Volume 2021 (4002), Number 1 pp. 1–7

Volumetric Ray Tracing

Matthias Eberhardt[1]

[1]OTH Regensburg, Germany

Abstract

1. Introduction

In Computer Graphics, objects usually are represented as a set of geometric primitives (e.g. triangles), displaying the surface of the object. However, this approach is not always suitable. For example, if the original data representation of the object is volumetric, the traditional rendering technique would necessitate the creation of an intermediate surface representation that might introduce unwanted artifacts [Lev88]. Another issue arises if the object has no well-defined surfaces to which geometric primitives could be fitted, such as a cloud or fog [KVH84]. In such cases, volumetric ray tracing might be used, a technique in which rays are cast through a volume that contains information about its optical properties(e.g color and opacity), sampled at various points within the volume, accumulated, and projected on a 2D image (see figure 1) [Lev88].

2. Derivation Of The Rendering Equation

Volumetric ray tracing follows the same principle as classical ray tracing in which an image is rendered by spanning a pixel plane in front of the camera and casting one or multiple rays through each pixel. That means for a point $\vec{x}$ on the plane we cast a ray in direction $-\omega$ and calculate the amount of light $\vec{x}$ receives from direction ω, called $L(\vec{x},\omega)$ [Gla89]. To do this, we calculate $\vec{y}$, the closest intersection point of the ray with a piece of geometry. At $\vec{y}$, some amount of light is transported from $\vec{y}$ in direction ω (either through reflection or emission), called $L_e(\vec{y},\omega)$ (The method for calculating L_e is a topic of classical raytracing [Gla89] and will not be further evaluated in this paper. In the following, we assume L_e to be known). Thus, we get the equation

$$L(\vec{x},\omega) = L_e(\vec{y},\omega) \tag{1}$$

However, this equation assumes that the light has no interactions between $\vec{x}$ and $\vec{y}$, which is only true if the ray travels through a vacuum. If it travels through a medium that interacts with it (called a participating medium), the interactions change the light along the ray [LW96] (see figure 1). The types of interactions we need to

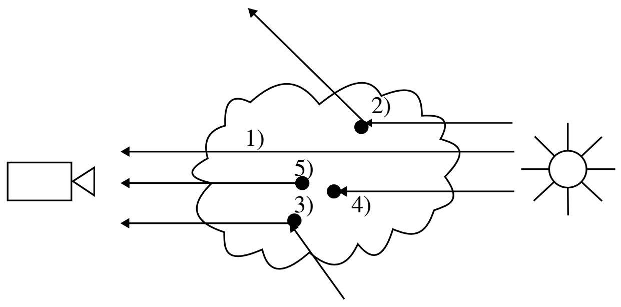

Figure 1: An overview of the possible interactions a light ray can experience in a participating medium. The light might travel through the medium unimpeded (1, e.g. in a clear atmosphere), might be out- or in-scattered (2 and 3, e.g. in clouds), or absorbed (4, e.g. in smoke). Additionally, light might be emitted from inside the medium (5, e.g. in a fire).

consider are absorption, emission, in-scattering, and out-scattering [Max95].

Absorption and out-scattering attenuate the light intensity along the ray, in-scattering and emission add to it.

2.1. Absorbtion And Out-Scattering

At first, let us consider only out-scattering and absorption, which occur due to tiny particles floating around in the volume (such as water droplets in clouds and fog). Of course, these particles are far too numerous to be directly simulated, but their distribution in 3D space can be stochastically modeled (similar to how detailed surface structures can be modeled by microfacets in surface ray tracing [CITE!]).

We follow Max [Max95] in our development of the stochastical model.

