Real-Time Ray Tracing and Denoising

Acceleration structures for dynamic scenes

Denoising

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

Many-Light Sampling on the GPU

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

Battlefield V (EA/DICE)

Real-Time vs. Offline Rendering

Different constraints:

- **Offline: render until quality achieved**
- **Real-time: render until time runs out**

Implications:

- **Performance eval: average vs. max time**
- **Many fewer rays/image in real-time**

Implications: Acceleration Structures

Time to build BVH/kd-tree matters much more for real-time ray tracing

- **For real-time, can allow a few ms / frame: e.g. @10M tris, 60fps, need 600M tris / sec.**
- **pbrt BVH construction: ~2.5M tris / second**

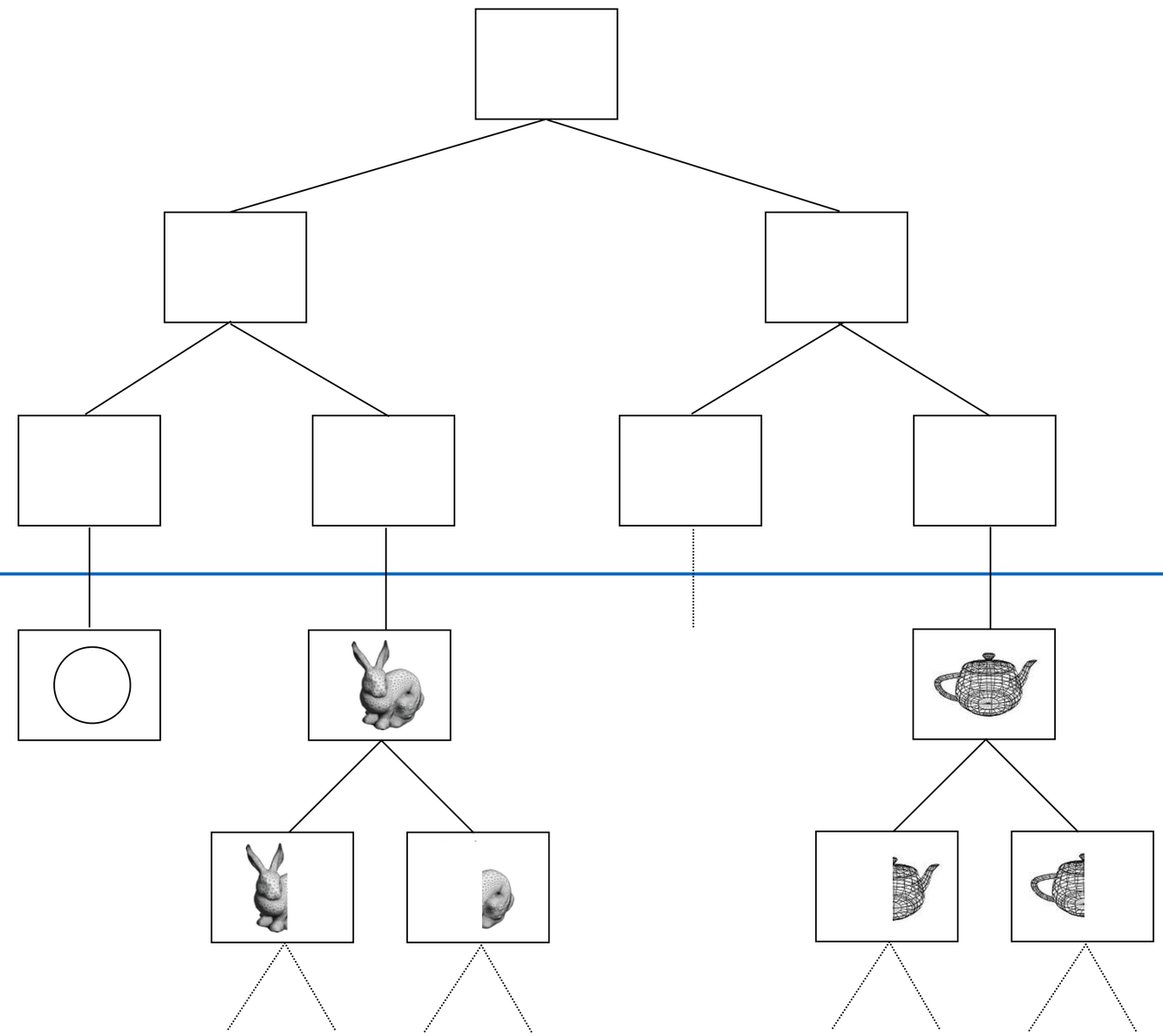
➡ Hierarchy construction efficiency really matters

➡ Hierarchy quality is (a little) less important

Battlefield V (EA/DICE)

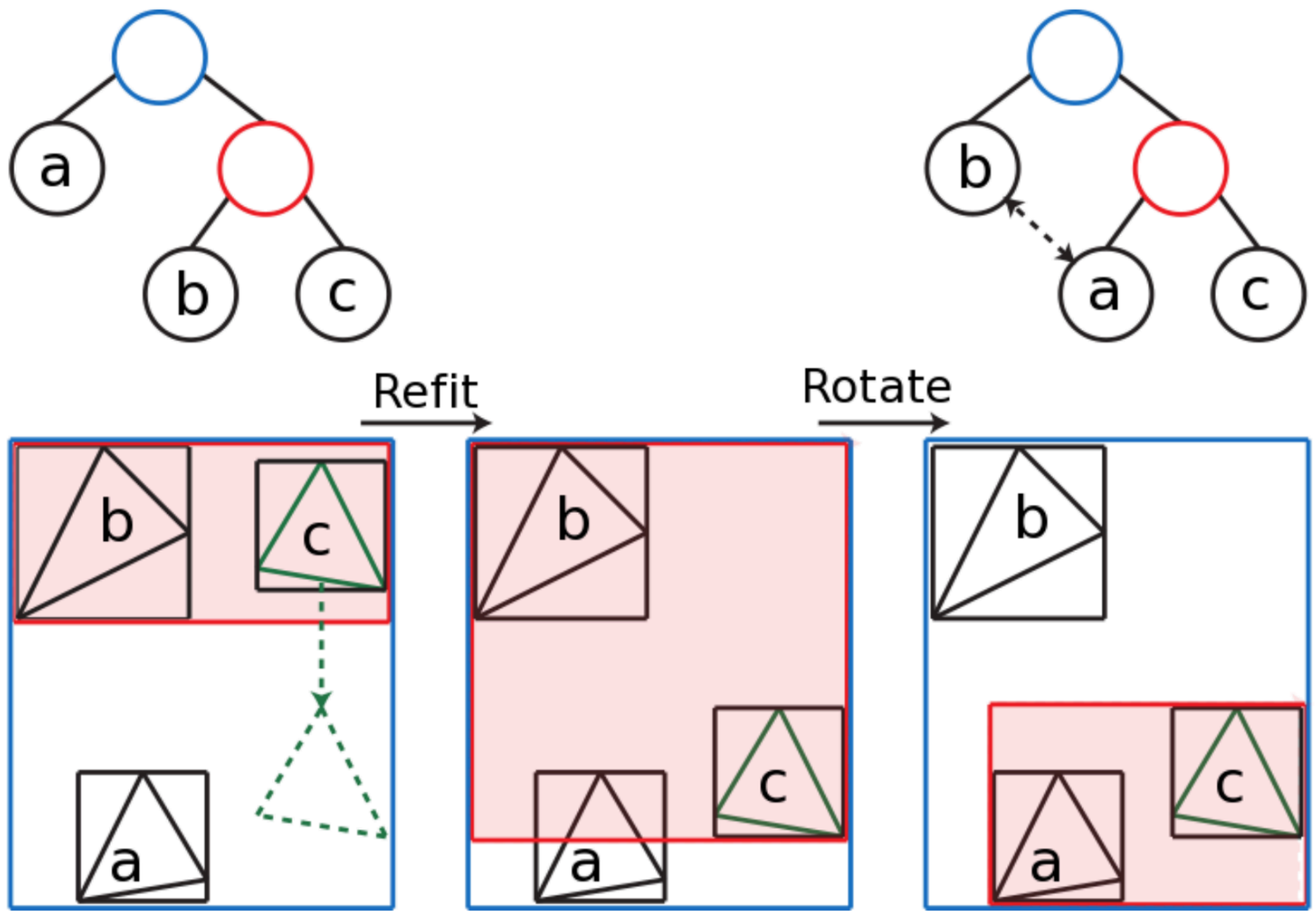
Two-level Acceleration Structures

Top-level acceleration structure



Bottom-level acceleration structures

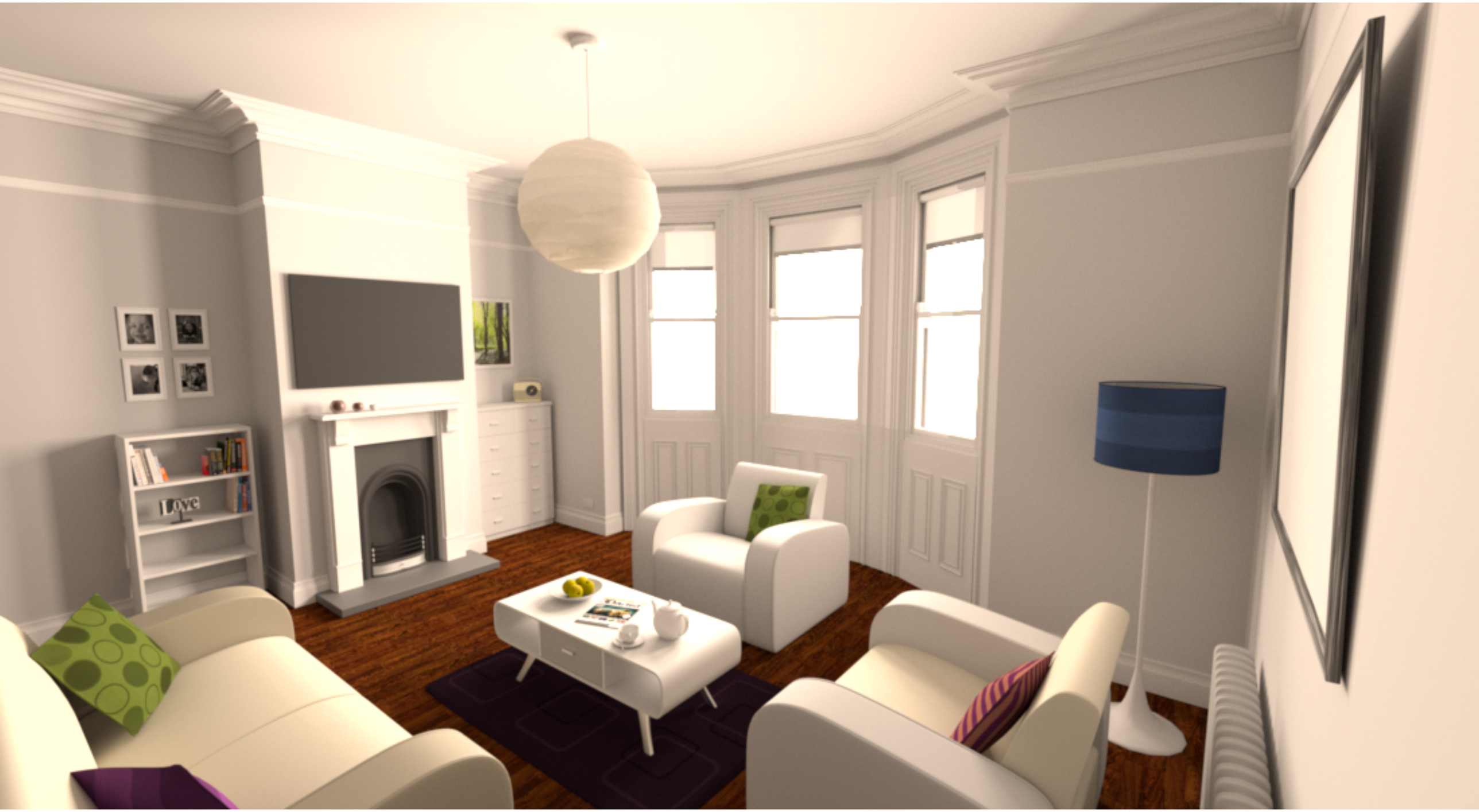
Refit BVH When Objects Move



[Kopta et al. 2012]

