

Supplemental

Filtering Environment Illumination for Interactive Physically-Based Rendering in Mixed Reality

Soham Uday Mehta^{1,2} Kihwan Kim² Dawid Pająk² Kari Pulli³ Jan Kautz² Ravi Ramamoorthi⁴

¹University of California, Berkeley ²NVIDIA Corporation ³Light ⁴University of California, San Diego

1 Two-mode Sampling Algorithm

We propose a two-mode path-tracer that traces OptiX rays to intersect only virtual geometry, and screen-space rays to intersect only real geometry. Our sampling algorithm is explained in Algorithm 1. For brevity, we omit obvious function arguments. Position and normal are abbreviated to ‘pos’ and ‘n’; real and virtual quantities are separated with suffixes ‘_r’ and ‘_v’ respectively.

The main blocks in the code are explained below:

Lines 3-12: The outer loop consists of 4 samples per pixel (spp) anti-aliasing, and we determine whether a real or a virtual object is visible at the current sample and update the mask M .

Line 13: This loop computes 4 secondary samples for each of the 4 primary samples, so we compute a total of 16 spp for each of direct and indirect illumination.

Lines 14-25: For direct illumination, we importance sample the environment map, as this gives the least amount of noise for very little overhead. In line 15, an environment map importance sample is obtained, and in lines 18-19, the functions *trace_optix_sray* and *trace_ss_sray* return hit distances (-1 if no hit) for OptiX and screen-space shadow rays respectively.

Lines 26-40: For indirect illumination, we sample the cosine hemisphere for diffuse surfaces (real and virtual) and a Phong lobe for glossy surfaces (virtual only). Line 27 samples the BRDF to produce a sampling direction, and in lines 28-29, *trace_optix_iray* and *trace_ss_iray* return the indirect radiance and hit distance for OptiX and screen-space indirect rays respectively. In line 28, for screen-space rays, radiance from the secondary hit is computed by L_{cam} image look-up. In line 29, radiance from the secondary hit for OptiX rays (that intersect only virtual surfaces) is computed by tracing a secondary shadow ray to an environment map sample.

Although not shown in Algorithm 1, we also save the (average) world location, normal, virtual texture k_V . We also record the minimum hit distance for direct and indirect illumination; these are required for filtering. Since texture is multiplied with irradiance after filtering, we require the approximation $\langle k \cdot E \rangle \approx \langle k \rangle \cdot \langle E \rangle$, where $\langle \rangle$ denotes the mean of the quantity at a pixel.

This algorithm can be implemented in a single Optix pixel-shader kernel execution. However, Optix kernels are optimized only for ray intersection testing, and thread divergence reduces speed of this one-kernel approach significantly due to the screen-space ray-tracing component. Hence, we implement the algorithm in two passes. A first Optix pass traces only the Optix rays, storing the intersection results in a buffer. A second CUDA pixel shader pass traces the screen-space rays and also combines the result of screen-space and Optix ray-tracing, per lines 21-26 and 32-42 of the pseudo-code.

Algorithm 1

```
1: for each pixel do
2:   E_dir_r = E_dir_rv = E_ind_r = E_ind_rv = 0, M = 0
3:   for i = 1 : 4 do ▷ Anti-aliasing
4:     is_real = false
5:     {pos_v,n_v,depth_v} = trace_primary_ray_optix(...)
6:     {pos_r,n_r,depth_r} = real_world_map(...)
7:     if depth_v = -1 OR depth_r < depth_v then
8:       is_real = true, M += 0.25
9:       pos = pos_r, n = n_r
10:    else
11:      pos = pos_v, n = n_v
12:    end if
13:    for j = 1 : 4 do ▷ Secondary Samples
14:      // Direct Illumination
15:      sample = get_importance_sample(...)
16:      L_env = env_map[sample.xy]
17:      ray_dir = {pos, sample_to_direction(sample.xy)}
18:      hit_dist_v = trace_optix_sray(ray_dir)
19:      hit_dist_r = trace_ss_sray(ray_dir)
20:      if is_real and hit_dist_r = -1 then
21:        E_dir_r += L_env * cos_theta / sample.pdf
22:      end if
23:      if hit_dist_r = -1 and hit_dist_v = -1 then
24:        E_dir_rv += L_env * cos_theta / sample.pdf
25:      end if
26:      // Indirect Illumination
27:      ray_ind = {pos, sample_BRDF(...)}
28:      {hit_dist_v, L_ind_v} = trace_optix_iray(ray_ind)
29:      {hit_dist_r, L_ind_r} = trace_ss_iray(ray_ind)
30:      if hit_dist_r > 0 and hit_dist_v > 0 then
31:        if hit_dist_r > hit_dist_v then
32:          E_ind_rv += L_ind_v
33:        else
34:          E_ind_rv += L_ind_r
35:        end if
36:      end if
37:      // Cases where dist = -1 not shown
38:      if is_real and hit_dist_r > 0 then
39:        E_ind_r += L_ind_r
40:      end if
41:    end for
42:  end for
43:  E_{dir|ind}_{r|rv} /= 16 // normalize quantities
44: end for
```