To do so, consider a close-up look at a ray section traveling

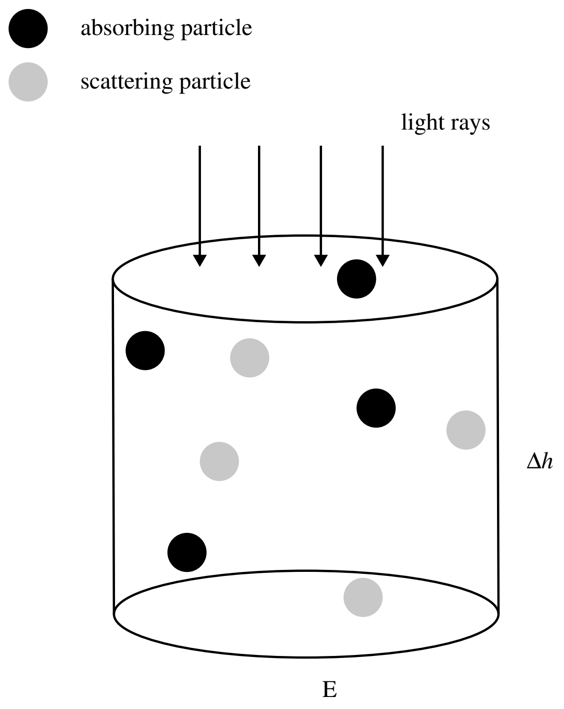

Figure 2: Image adapted from XXX.

through the volume. This section can be assumed to be a cylinder with a base area E and a height Δh (see figure 2), through which the ray travels. Within this cylinder, there exists a certain number of out-scattering and absorbing particles, defined as $n_s = \rho_s E \Delta h$ and $n_a = \rho_a E \Delta h$ respectively, where ρ_s and ρ_a are the densities of the particles. From the top-down view (see figure 3), the particles occupy an area of $n_s A = \frac{\rho_s A E \Delta h}{E}$ and $n_a A = \frac{\rho_a A E \Delta h}{E}$, respectively, if we assume that the particles do not overlap each other, which is reasonable if the densities and the height do not become too large.

This can be simplified to $\rho_s A \Delta h$ and $\rho_s A \Delta h$, which gives us the fractions of light which are stopped in the cylinder through absorption and out-scattering. By letting Δh approach 0, we see that for each infinitesimally small cylinder slice with height dh, the change in intensity is proportional to $-(\rho_s A + \rho_a A)dh$. Formulated as a differential equation, this results in

$$dL = -(\rho_s(h)A + \rho_a(h)A)L(h)dh \tag{2}$$

At this point, we define the scattering coefficient $\mu_s = \rho_s(h)A$, the absorption coefficient $\mu_a = \rho_a(h)A$ and the extinction coefficient $\mu_t = \mu_s + \mu_a$, which give a measure of how much light is lost due to scattering, to absorption, and in total. Thus, equation 2 can be simplified to

$$dL = -\mu_t(h)L(h)dh \tag{3}$$

which solution is

$$L(h) = L_0 e^{-\int_0^h \mu_t(s)ds} \tag{4}$$

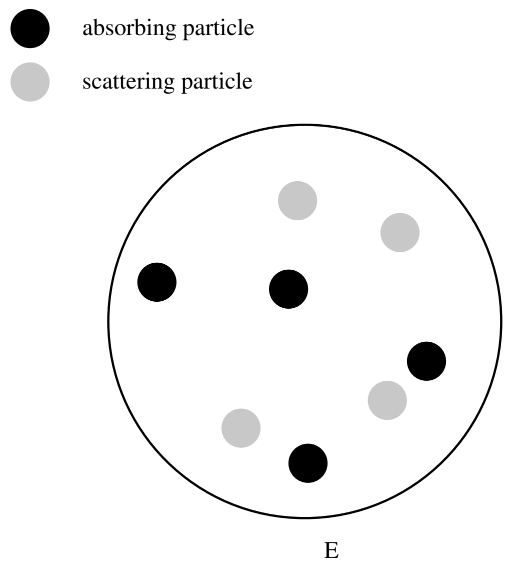

Figure 3: Top-down view of the cylinder. The cylinder has the base area E, a fraction of which is occluded by absorbing and scattering particles. Image inspired by XXX.

Rearranging this results in the equation

$$\frac{L(h)}{L_0} = e^{-\int_0^h \mu_t(s)ds} \tag{5}$$

describing the ratio of the light traveling a distance h unimpeded. In the following, we call this quantity the transmittance and refer to it as $\tau(\vec{x}, \vec{x'})$. Thus, going back to equation 1, we can now specify how much L_e gets attenuated:

$$L(\vec{x}, \omega) = \tau(\vec{x}, \vec{y}) L_e(\vec{y}, \omega) \tag{6}$$

This equation is still not accurate since we also need to consider in-scattering and emission along the ray.

