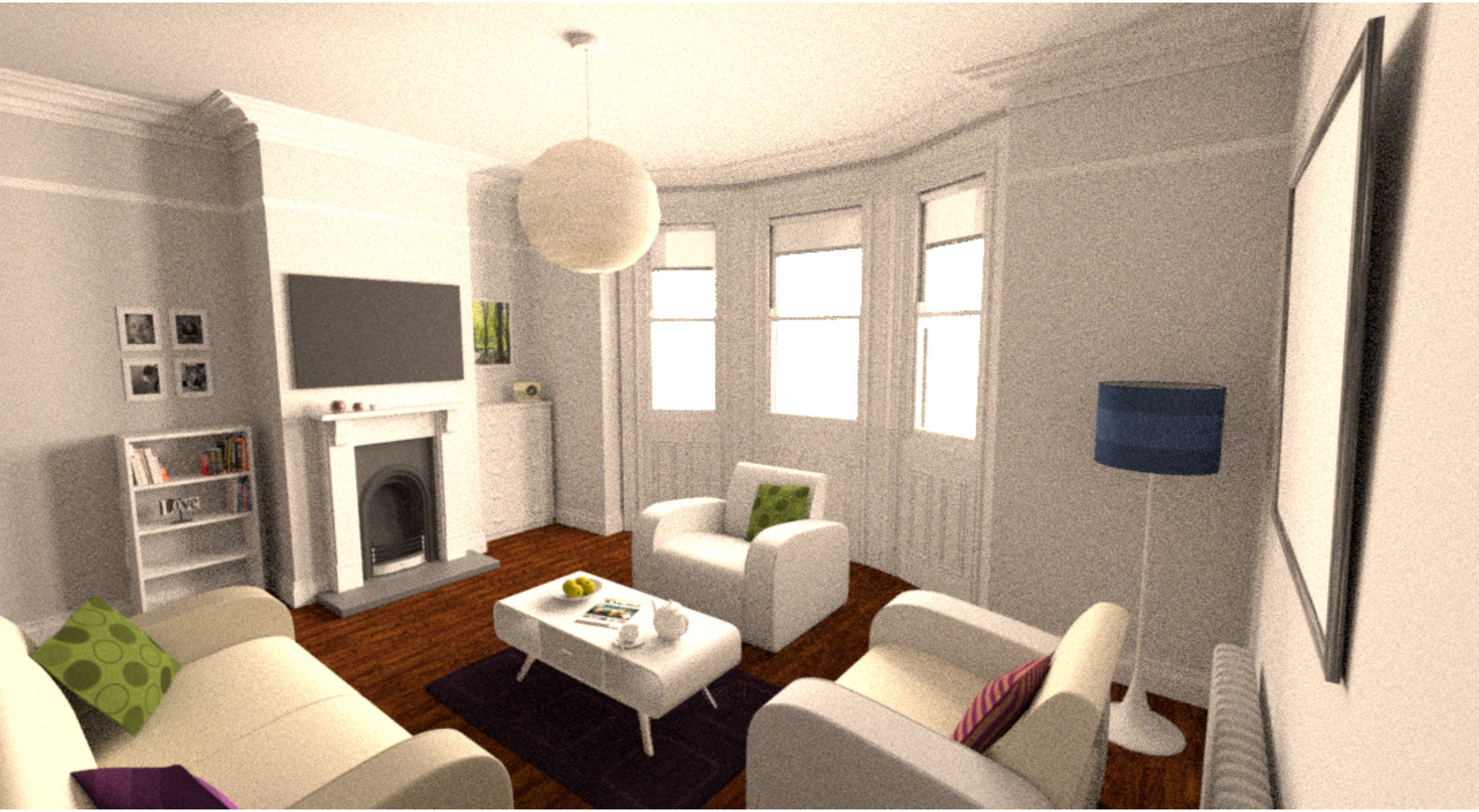
$$L_{\mathrm{o}}(\omega_{\mathrm{o}}) = \int_{H^2} f_r(\omega_{\mathrm{i}} \to \omega_{\mathrm{o}}) \, L_{\mathrm{i}}(\omega_{\mathrm{i}}) \, \cos\theta_{\mathrm{i}} \, \mathrm{d}\omega_{\mathrm{i}}$$

Ratio approximates incident radiance:

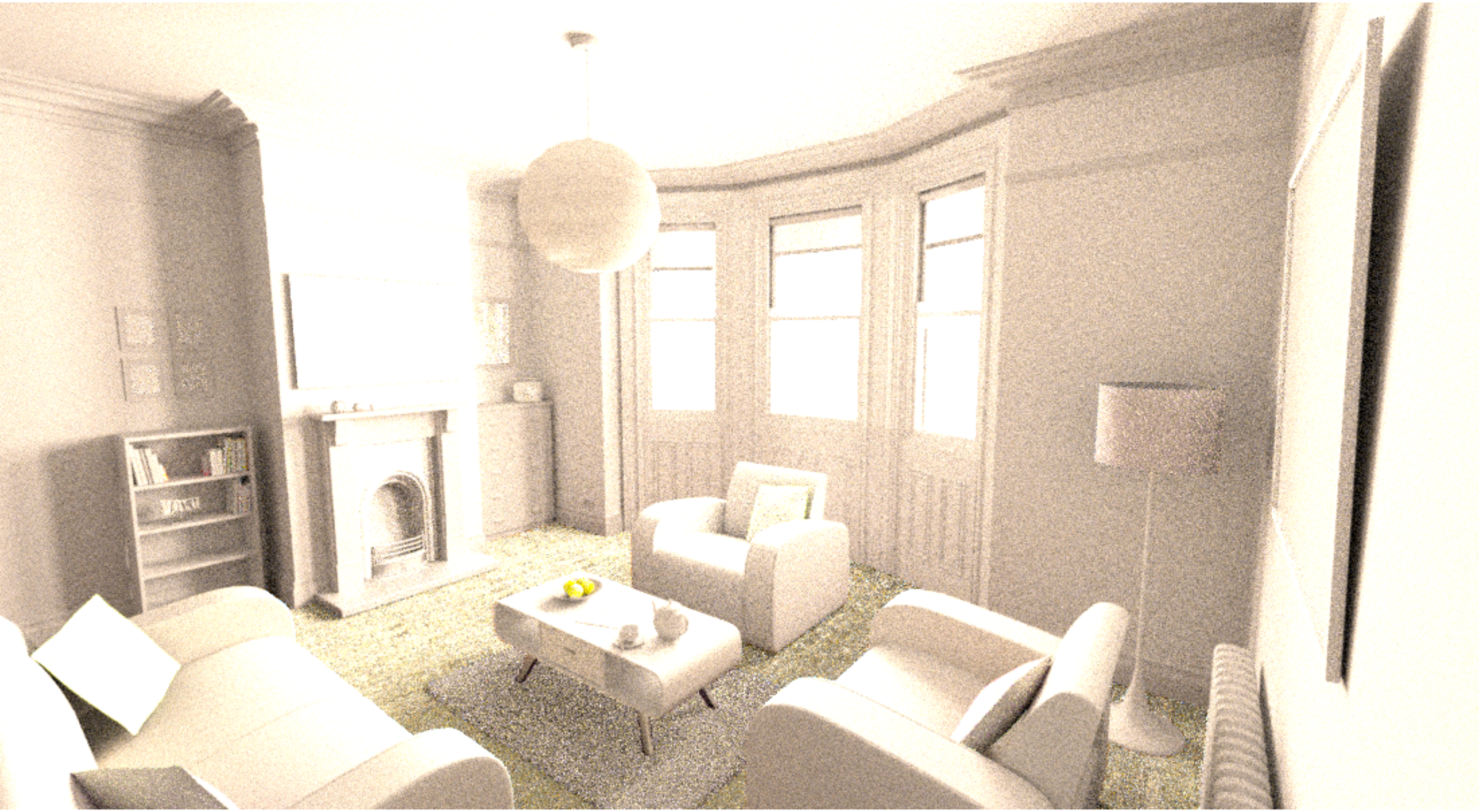
$$\frac{L_{\mathrm{o}}(\omega_{\mathrm{o}})}{\rho_{\mathrm{hd}}(\omega_{\mathrm{o}})} = \frac{\int_{H^2} f_r(\omega_{\mathrm{i}} \to \omega_{\mathrm{o}}) \, L_{\mathrm{i}}(\omega_{\mathrm{i}}) \, \cos\theta_{\mathrm{i}} \, \mathrm{d}\omega_{\mathrm{i}}}{\int_{H^2} f_r(\omega_{\mathrm{i}} \to \omega_{\mathrm{o}}) \, \cos\theta_{\mathrm{i}} \, \mathrm{d}\omega_{\mathrm{i}}} \approx \mathrm{avg} \; L_{\mathrm{i}}$$

Hemi-Directional Reflectance (Albedo)