Denoising

White Room, 4096 pixel samples

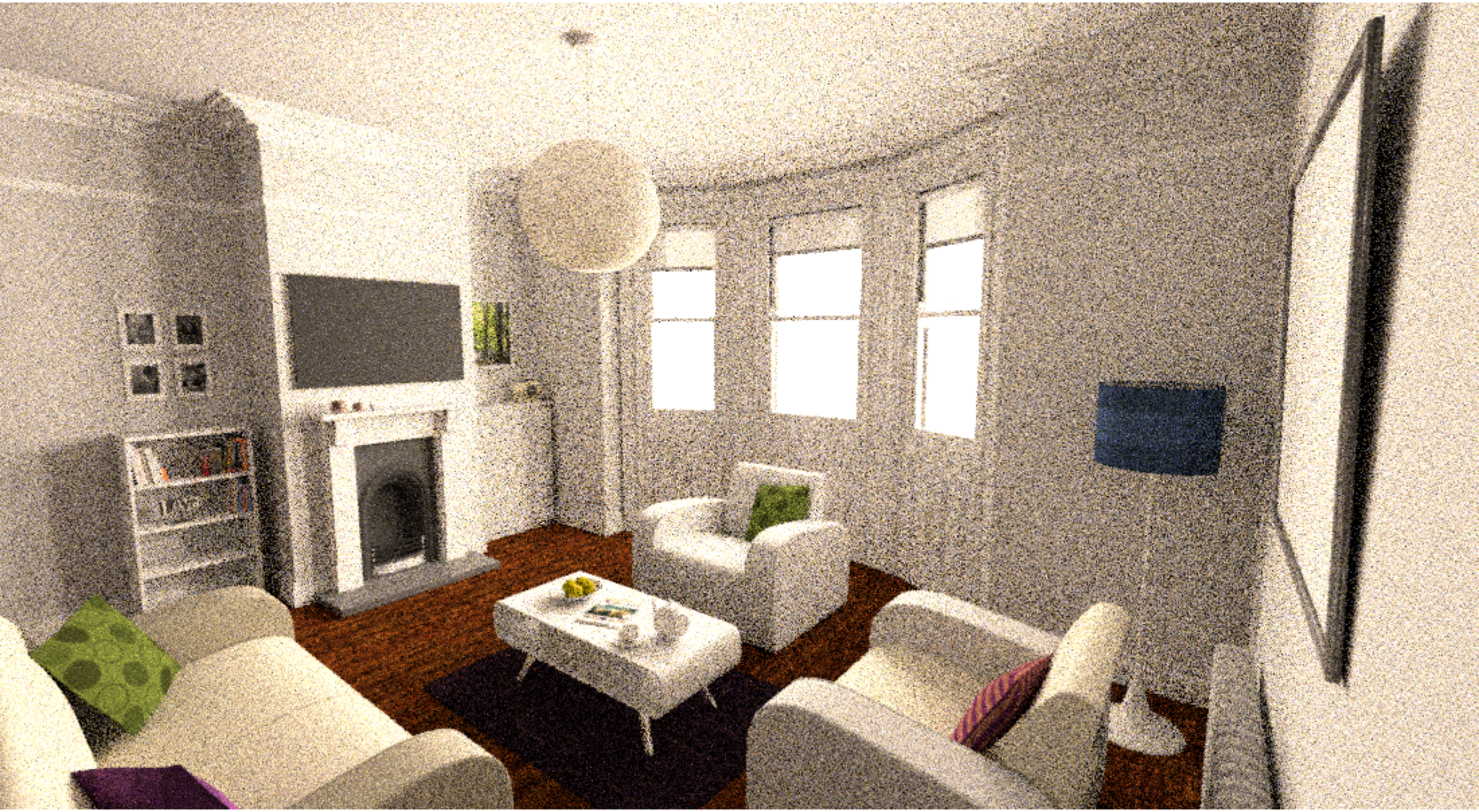


White Room, 1 pixel sample

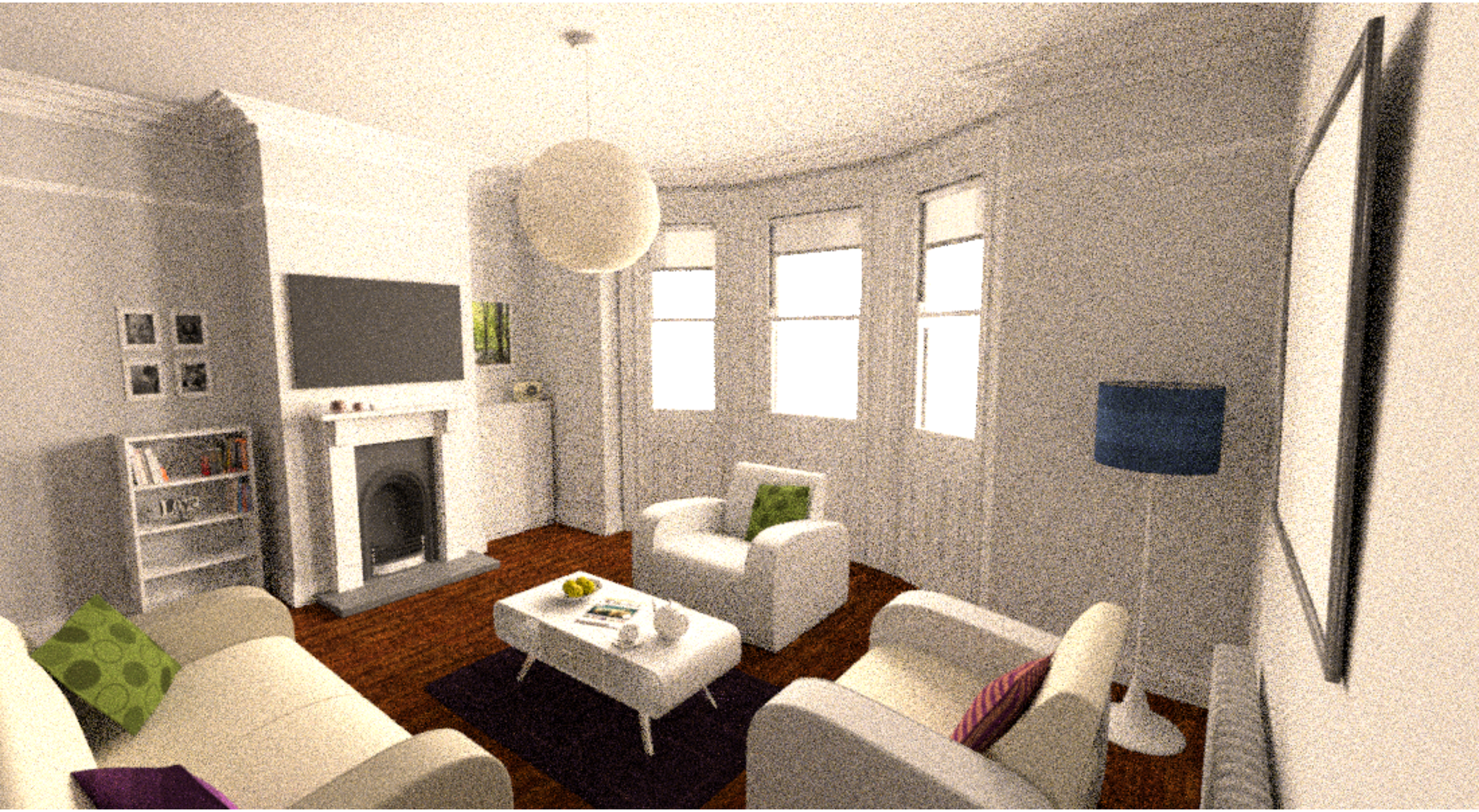
White Room, 2 pixel samples

White Room, 4 pixel samples

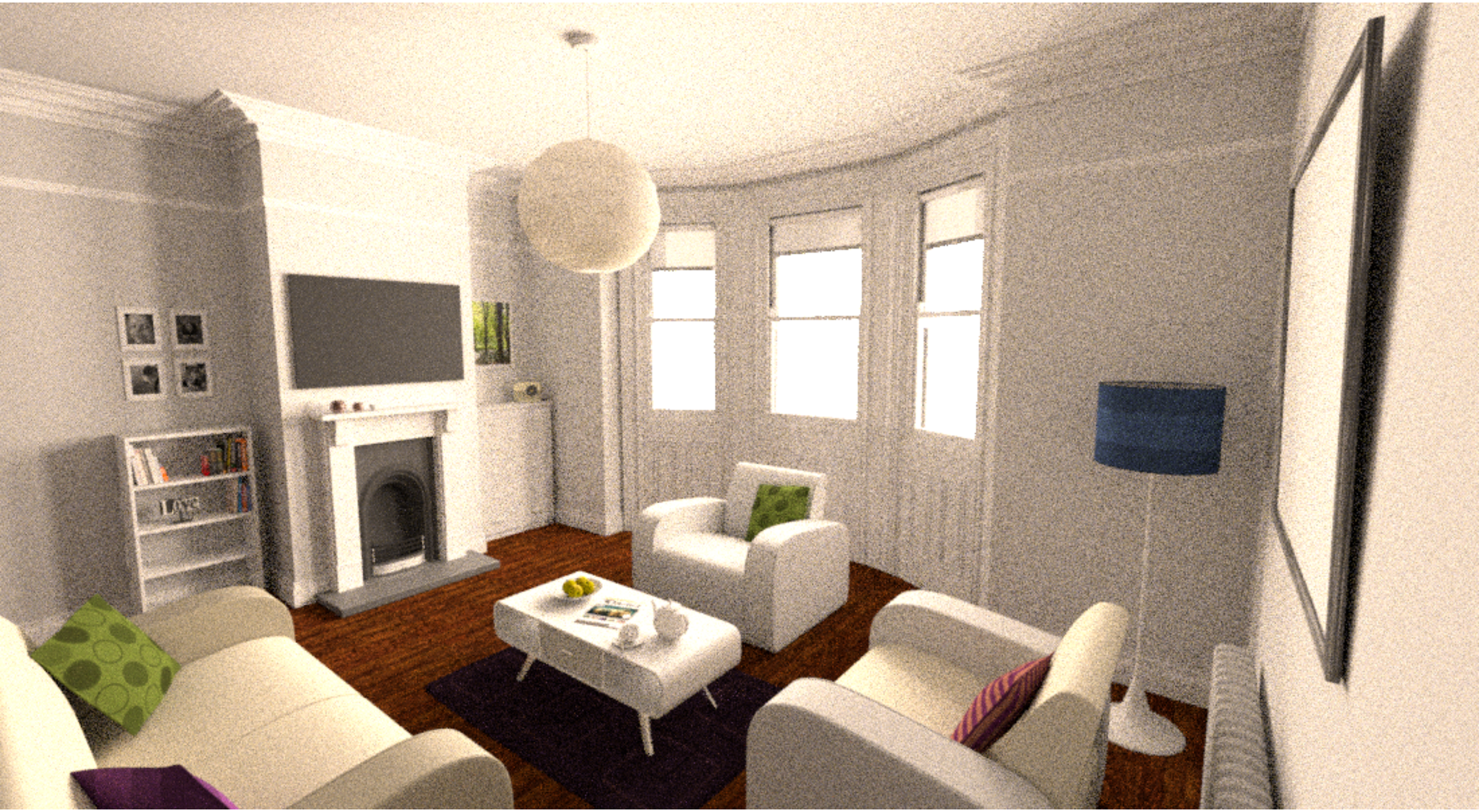
White Room, 8 pixel samples



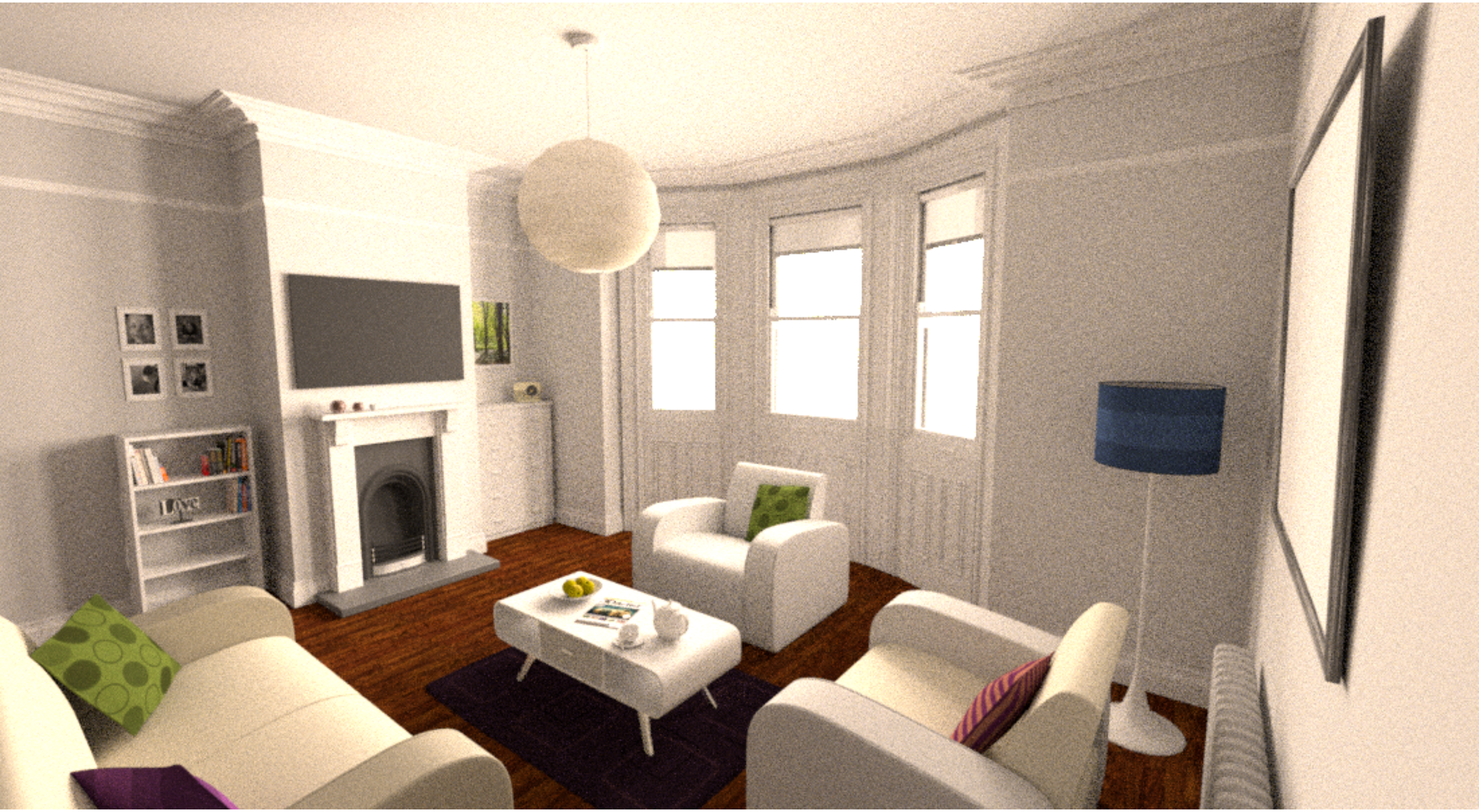
White Room, 16 pixel samples



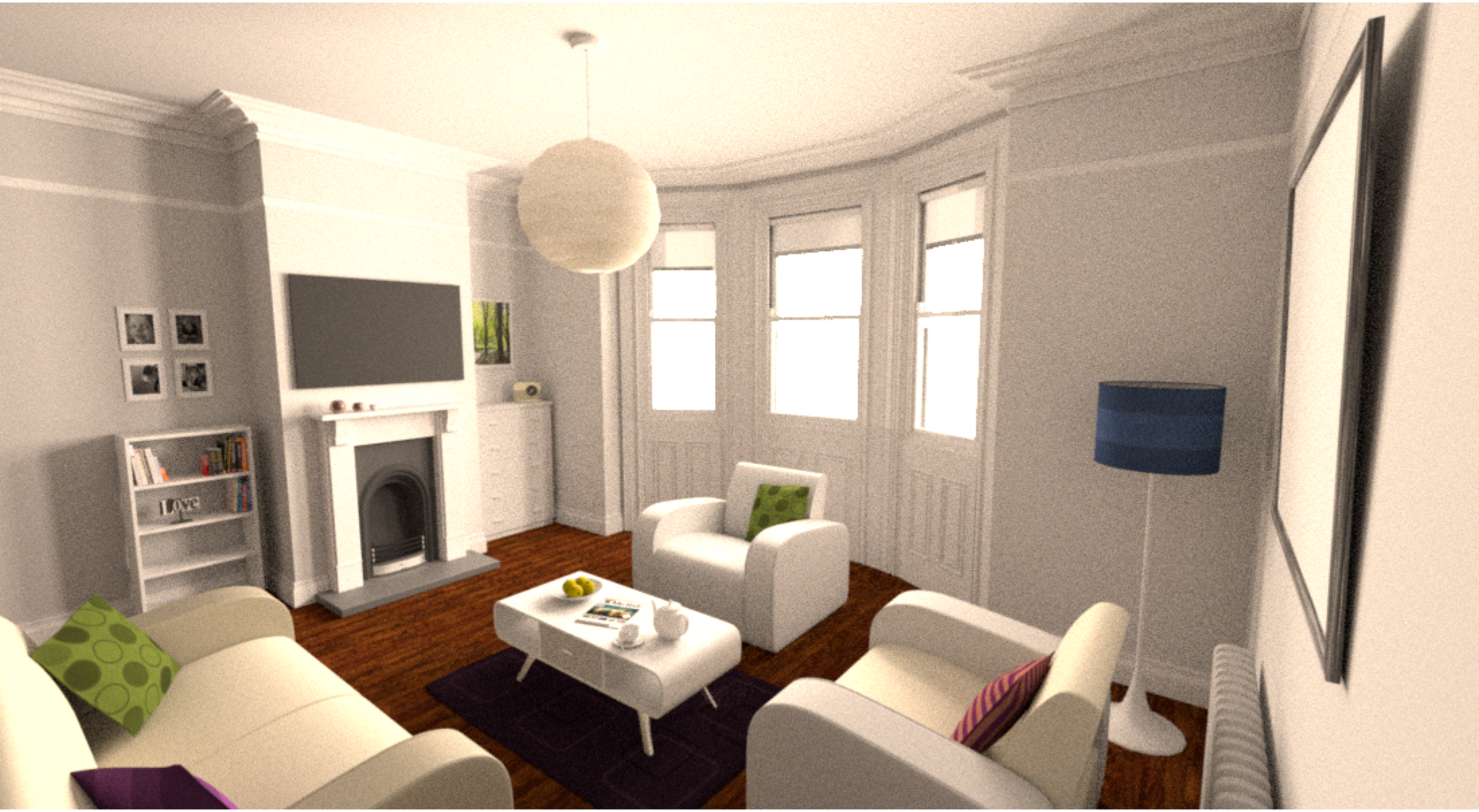
White Room, 32 pixel samples



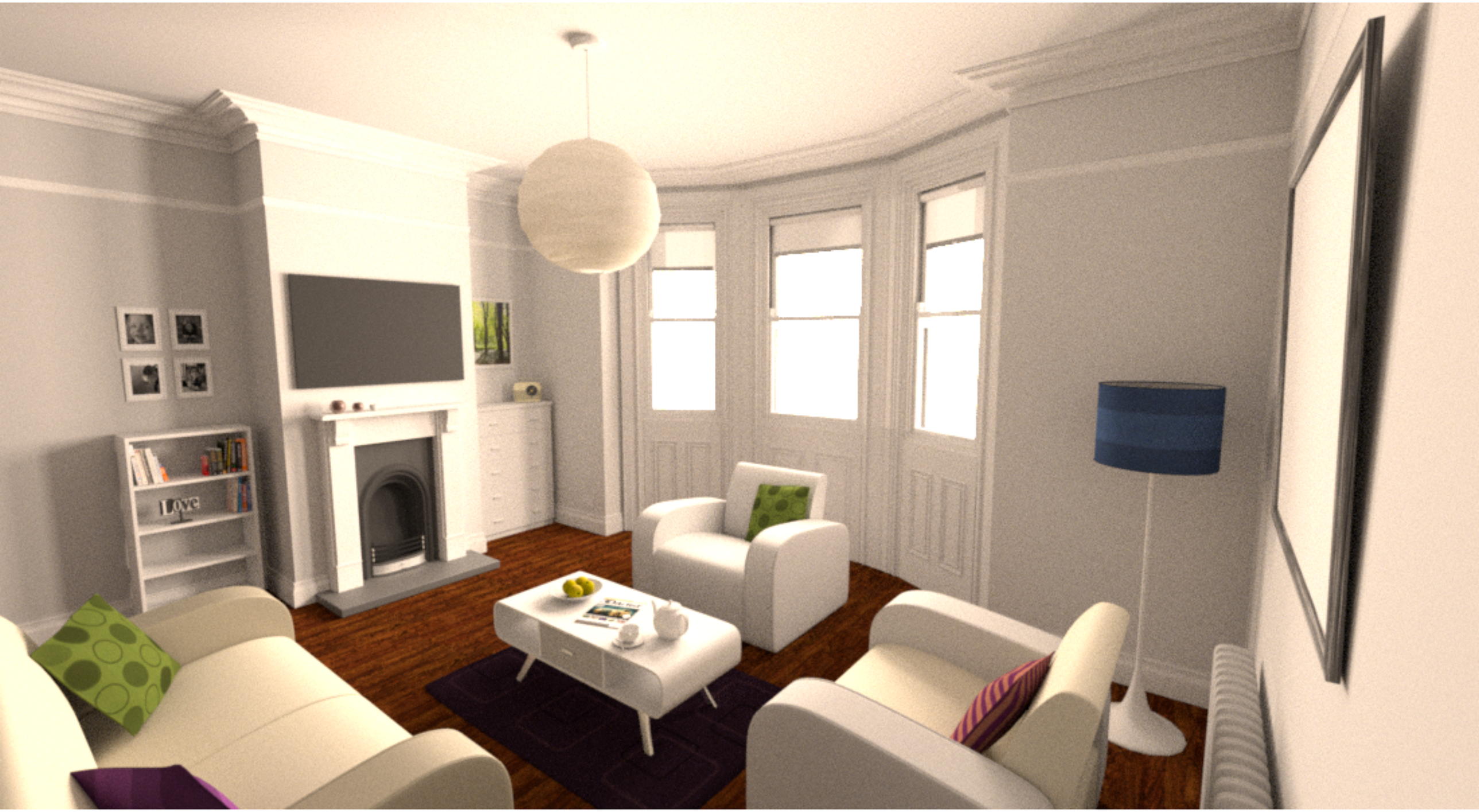
White Room, 64 pixel samples



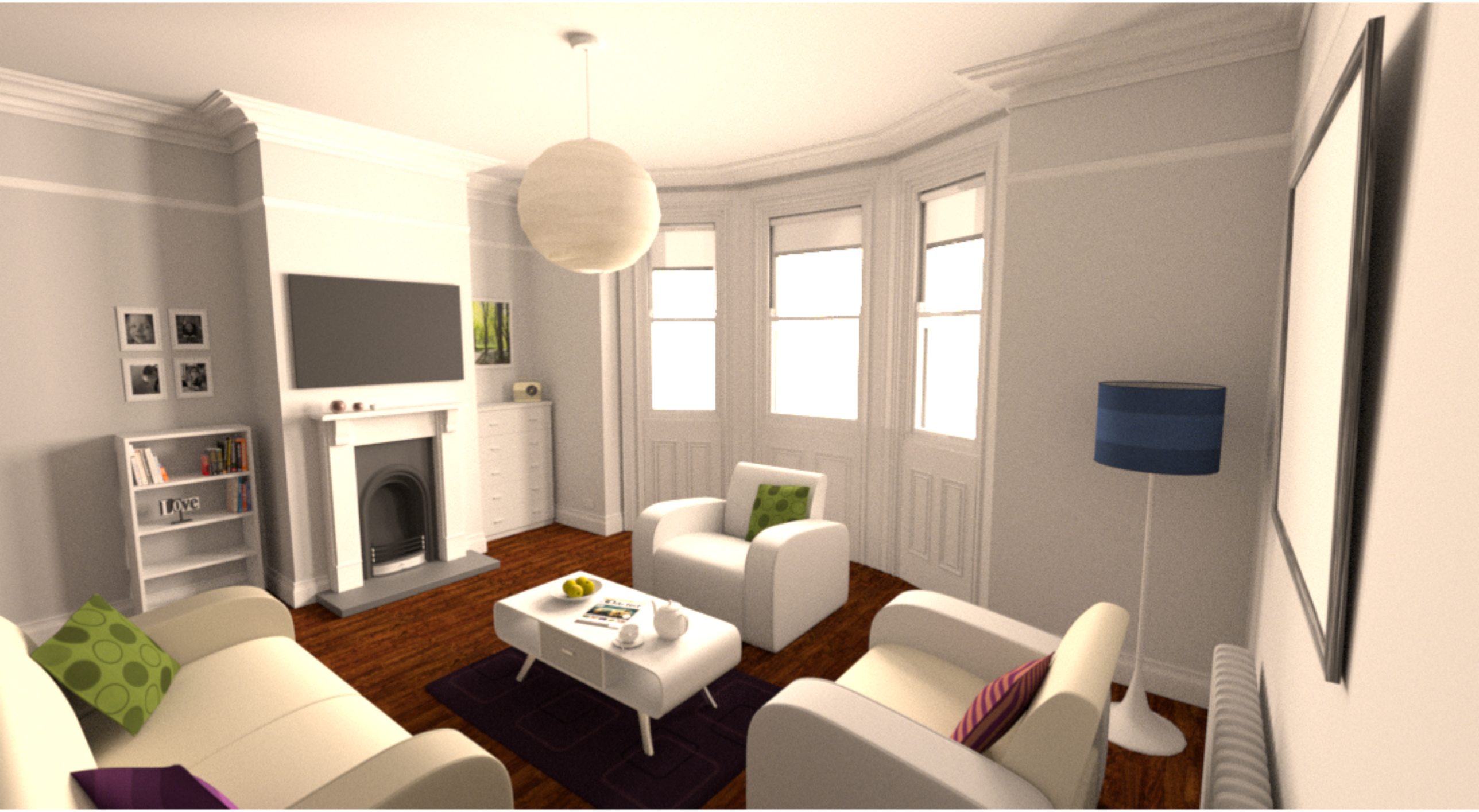
White Room, 128 pixel samples



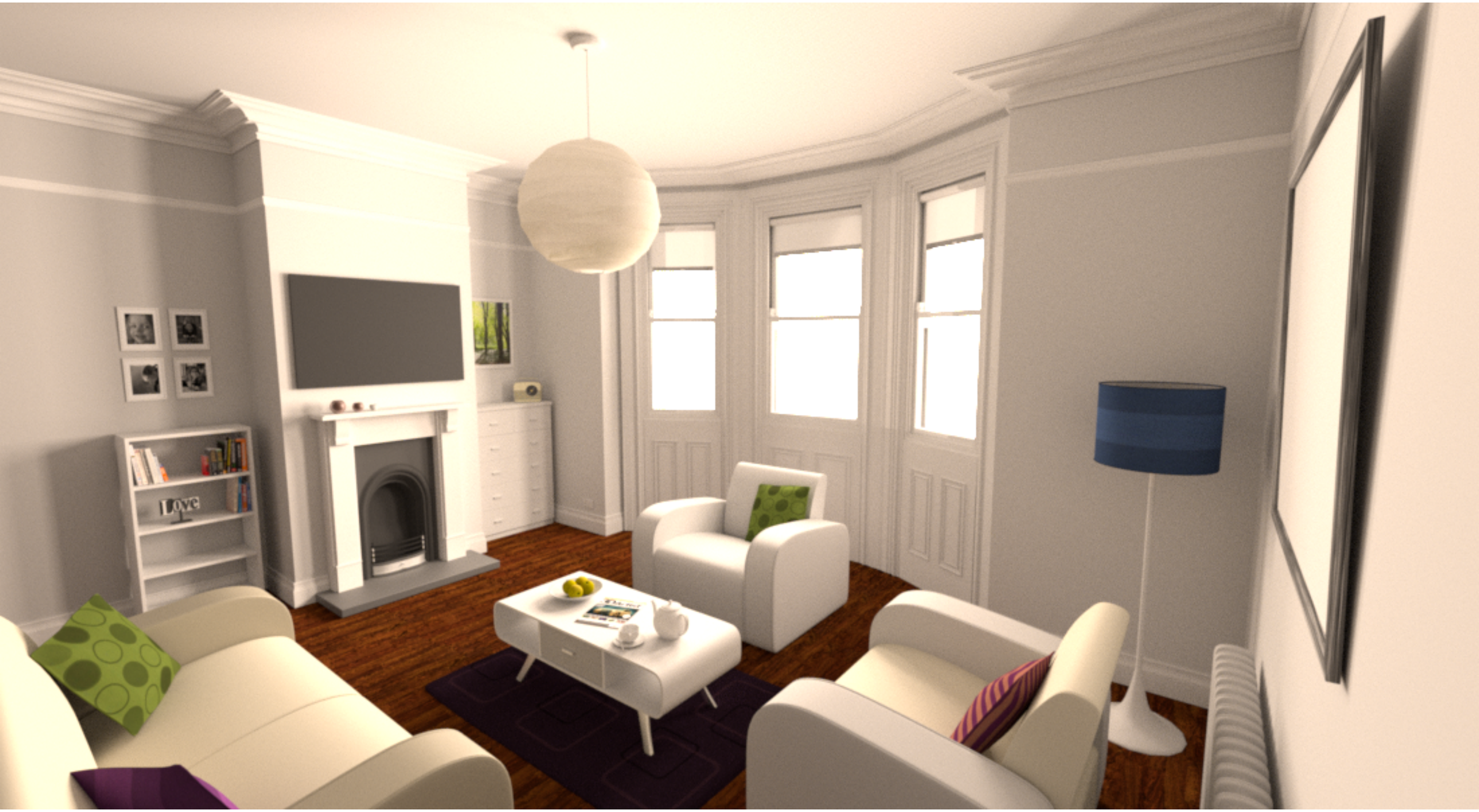
White Room, 256 pixel samples



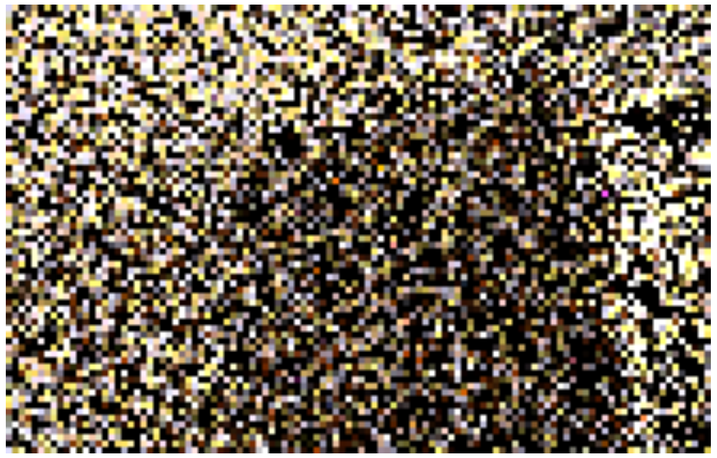
White Room, 512 pixel samples



White Room, 1024 pixel samples



Zoom-in by Bookshelf

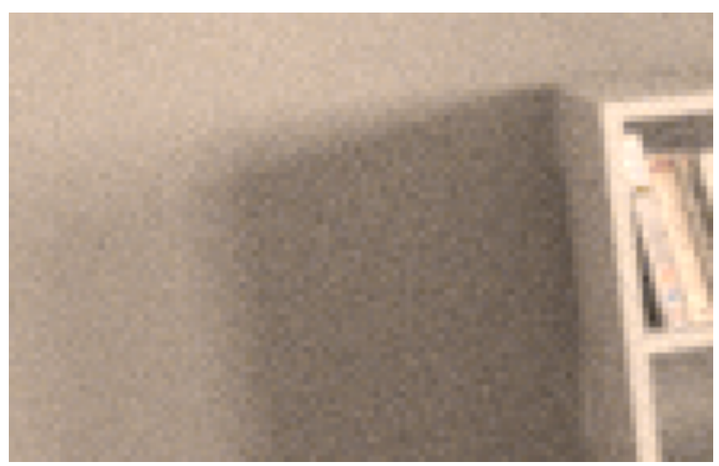


1 sample

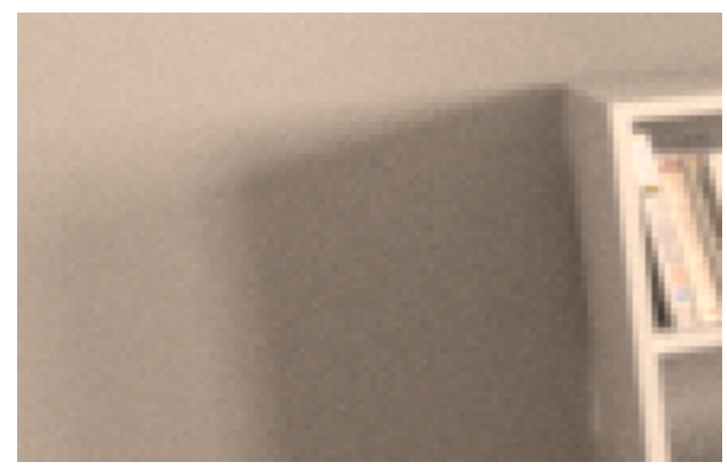
4 samples

16 samples

64 samples



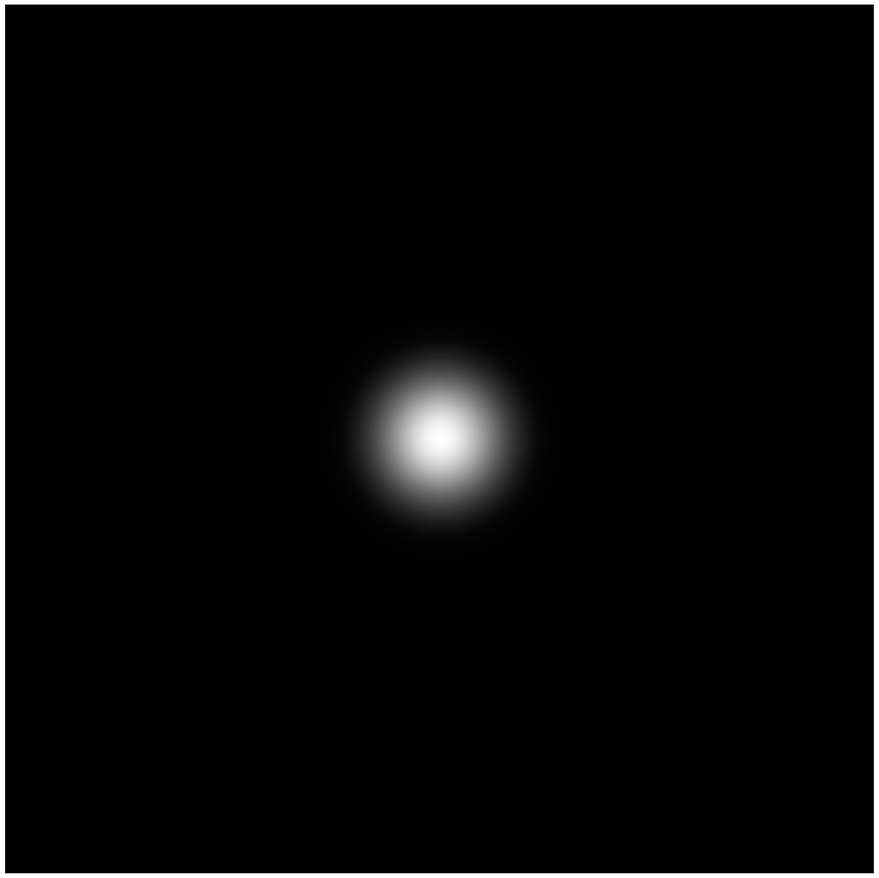
256 samples



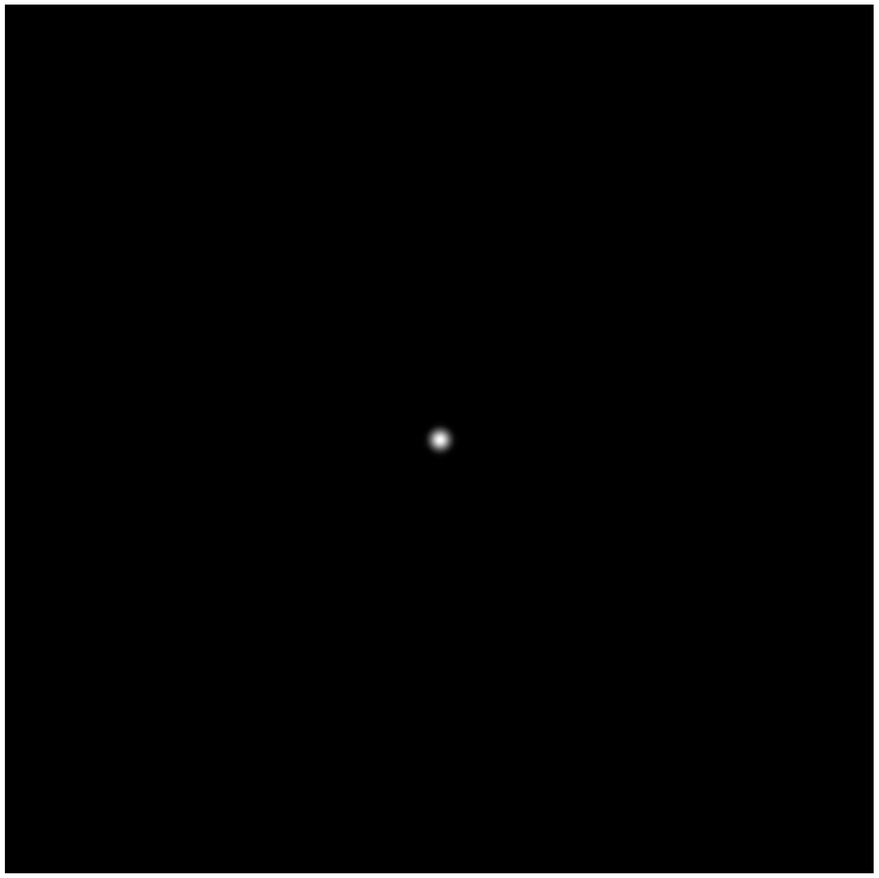
1024 samples