2.2. Emission And In-Scattering

Consider any arbitrary point $\vec{u}$ on the ray from $\vec{x}$ to $\vec{y}$ (see figure 4). At $\vec{u}$, the particles of the medium emit some light towards ω [Max95], which we call $\varepsilon(\vec{u}, \omega)$. In the cylindrical model, the probability of finding an absorbing particle at position $\vec{u}$ is $\mu_a(\vec{u})$. We assume that the particles responsible for absorption are also responsible for emission [Max95], meaning the amount of light emitted at $\vec{u}$ towards ω is $\mu_a(\vec{u})\varepsilon(\vec{u}, \omega)$. In-scattered light can arrive at $\vec{u}$ from every direction, therefore we need to integrate over the unit sphere Ω surrounding $\vec{u}$ [JC98]:

$$\int_\Omega L(\vec{u}, \omega`) d\omega` \tag{7}$$

However, not all light arriving at $\vec{u}$ is scattered towards ω. The so called phase function $f_p(\vec{u}, \omega, -\omega`)$ describes the probality that light hitting a particle at $\vec{u}$ from $\omega`$ is reflected towards ω [JC98].

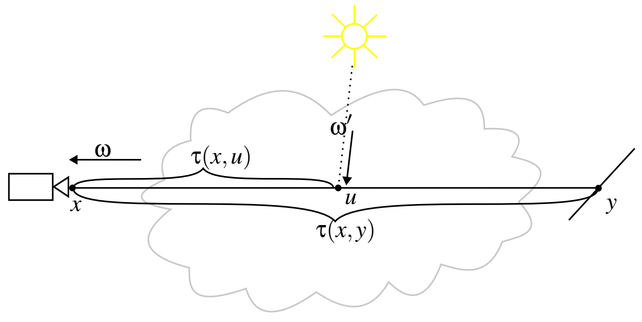

Figure 4: Simplified illustration of light transport in a participating medium. Light is transported from y to x in directionω, but is attenuated by $\tau(x,y)$. During the travel, the light ray also receives light, such as at point u, which receives some light from ω', some of which is scattered towards ω. Image adapted from Zhou et al. [ZRL*07].

In this sense, it is analogous to the BRDF in surface ray tracing. Furthermore, the amount of light scattered towards ω also depends on the probability that in-scattering particles are present at $\vec{u}$ [JC98]. We assume that in-scattering particles are the same as out-scattering particles and get the equation

$$L_e^s(\vec{u},\omega) = \mu_s(\vec{u})\int_\Omega f_p(\vec{u},\omega,-\omega`)L(\vec{u},\omega`)d\omega` + \mu_a(\vec{u})\varepsilon(\vec{u},\omega) \quad (8)$$

describing the total amount of light being added to the ray at $\vec{u}$. We call this quantity L_e^s and use the superscript s to differentiate it from L_e (which describes light reflection at solid surfaces). Not all of $L_e^s(\vec{u},\omega)$ arrives at $\vec{x}$ since the newly added light also needs to travel through the volume where it is attenuated by the transmittance $\tau(\vec{x},\vec{u})$. To get the total amount of light added to the ray, we sum up the light contribution of all points along the ray by integrating over it [ZRL*07]:

$$\int_{\vec{x}}^{\vec{y}} \tau(\vec{x},\vec{u})L_e^s(\vec{u},\omega)d\vec{u} \quad (9)$$

2.3. The Rendering Equation

Adding this to the light arriving from $\vec{y}$ gives the total amount of light arriving at $\vec{x}$ from ω.

$$L(\vec{x},\omega) = \int_{\vec{x}}^{\vec{y}} \tau(\vec{x},\vec{u})L_e^s(\vec{u},\omega)d\vec{u} + \tau(\vec{x},\vec{y})L_e(\vec{y},\omega) \quad (10)$$

This integral is difficult to solve, especially since the expression $L(\vec{x},\omega)$ appears on both sides of the equation. In the following, we describe several methods for approximating a solution.

3. Ray Marching Algorithm

In this chapter, we discuss an algorithmic solution [Lev88, DH92] to a simplified version of the rendering equation we derived in the previous section.

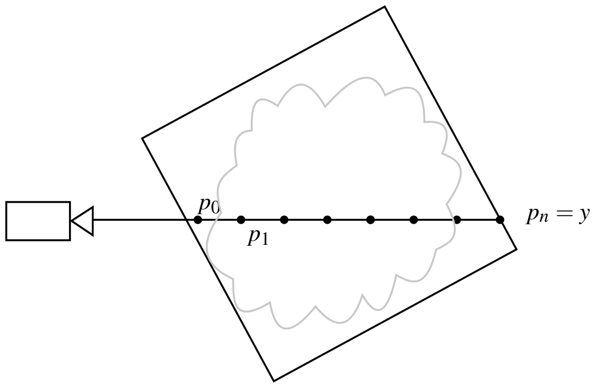

Figure 5: Illustration of the ray marching process. Image adapted from XXX(Compensated ray marching).