White Room, 64 pixel samples

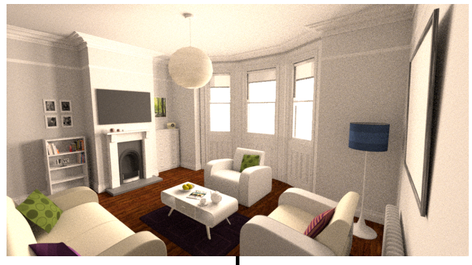


Final Image / Albedo ~ Illumination



General Pipeline

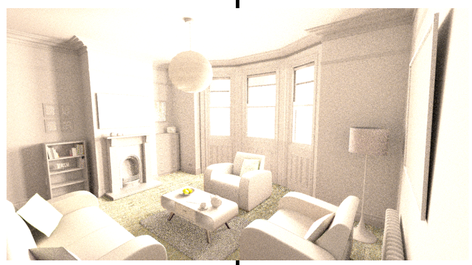
Noisy image



/

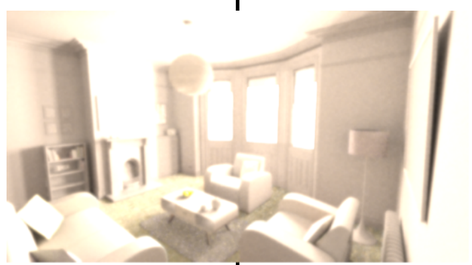
Albedo

Noisy Illumination



Denoising

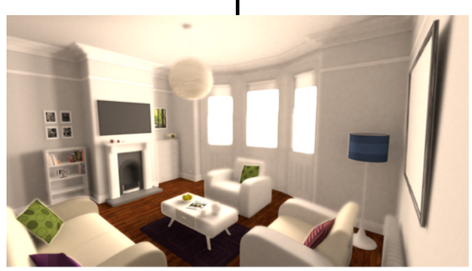
Denoised Illumination



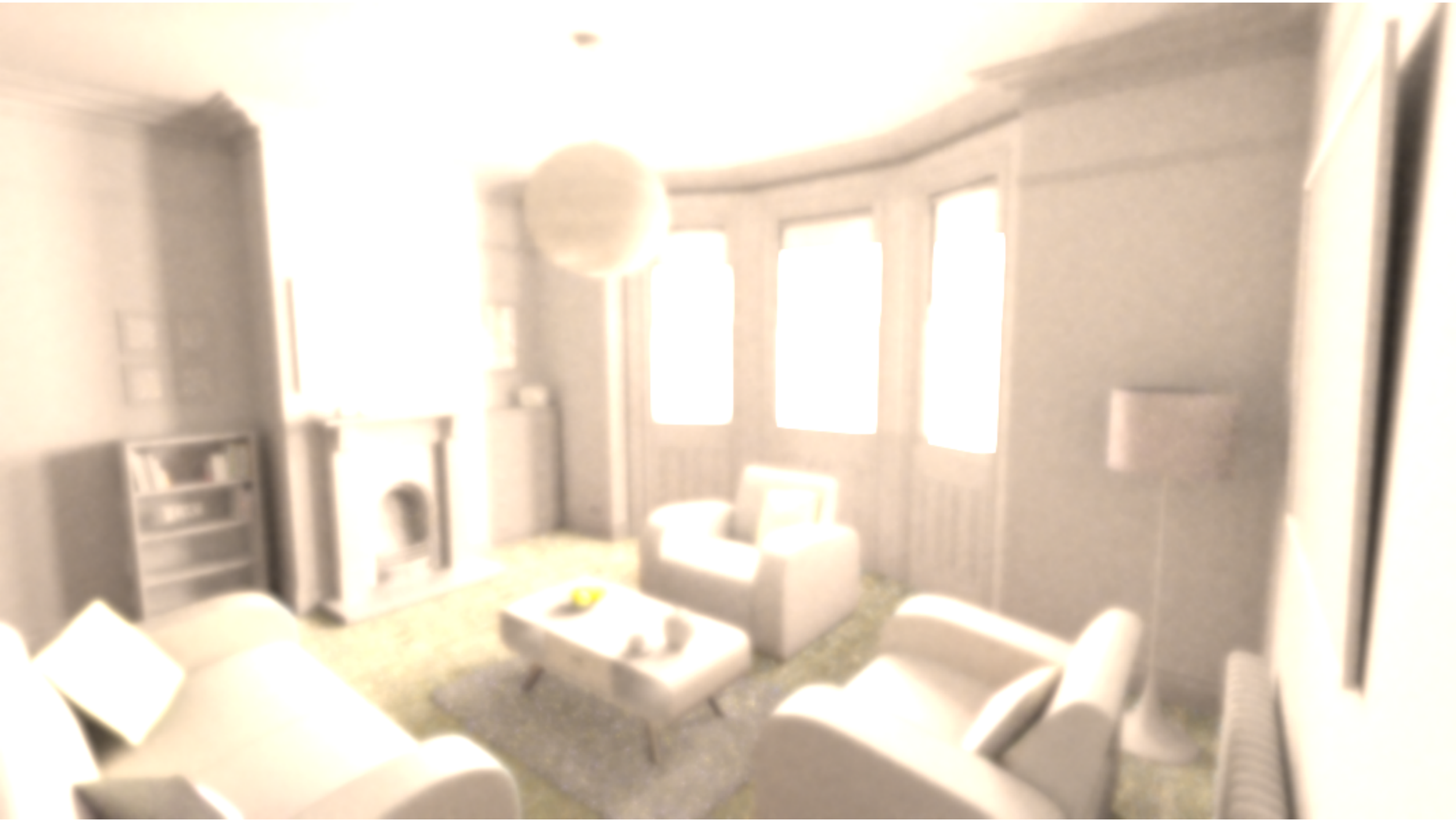
*

Albedo

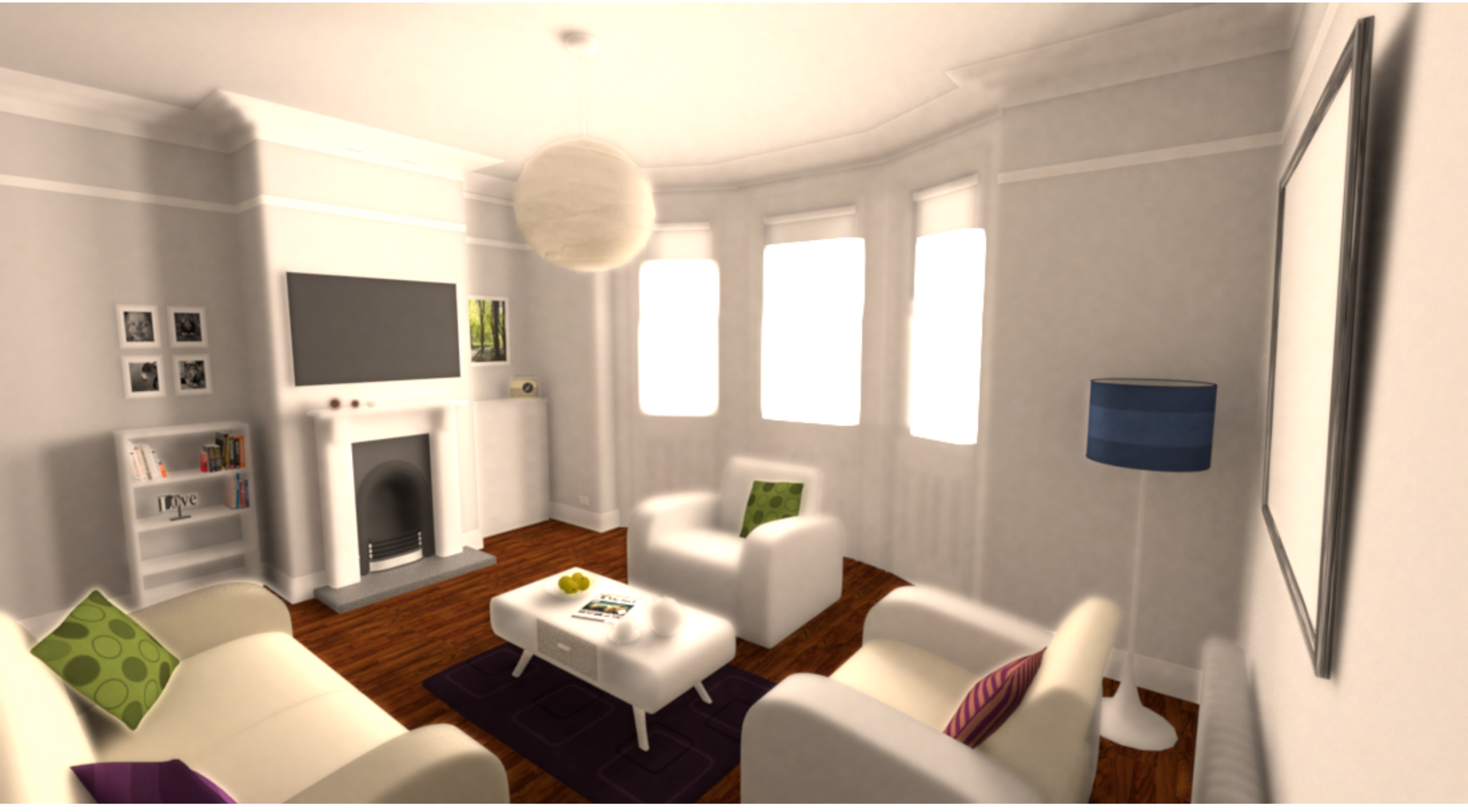
Denoised image



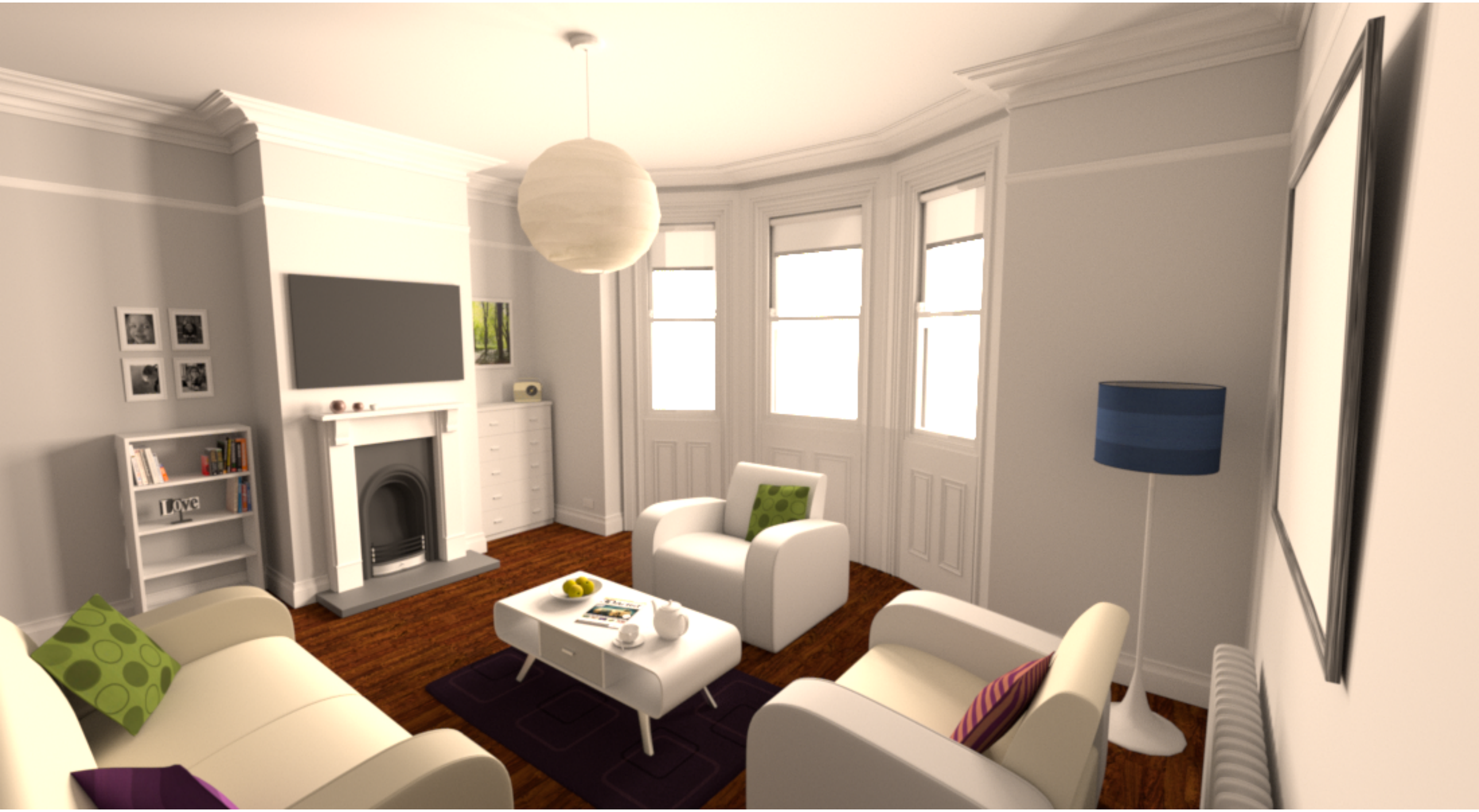
Gaussian Blurred Illumination



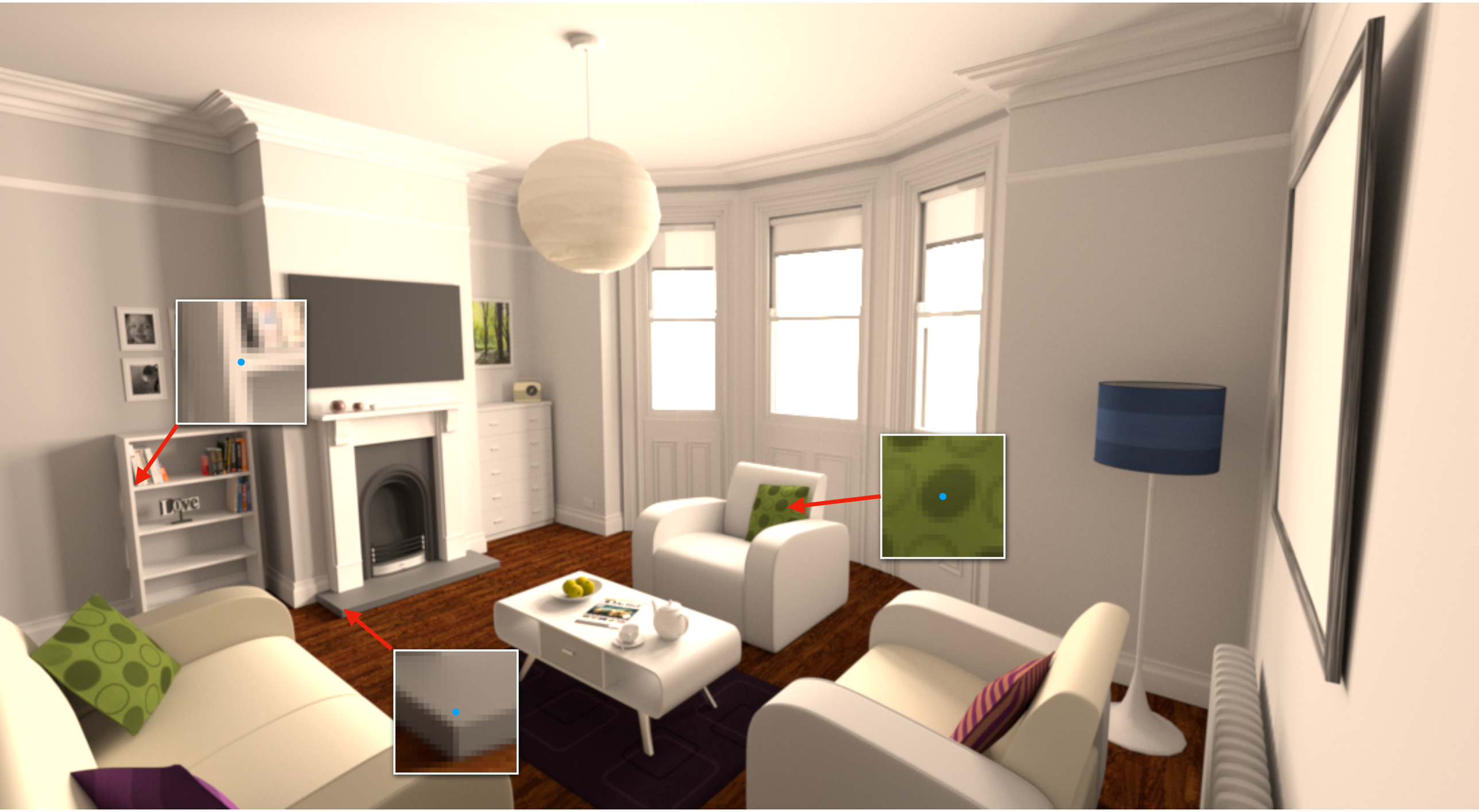
Blurred Illumination * Albedo



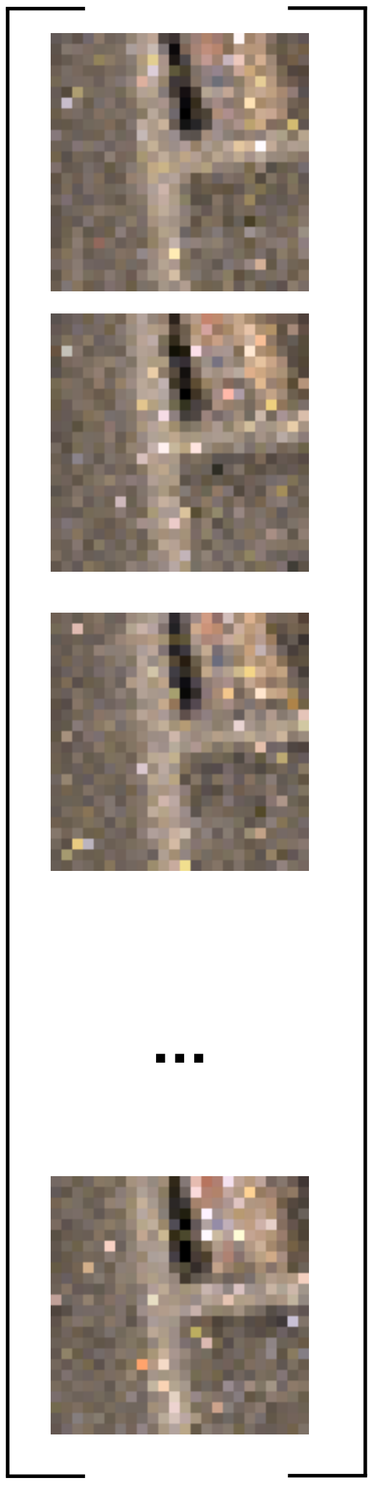
White Room, 4096 pixel samples



Ideal Denoising Filters (via Brute Force)



Ideal Denoising Filters (via Brute Force)

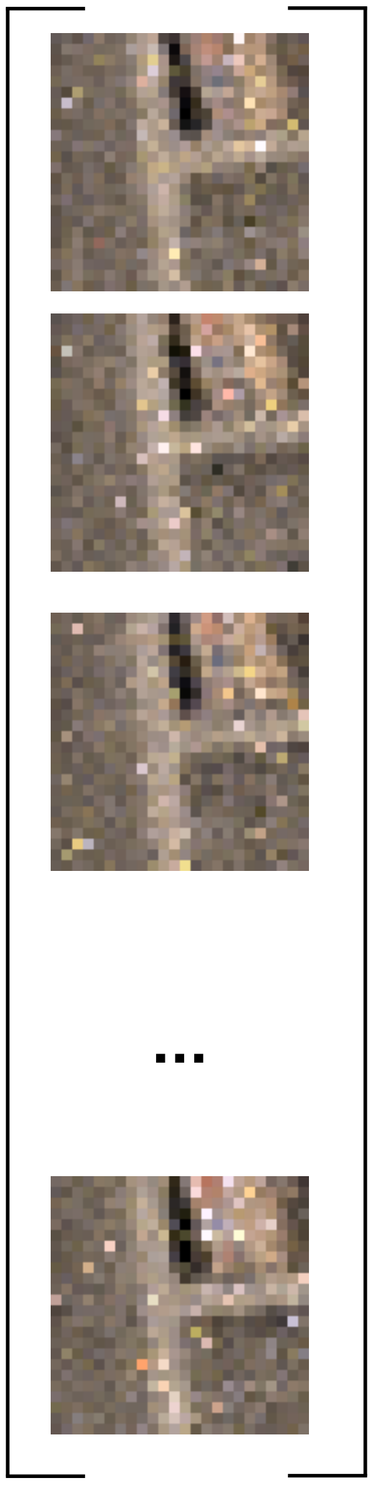


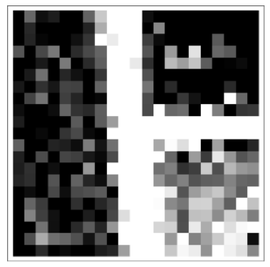
← Independent rendered images

$$\left[\ \mathbf{f}\ \right] = $$

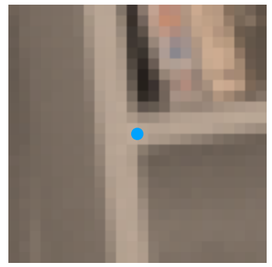
Solve for filter using linear least squares

Ideal Denoising Filters (via Brute Force)

