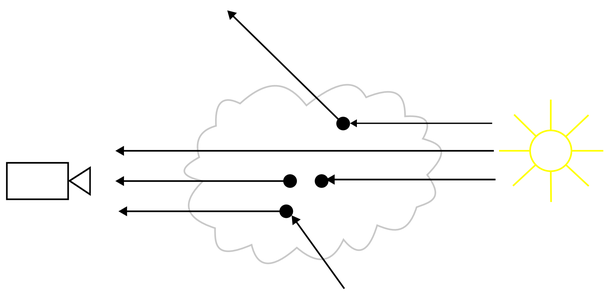
Volumetric Ray Tracing

Matthias Eberhardt
Ostbayerische Technische Hochschule Regensburg

Motivation

In Computer Graphics, objects usually are represented as a set of geometric primitives (e.g. triangles), displaying the surface of the object. However, if a volumetric object (e.g a cloud), is to be rendered, the traditional rendering technique requires an intermediate surface representation that can introduce unwanted artifacts [Lev88]. Volumetric ray tracing provides a mechanism to directly render such 3D data without the need for a surface representation.

Participating Media and the Beer-Lambert Law



Surface ray tracing assumes that the light which arrives at a point x from another point y in direction $-\omega$ experiences no interactions during its travel:

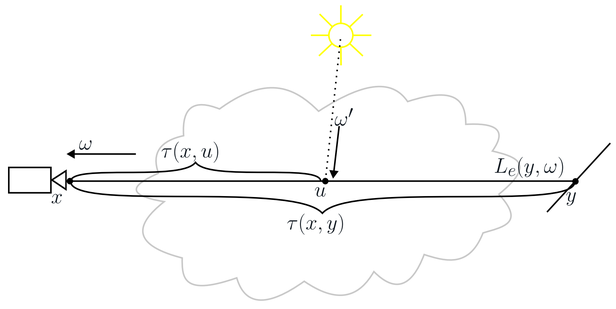
$$L(x, \omega) = L_e(y, \omega)$$

This assumption holds only true if the light travels through a vacuum, if it travels through a medium that interacts with it (called a participating medium), the intensity changes. The possible interactions (see left) include absorption (e.g. in smoke), in and out-scattering (e.g. in clouds), and emission from within the medium that adds light to the ray (e.g. in fire). The goal of volumetric ray tracing is to correctly model those interactions. The interactions enumerated above occur due to tiny particles suspended within the medium, which are stochastically modeled by the absorption coefficient $\mu_a(x)$ and scattering coefficient $\mu_s(x)$ [Max95], providing a measure for the light loss lost due to out-scattering and absorption respectively. Combining μ_a and μ_s to the extinction coefficient μ_t gives a measure for the total amount of light lost. Using the Beer-Lambert law, the attenuation or transmittance between x and y is calculated as

$$\tau(x, y) = e^{-\int_x^y \mu_t(s) ds}$$

This describes the percentage of light that "survives" the travel from x to y. Thus, L_e gets attenuated by $\tau(x, y)$.

Mathematical Model



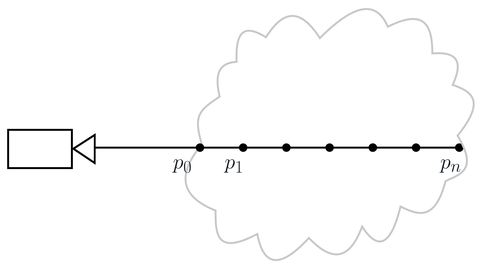
Due to in-scattering and emission, light intensity changes for every point u on the ray (see above) [Max95]. Emission is calculated as $\mu_a(u)\varepsilon(u, \omega)$, in-scattering as $\mu_s \int_\Omega f_p(u, \omega, -\omega')L(u, \omega')d\omega'$, where $\varepsilon(u, \omega)$ is the light emitted by a particle at u in direction ω, and f_p is the phase function, which describes the probability that light arriving from $-\omega'$ is scattered towards ω.
Therefore, the light send from u to x is

$$L_e^s(u, \omega) = \mu_s(u) \int_\Omega f_p(u, \omega, -\omega')L(u, \omega')d\omega' + \mu_a(u)\varepsilon(u, \omega)$$

The total added light intensity to the ray between x and y is computed by the line integral of L_e^s. Adding this to the attenuated light from y results in the rendering equation

$$L(x, \omega) = \int_x^y \tau(x, u)L_e^s(u, \omega)du + \tau(x, y)L_e(y, \omega) \quad (1)$$