Gaussian Filter

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

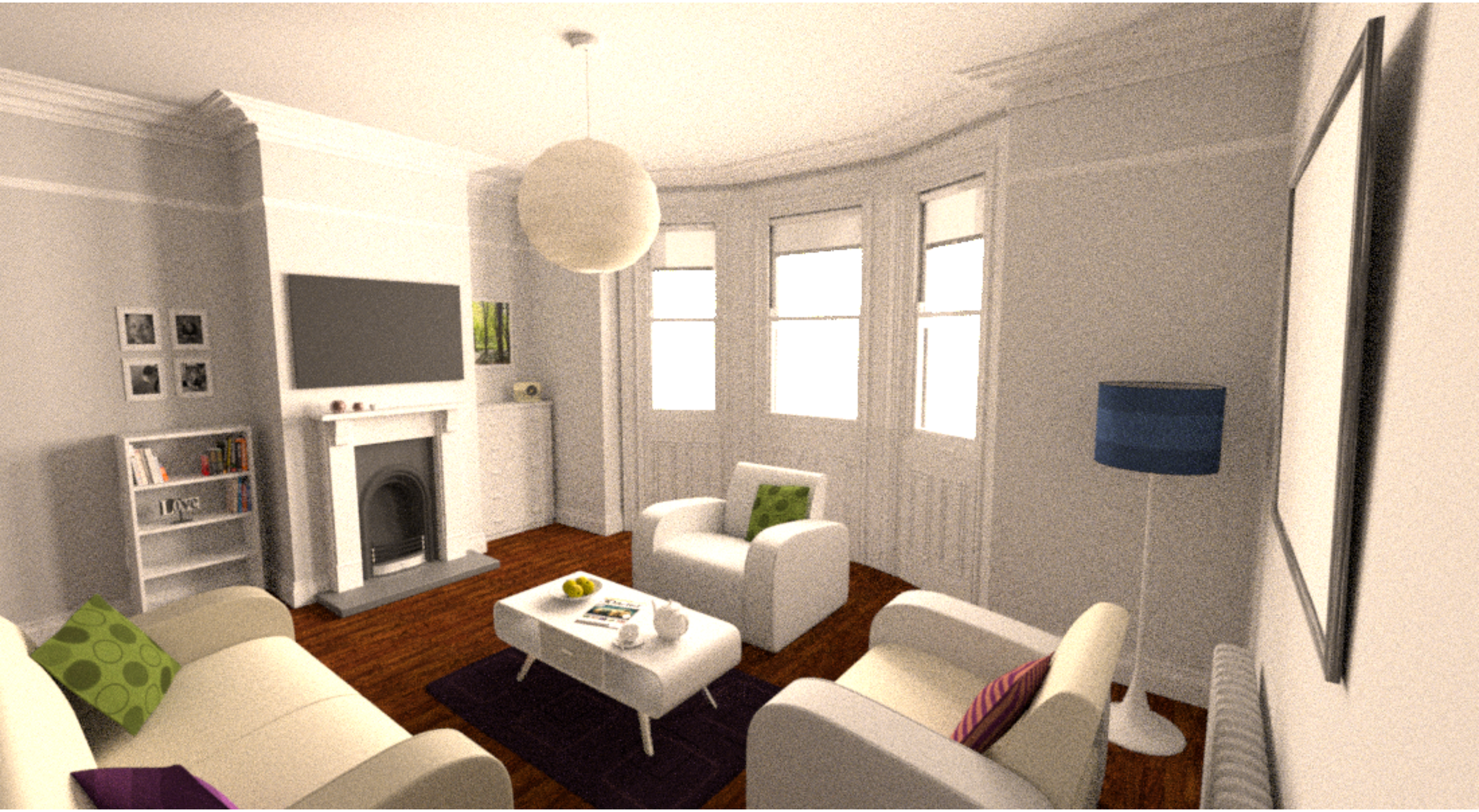


Spatial Domain

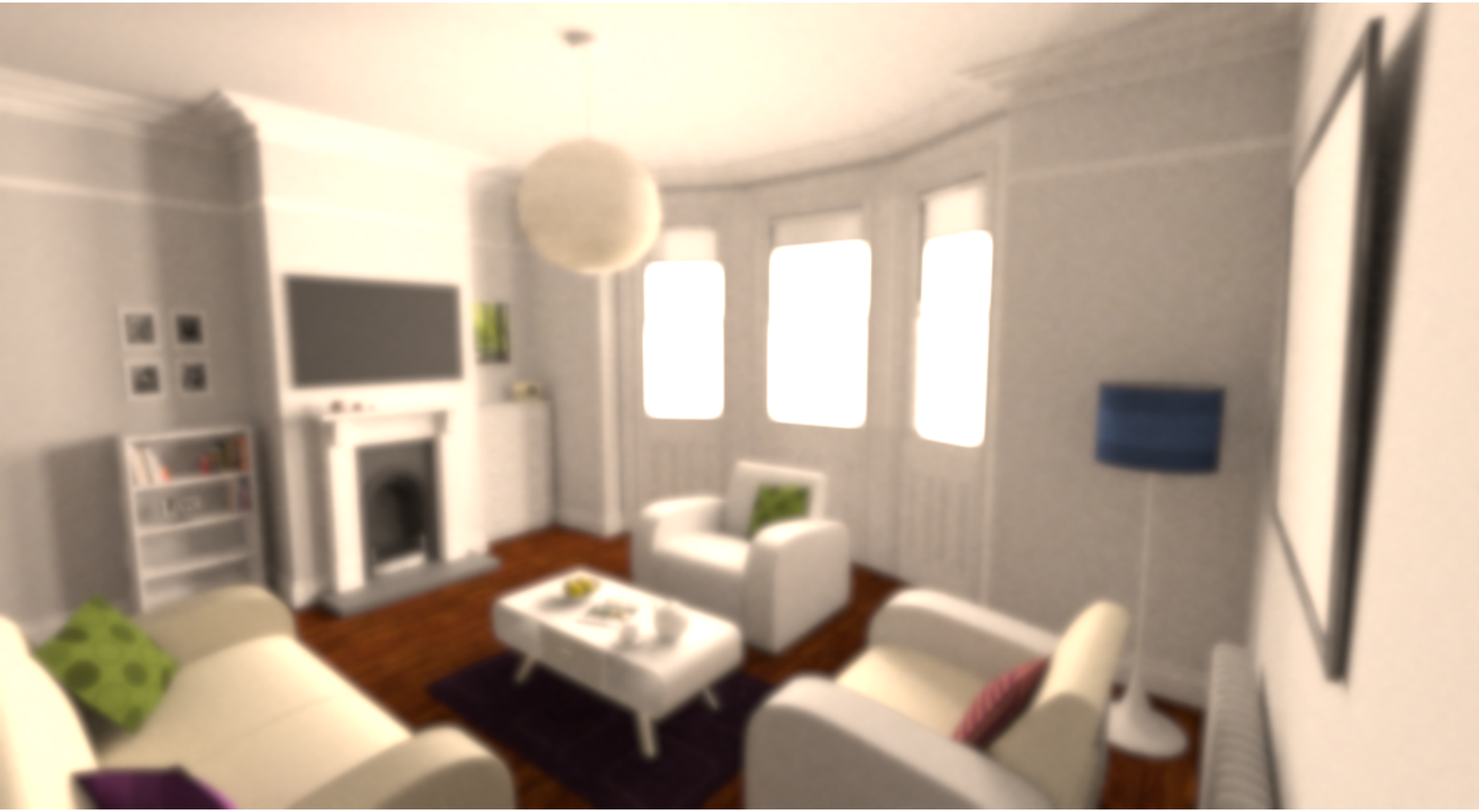


Frequency Domain

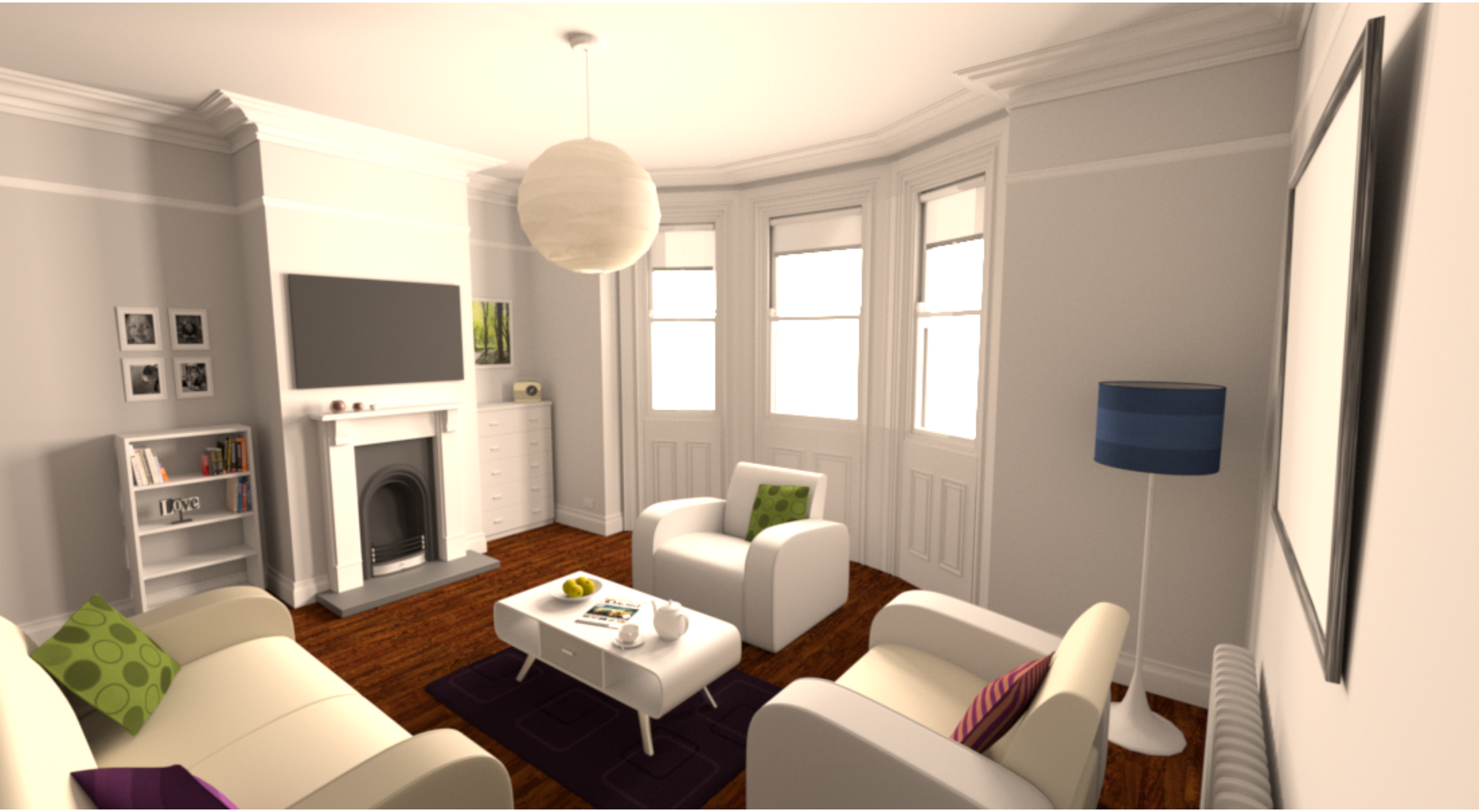
64 pixel samples



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:

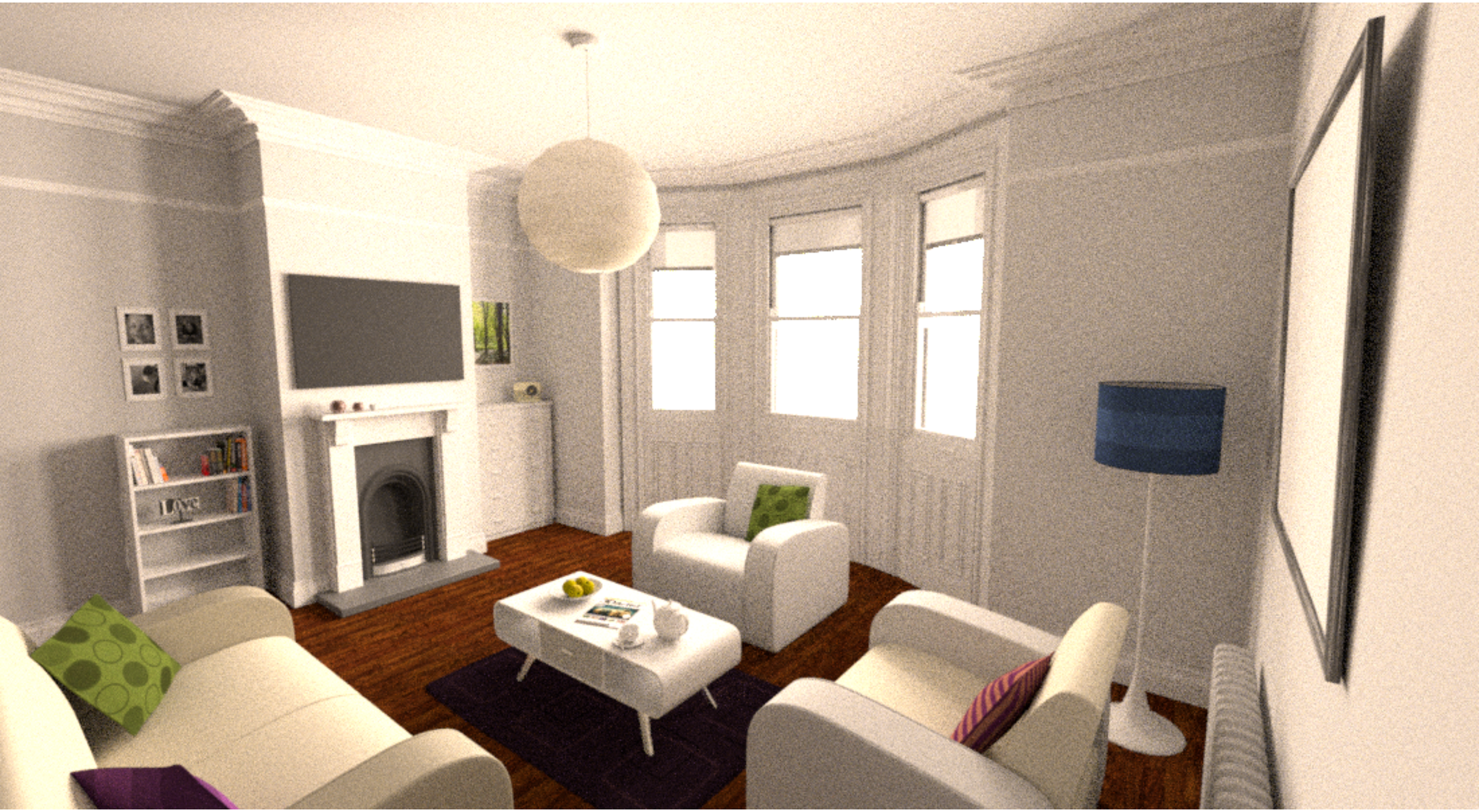
$$L_{\mathrm{o}}(\omega_{\mathrm{o}}) = \int_{H^2} f_r(\omega_{\mathrm{i}} \to \omega_{\mathrm{o}}) \, L_{\mathrm{i}}(\omega_{\mathrm{i}}) \, \cos\theta_{\mathrm{i}} \, \mathrm{d}\omega_{\mathrm{i}}$$

Ratio approximates incident radiance:

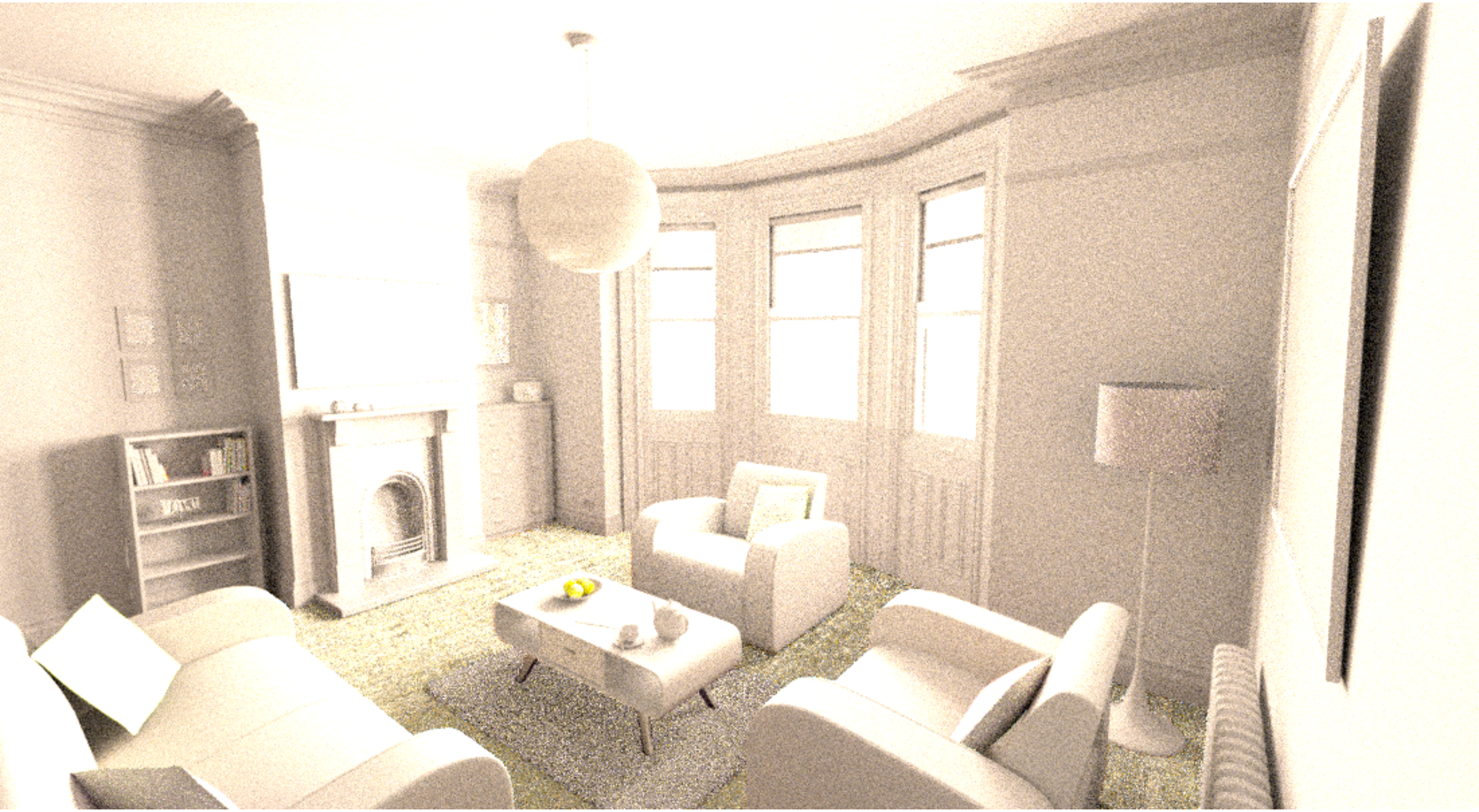
$$\frac{L_{\mathrm{o}}(\omega_{\mathrm{o}})}{\rho_{\mathrm{hd}}(\omega_{\mathrm{o}})} = \frac{\int_{H^2} f_r(\omega_{\mathrm{i}} \to \omega_{\mathrm{o}}) \, L_{\mathrm{i}}(\omega_{\mathrm{i}}) \, \cos\theta_{\mathrm{i}} \, \mathrm{d}\omega_{\mathrm{i}}}{\int_{H^2} f_r(\omega_{\mathrm{i}} \to \omega_{\mathrm{o}})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

Hemi-Directional Reflectance (Albedo)