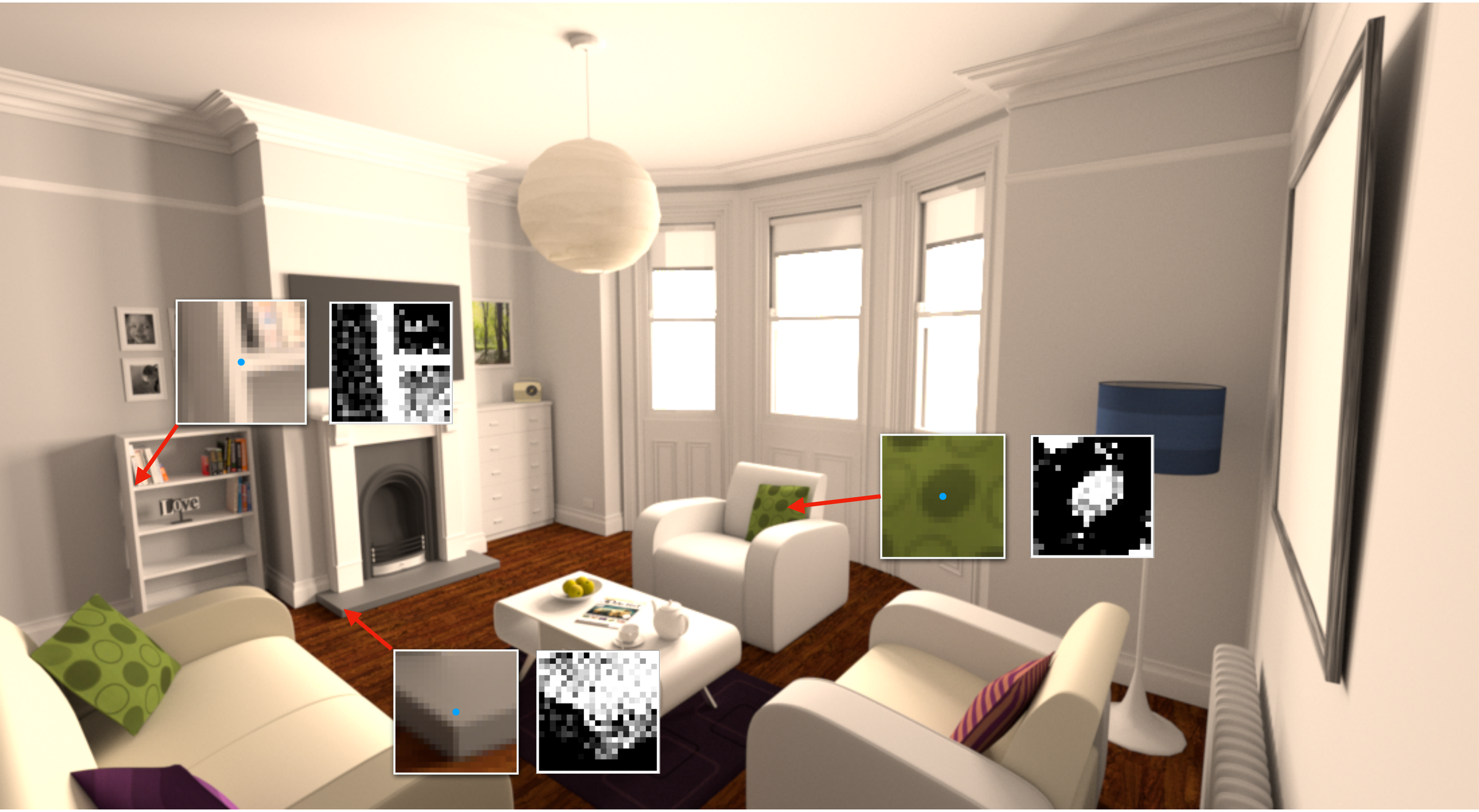




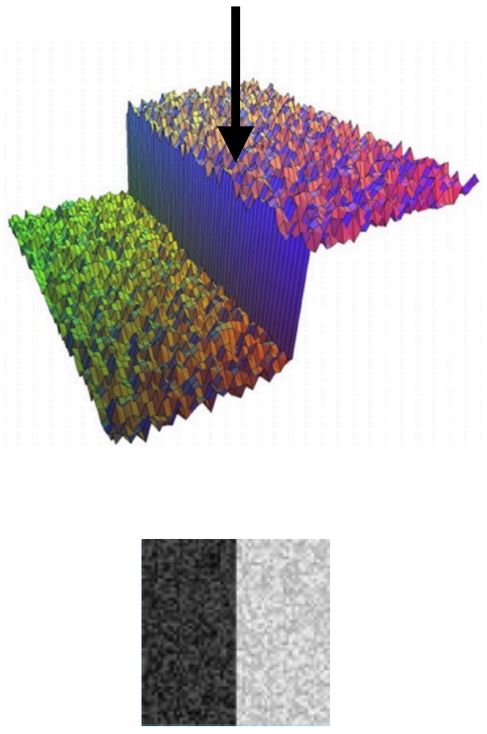
≈

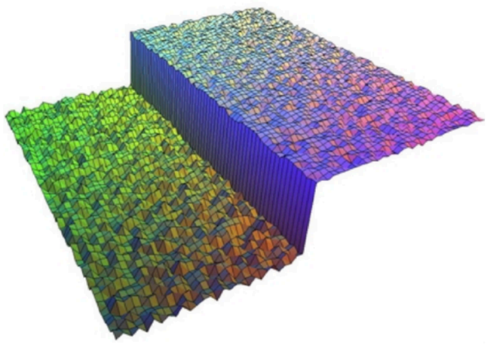


Ideal Denoising Filters (via Brute Force)



Better Filter: Bilateral

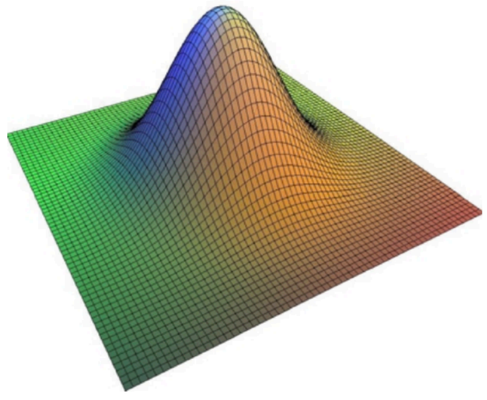




Pixel intensity kernel

$$G_p(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{d((0,0),(x,y))^2}{2\sigma^2}}$$

e.g. $d(p,p') = |I(p) - I(p')|$

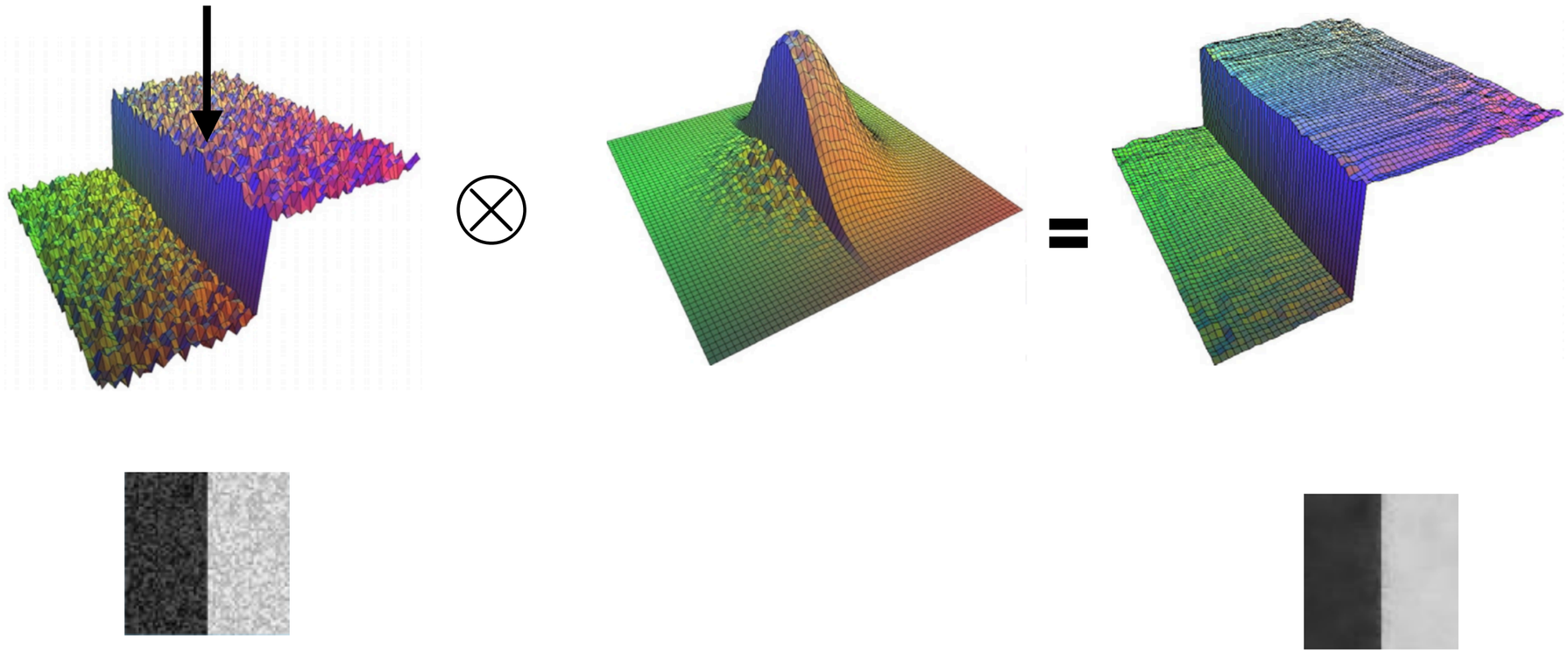


Spatial kernel

$$G_s(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}}$$

[Durand and Dorsey 2002]

Better Filter: Bilateral



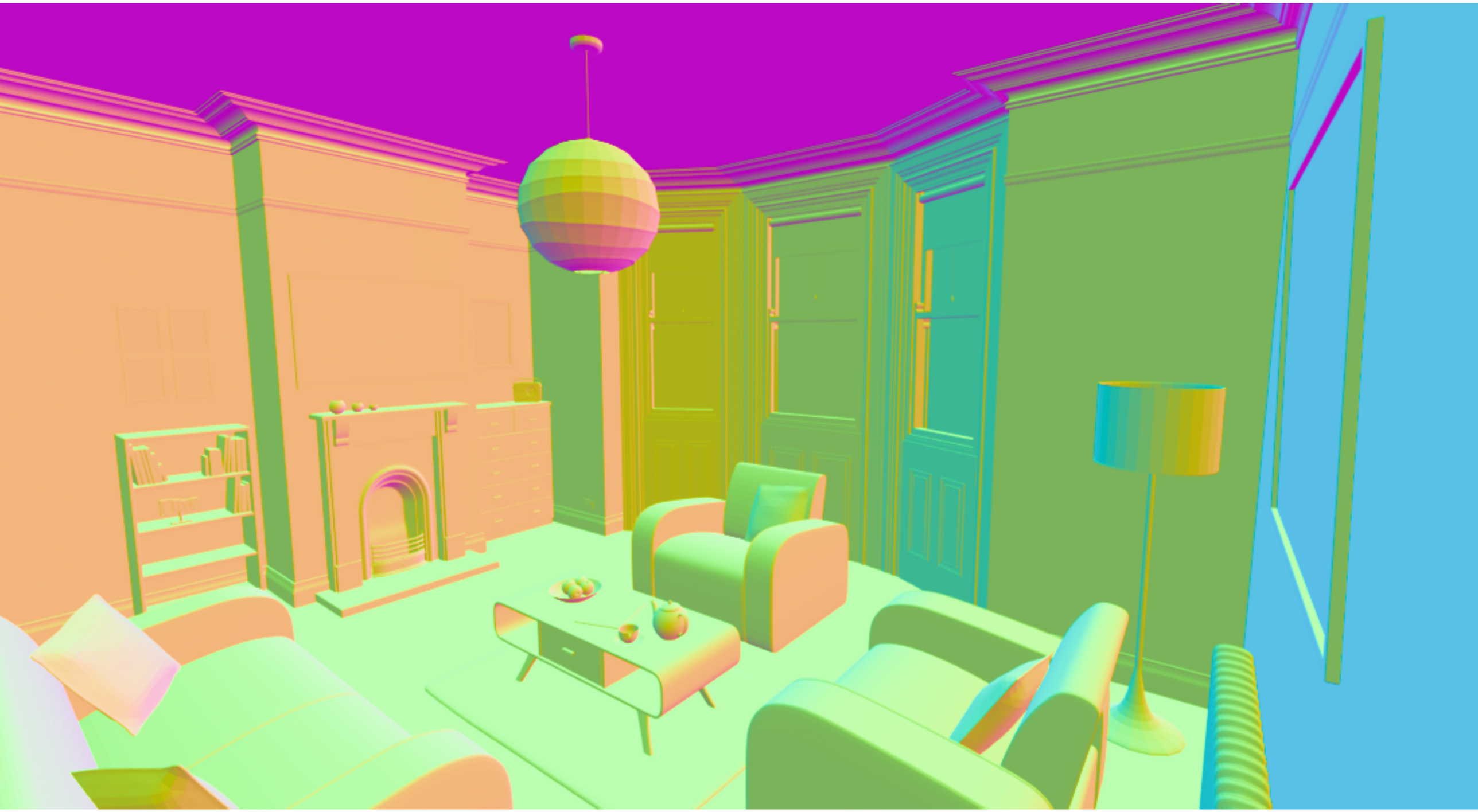
[Durand and Dorsey 2002]

Even Better Filter: Joint Bilateral

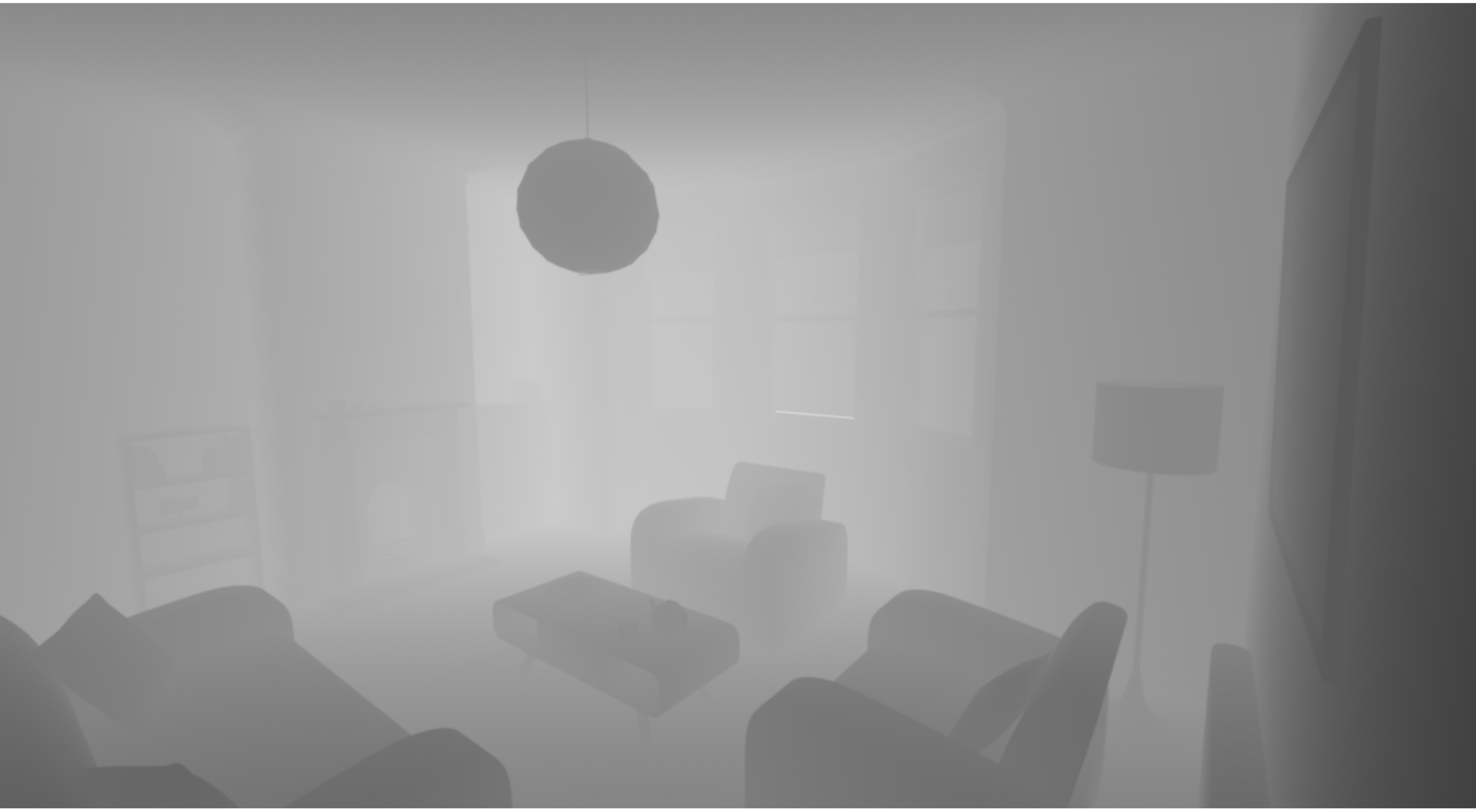
Include additional, non-visible features in the filter

