

Adaption of the Energy Term in the Paper “Importance Sampling of Many Lights with Adaptive Tree Splitting”

October 30, 2022

1 Preliminaries

Eq. (3) in the original paper^a is an estimation of the outgoing radiance, integrated over the emitter or cluster’s all possible incoming angles. The incident radiance in Blender is computed as^b

$$L_i = \text{light_eval} = \text{eval_fac} \times \text{strength}.$$

The radiance is divided by the pdf^c, which is usually $\frac{d^2}{A \cos \theta}$, with d being the distance between the shading point and the point on the light, θ the angle between the direction of the ray from the point on the light to the shading point and the light’s surface normal, and A the area of the emitter. For simplicity, we denote `eval_fac` as F , `strength` as S .

2 The Energy (Power) Term E

Area Light For area light, $F = \frac{1}{4A}$ ^d, ignoring light spread angle. Therefore

$$\begin{aligned} L_o &= \int f_a L_i(\omega_i) \cos \theta_i \, d\omega_i \\ &= \int f_a \frac{S}{4A} \cos \theta_i \frac{\cos \theta}{d^2} \, dA \\ &\approx f_a \frac{S}{4} \cos \theta_i \frac{\cos \theta}{d^2}, \end{aligned}$$

^aConty Estevez, A., & Kulla, C. (2018). Importance sampling of many lights with adaptive tree splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2), 1-17.

^b`cycles/kernel/light/sample.h`, function `light_sample_shader_eval()`

^c`cycles/kernel/integrator/shade_surface.h`, line 218: `bsdf_eval_mul(&bsdf_eval, light_eval / ls.pdf);`

^d`cycles/kernel/light/area.h`, line 213, `ls->eval_fac = 0.25f * invarea;`

where θ_i is the incidence angle. In the original paper, Conty et al. use the upper bound as an estimation of the two cosine terms, and denote them as θ'_i, θ' , respectively. The \approx sign comes from the fact that we can only estimate θ and θ_i . Thus, $E = \frac{S}{4}$.

Triangle Light For triangle (mesh) light, $F = 1^e$, therefore

$$\begin{aligned} L_o &= \int f_a L_i(\omega_i) \cos \theta_i d\omega_i \\ &= \int f_a S \cos \theta_i \frac{\cos \theta}{d^2} dA \\ &\approx f_a S A \cos \theta_i \frac{\cos \theta}{d^2}. \end{aligned}$$

Thus, $E = SA$.

Point Light For point light, $F = \frac{1}{4\pi A}^f$, therefore

$$\begin{aligned} L_o &= \int f_a L_i(\omega_i) \cos \theta_i d\omega_i \\ &= \int f_a \frac{S}{4\pi A} \cos \theta_i \frac{\cos \theta}{d^2} dA \\ &\approx f_a \frac{S}{4\pi} \cos \theta_i \frac{\cos \theta}{d^2}. \end{aligned}$$

Thus, $E = \frac{S}{4\pi}$.

The axis of a point light always points to the shading point, but the shading point is unknown during tree construction; therefore, the original paper use arbitrary axis and set the bounding cone (θ_o) to π . During traversal, the shading point is known, we could then realign the axis to be pointing to the shading point, and set θ_o to 0. We could certainly recompute the bounding cone when computing the importance of one emitter, it remains a question whether this is also meaningful when traversing the clusters. Expensive operations would be `acos()`, potentially `rotate_around_axis()`^g.

Spot Light For spot light, $F = \frac{\cos \theta}{4\pi A}^h$, ignoring spread angle and blend. Therefore,

$$L_o \approx f_a \frac{S \cos \theta}{4\pi} \cos \theta_i \frac{\cos \theta}{d^2}.$$

Thus, $E = \frac{S \cos \theta}{4\pi}$.

^ecycles/kernel/light/triangle.h, line 152, `ls->eval_fac = 1.0f;`
^fcycles/kernel/light/point.h, line 34, `ls->eval_fac = M_1_PI_F * 0.25f * klight->spot.invarea;`
^gcycles/scene/light/tree.cpp, function `merge()`
^hcycles/kernel/light/spot.h, line 55: `ls->eval_fac = (0.25f * M_1_PI_F) * invarea;` line 58: `ls->eval_fac *= spot_light.attenuation();`

I am not sure whether the extra $\cos \theta$ term makes sense. In PBRTⁱ, the falloff only exists outside of `falloffStart`, for which we have the `blend` term. The extra $\cos \theta$ is currently ignored in the light tree.