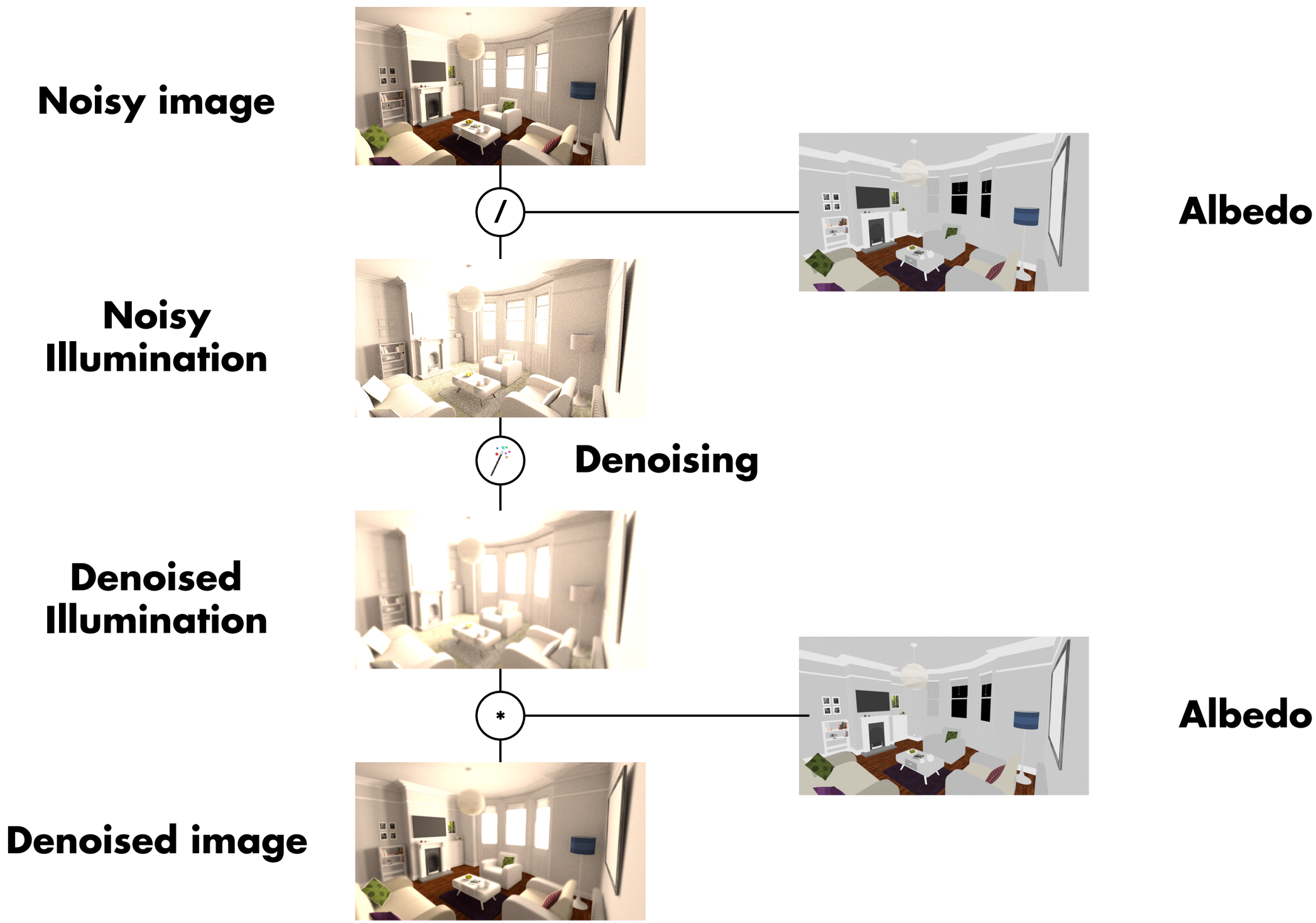
White Room, 64 pixel samples



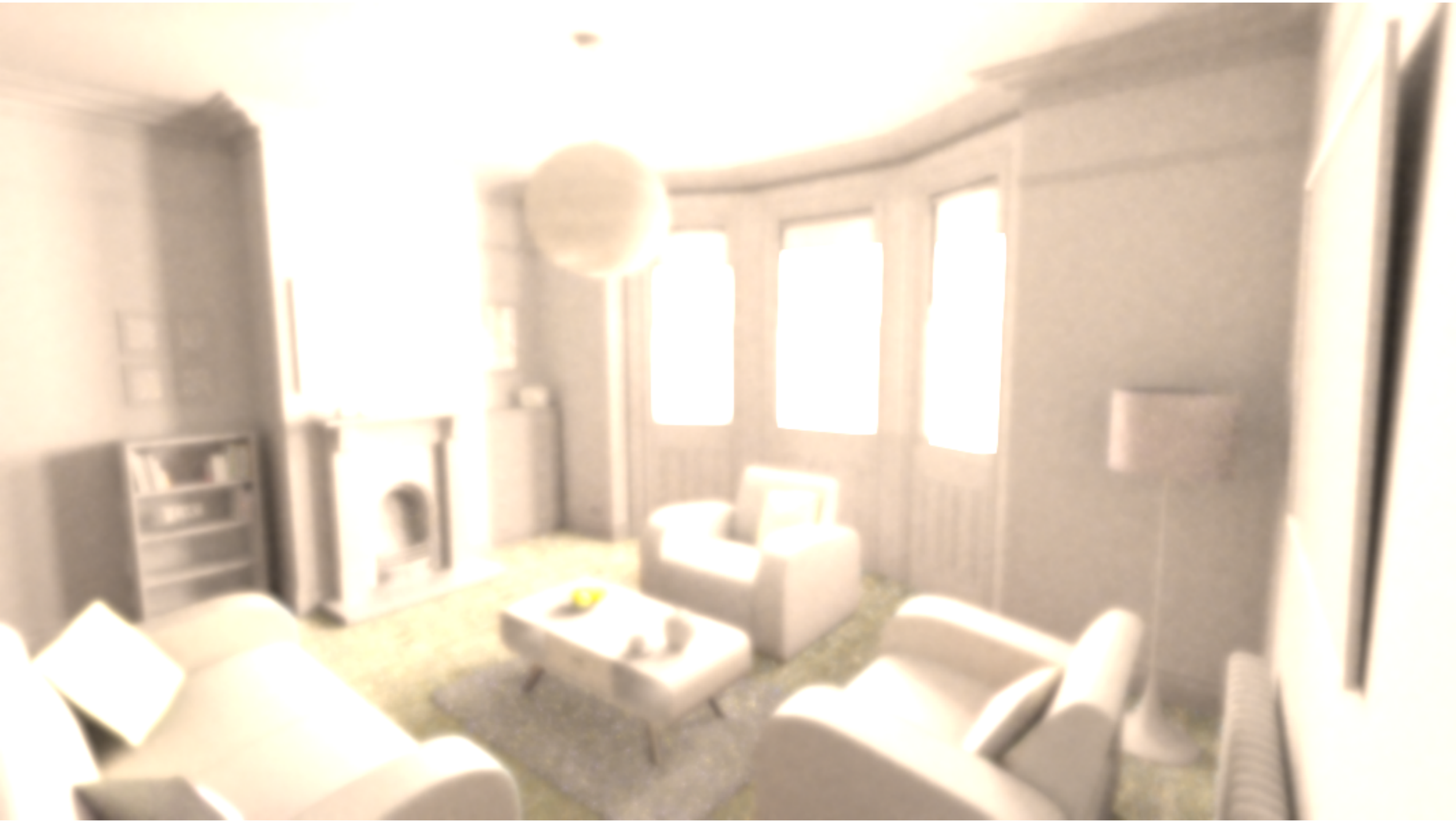
Final Image / Albedo ~ Illumination



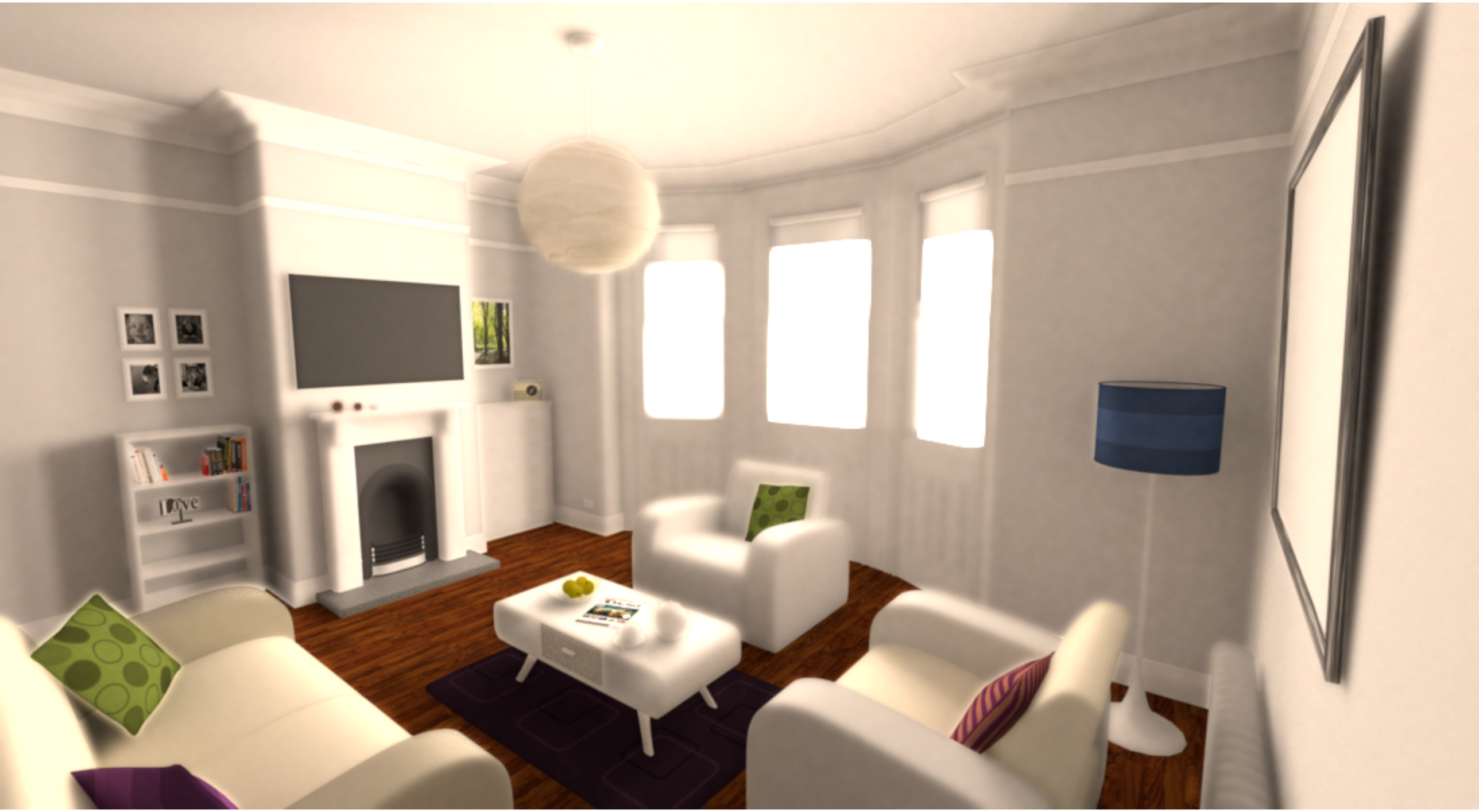
General Pipeline



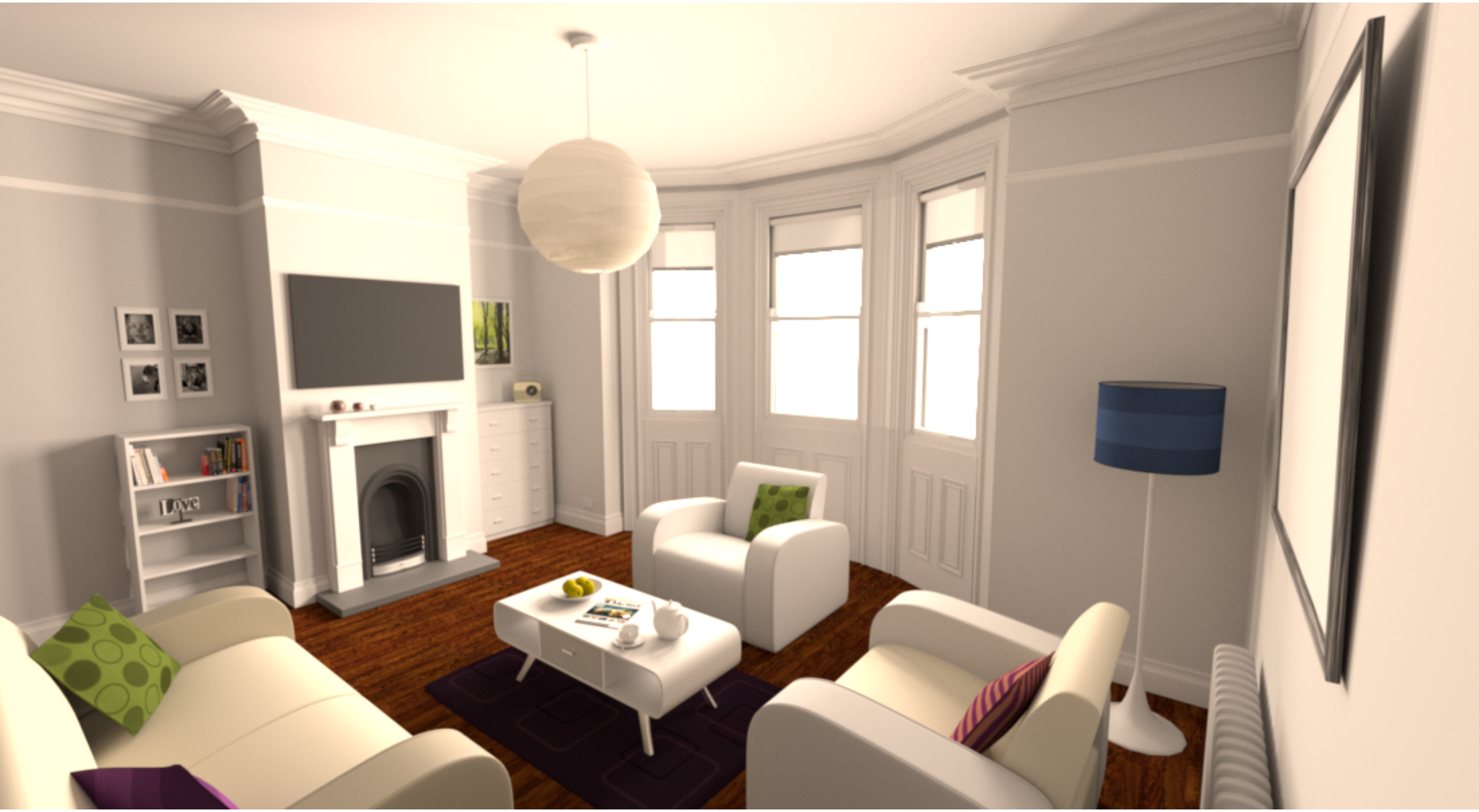
Gaussian Blurred Illumination



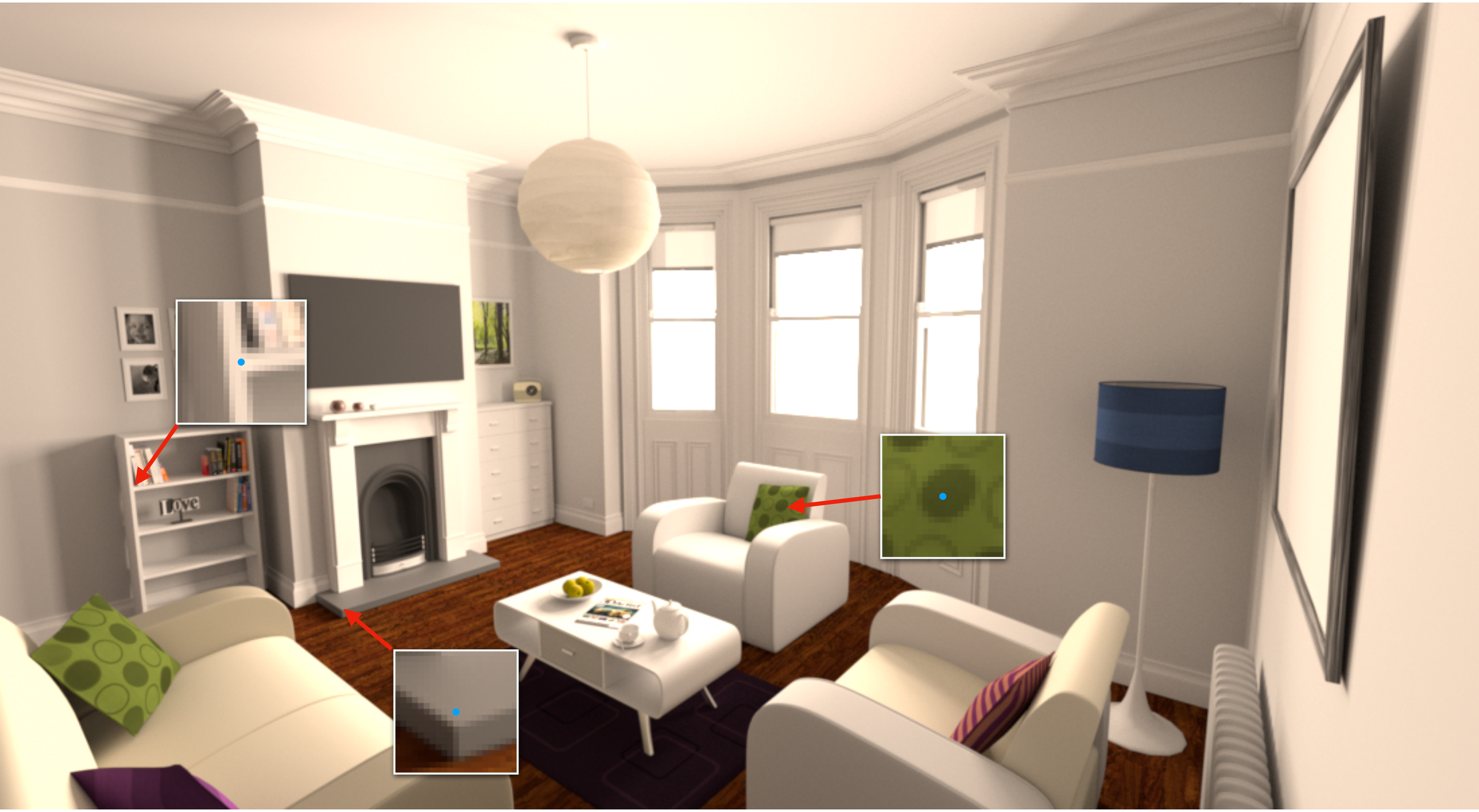
Blurred Illumination * Albedo



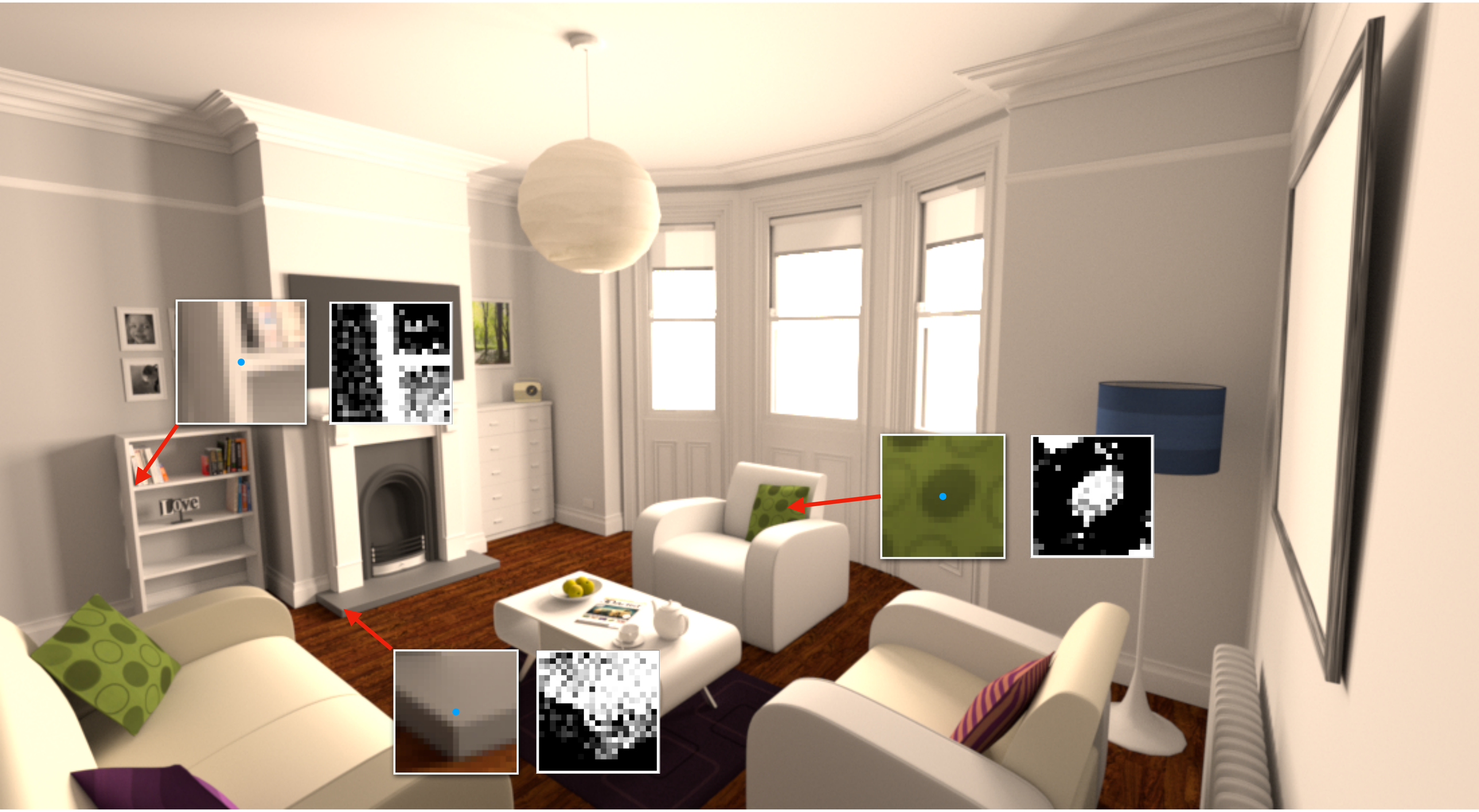
White Room, 4096 pixel samples



Ideal Denoising Filters (via Brute Force)



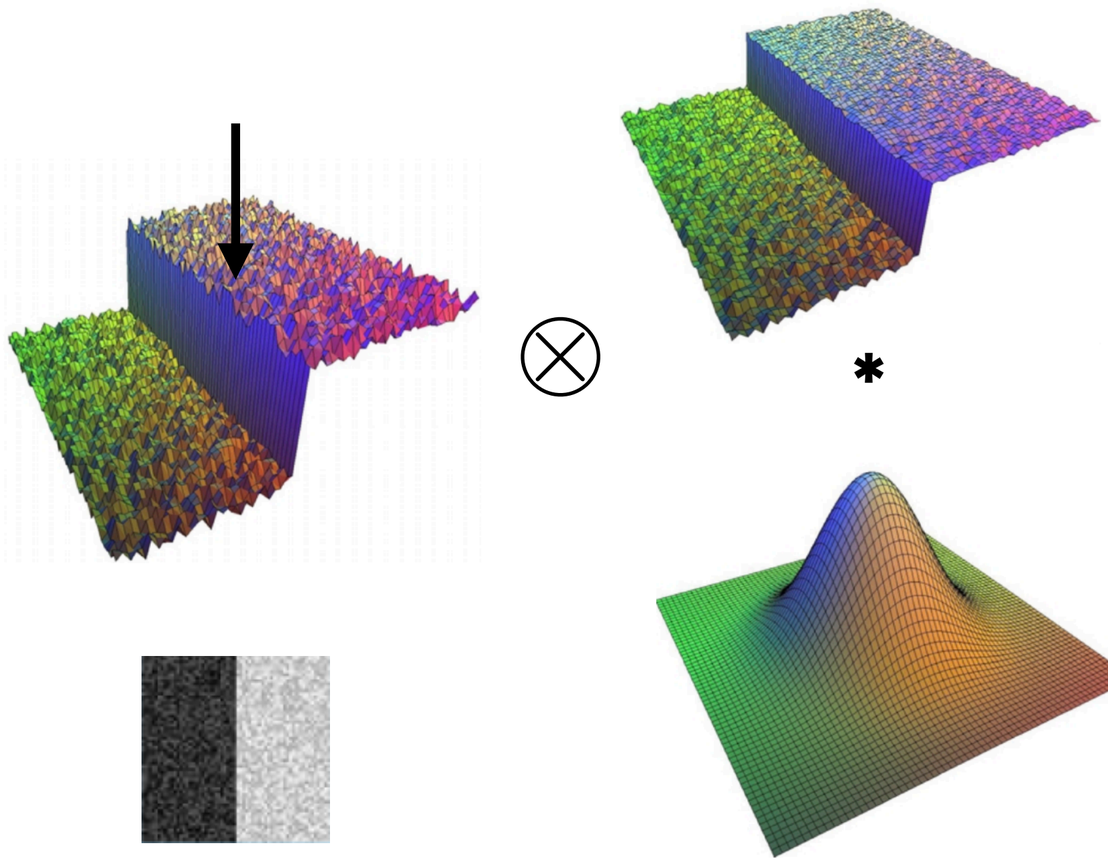
Ideal Denoising Filters (via Brute Force)