- Pixel depth
- Surface normal
- BRDF features—roughness, ...
- Object id

Surface Normal



Camera Space “z”



Approximating Local Planar Surface

Given camera space z at a pixel, can approximate the local planar surface as:

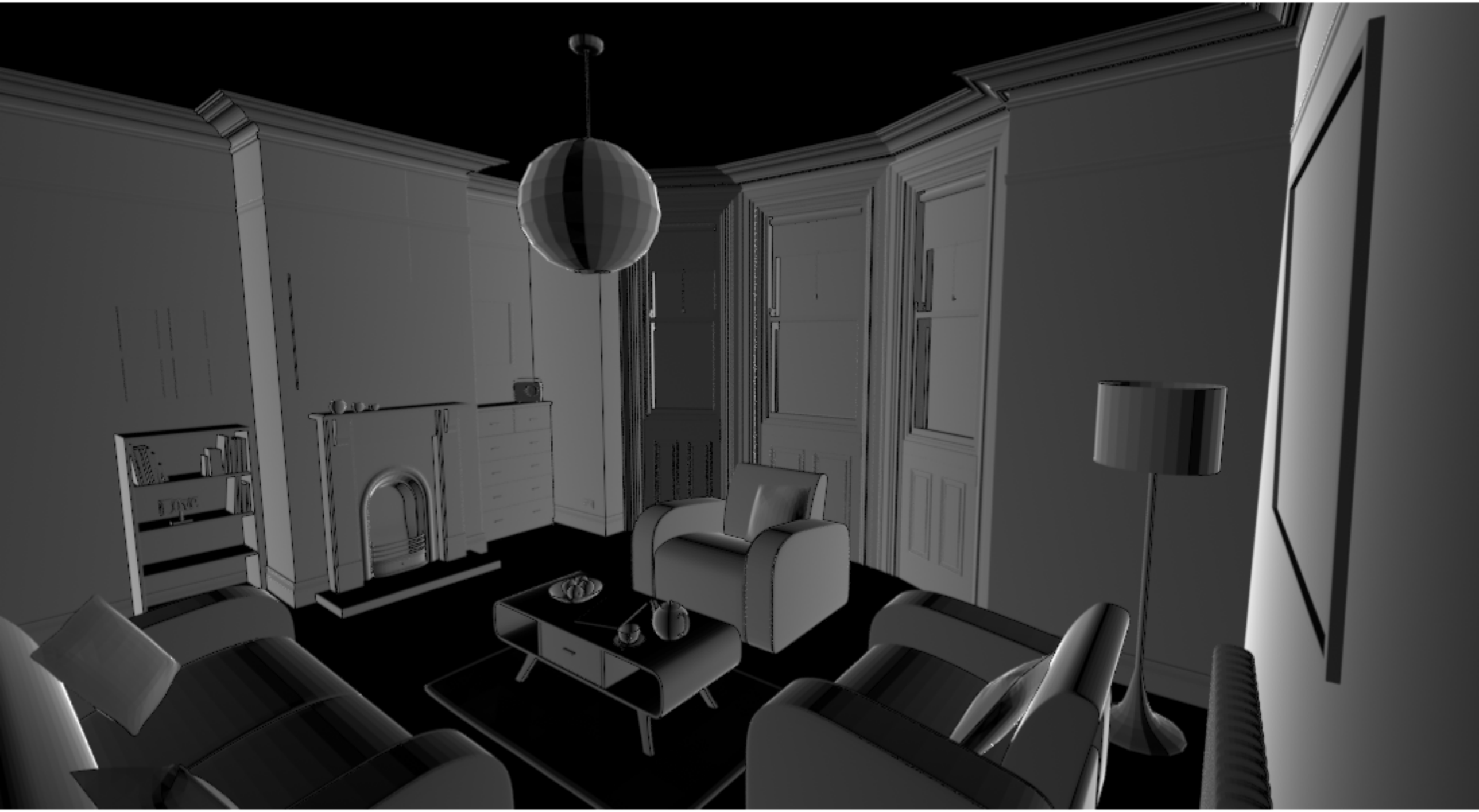
$$z(\Delta x, \Delta y) \approx z + \Delta x \frac{\partial z}{\partial x} + \Delta y \frac{\partial z}{\partial y}$$

Where $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$ are partial derivatives of z in terms of pixel coordinates (x, y).

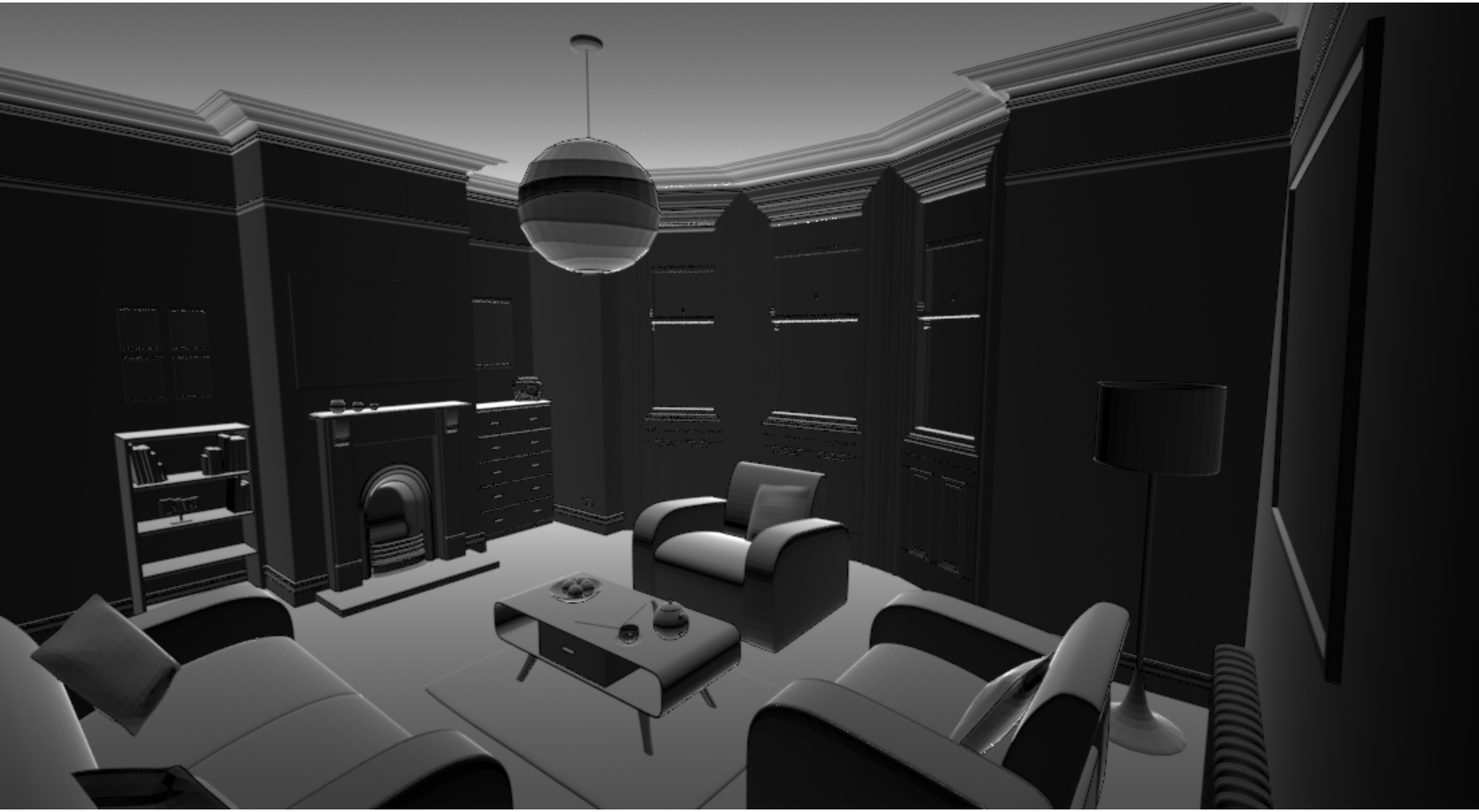
Given a depth z' at a nearby pixel, can compute distance from planar approximation,

$$z' - z(\Delta x, \Delta y)$$

dz/dx



dz/dy



Joint Bilateral Filter Function

$$G = G_s\, G_p\, G_n\, G_z$$

Spatial: $G_s(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}}$

Intensity: $G_p(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{d((0,0),(x,y))^2}{2\sigma^2}}$

Normal difference: $G_n(x,y) = \max\left(\vec{n}(x,y) \cdot \vec{n}(0,0), 0\right)^n$

Depth difference: $G_z(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{(\hat{z}(x,y)-z(0,0))^2}{2\sigma^2}}$

with $\hat{z}(\Delta x, \Delta y) \approx z + \Delta x \frac{\partial z}{\partial x} + \Delta y \frac{\partial z}{\partial y}$

About all those sigmas...

Pixel intensity contribution $G_p(x,y) = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{d((0,0),(x,y))^2}{2\sigma^2}}$
should be based on pixel's variance

Uniform variance:

$$d(p,p')^2 = \frac{\max(0,(I(p)-I(p'))^2 - 2\bar{\sigma}^2)}{\epsilon + k^2 2\bar{\sigma}}$$

Non-uniform variance:

$$d(p,p')^2 = \frac{\max(0,(I(p)-I(p'))^2 - (\mathrm{Var}[p] + \min(\mathrm{Var}[p],\mathrm{Var}[p'])))}{\epsilon + k^2(\mathrm{Var}[p] + \mathrm{Var}[p'])}$$

Direct Illumination Sample Variance

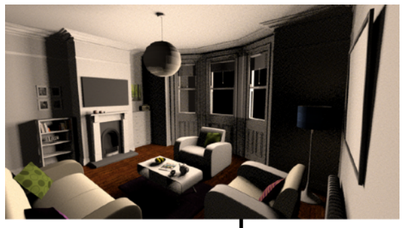
Indirect Illumination Sample Variance

Filtered Direct Variance

Filtered Indirect Variance

Revised Pipeline

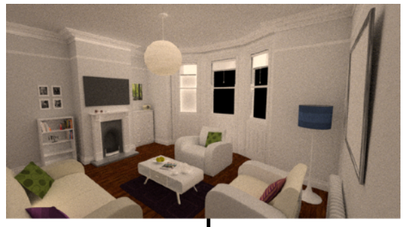
Noisy direct



/

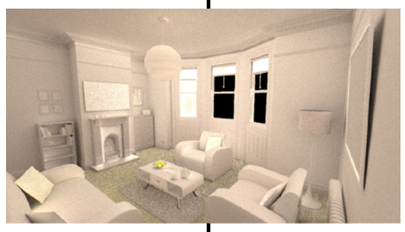
Albedo

/



Noisy indirect

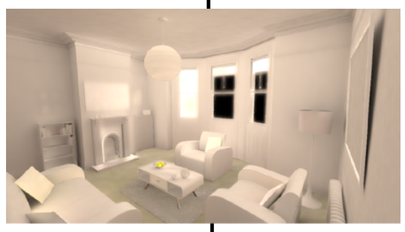
Denoising



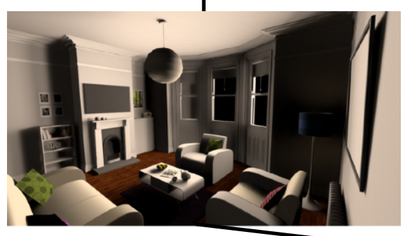
Denoising

*

*



Denoised direct



+

Denoised indirect

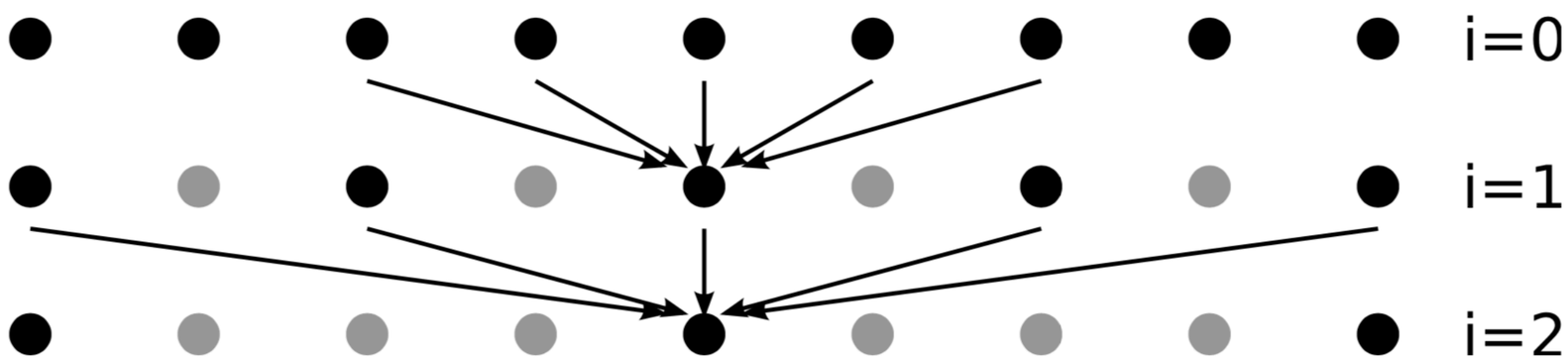
Multi-Level Filtering: À-Trous

Want wide filter kernels to eliminate noise

- **Recall Gaussian 19x19 was still blotchy**

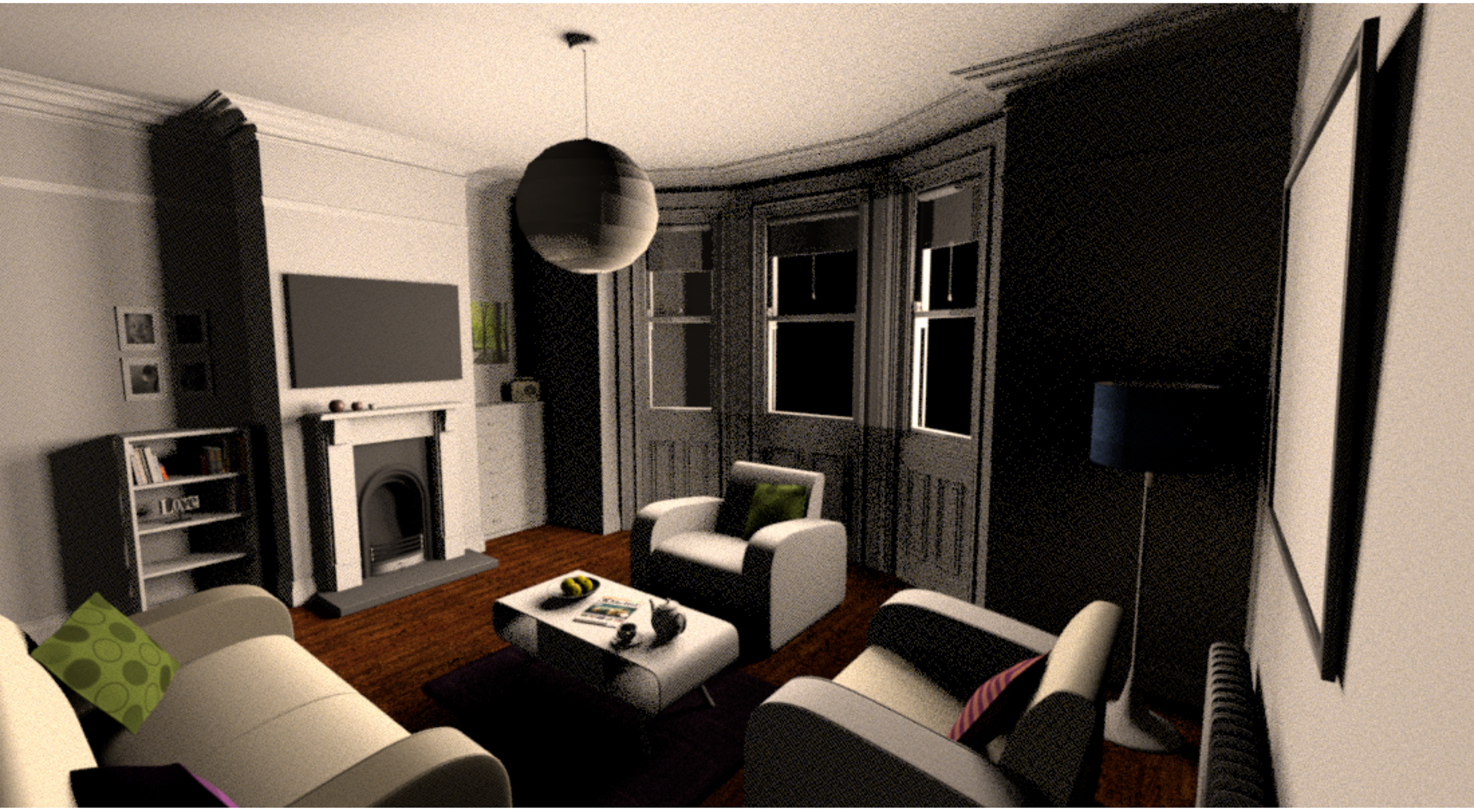
Wide kernels are expensive...