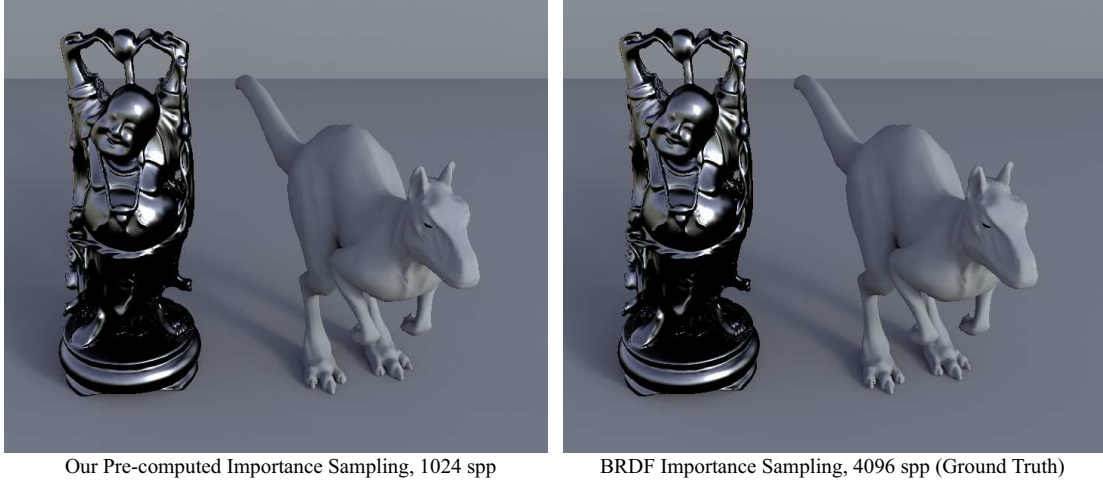


Figure 1

1.1 Comparing our Importance Sampling to Ground Truth

As explained in Sec 5.2 in the paper, we pre-compute 4096 importance samples and render the image by choosing samples from this pre-computed set. In Fig. 1, we compare a 1024 spp image rendered with the pre-computed importance samples, to a ground truth image with 4096 spp BRDF importance sampling. There is no visual difference, and the numeric (RMS, per channel) error is 10^{-5} .

2 Derivation of equation 11

Here we prove eqn. 11. Taking the 1D Fourier transform of eqn. 10 gives:

$$\begin{aligned}
 \hat{E}(\Omega_x) &= \iint L_e(\theta + \kappa x) H(\theta - \theta_{\text{occ}} + \lambda x) f(\theta) e^{-jx\Omega_x} d\theta dx \\
 &= \int \left(\iint L_e(\theta + \kappa x) H(\theta - \theta_{\text{occ}} + \lambda x) e^{-jx\Omega_x} e^{-j\theta\Omega_\theta} dx d\theta \right) \hat{f}(-\Omega_\theta) d\Omega_\theta \quad (1) \\
 &= \int \hat{G}(\Omega_x, \Omega_\theta) \hat{f}(\Omega_\theta) d\Omega_\theta
 \end{aligned}$$

In the second step, we have used the inverse Fourier transform $f(\theta) = \int \hat{f}(-\Omega_\theta) e^{-j\theta\Omega_\theta} d\theta$, and the Fourier transform of f satisfies $\hat{f}(-\Omega_\theta) = \hat{f}(\Omega_\theta)$.

Since both L_e and H are 1D functions sheared along constant slopes, their Fourier transforms are straight lines through the origin. The spectrum of the product $\hat{G}(\Omega_x, \Omega_\theta)$ is then a convolution of two lines of different slopes. We now derive \hat{G} explicitly:

$$\begin{aligned}
 \hat{G}(\Omega_x, \Omega_\theta) &= \iint L_e(\theta + \kappa x) H(\theta - \theta_{\text{occ}} + \lambda x) e^{-jx\Omega_x} e^{-j\theta\Omega_\theta} dx d\theta \\
 &= \mathcal{F}\{L_e(\theta + \kappa x)\} * \mathcal{F}\{H(\theta - \theta_{\text{occ}} + \lambda x)\} \quad (2) \\
 &= \iint \hat{L}_e(\omega_\theta) \delta(\omega_x - \kappa\omega_\theta) \\
 &\quad \hat{H}(\Omega_\theta - \omega_\theta) \delta(\Omega_x - \omega_x - \lambda(\Omega_\theta - \omega_\theta)) e^{-j\theta_{\text{occ}}\Omega_\theta} d\omega_x d\omega_\theta
 \end{aligned}$$

We use the property $\int f(x)\delta(ax-b)dx = f(b/a)/|a|$ to simplify the integral of the product of two delta functions.

$$\int \delta(\omega_x - \kappa\omega_\theta)\delta(\Omega_x - \omega_x - \lambda(\Omega_\theta - \omega_\theta))d\omega_x = \delta(\Omega_x - \kappa\omega_\theta - \lambda(\Omega_\theta - \omega_\theta)) \quad (3)$$