Better Filter: Bilateral

Pixel intensity kernel

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



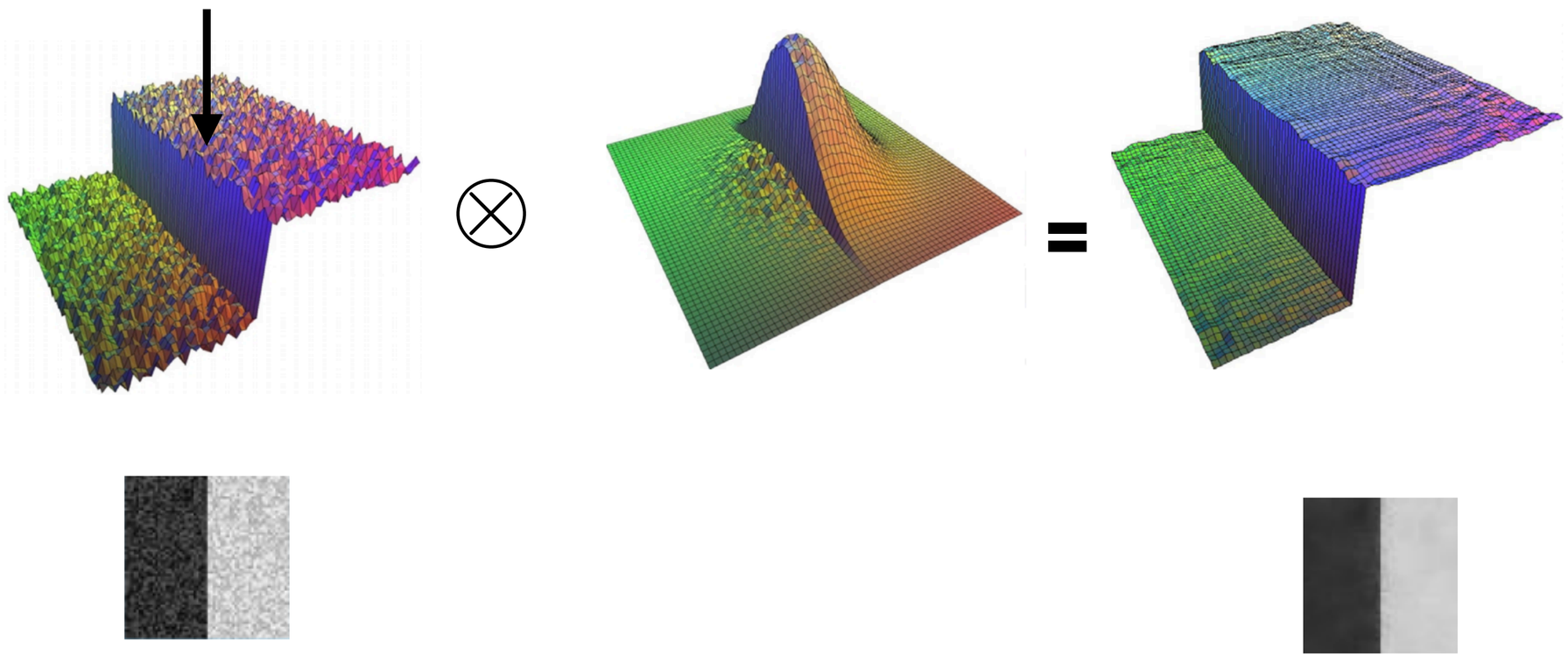
e.g. $d(p,p') = |I(p) - I(p')|$

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



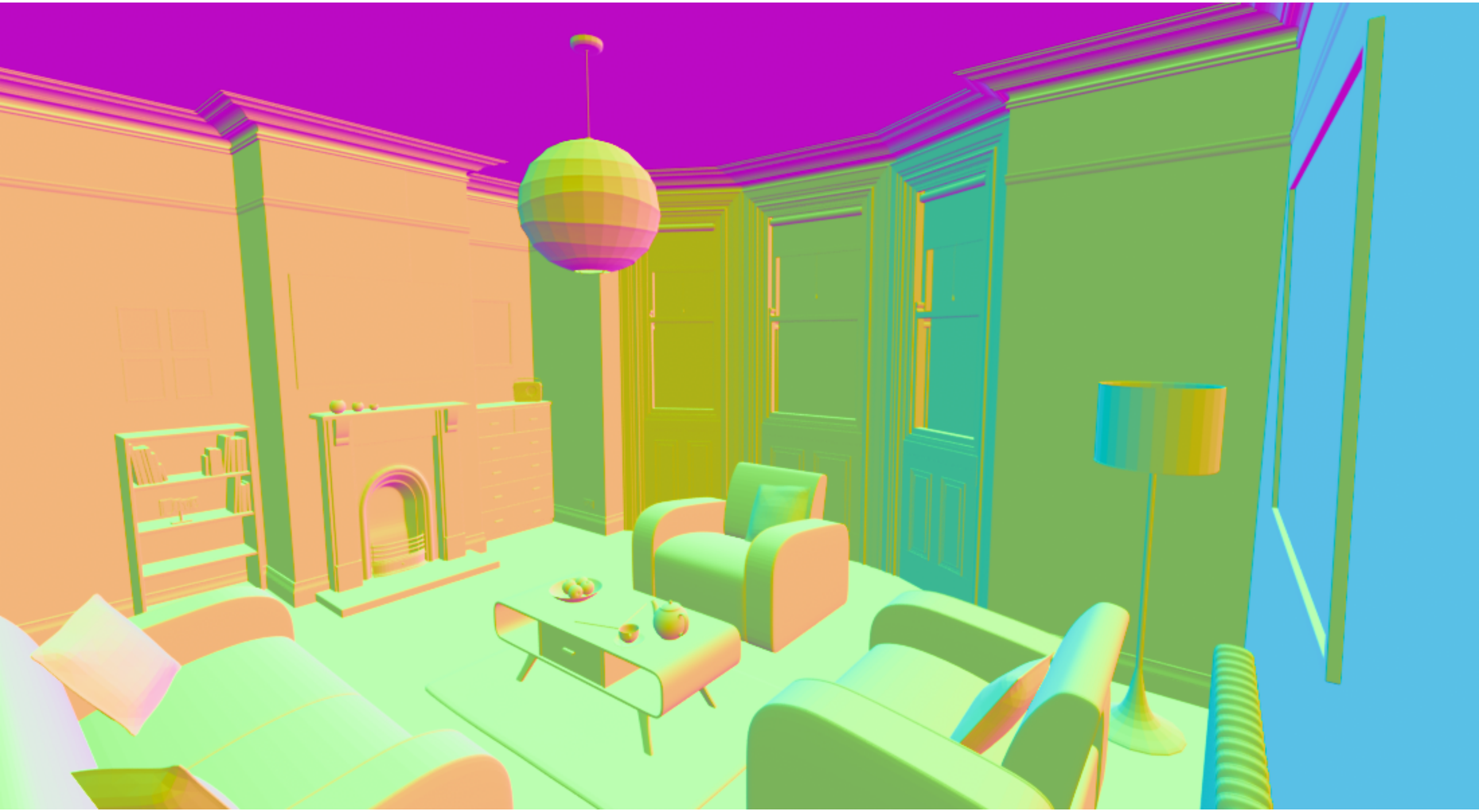
[Durand and Dorsey 2002]

Even Better Filter: Joint Bilateral

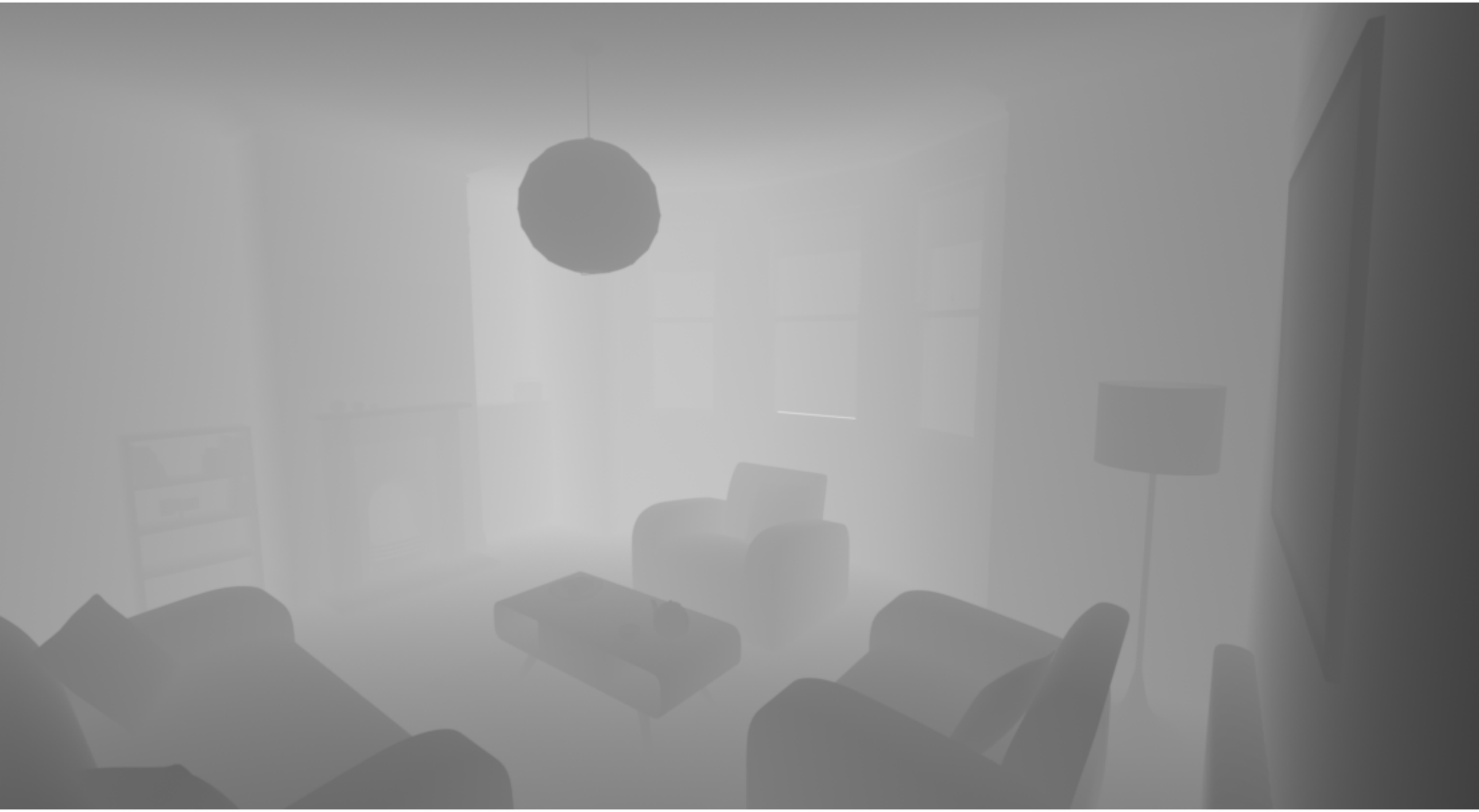
Include additional, non-visible features in the filter

- **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:

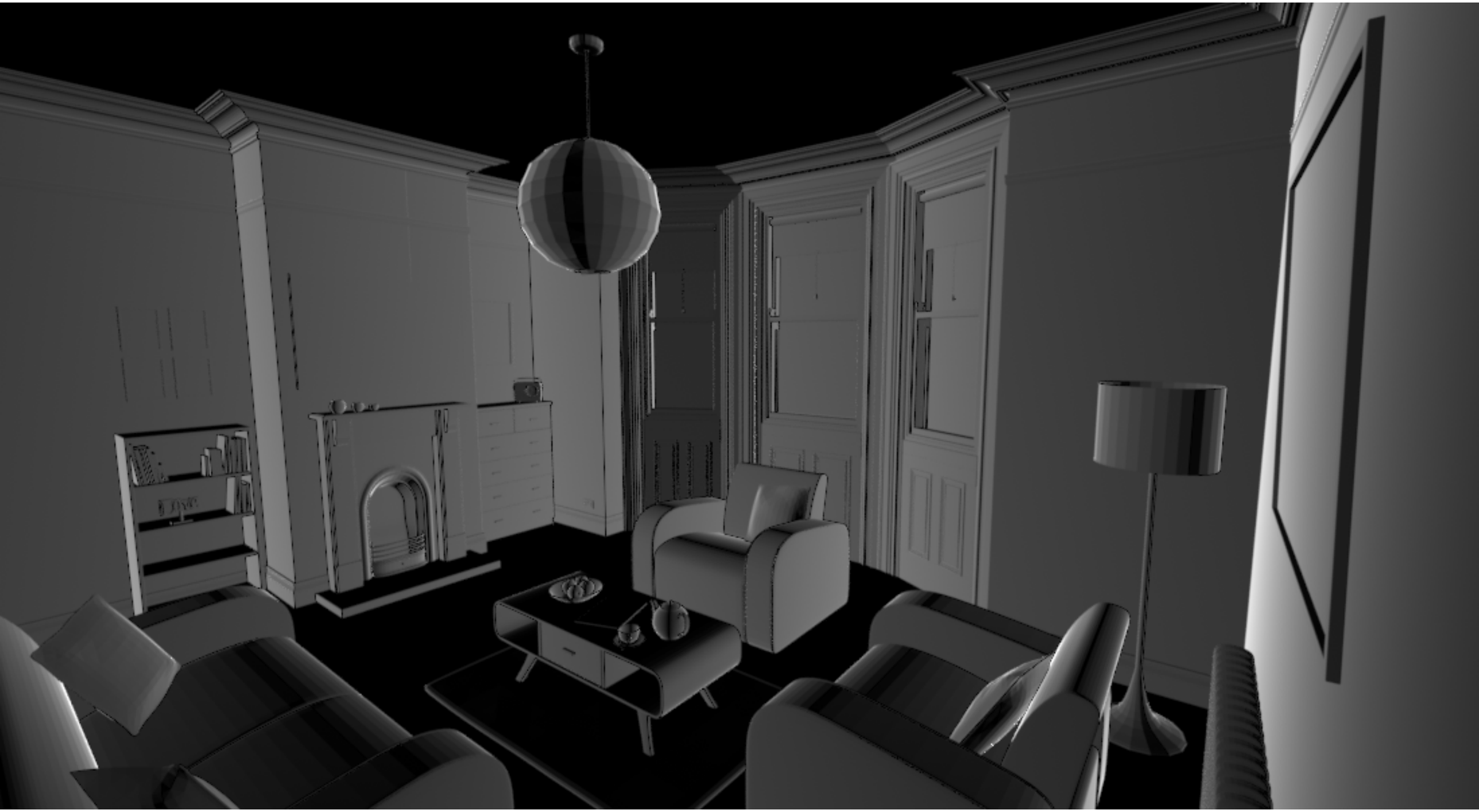
$$z(\Delta x, \Delta y) \approx z + \Delta x \frac{\partial z}{\partial x} + \Delta y \frac{\partial z}{\partial y}$$

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

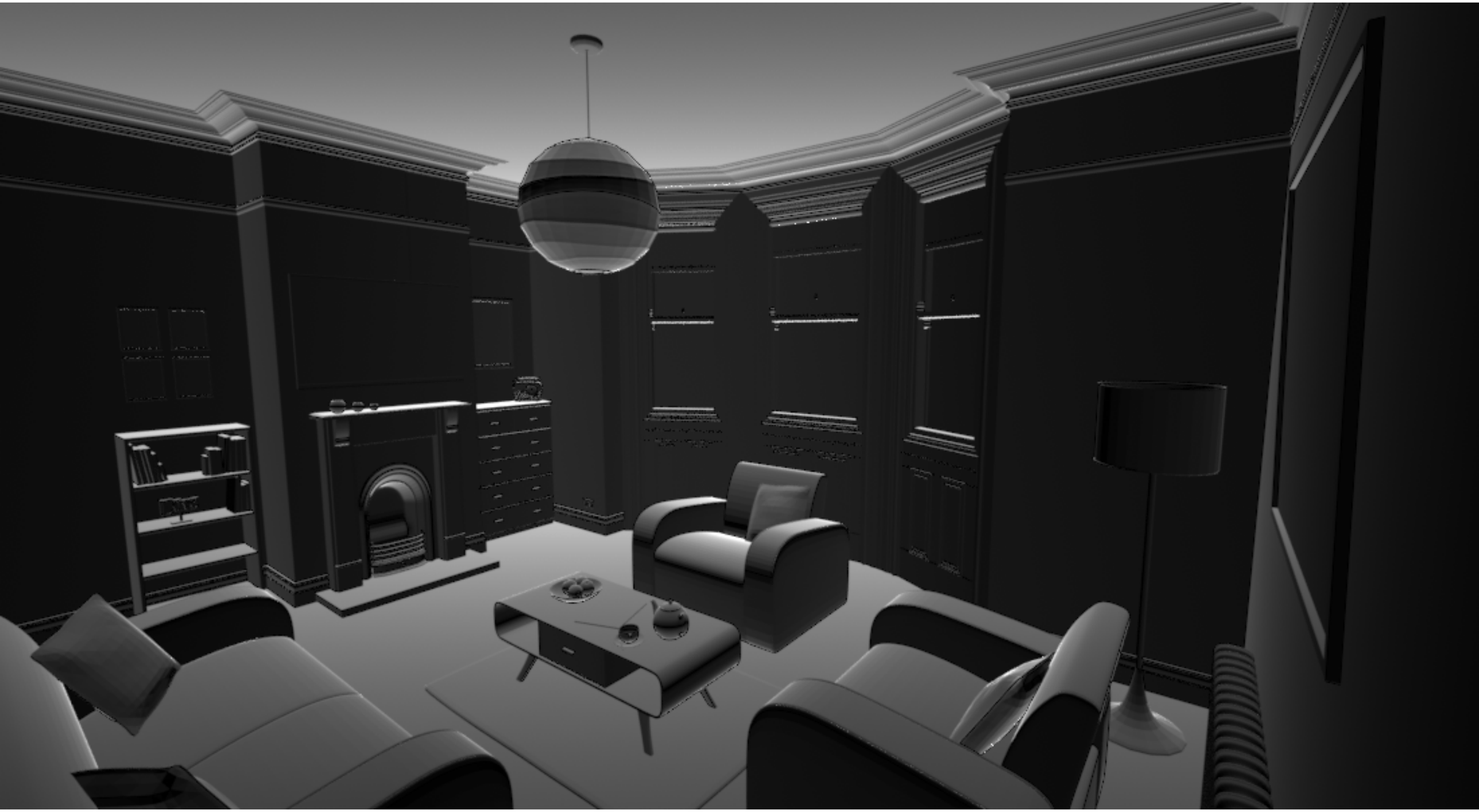
Given a depth z' at a nearby pixel, can compute distance from planar approximation,

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

dz/dx



dz/dy



Pat Hanrahan / Matt Pharr, Spring 2020

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

Pat Hanrahan / Matt Pharr, Spring 2020

Revised Pipeline

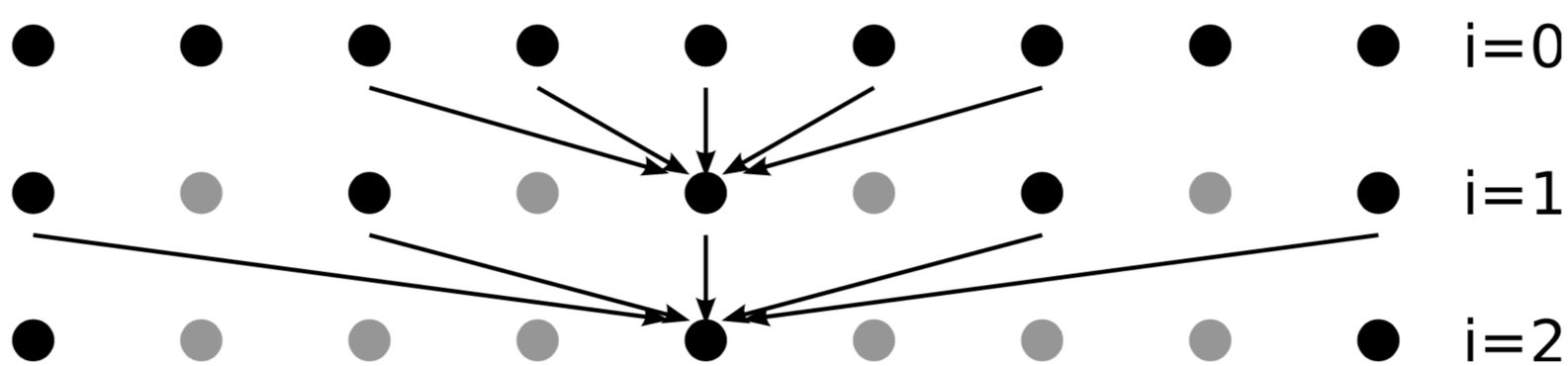
Multi-Level Filtering: À-Trous

Want wide filter kernels to eliminate noise

- Recall Gaussian 19x19 was still blotchy

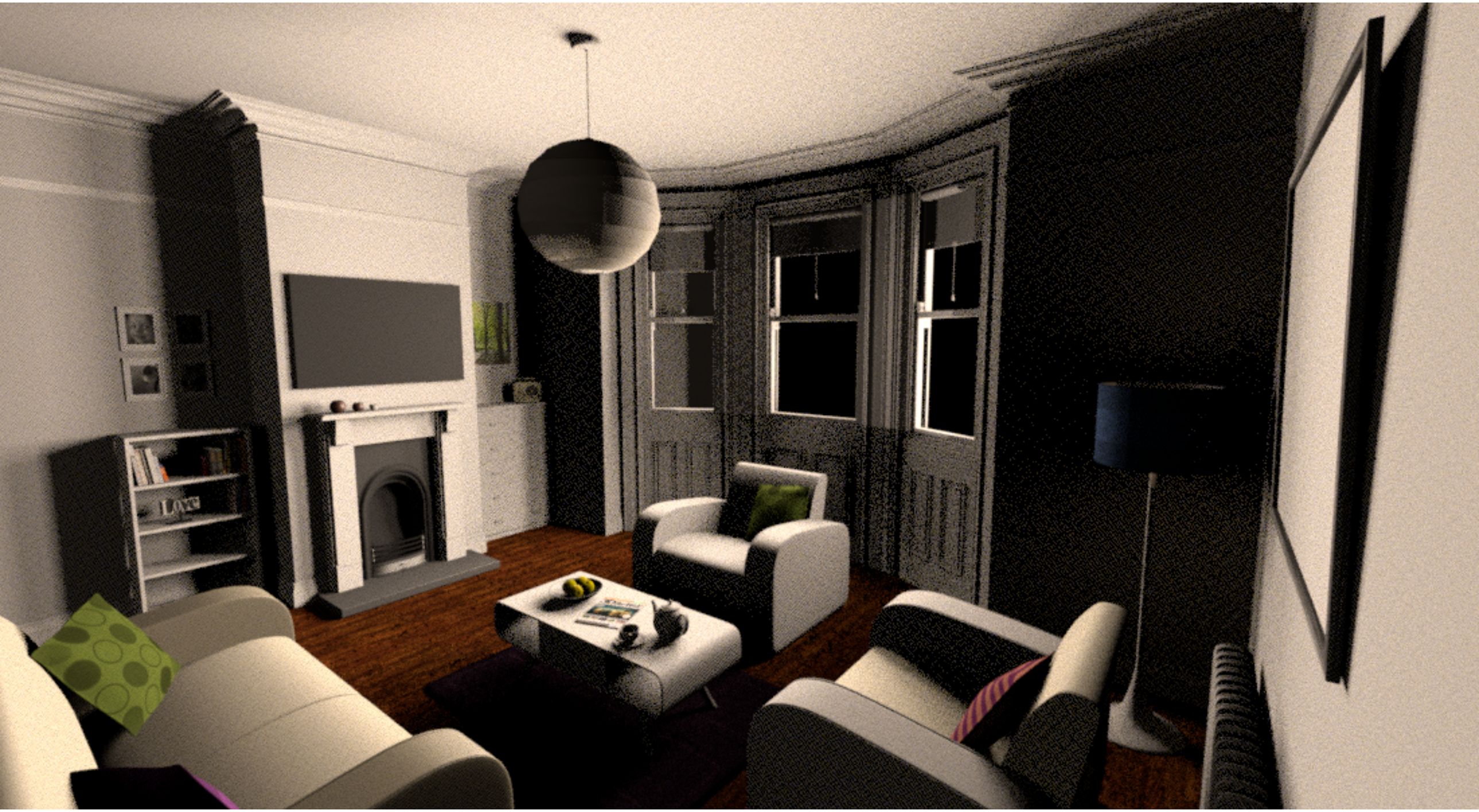
Wide kernels are expensive...