Multi-scale filtering:

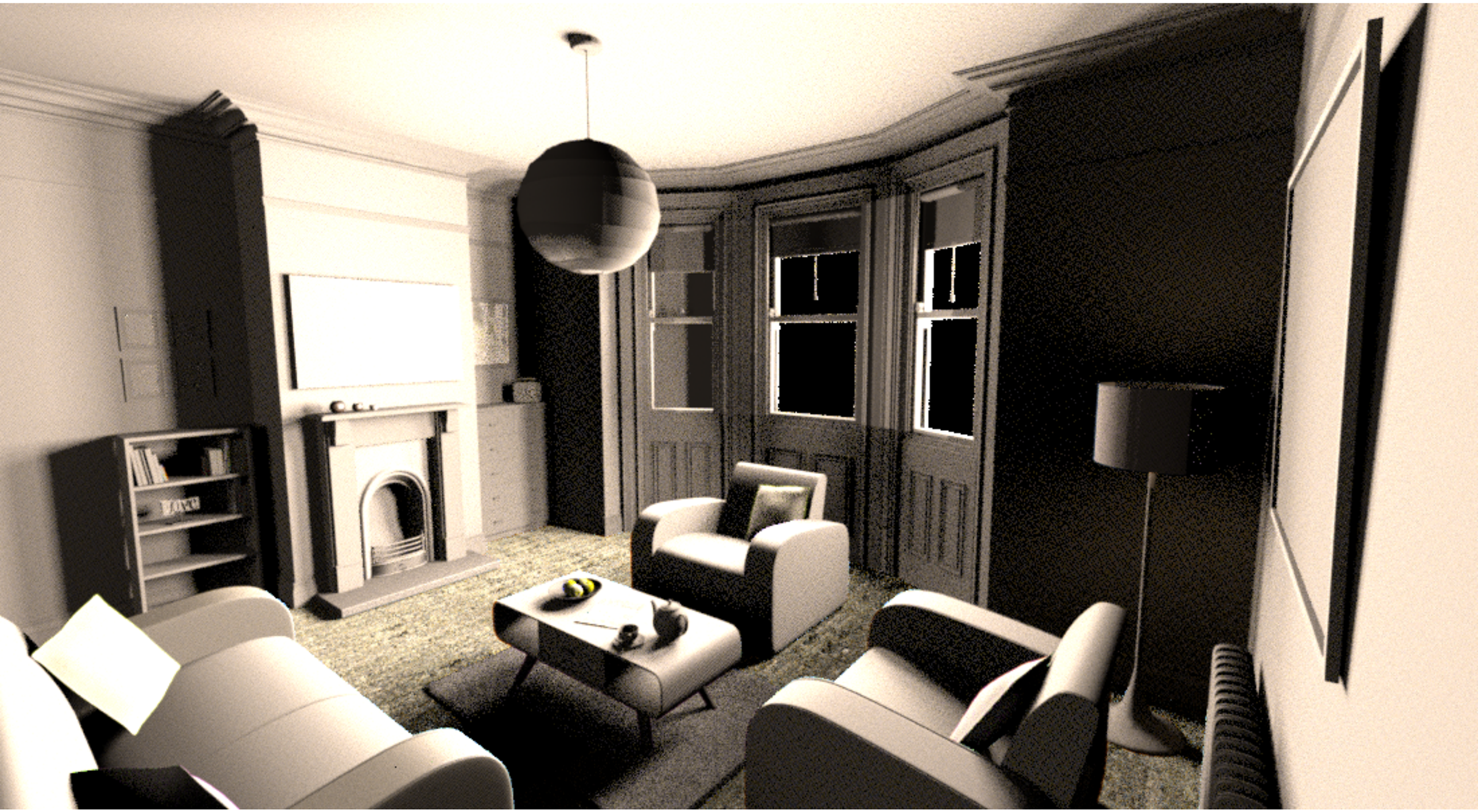


[Dammertz et al. 2010]