Substituting this into eqn. 2 and applying the delta function integral property once again, we get:

$$\begin{aligned} \hat{G}(\Omega_x, \Omega_\theta) &= e^{-j\theta_{\text{occ}}\Omega_\theta} \int \hat{L}_e(\omega_\theta)\hat{H}(\Omega_\theta - \omega_\theta)\delta(\Omega_x - \kappa\omega_\theta - \lambda(\Omega_\theta - \omega_\theta))d\omega_\theta \\ &= \frac{e^{-j\theta_{\text{occ}}\Omega_\theta}}{\lambda - \kappa} \hat{L}_e\left(\frac{-\Omega_x + \lambda\Omega_\theta}{\lambda - \kappa}\right) \hat{H}\left(\Omega_\theta - \frac{-\Omega_x + \lambda\Omega_\theta}{\lambda - \kappa}\right) \\ &= \frac{e^{-j\theta_{\text{occ}}\Omega_\theta}}{\lambda - \kappa} \hat{L}_e\left(-\frac{\Omega_x - \lambda\Omega_\theta}{\lambda - \kappa}\right) \hat{H}\left(\frac{\Omega_x - \kappa\Omega_\theta}{\lambda - \kappa}\right) \end{aligned} \quad (4)$$

3 Verification of equation 17

In the main paper, Fig. 4 verifies eqn. 13 for a particular diffuse flatland set-up. Here, we provide a similar verification for eqn. 17, the glossy case. As shown in Fig. 2 below, eqn. 17 over-estimates the true bandwidth, since we simply combine eqn. 16 and eqn. 13. However, this estimate works well for filtering glossy surfaces as shown in Fig. 6.

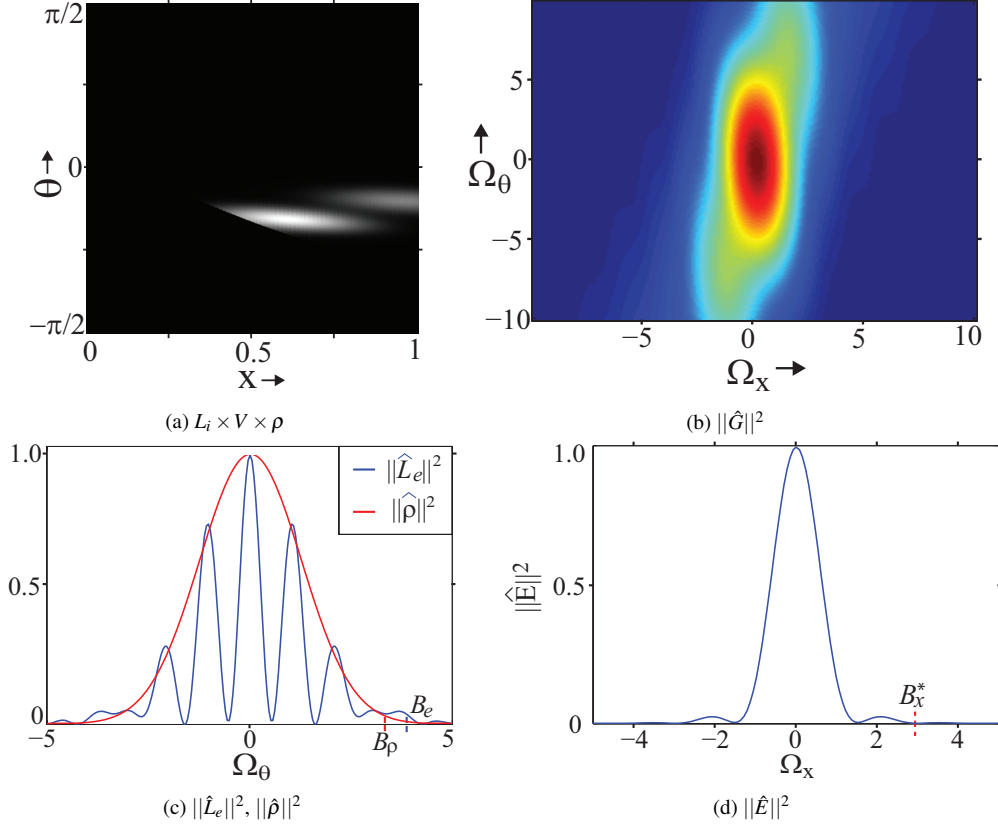


Figure 2: Verification of eqn. 17 for the simple flatland setup of Fig. 3(c) with a glossy ($n = 32$) surface with $\kappa = 0.5$ and one occluder at $\theta_{\text{occ}} = \pi/4$ and $z = 2$ ($\cos^2 \theta_{\text{occ}}/z = 0.25$), under high-frequency illumination. (a) shows the product $L_i \times V \times \rho$ for this setup. (b) shows the power spectrum $\|\hat{G}\|^2$ of (a). In (c) we show the 1D power spectra of L_e and ρ , showing bandlimits $B_e = 4$ and $B_\rho = 3.5$. (d) shows the 1D power spectrum \hat{E} of the surface irradiance, showing the true bandwidth $B_x^* \approx 3$. Eqn. 17 ($B_f = 1$) gives $B_x = 4 + 0.25 \times 12 = 7$. Our estimate is conservative but not tight.