Multi-scale filtering:

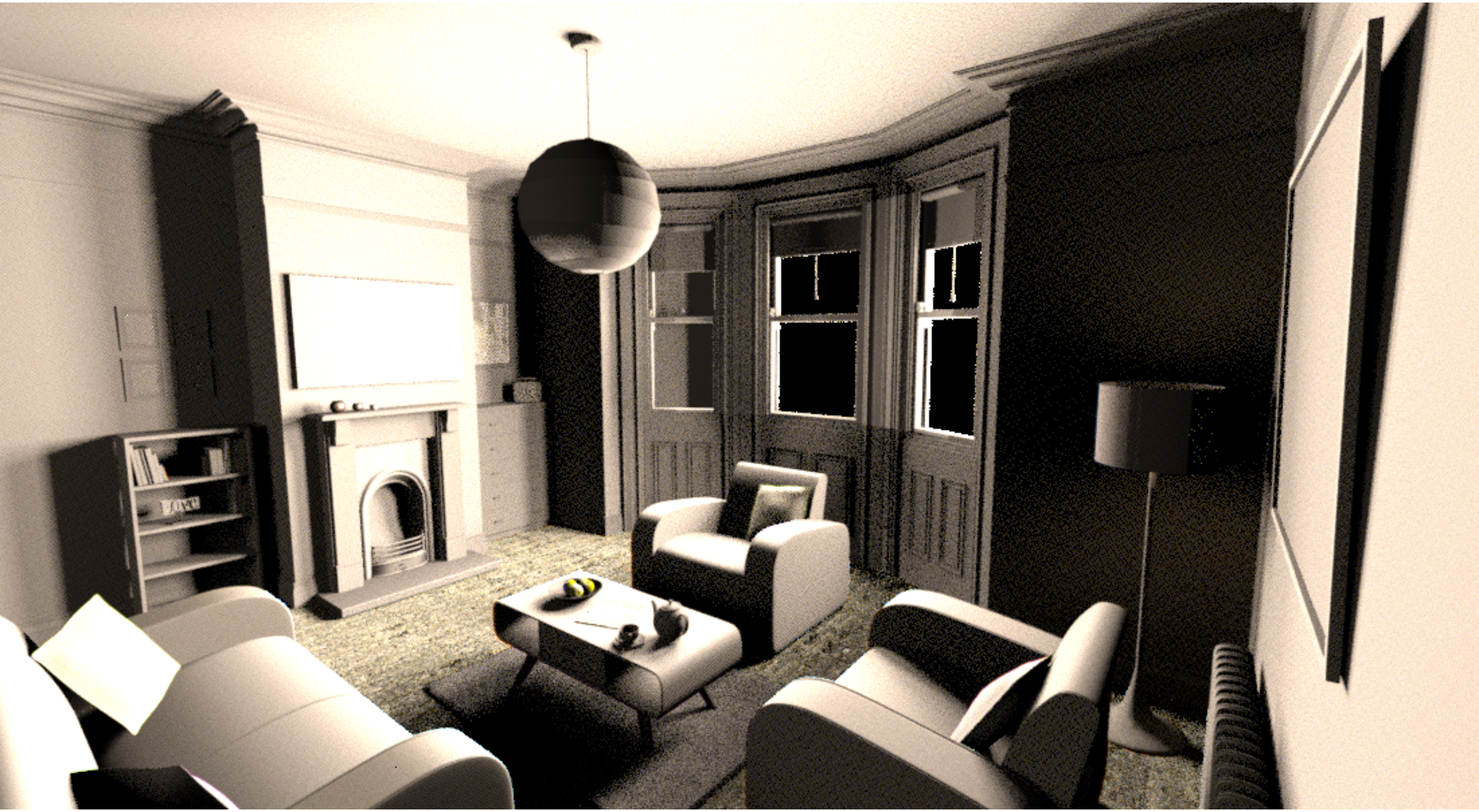


[Dammertz et al. 2010]