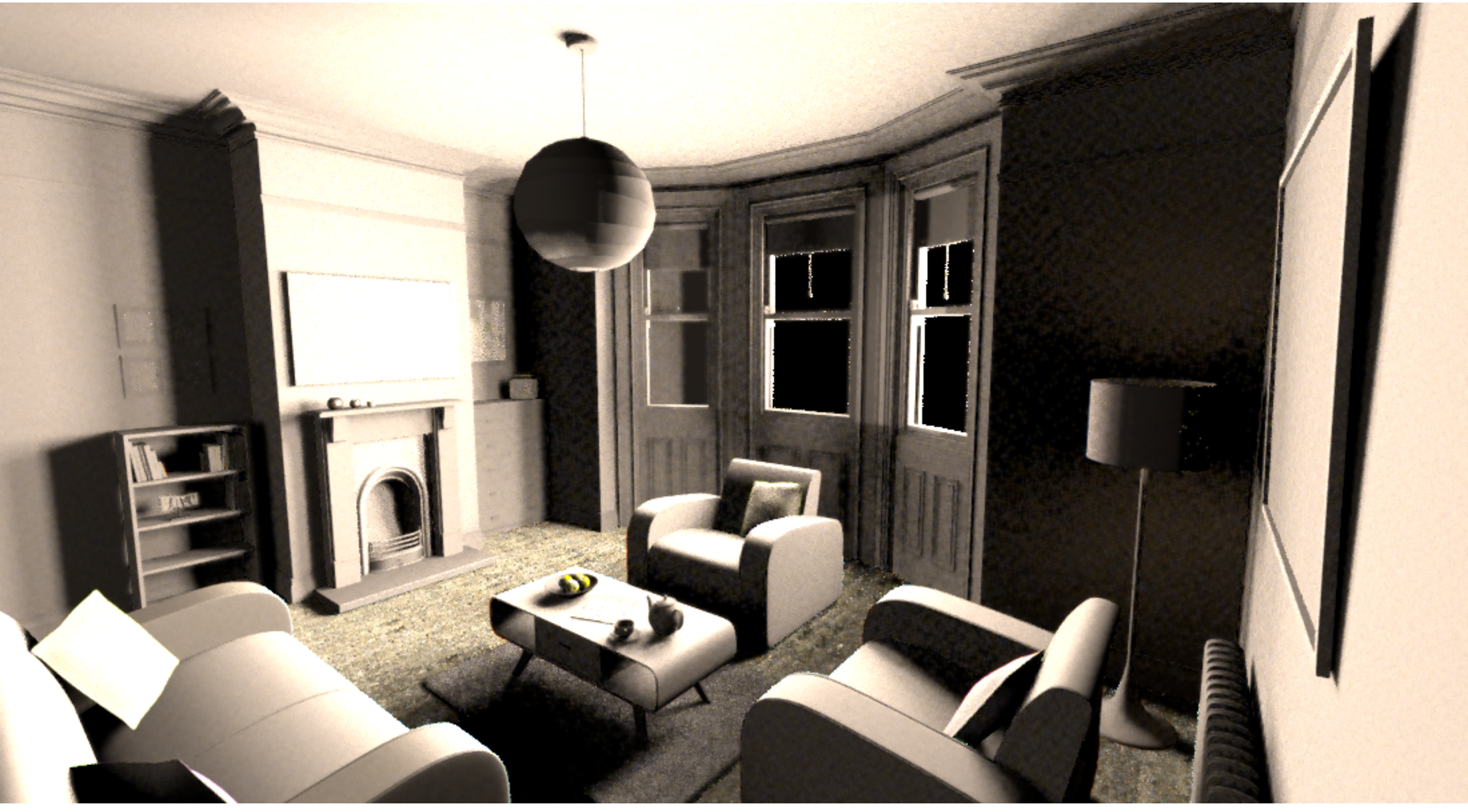
Direct Lighting



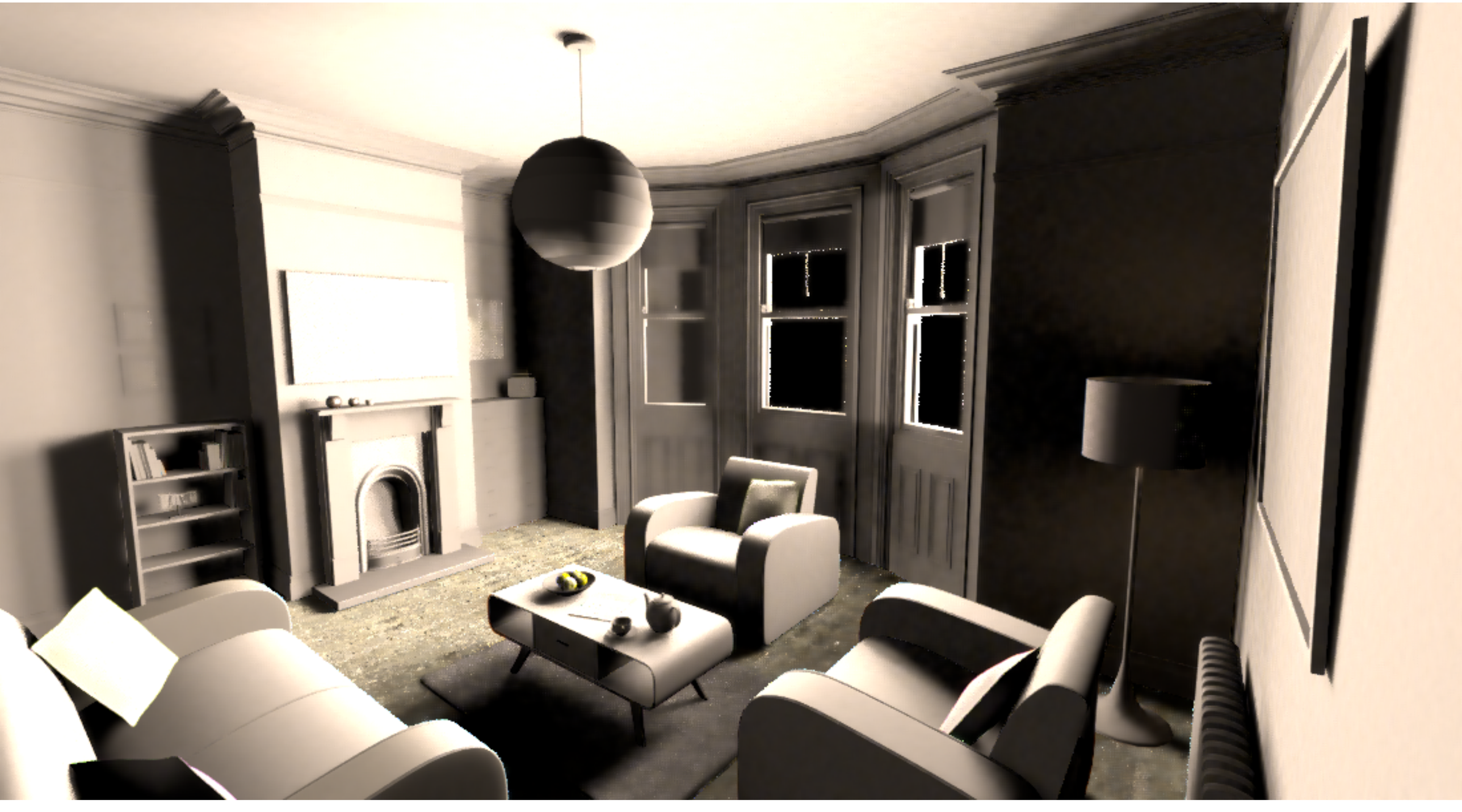
Direct Lighting / Albedo



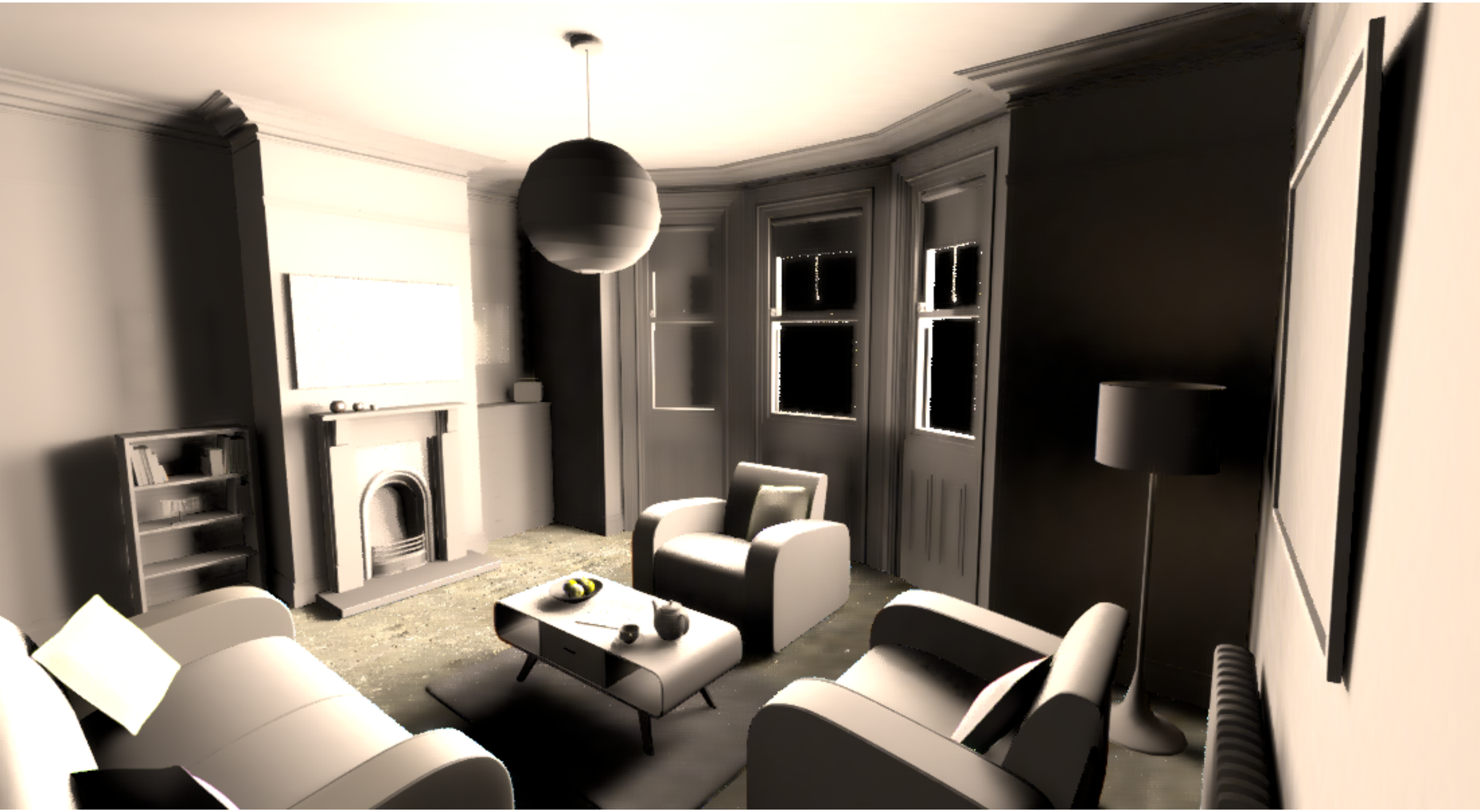
One Iteration



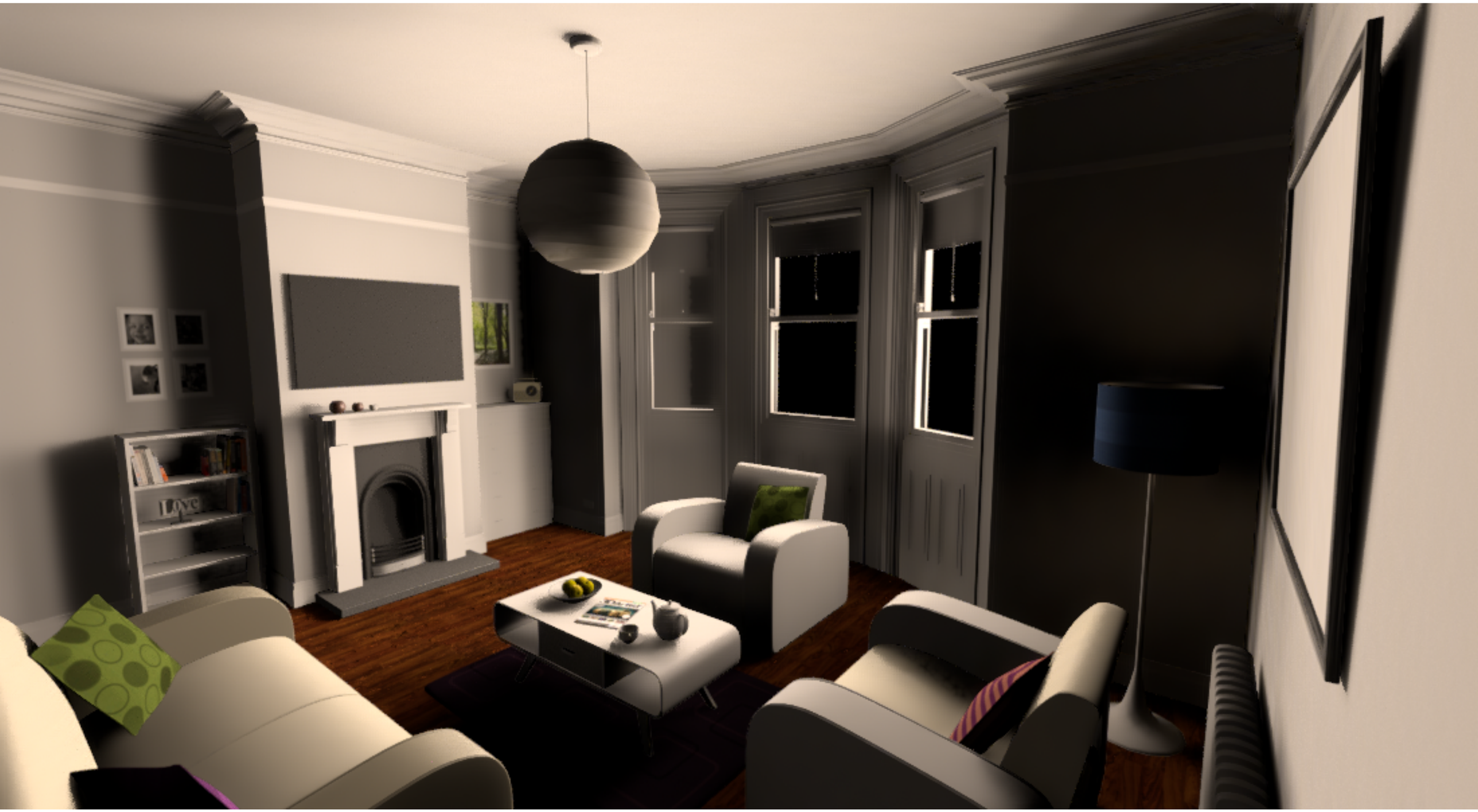
Two Iterations



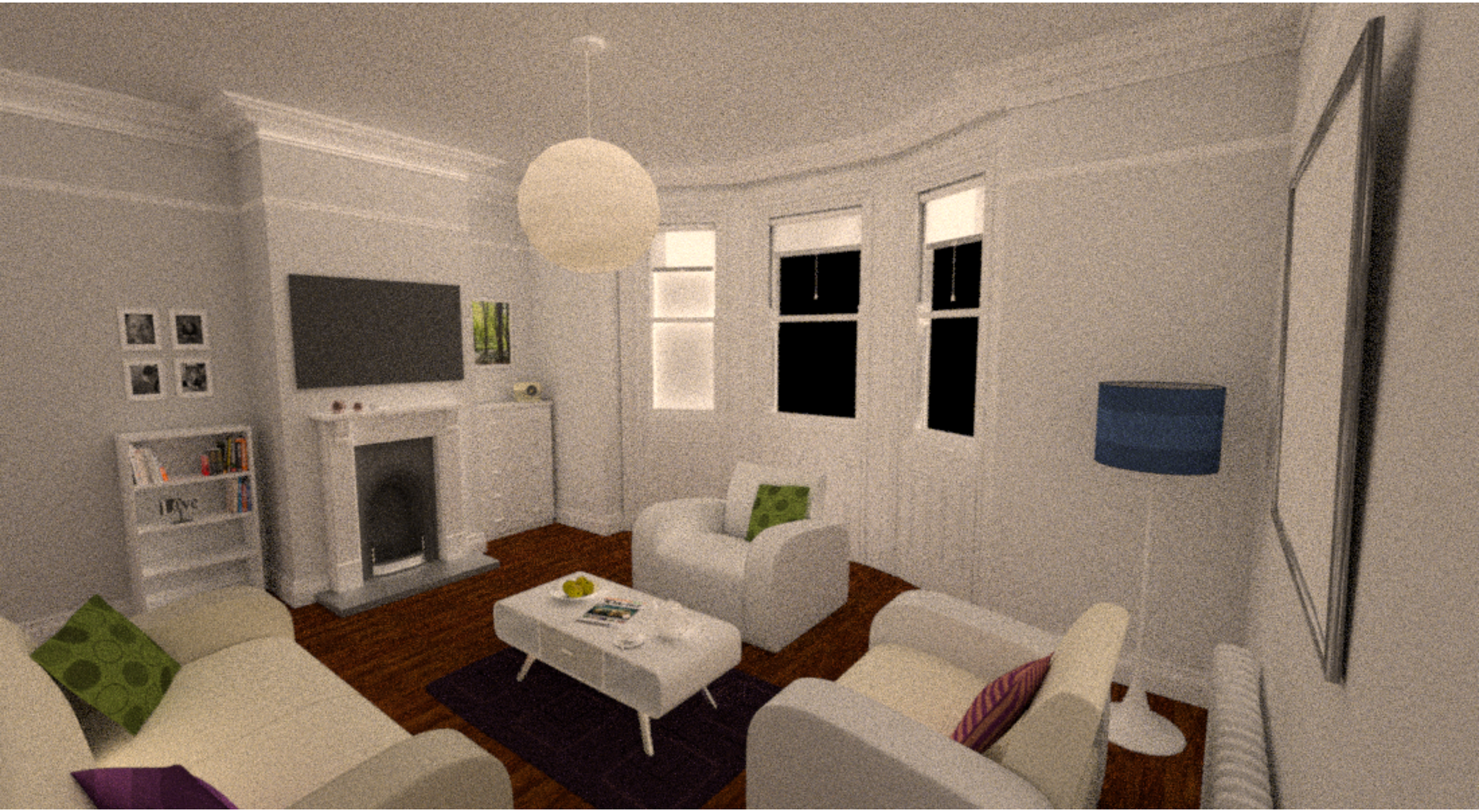
Three Iterations



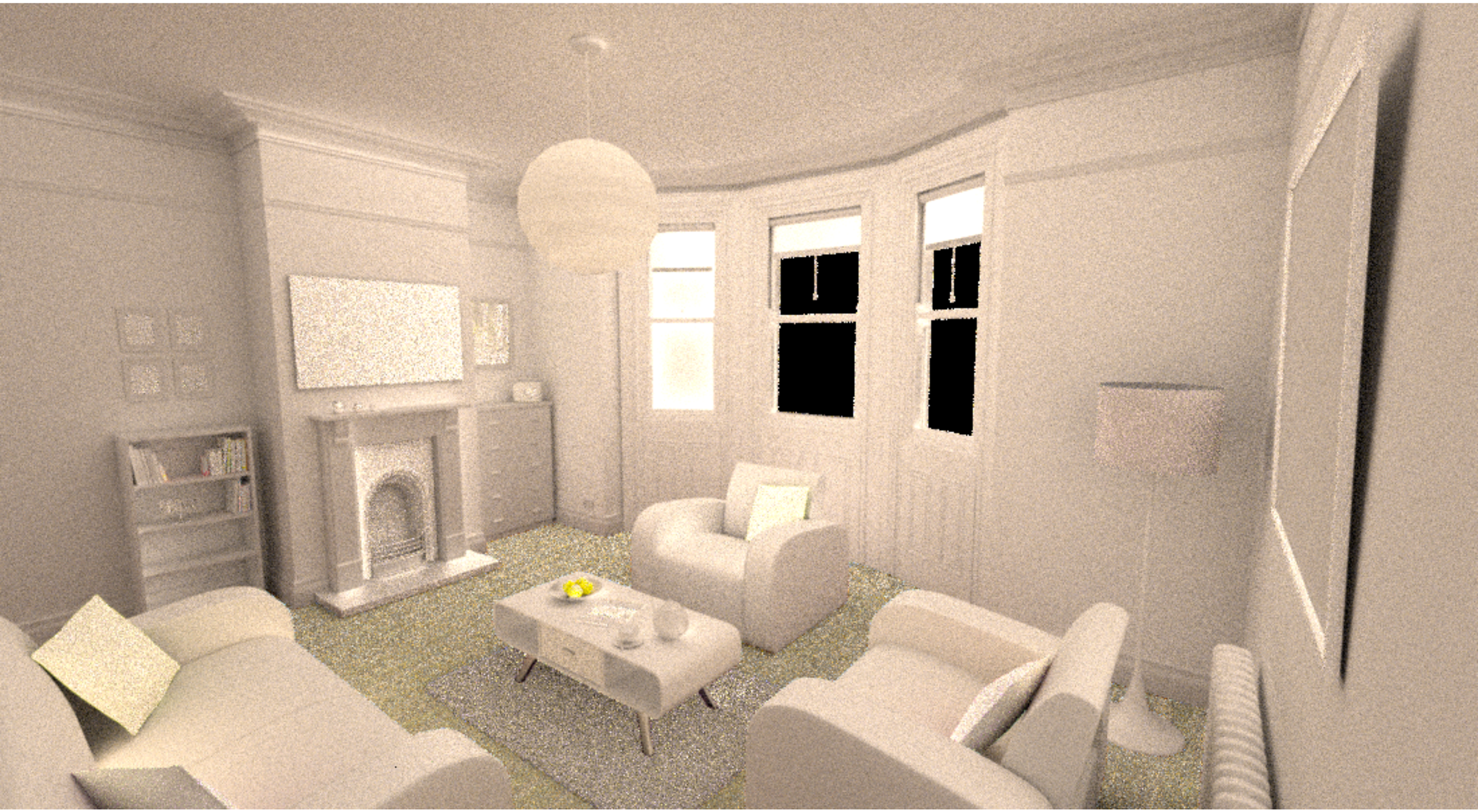
Filtered Direct Lighting



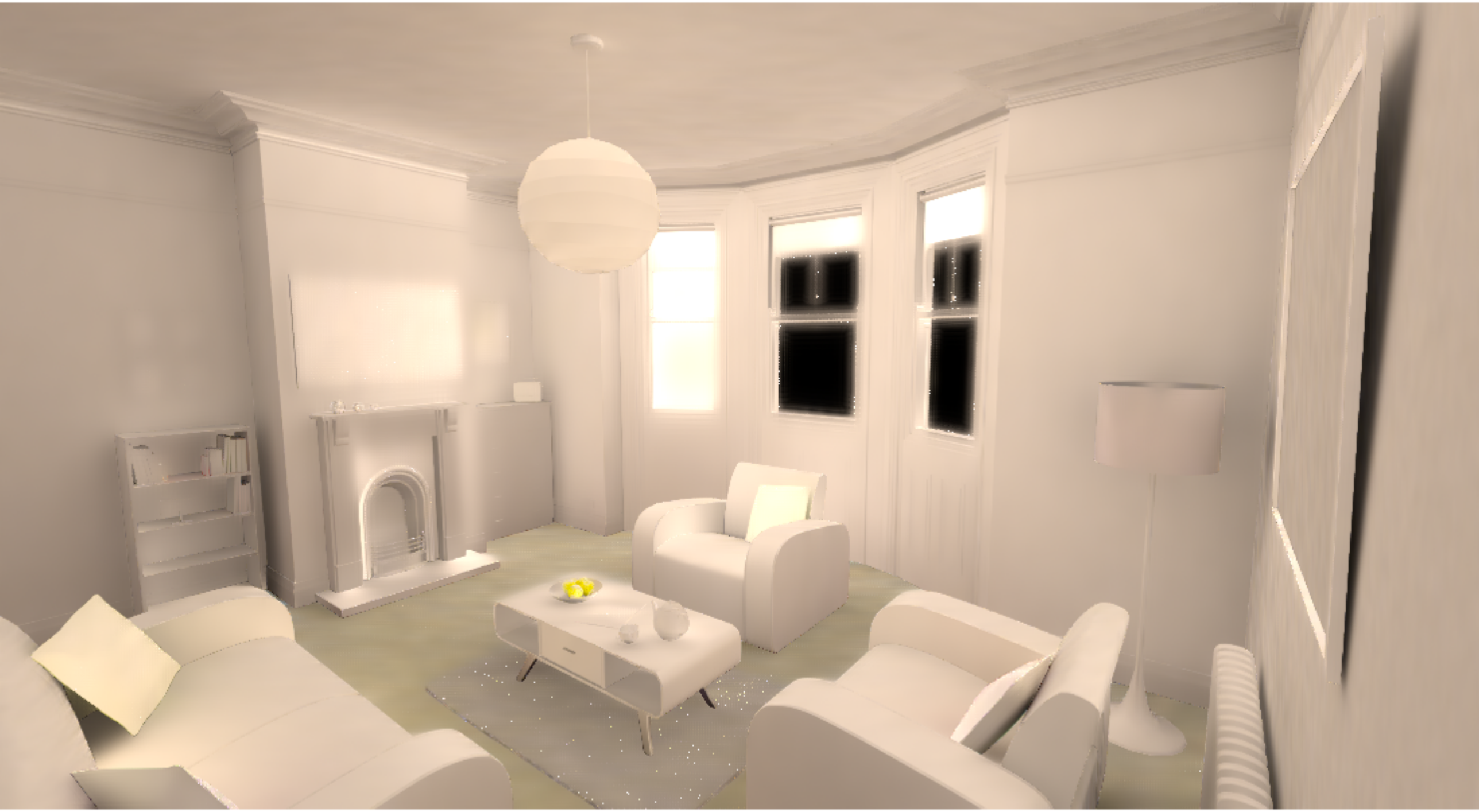
Indirect Lighting (64 samples)