Distant Light In blender, distant light is claimed to be equivalent to a disk area light 1 unit away from the shading point^j. However, the equivalency is questionable, as a diffuse area light has constant radiance, but the radiance of the implemented distant light depends on the distance and the orientation to the shading point. The `pdf` is also the same as `eval_fac`, as the same sampling technique as area light is used. Therefore, `light_eval / pdf` always returns the strength. Following the current implementation, $F = \frac{1}{A \cos^3 \theta}$, therefore,

$$L_o \approx f_a \frac{S}{\cos^3 \theta} \cos \theta_i \frac{\cos \theta}{d^2}.$$

Thus, $E = \frac{S}{\cos^3 \theta}$.

Background Light

$$\begin{aligned} L_o &= \int f_a L_i(\omega_i) \cos \theta_i d\omega_i \\ &= \int_0^{2\pi} \int_0^{\frac{\pi}{2}} f_a L_i(\omega_i) \cos \theta_i \sin \theta d\theta d\varphi_i \\ &= \int_0^{2\pi} 0.5 f_a L_i d\varphi_i \\ &\approx \pi f_a L_i, \end{aligned} \tag{1}$$

where L_i is the average radiance of the background. The current implementation^k simply averages all the pixel values, I think a $\sin \theta$ weight should be applied. Denoting the resolution in θ and φ directions as R_θ and R_φ , respectively, the average radiance is computed as

$$\begin{aligned} L_i &= \frac{1}{4\pi} \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} L(\theta, \varphi) \sin \theta d\theta d\varphi \\ &\approx \frac{1}{4\pi} \sum L(\theta, \varphi) \sin \theta \frac{\pi}{R_\theta} \frac{2\pi}{R_\varphi} \\ &= \frac{\pi}{2R_\theta R_\varphi} \sum L(\theta, \varphi) \sin \theta. \end{aligned}$$

When traversing the light tree, eq. (1) should be used for background light instead of eq. (3) in the original paper.

ⁱhttps://www.pbr-book.org/3ed-2018/Light_Sources/Point_Lights

^j`cycles/kernel/light/distant.h`, the illustration starting line 59

^k`cycles/scene/light.cpp`, function `average_background_energy()`

3 Result

After applying the above scaling (ignoring all $\cos \theta$ terms), the variance is reduced by a small amount. Min-max importance is not applied on mesh and area lights, so there is potential for further improvements.

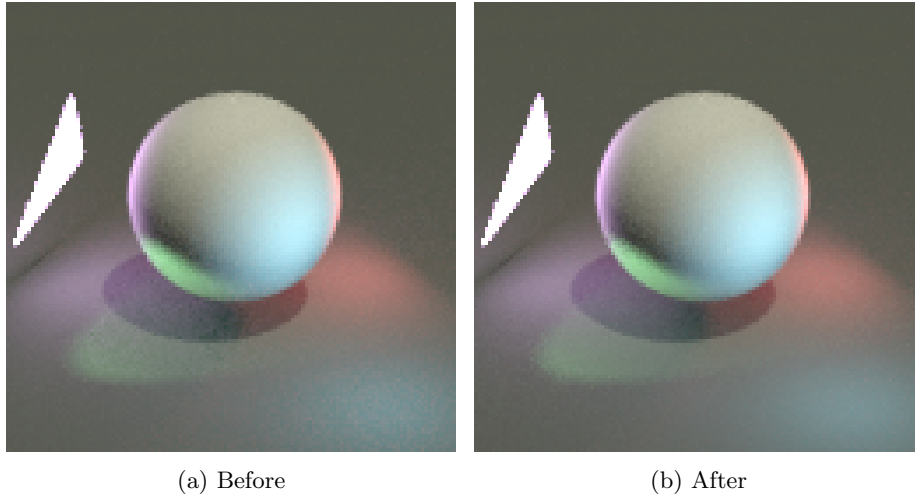


Figure 1: Comparison before and after scaling the energy