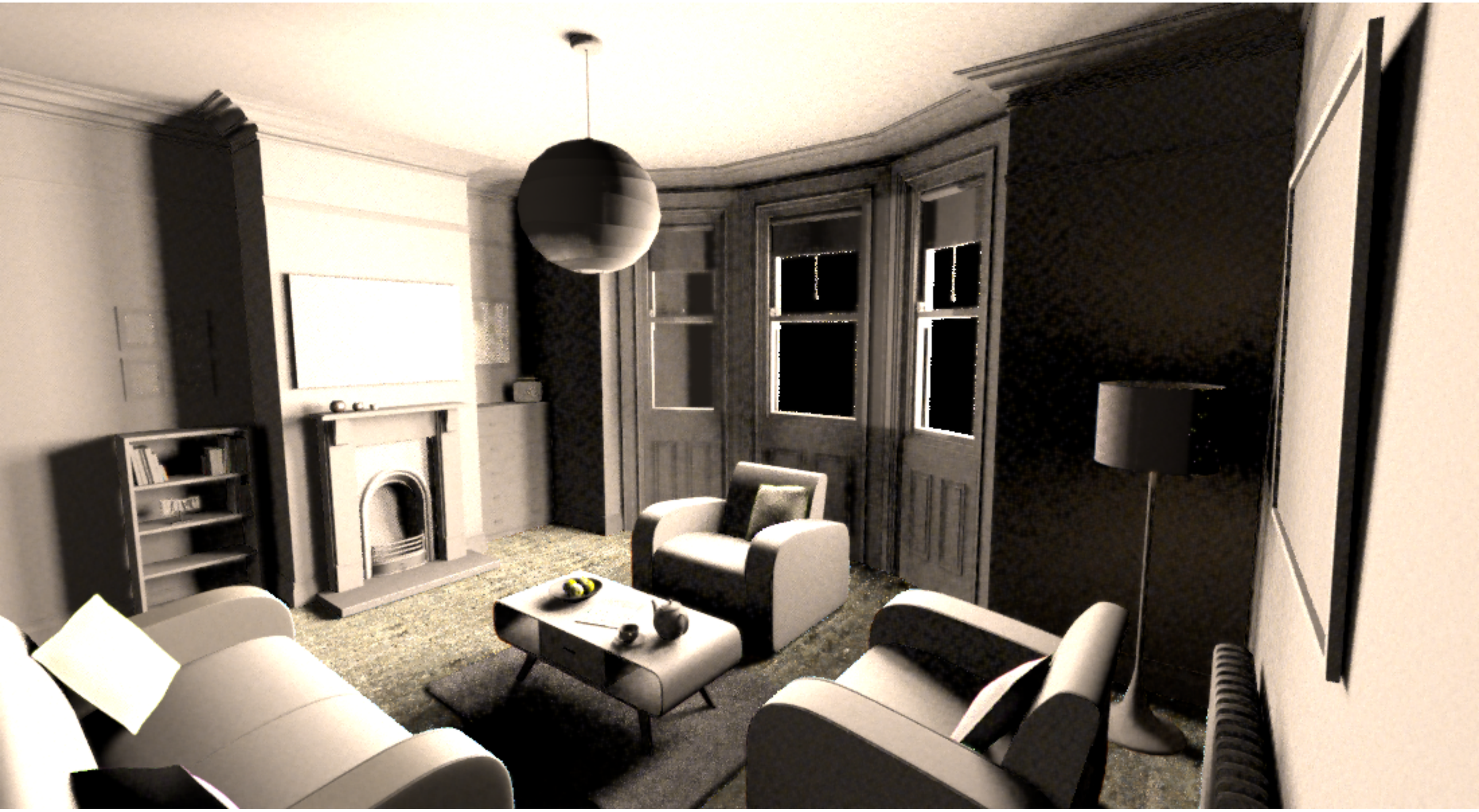
Direct Lighting



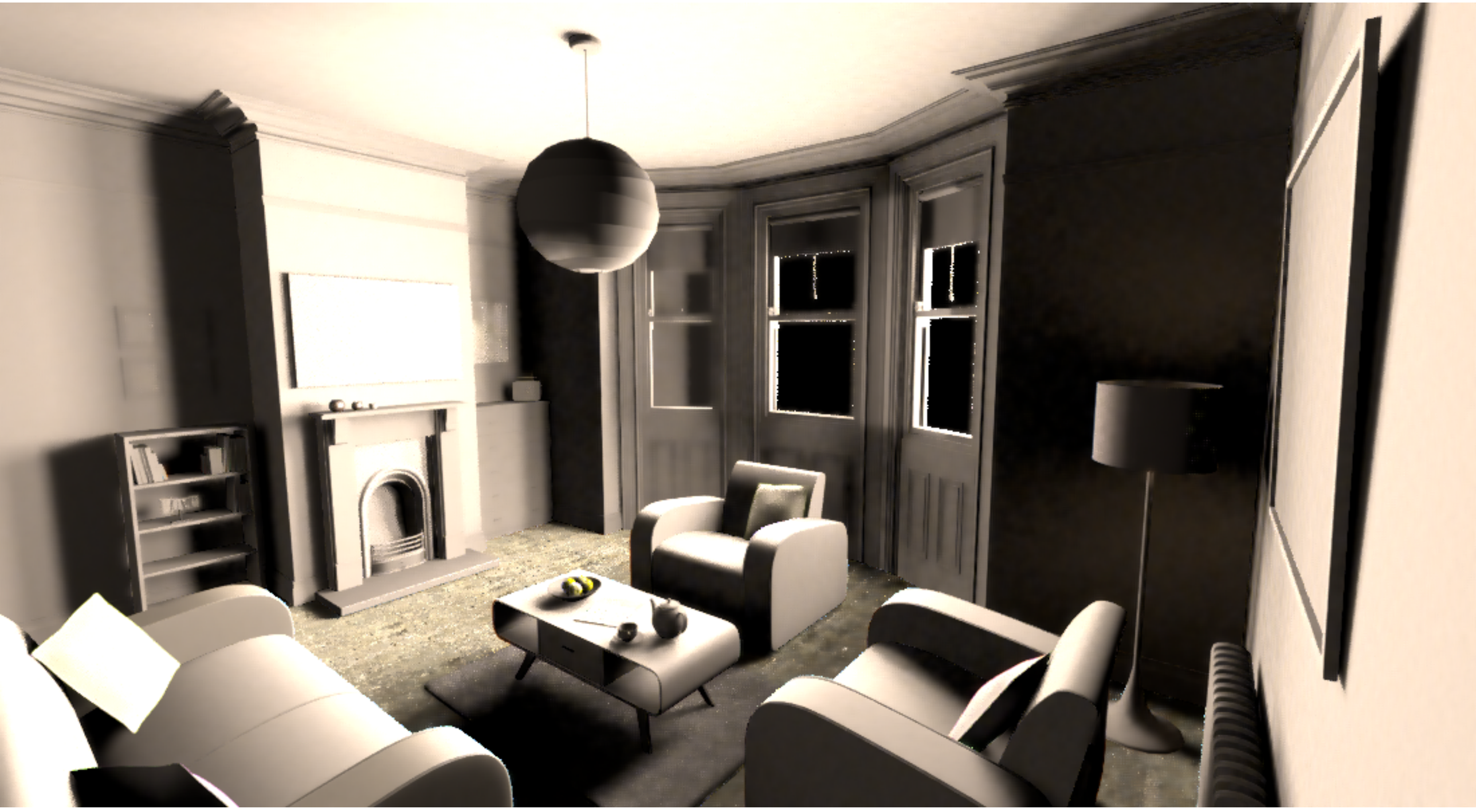
Direct Lighting / Albedo



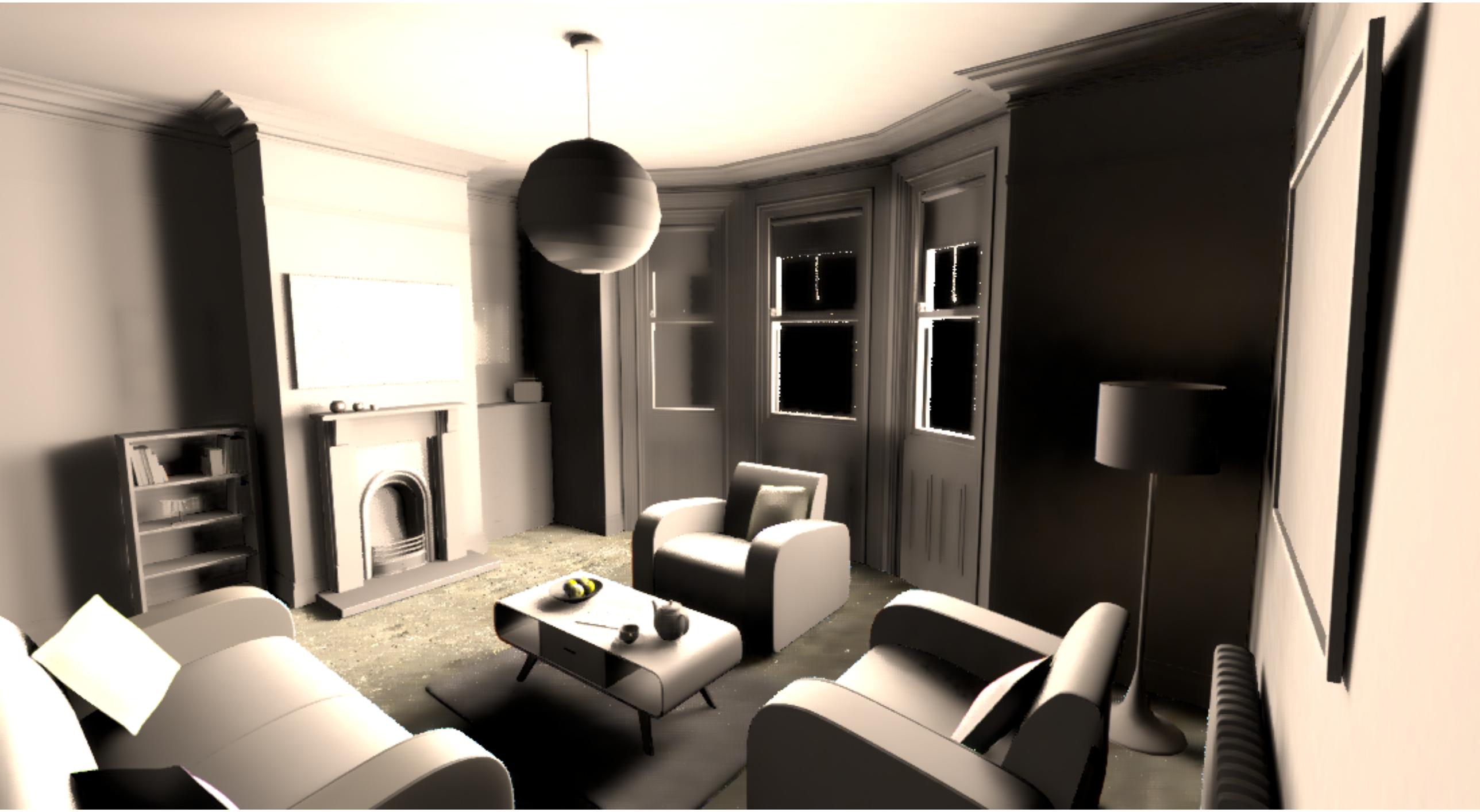
One Iteration



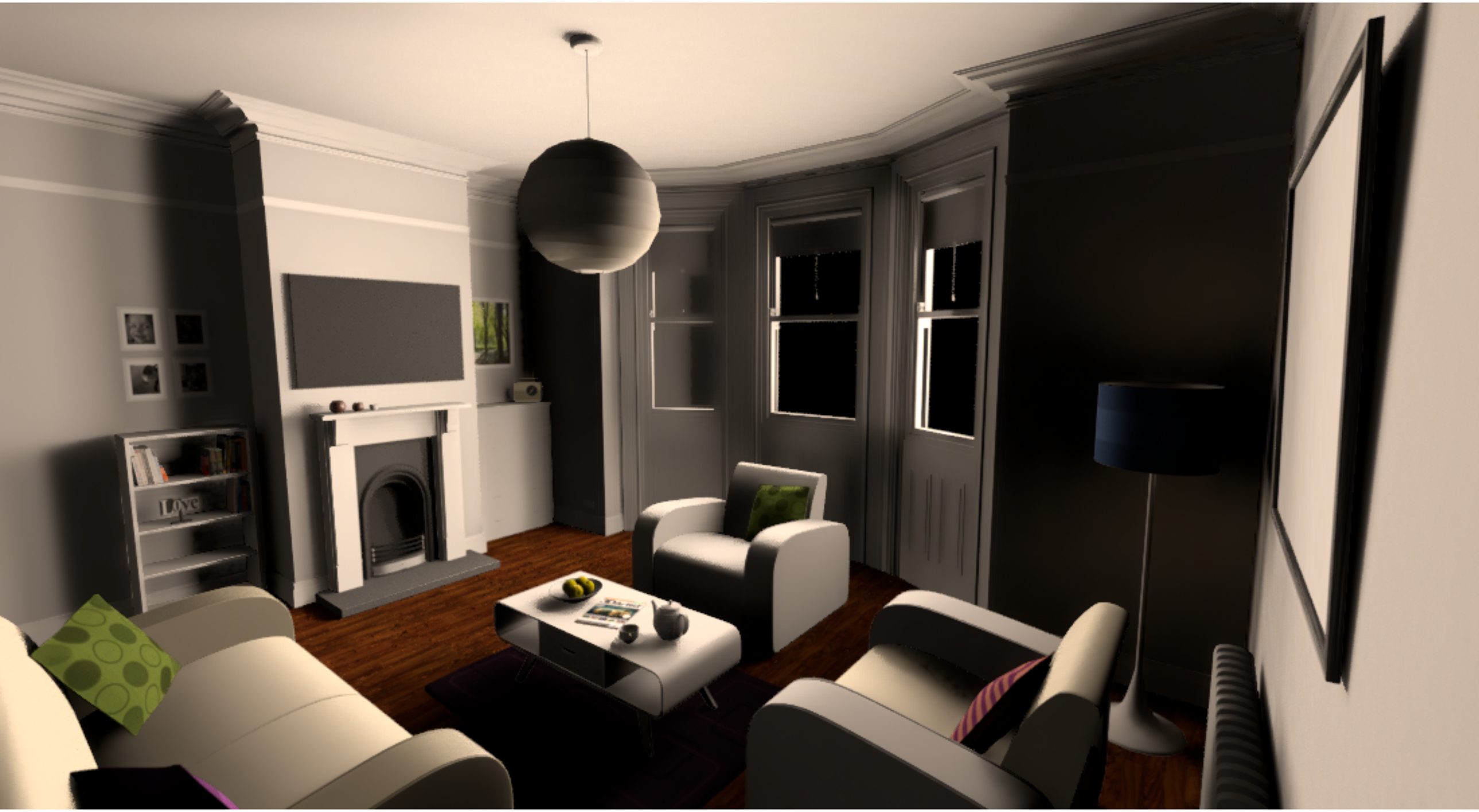
Two Iterations



Three Iterations

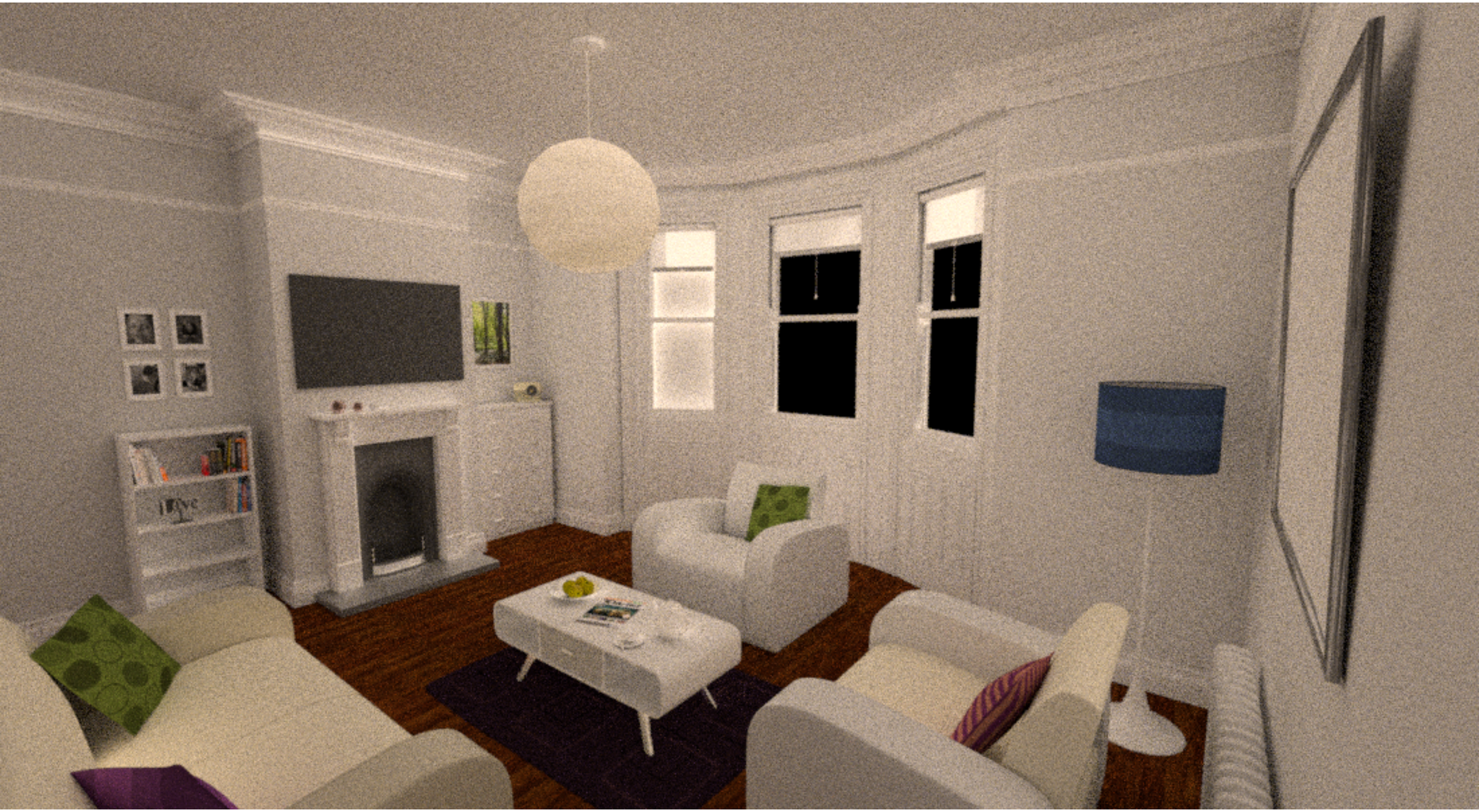


Filtered Direct Lighting

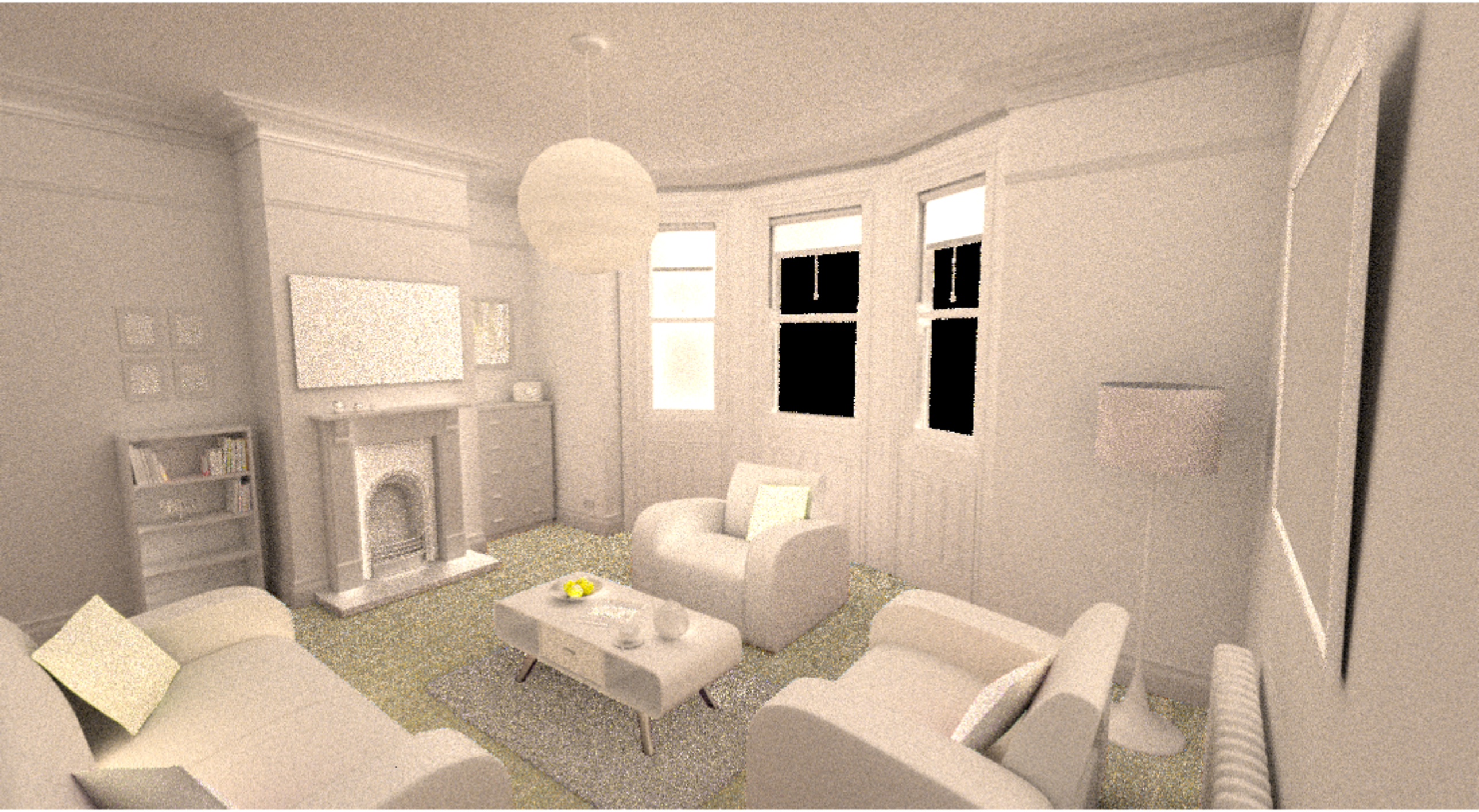


Pat Hanrahan / Matt Pharr, Spring 2020

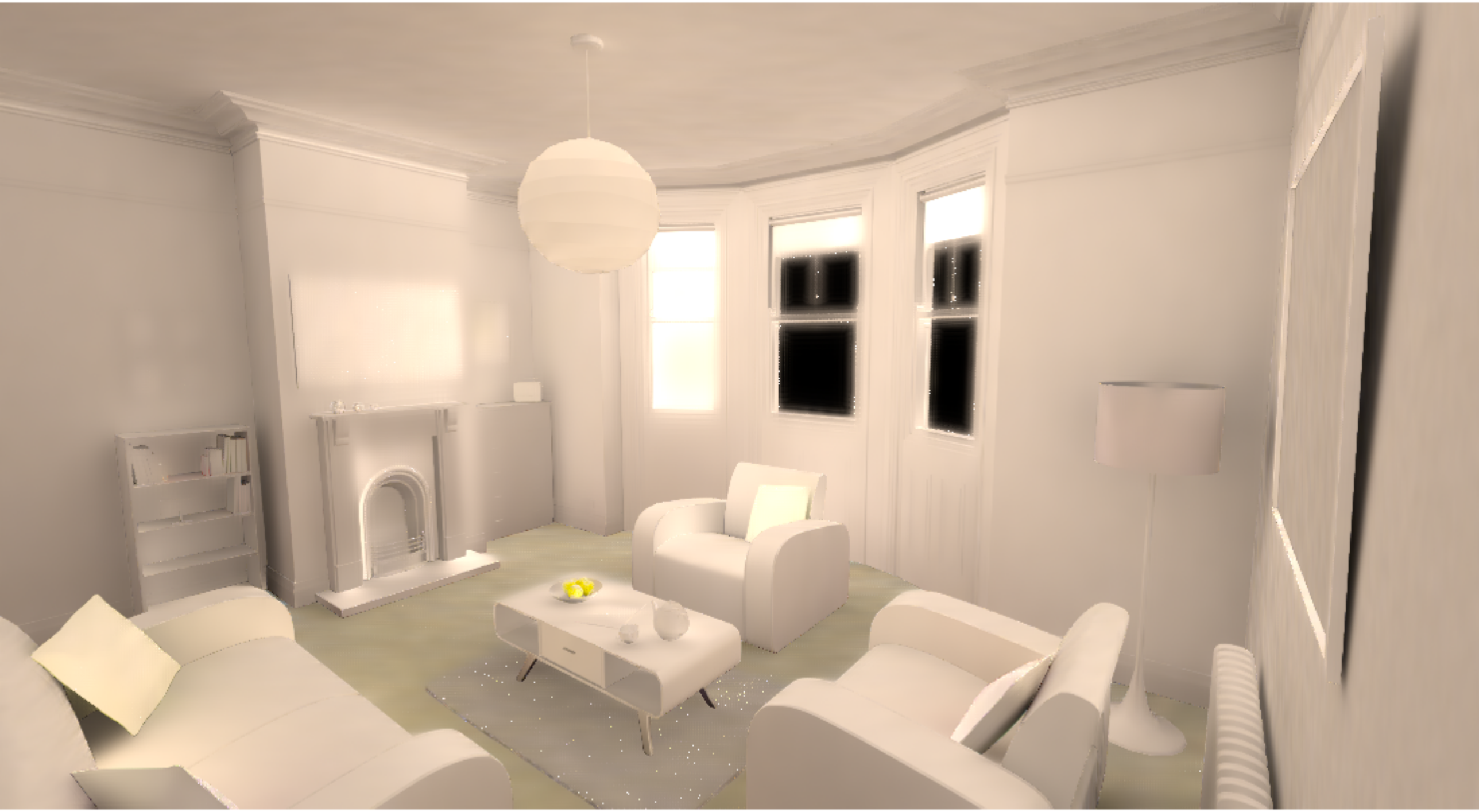
Indirect Lighting (64 samples)



Indirect Lighting / Albedo



Filtered Indirect Lighting / Albedo

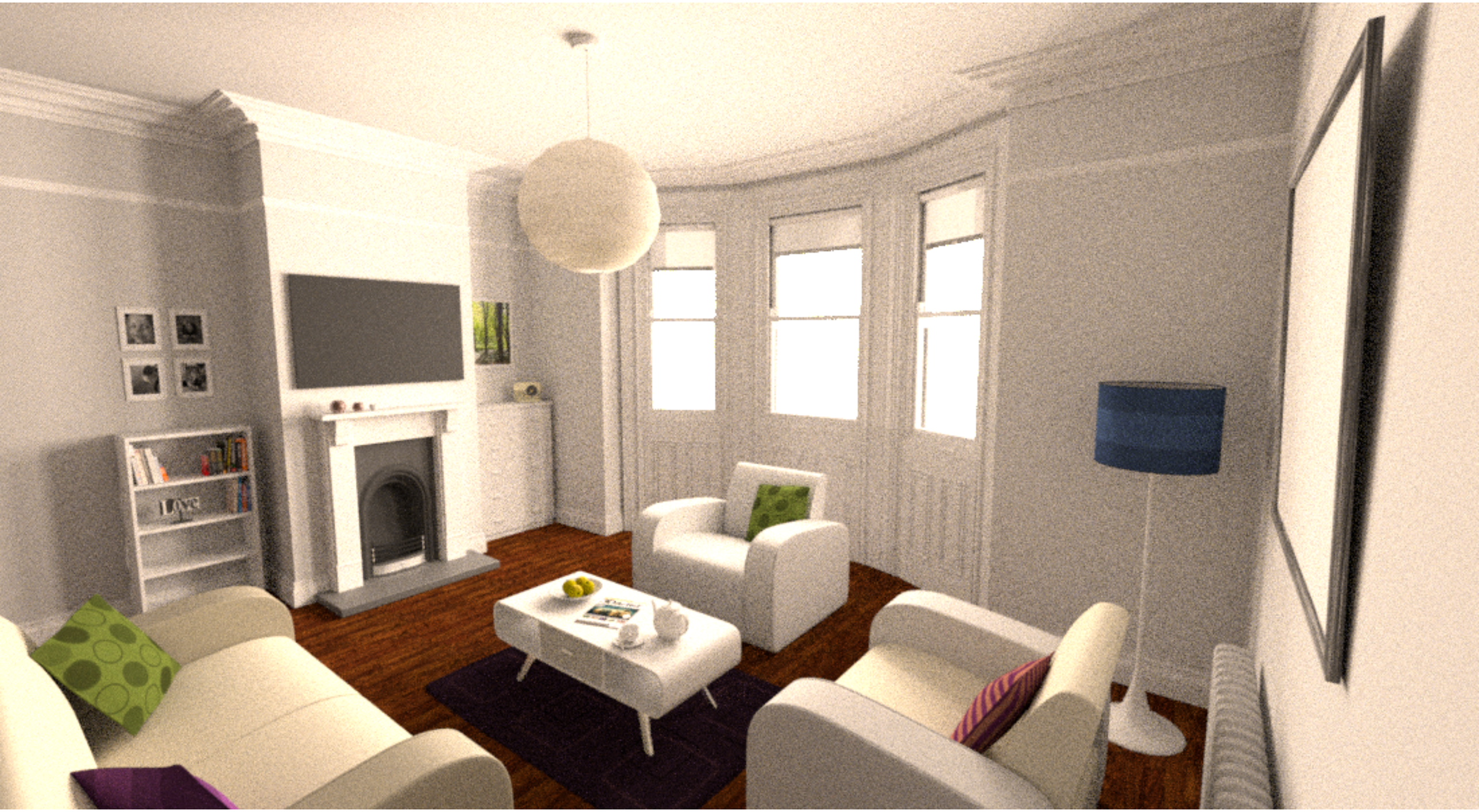


Pat Hanrahan / Matt Pharr, Spring 2020

Filtered Indirect Lighting

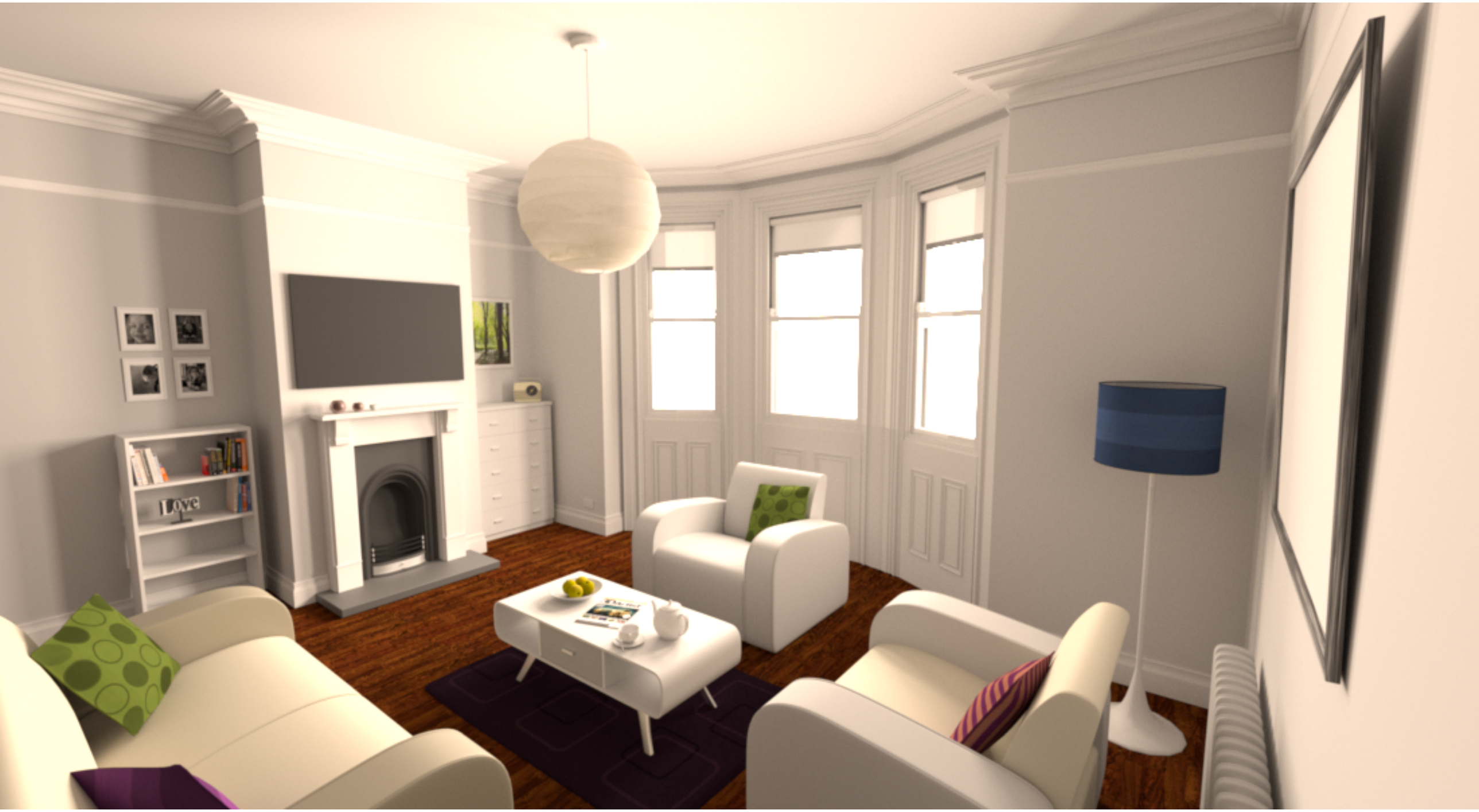
Pat Hanrahan / Matt Pharr, Spring 2020

64 samples per pixel, MSE = 3.44



64 denoised samples, MSE = 2.16

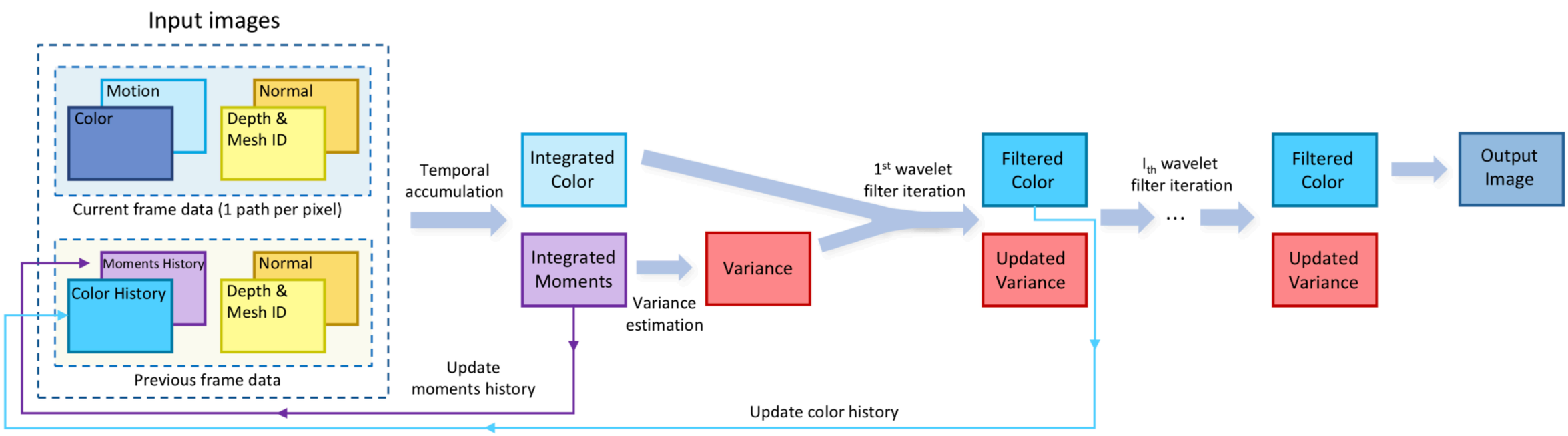
Reference (4096 samples)



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

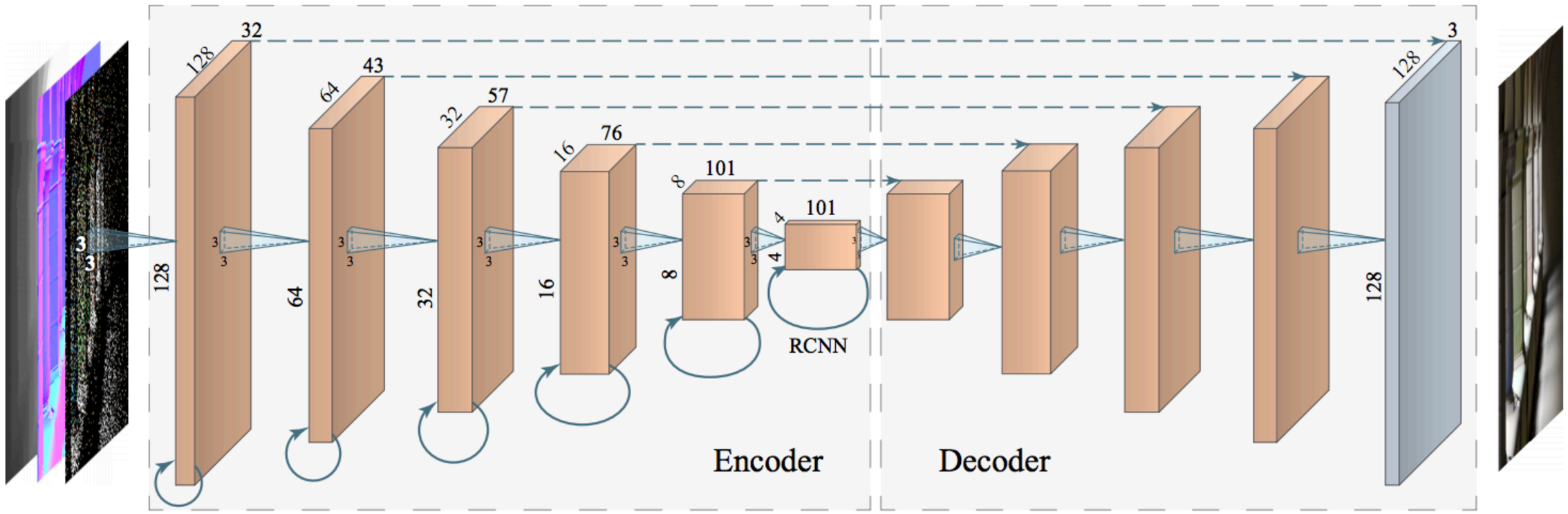
Our result with a cross-bilateral filter (4 spp)

[Kalantari et al. 2015]

3-layer fully connected MLP

Computes per-pixel filter parameters

Deep Denoising (NVIDIA)

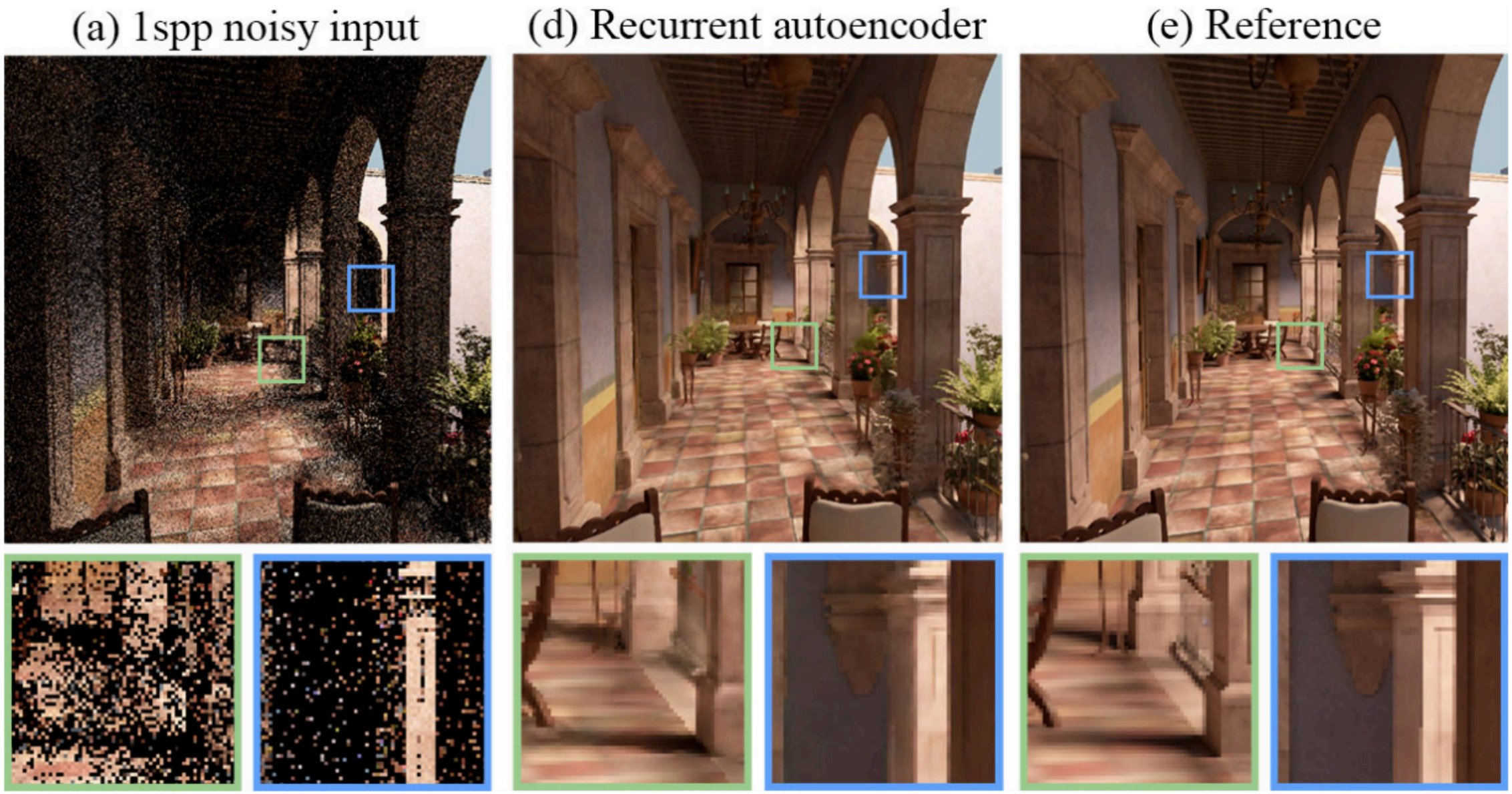


Features: illumination, normals, depth, roughness

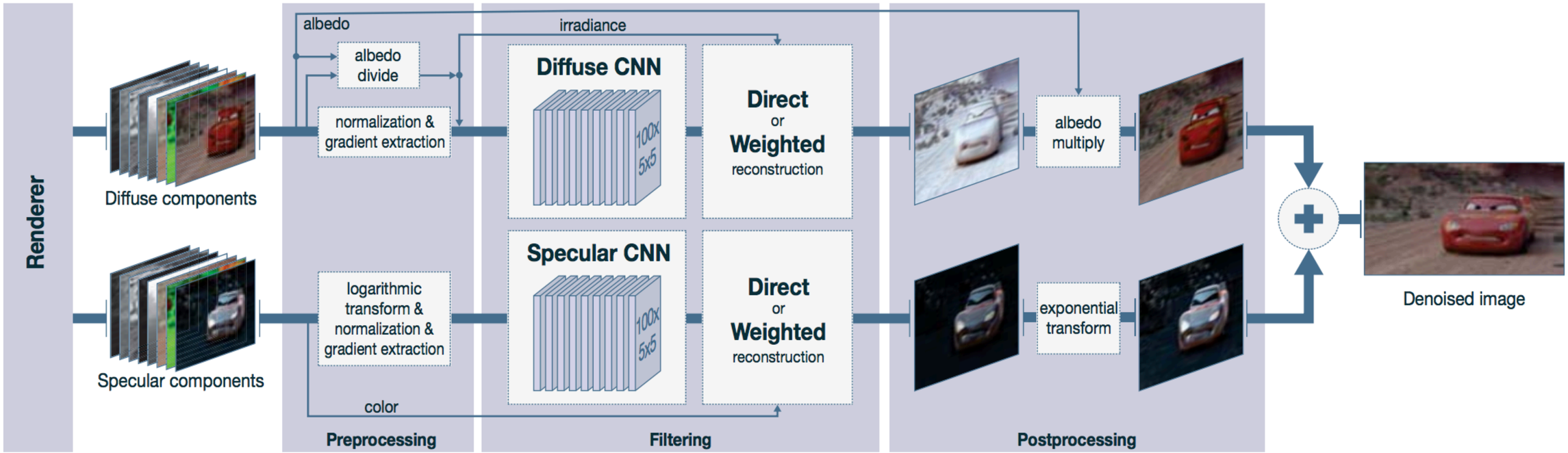
3.2M trainable parameters

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

Results



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

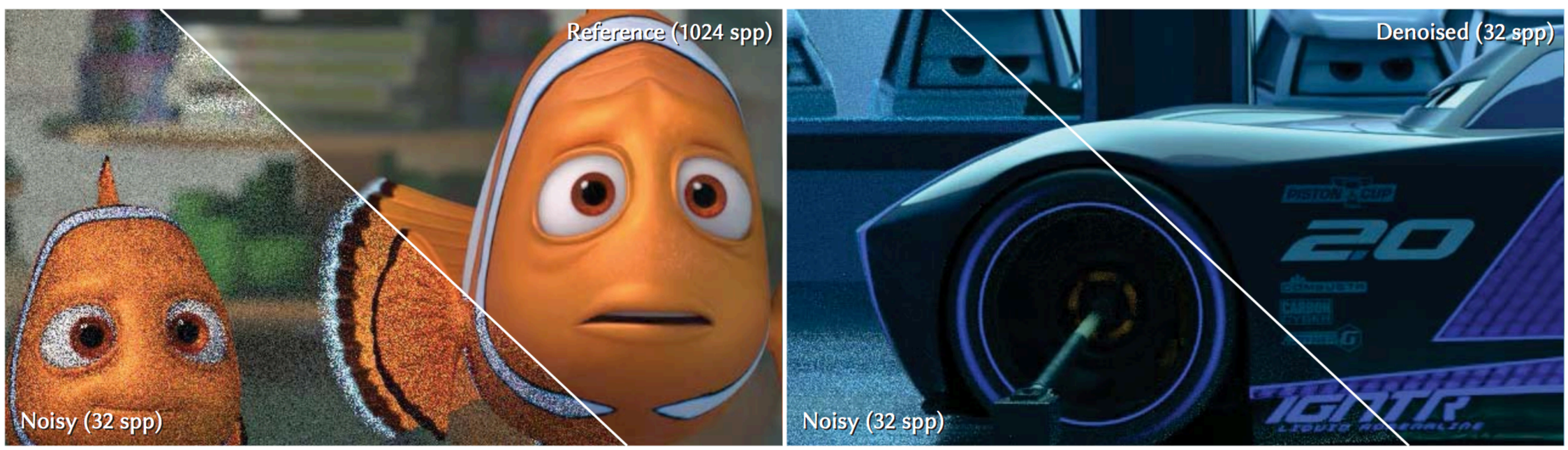
Network generates a stencil of 21x21 filter weights at each pixel