Indirect Lighting / Albedo

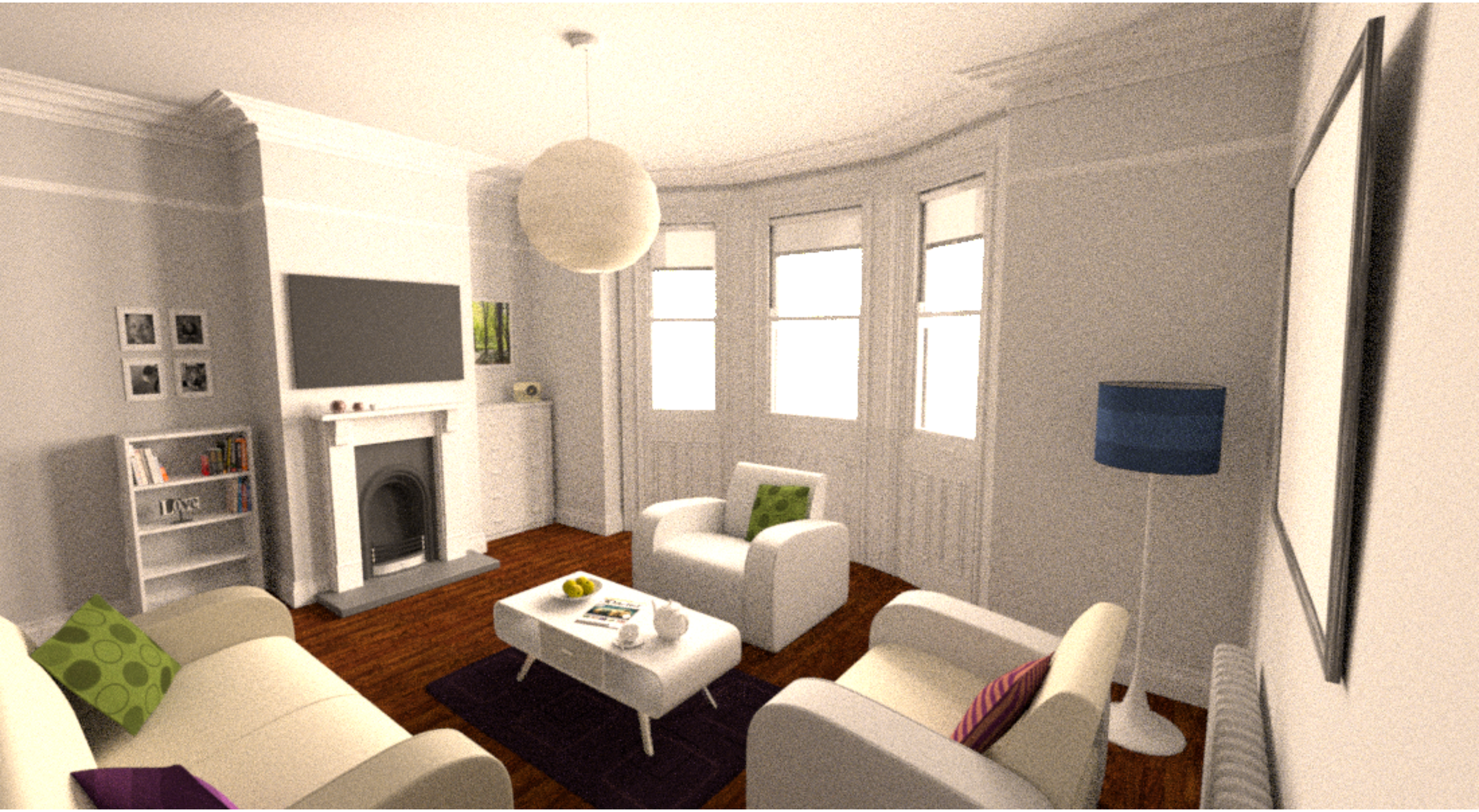


Filtered Indirect Lighting / Albedo



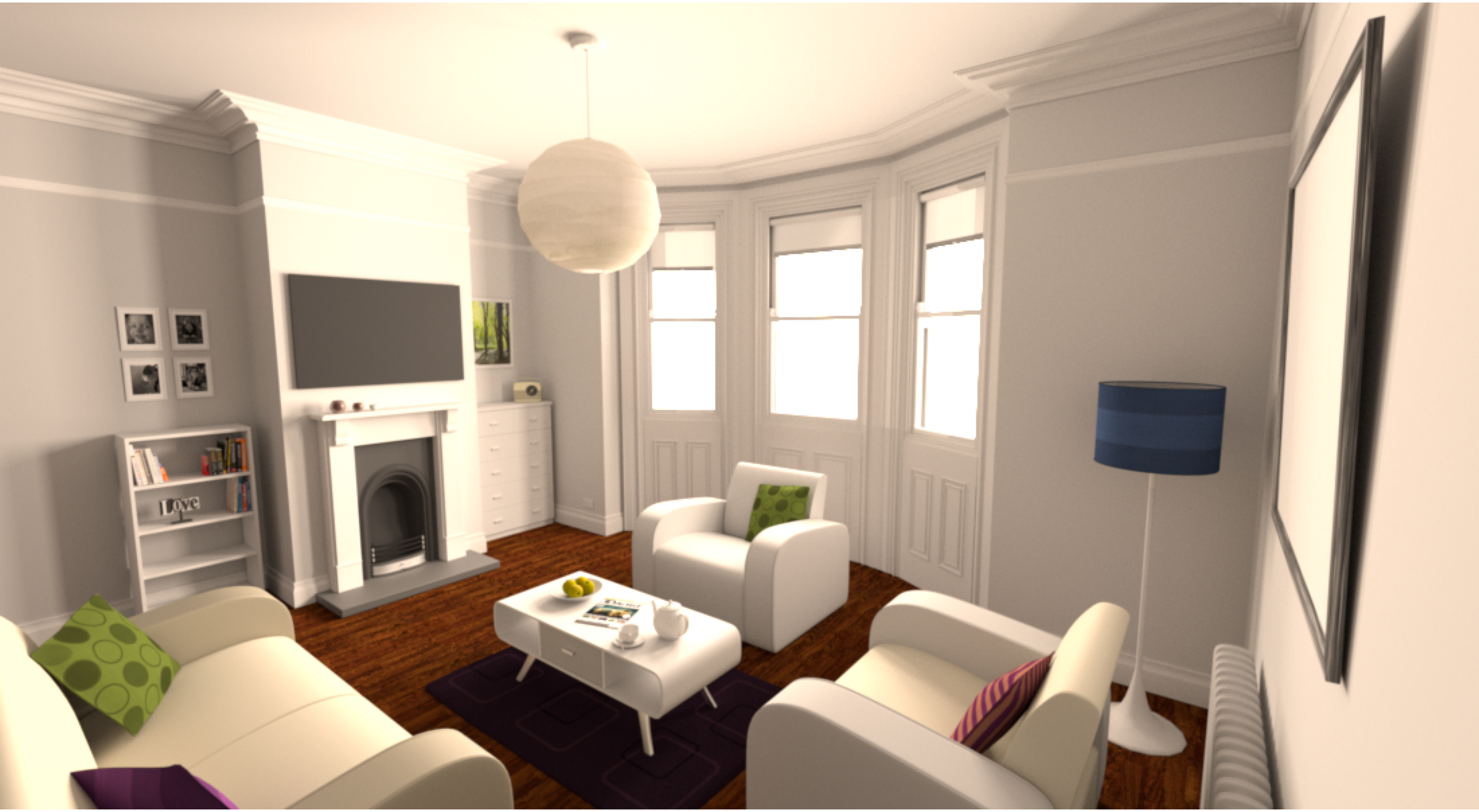
Filtered Indirect Lighting

64 samples per pixel, MSE = 0.005212



64 denoised samples, MSE = 0.000327

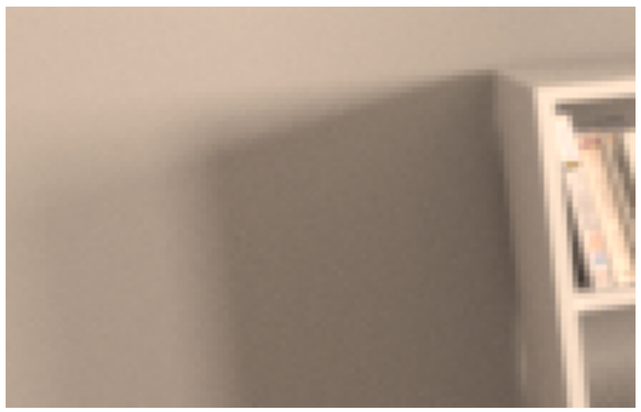
Reference (4096 samples)



Close-ups

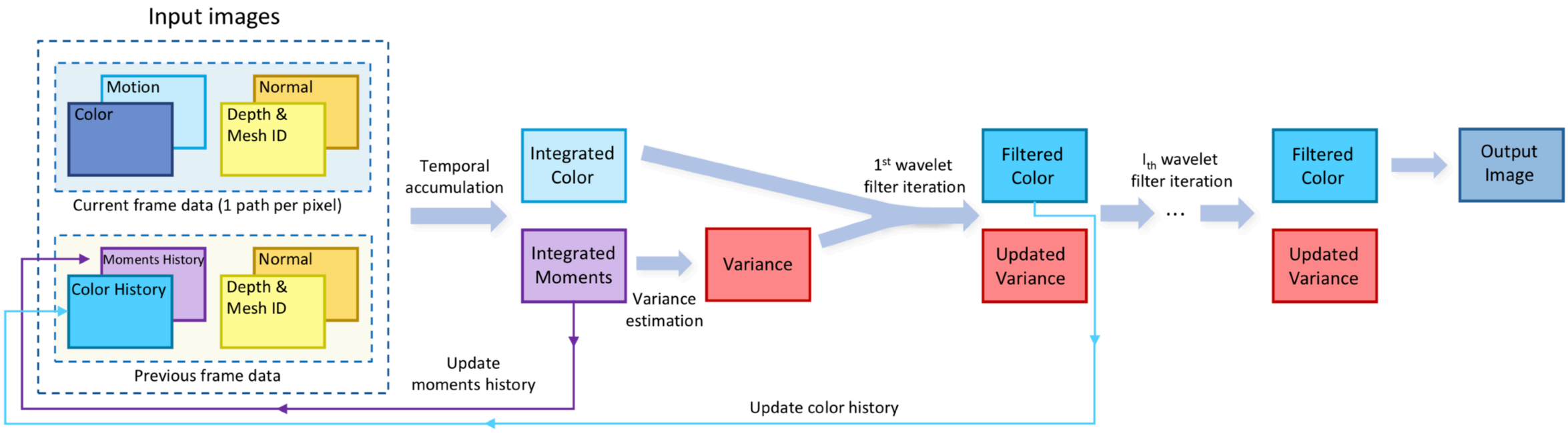
64 samples

4096 samples



64 samples, denoised

Temporal Filtering



Spatiotemporal Variance-Guided Filtering [Schied et al. 2017]

Summary

Estimate variance, then blur it

- **Using which filter? Blur how much?**

Choose additional features for joint filter

- **Which ones? Measure distance how? How much weight does each one get?**

Blur illumination using joint filter

- **Which filter? Blur how much?**

Multiply filtered illumination by albedo

Learning Filter Parameters

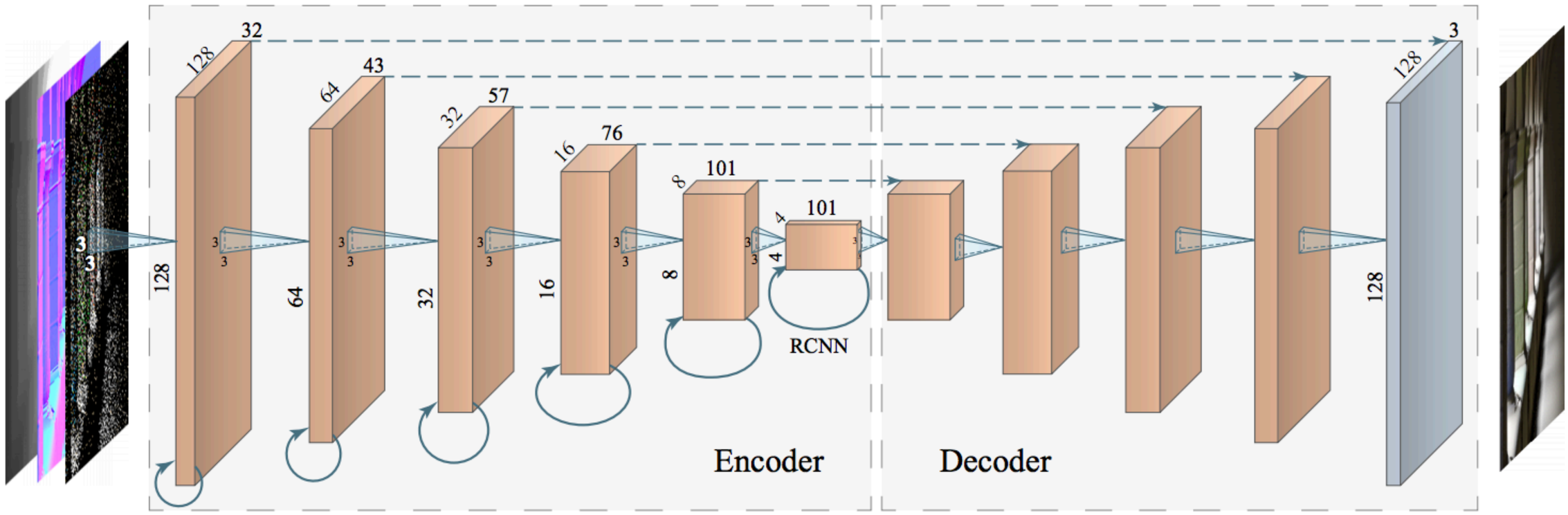
Our result with a cross-bilateral filter (4 spp)

[Kalantari et al. 2015]

3-layer fully connected MLP

Computes per-pixel filter parameters

Deep Denoising (NVIDIA)

