Point cloud method for rendering BSSRDFs

Alessandro Dal Corso
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

March 2018

This is a short note on rendering of a triangle mesh onto which we apply a BSSRDF model. It is a way to implement the technique referred to by Frisvad et al. [1] as direct Monte Carlo integration. The method works for arbitrary triangle meshes, and it is unbiased. The configuration of the method is illustrated in Figure 1.

We first define some quantities relative to our mesh. We consider a triangle mesh a set $M = \{T_\Delta, \Delta \in [0, K-1]\}$ composed of K triangles. Each triangle T_Δ is composed of three vertices:

$$T_\Delta = \{\mathbf{v}_0^\Delta, \mathbf{v}_1^\Delta, \mathbf{v}_2^\Delta\}\,.$$

From these quantities, it straightforward to define two derived quantities, the normal $\vec{n}_\Delta$ and the area A_Δ of the triangle.

$$\vec{n}_\Delta = \frac{(\mathbf{v}_1^\Delta - \mathbf{v}_0^\Delta) \times (\mathbf{v}_2^\Delta - \mathbf{v}_0^\Delta)}{\|(\mathbf{v}_1^\Delta - \mathbf{v}_0^\Delta) \times (\mathbf{v}_2^\Delta - \mathbf{v}_0^\Delta)\|}$$

$$A_\Delta = \frac{1}{2}\|(\mathbf{v}_1^\Delta - \mathbf{v}_0^\Delta) \times (\mathbf{v}_2^\Delta - \mathbf{v}_0^\Delta)\|\,.$$

A triangle in a triangle mesh may have a different normal for each vertex. In this case, the normal $\vec{n}$ of a particular point $\mathbf{x}$ of the triangle is given by linear interpolation based on the barycentric coordinates of $\mathbf{x}$ in T_Δ. With our triangles defined, we can start describing our solution.

Theoretically, we solve the equation of reflected radiance for BSSRDFs:

$$L_r(\mathbf{x}_o, \vec{\omega}_o) = \int_A \int_\Omega S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n}_i \cdot \vec{\omega}_i) dA d\omega_i\,,$$

where L_i and L_r are incident and reflected radiance, S is the BSSRDF, A is the surface area of the object represented by the triangle mesh, and Ω is the hemisphere around the surface normal $\vec{n}_i$ at the surface position $\mathbf{x}_i$. We now solve the integral using Monte Carlo integration with importance sampling. Taking M surface position samples for the area integral and N direction samples for the direction integral, we have

$$\begin{aligned}\hat{L}_r(\mathbf{x}_o, \vec{\omega}_o) &\approx \frac{1}{NM}\sum_{p=1}^{M}\sum_{q=1}^{N} \frac{S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})(\vec{n}_{i,p} \cdot \vec{\omega}_{i,q})}{\mathrm{pdf}(\mathbf{x}_{i,p})\mathrm{pdf}(\vec{\omega}_{i,q})} \\ &= \frac{1}{NM}\sum_{p=1}^{M}\sum_{q=1}^{N} S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o)\Phi_{i,p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})\,. \qquad (1)\end{aligned}$$

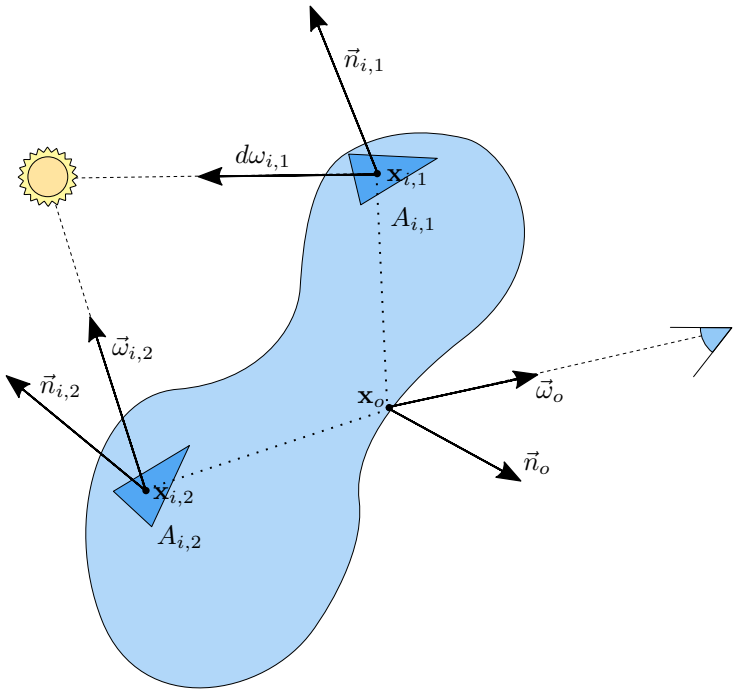

Figure 1: Diagram illustrating the configuration. Two points $\mathbf{x}_{i,p}$, $p = 1, 2$, are selected on the surface, and the irradiance stored in a buffer. The contribution is then gathered from the various points of emergence $\mathbf{x}_o$ in the outgoing direction $\vec{\omega}_o$ towards the eye.