Discrete Ray Marching

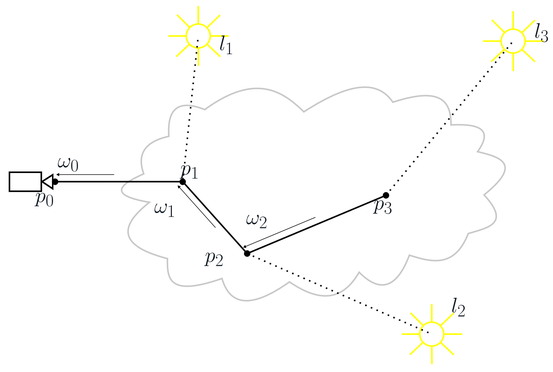


A simple approximation to equation 1 that ignores all global illumination effects is the so-called ray marching algorithm (left), which works by sampling each ray at multiple, evenly spaced points p_i and compositing those samples to a final value [Lev88]. If $c(i)$ is the color of p_i and $\alpha(i)$ its opacity, this composition is given by

$$\sum_{1 \le i \le n} \left(1 - \prod_{1 \le j \le i} (1 - \alpha(j))\right) \cdot c(i)$$

As a simplification, the values of c and α are assumed to be piecewise constant between the different p_i. An extension of the method was described by Blinn [Bli82] and further developed by Kajiya and Von Herzen [KV84], which improves the algorithm by also considering direct illumination. This works as a two-pass process: In the first step, at each voxel of the original volume, the intensity of direct illumination is calculated and saved in an intermediate 3D array. In the second step, the ray marching algorithm is used to render the output of the first step.

Monte Carlo Ray Tracing



A more sophisticated approach is the so-called monte carlo ray tracing [Hof+21]. This method works by casting a ray from the camera (p_0) in direction $-\omega_0$ for a randomly sampled distance t_0. At $p_1 = p_0 - \omega_0 t_0$ a new direction ω_1 is randomly sampled, and the process is repeated, until either a light is hit or the ray is randomly terminated. At each point p_i, next event estimation is done by sending shadow rays towards randomly sampled points on light sources (called l_i). The final result is computed by adding the contributions along the path together (while attenuating by the transmittance τ along the way). The light sources l_i and the directions ω_i are calculated using importance sampling [VG95]. For the sampling of the distances t_i delta tracking (also called woodcock tracking) is usually used [NSJ14]. To estimate the transmittance τ, several methods related to delta tracking might be used [NSJ14], e.g. ratio tracking.

References

[Bli82] James F. Blinn. "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces". In: *SIGGRAPH Comput. Graph.* 16.3 (July 1982), pp. 21–29. ISSN: 0097-8930. DOI: 10.1145/965145.801255. URL: https://doi.org/10.1145/965145.801255.

[Hof+21] Nikolai Hofmann et al. "Interactive Path Tracing and Reconstruction of Sparse Volumes". In: *Proc. ACM Comput. Graph. Interact. Tech.* 4.1 (Apr. 2021). DOI: 10.1145/3451256. URL: https://doi.org/10.1145/3451256.

[KV84] James T. Kajiya and Brian P Von Herzen. "Ray Tracing Volume Densities". In: *SIGGRAPH Comput. Graph.* 18.3 (Jan. 1984), pp. 165–174. ISSN: 0097-8930. DOI: 10.1145/964965.808594. URL: https://doi.org/10.1145/964965.808594.

[Lev88] M. Levoy. "Display of surfaces from volume data". In: *IEEE Computer Graphics and Applications* 8.3 (1988), pp. 29–37. DOI: 10.1109/38.511.

[Max95] N. Max. "Optical models for direct volume rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108. DOI: 10.1109/2945.468400.

[NSJ14] Jan Novák, Andrew Selle, and Wojciech Jarosz. "Residual Ratio Tracking for Estimating Attenuation in Participating Media". In: *ACM Trans. Graph.* 33.6 (Nov. 2014). ISSN: 0730-0301. DOI: 10.1145/2661229.2661292. URL: https://doi.org/10.1145/2661229.2661292.

[VG95] Eric Veach and Leonidas J. Guibas. "Optimally Combining Sampling Techniques for Monte Carlo Rendering". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, 1995, pp. 419–428. ISBN: 0897917014. DOI: 10.1145/218380.218498. URL: https://doi.org/10.1145/218380.218498.