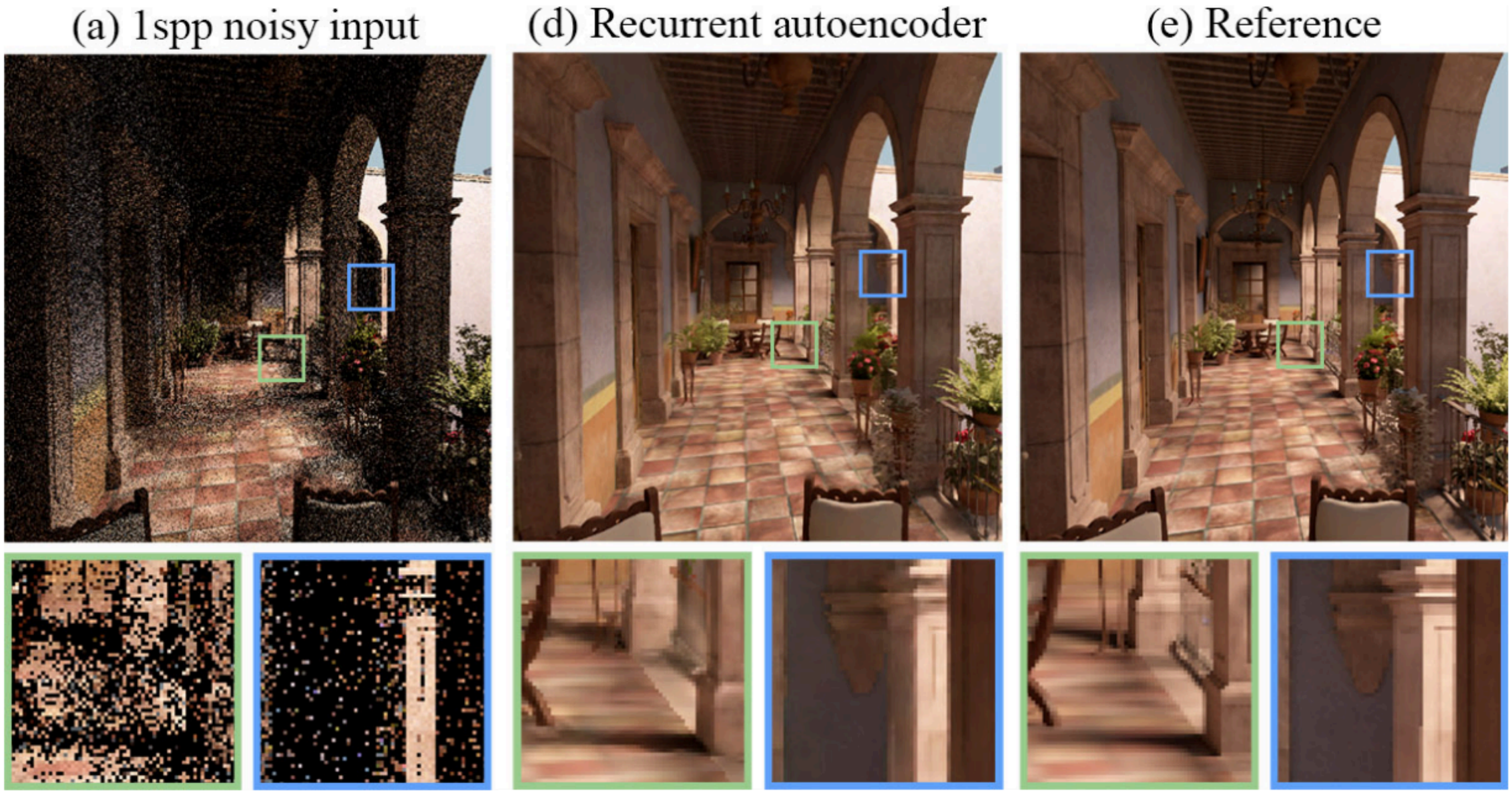


Features: illumination, normals, depth, roughness

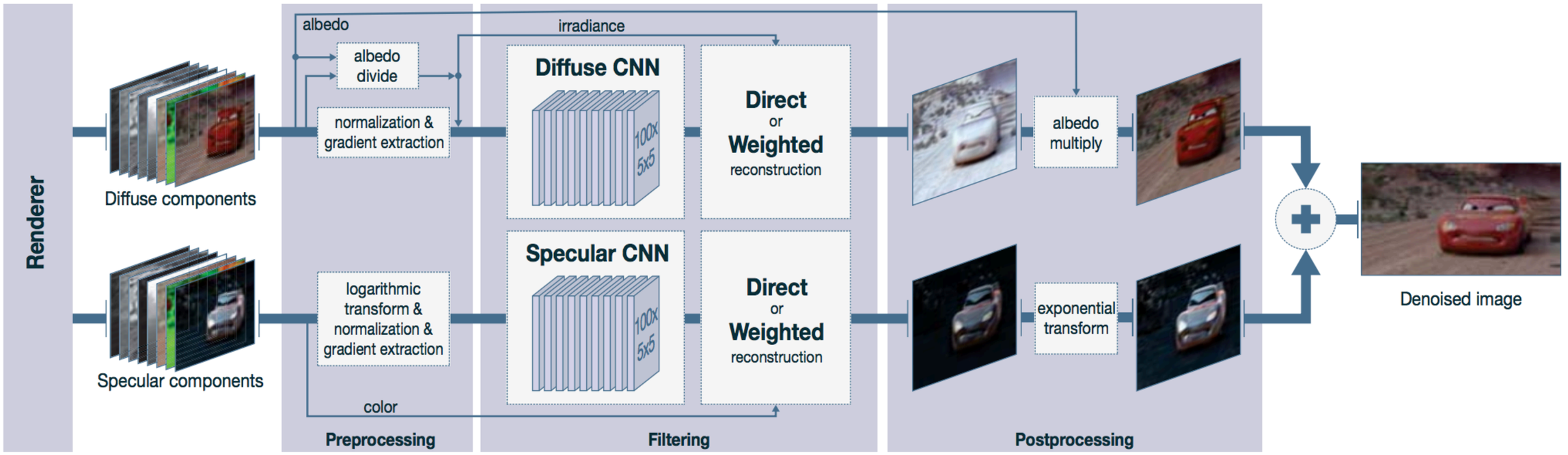
3.2M trainable parameters

Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder

Results



Deep Denoising (Pixar/Disney)



Deep CNN, 8 layers, 100 5x5 kernels

Features: illumination, normals, depth, and their variances

Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings

Kernel Prediction

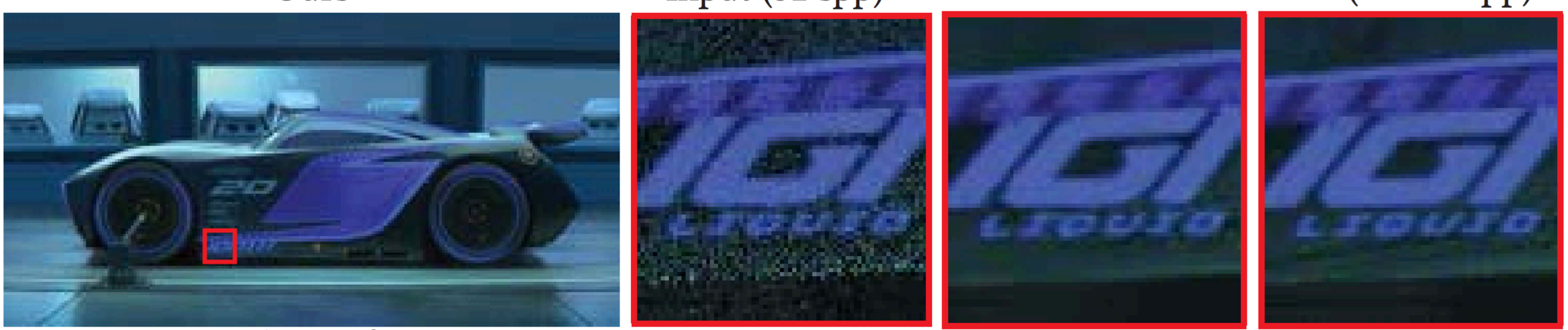
Network generates a stencil of 21x21 filter weights at each pixel

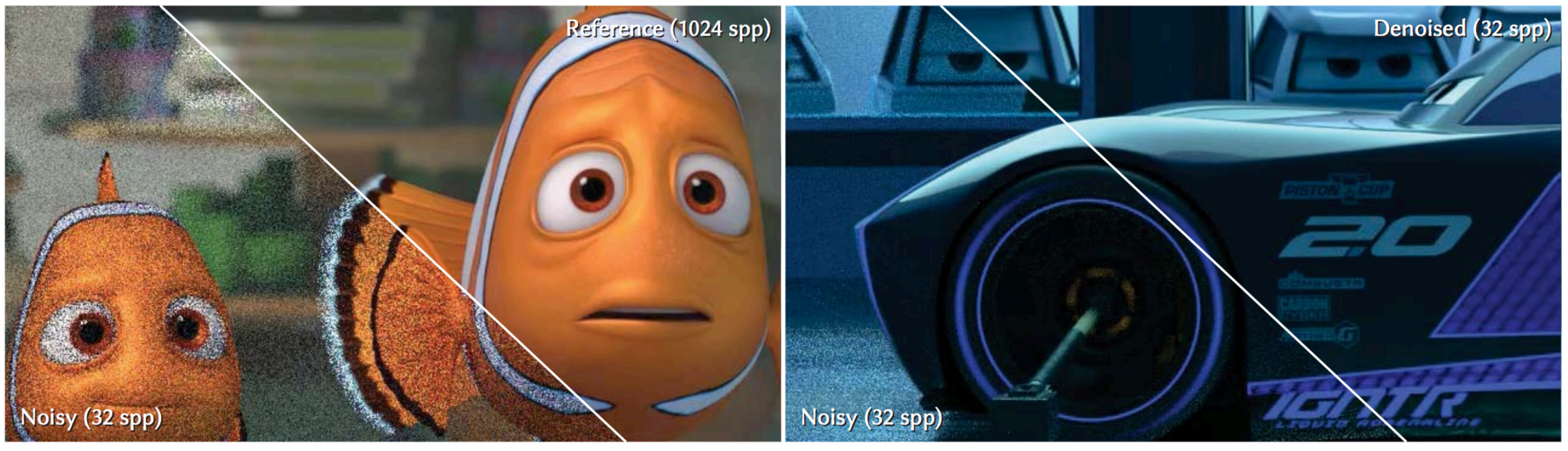
- Normalized to sum to one (~softmax)
- Weights are then applied to the noisy image

Advantages:

- Result always in convex hull of input
- Better scale invariance (HDR)
- 5-6x faster convergence than direct reconstruction

Results


Ours
Input (32 spp)
Ours
Ref. (1K-4K spp)
relative ℓ_2
19.21e-3
1.16e-3
$1 - \text{SSIM}$
0.354
0.032

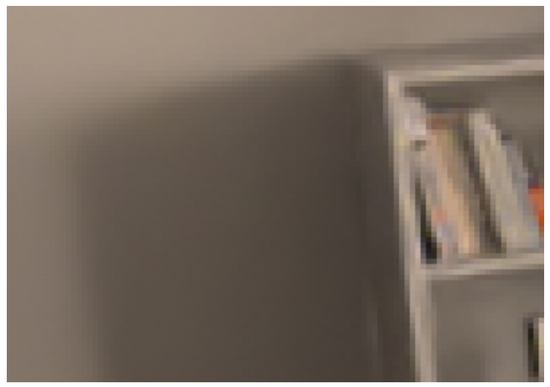

Reference (1024 spp)
Noisy (32 spp)
Denoised (32 spp)
Noisy (32 spp)
TRAINING
TEST

64 samples per pixel, MSE 0.03366

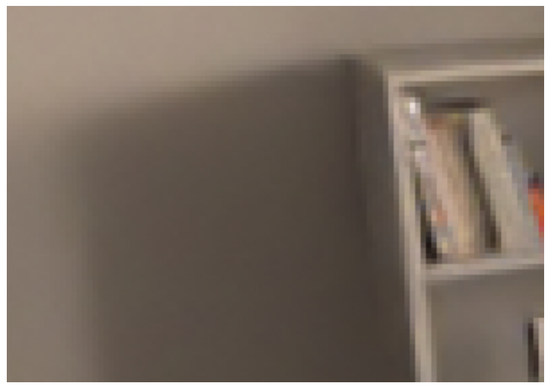
64 spp, CNN-denoised: MSE 0.005048

Reference (16,384 samples)

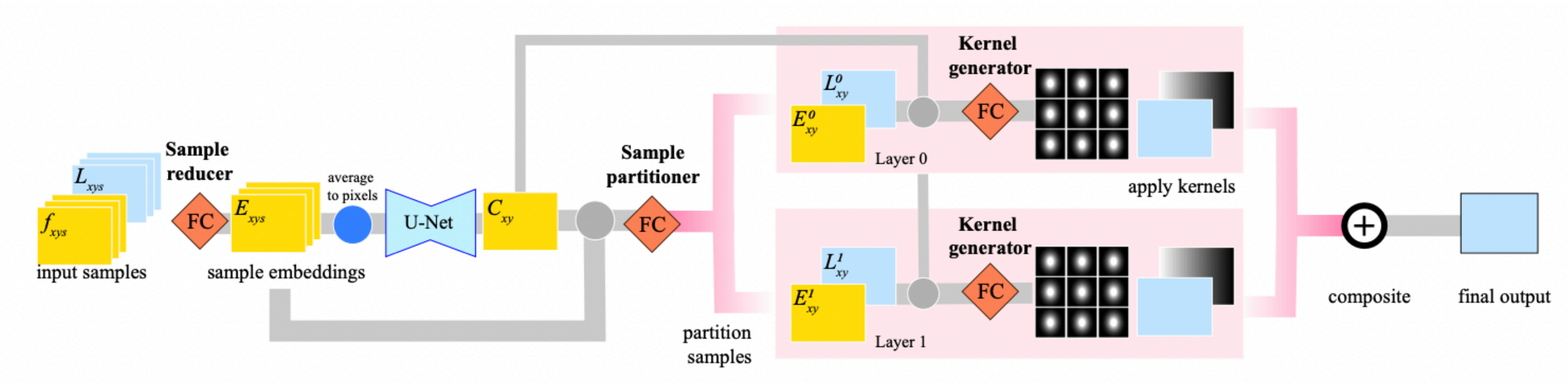
Zoom-ins: CNN denoising

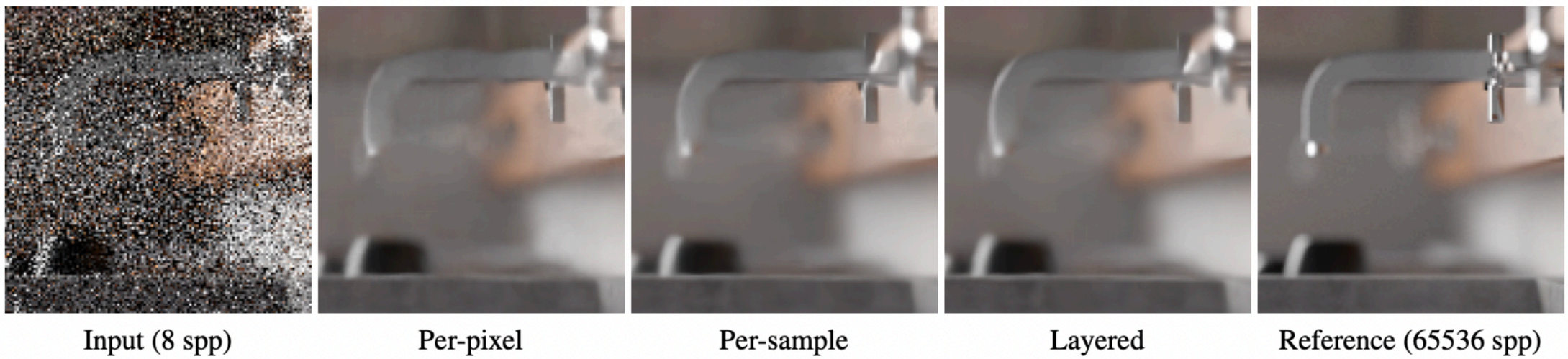
64 samples	64 samples, denoised	16,384 samples
		
		
		

Zoom-ins: CNN denoising

16 samples	16 samples, denoised	16,384 samples
		
		
		

Denoising Samples, Not Pixels





Input (8 spp) | Per-pixel | Per-sample | Layered | Reference (65536 spp)

Munkberg and Hasselgren. 2020.
Neural Denoising with Layer Embeddings.

Acknowledgement

Benedikt Bitterli