- 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

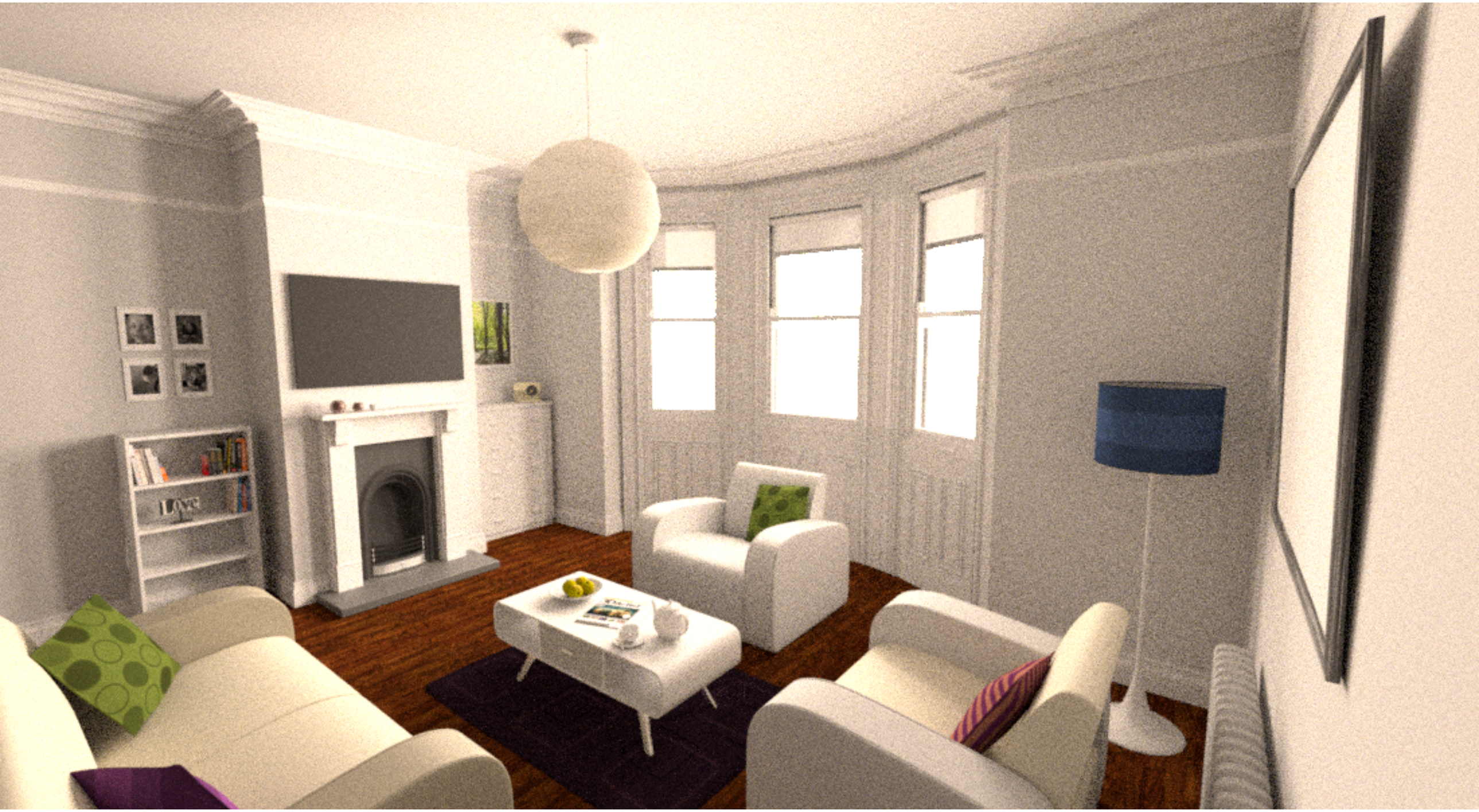
Results



TRAINING

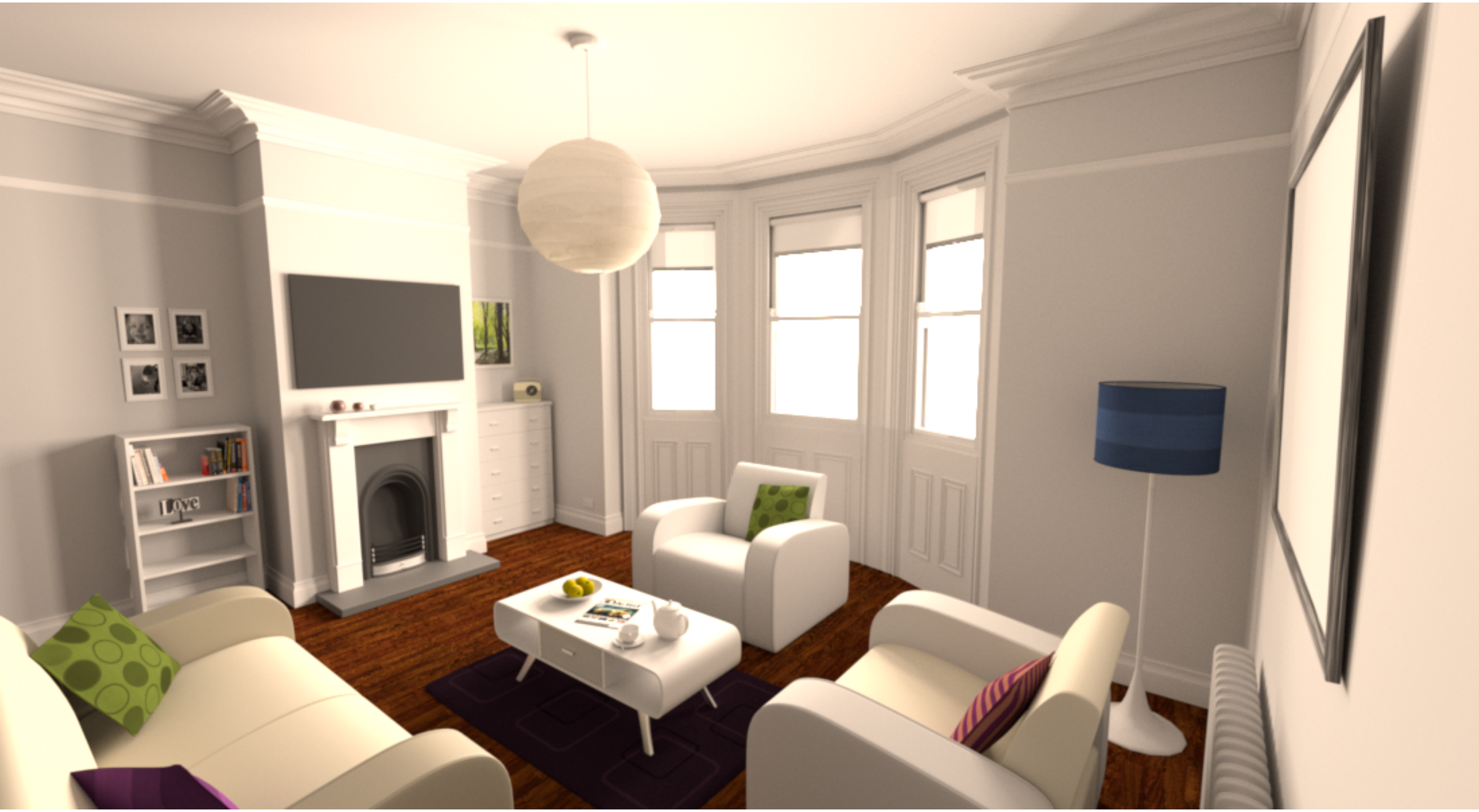
TEST

64 samples per pixel, MSE = 3.44



64 CNN-denoised samples, MSE = 2.38

Reference (4096 samples)

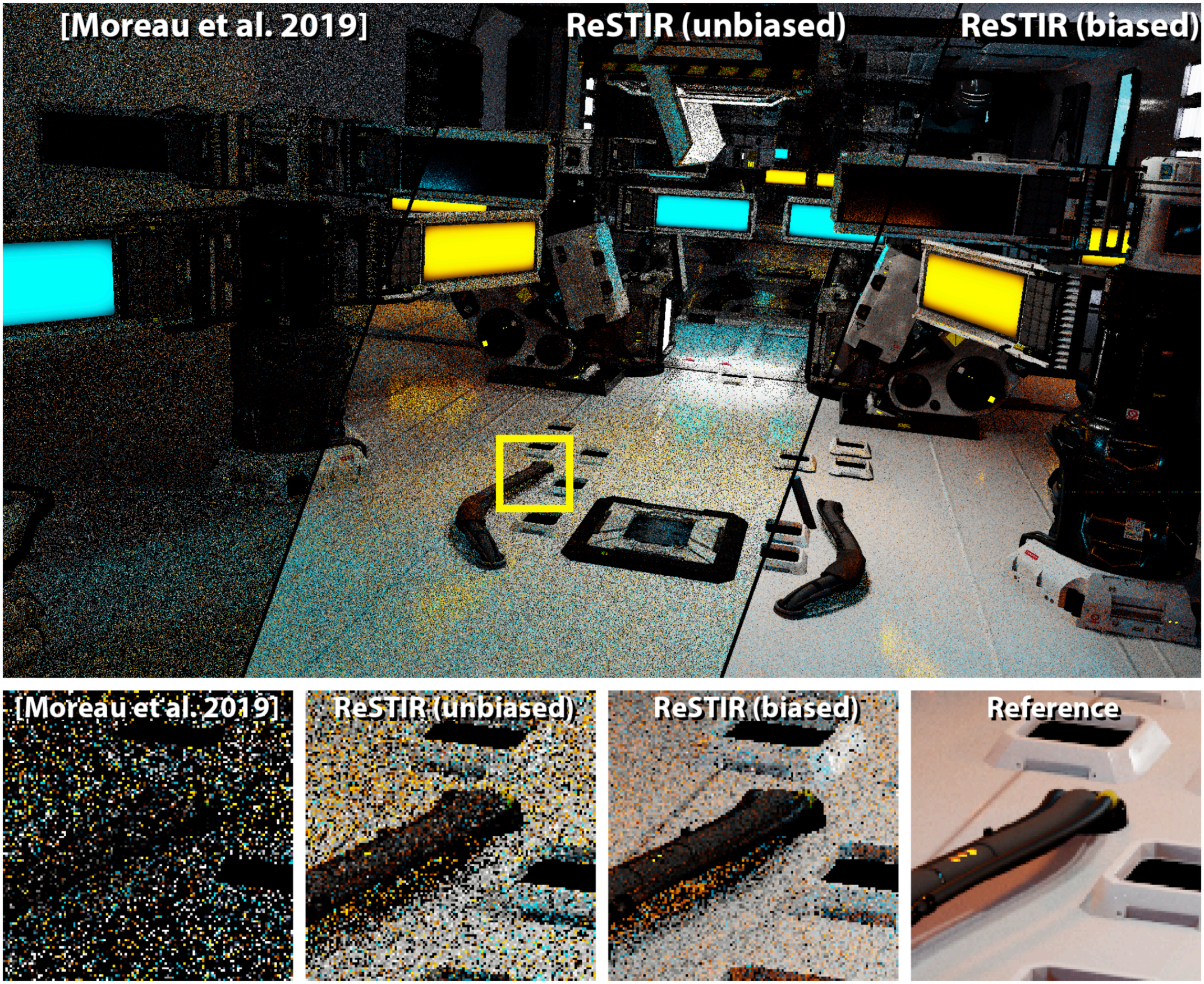


Zoom-ins: CNN denoising

64 samples | **4096 samples** | **64 samples, denoised**

ReSTIR

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


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


Boarding Area
Boarding Area
Boarding Area
Subway Station
City
Subway
T 149 - 12416 - 540-1369
T 149 - 12252 - 540-1373

Resampled Importance Sampling (RIS)

Generate M samples $x_i \sim p$

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

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

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

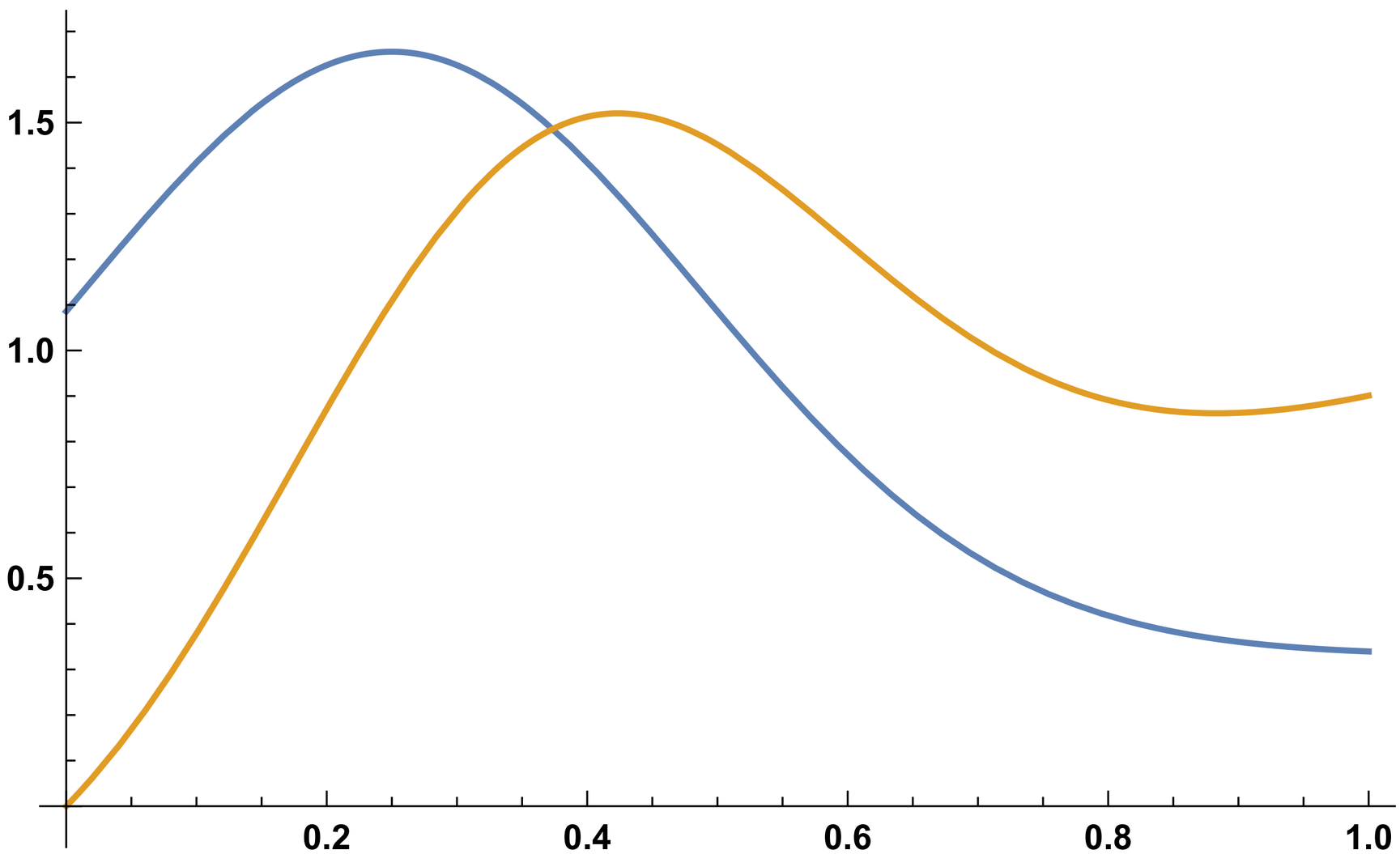
RIS for Product Sampling

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** $f = f_r\, L_e\, V$**:**

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

"Approximately"?



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

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

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**:**

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

Merging Reservoirs

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

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

Sharing Samples Across Pixels

Keep a small number of independent reservoirs at each pixel

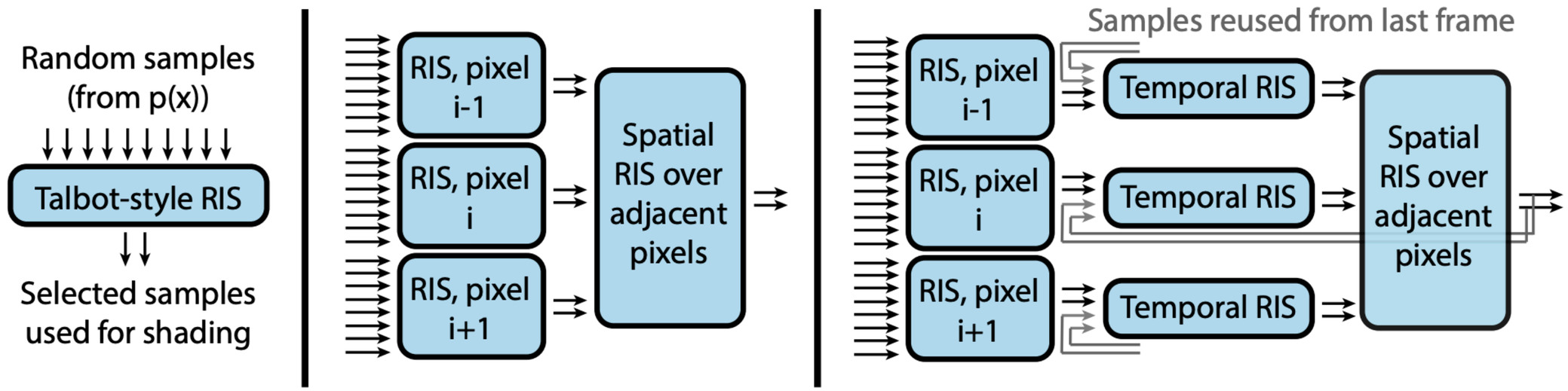
At each pixel, choose k neighbor pixels

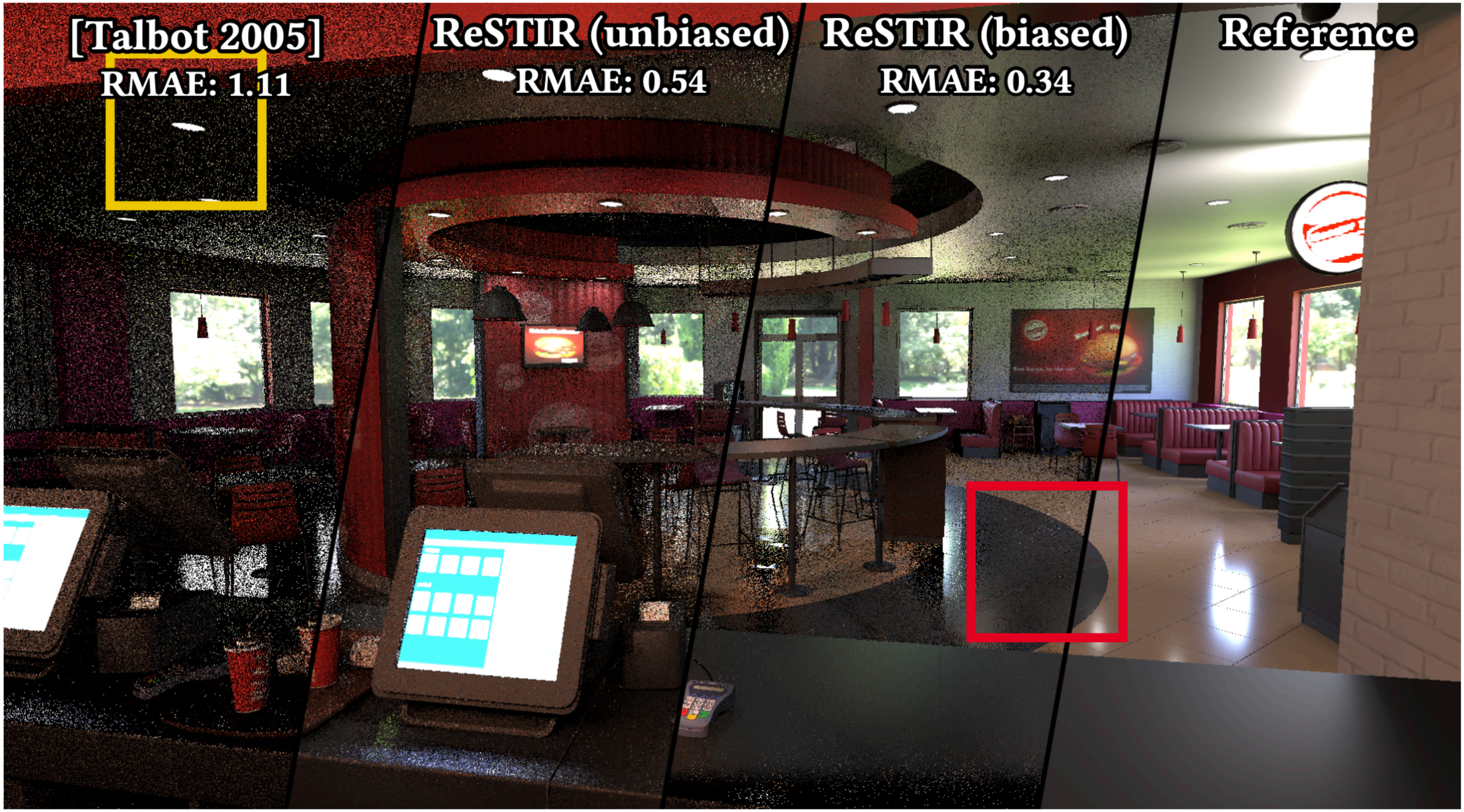
- **Merge reservoir with neighbor's reservoir**
- **Get equivalent of kM samples**

Repeat n times

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

Temporal Reuse




[Talbot 2005]
RMAE: 1.11
ReSTIR (unbiased)
RMAE: 0.54
ReSTIR (biased)
RMAE: 0.34
Reference