The algorithm is composed of two phases. In the first phase, we generate the NM samples on the surface of the model, storing the incoming flux $\Phi_{i,p,q}$ for each position $\mathbf{x}_{i,p}$ and direction $\vec{\omega}_{i,q}$. In the second phase, we render the final image evaluating the contribution at each pixel.

In phase one, we produce NM samples. Sample p, q is composed of a position $\mathbf{x}_{i,p}$, a direction $\vec{\omega}_{i,q}$, and a flux $\Phi_{i,p,q}$. First, we randomly sample a triangle T_Δ. This is done by uniformly sampling a triangle index Δ, so given the continuous uniform random variable $\xi \in [0, 1)$, we have $\Delta = \lfloor \xi K \rfloor$. The triangle index could be importance sampled using A_Δ / A as the probability of sampling index Δ, but this requires a binary search, which turned out to be less efficient on a GPU. To get $\mathbf{x}_{i,p}$, we uniformly sample a point in the triangle T_Δ using barycentric coordinates:

$$\mathbf{x}_{i,p} = (1 - \sqrt{\xi_0})\mathbf{v}_0^j + (1 - \xi_1)\sqrt{\xi_0}\mathbf{v}_1^j + \xi_1\sqrt{\xi_0}\mathbf{v}_2^j\,,$$

where $\xi_0, \xi_1 \in [0, 1)$ are again continuous uniform random variables.

Now, the sampling direction $\vec{\omega}_{i,q}$ and the flux $\Phi_{i,p,q}$ need to be evaluated. We leave the choice of the sampling distribution for $\vec{\omega}_{i,q}$ to implementation, since it is not important for the method. From sampling the light source, we obtain $\vec{\omega}_{i,q}$ and an irradiance $E_{p,q}$:

$$E_{p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) = \frac{L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})(\vec{n}_{i,p} \cdot \vec{\omega}_{i,q})}{\mathrm{pdf}(\vec{\omega}_{i,q})}$$

We need the pdf for the sampling of the point of incidence $\mathrm{pdf}(\mathbf{x}_{i,p})$. Since we uniformly sampled a triangle index and then uniformly a point within the triangle, the pdf is

$$\mathrm{pdf}(\mathbf{x}_{i,p}) = \frac{1}{KA_\Delta}$$

Putting it all together:

$$\Phi_{i,p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) = \frac{E_{p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})}{\mathrm{pdf}(\mathbf{x}_{i,p})} = KA_\Delta E_{p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})\,.$$

Once we have stored this quantity in the sample, we are ready to proceed to the second and final phase. For each pixel, we ray trace a camera ray, obtaining a point $\mathbf{x}_o$ and a direction towards the camera $\vec{\omega}_o$. We consider the case of a participating medium with scattering properties σ_s, σ_a, g, relative index of refraction η, and a perfectly smooth surface so that the Fresnel equations for reflection describe the scattering at the surface.

In our path tracing implementation, we perform a Russian roulette using the Fresnel reflectance R as the probability of reflection when choosing reflection or refraction. In the case of reflection, we continue tracing the reflected ray. In the case of refraction, we calculate the final radiance as:

$$\hat{L}_o(\mathbf{x}_o, \vec{\omega}_o) = L_e(\mathbf{x}_o, \vec{\omega}_o) + L_t(\mathbf{x}_o, \vec{\omega}_o) + \hat{L}_r(\mathbf{x}_o, \vec{\omega}_o)\,,$$

where L_e is the emitted radiance from the medium, L_t is the direct transmission (or reduced intensity) term, and $\hat{L}_r$ comes from the Monte Carlo estimator (1).

To efficiently render the model and save expensive BSSRDF evaluations, we do a last optimization. We probabilistically include or reject samples at position $\mathbf{x}_{i,p}$ with $\exp(-\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_k\|)$ being the probability of inclusion, where $\sigma_{tr} = \sqrt{3\sigma_a(\sigma_a + (1-g)\sigma_s)}$ is the effective transport coefficient. So, only samples that are very close to the exit point $\mathbf{x}_o$ will actually be evaluated. In a formula,

$$\hat{L}_r(\mathbf{x}_o, \vec{\omega}_o) \approx \frac{1}{NM}\sum_{p=1}^{M}\sum_{q=1}^{N} S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o)\Phi_{i,p,q}(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) V(\xi, e^{-\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_{i,p}\|})\, e^{\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_{i,p}\|}\,,$$

where:

$$V(\xi, d) = \begin{cases} 1 & \text{if } \xi < d \\ 0 & \text{otherwise}\,. \end{cases}$$

We always use $N = 1$, and we generate a new set of M samples for each frame in a progressive path tracing. In case of RGB rendering, we use the mean of the three color components in σ_{tr} when sampling for inclusion or rejection.

References

[1] Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics*, 34(1):5:1–5:12, November 2014.