3.1. Simplified Rendering Equation

In the following, we ignore all scattering effects in the medium (meaning μ_s is 0) and assume that the medium only absorbs and emits light [DH92], which can be understood as a scenario where all external light has been evenly distributed within the volume. This is analogous to only rendering ambient lighting in classical ray tracing, which assumes that the light is evenly distributed in the scene. Furthermore, we also ignore all other geometry in the scene and only focus on the volume. The term $\tau(\vec{x},\vec{y})L_e(\vec{y},\omega)$, which describes the light contribution from the nearest intersection with a piece of geometry, is omitted. Thus, we arrive at the equation

$$L(\vec{x},\omega) = \int_{\vec{x}}^{\vec{y}} \tau(\vec{x},\vec{u})\mu_a(\vec{u})\varepsilon(\vec{u},\omega)d\vec{u} \quad (11)$$

For this chapter, we assume that the medium is contained within a cuboid boundary [DH92, Lev90]. Since there is no geometry, we can't cast the ray until it intersects with the geometry. Instead, the endpoint $\vec{y}$ of the ray is the point where the ray leaves the boundary(see figure 5).

3.2. Discrete Approximation To The Simplified Equation

$L(\vec{x},\omega)$ is approximated by casting a ray in direction $-\omega$ and sampling it at evenly spaced points along the ray (see figure 5) and summing up their contributions [Lev88]. The distance s between the sampling points should be smaller than the volumes Nyquist limit[CITE], otherwise important details are missed[CITE]. We start sampling at the first point within the volume, which has a distance of s_0 to $\vec{x}$. The i-th sample point then is described by

$$p_i = \vec{x} + s_0(-\omega) + si(-\omega) \quad (12)$$

The light contribution of a point p_i is considered to be the light contribution of the ray segment between p_{i-1} and p_i.

$$\int_{p_{i-1}}^{p_i} \tau(p_0,p_{i-1})\tau(p_{i-1},p_{i'})\mu_a(p_{i'})\varepsilon(p_{i'},\omega)dp_{i'} \quad (13)$$

As a simplification, we assume all variables to be piecewise constant [DH92] on the ray segments. This yields

$$\prod_{1\leq j\leq i} (\tau(p_{j-1},p_j)) \cdot \mu_a(p_i)\varepsilon(p_i,\omega)s \tag{14}$$

for one line segment between p_{i-1} and p_i. The quantity $\mu_a(p_i)\varepsilon(p_i,\omega)s$ describes the emitted light from p_i and will from now on be referred to as the color $c(i)$. The product is the total attenuation (transparency) from $\vec{x}$ to p_i, $\tau(p_{j-1},p_j)$ is the attenuation from p_{j-1} to p_j and can be calculated by

$$\tau(p_{j-1},p_j) = e^{-s\mu_t(j)} \tag{15}$$

However, in computer graphics, it is more common to use opacity instead of transparency. Therefore, we refer to $\tau(p_{j-1},p_j)$ as $1-\alpha(j)$ from now on, where $\alpha(j)$ is the opacity. Inserting this in formula for $\tau(\vec{x},p_i)$ yields $\prod_{1\leq j\leq i}(1-\alpha(j))$.

Again, we can refer to this quantity in terms of opacity rather than transparency by using the equality

$$1-\beta(i) = \prod_{1\leq j\leq i} (1-\alpha(j)) \tag{16}$$

where $\beta(i)$ is the accumulated opacity between $\vec{x}$ and p_i. Using these results, we can approximate the integral in equation 11 as the sum

$$L(\vec{x},\omega) = \sum_{1\leq i\leq n} (1-\beta(i))c(i) \tag{17}$$

which can be calculated in a single for loop [DH92]. Since $\beta(i)$ equals $\beta(i-1)(1-\alpha(i))$, it is not necessary to completely recalculate β for every step.

3.3. Underlying Volume Data

Due to our assumption that the variables μ_s, μ_a, μ_t, and ε are piecewise constant, opacity and color can easily be computed, if the volume provides such information. If the volume is already defined in terms of color and opacity α and c can be sampled directly from it. In some cases, the volume might contain other information such as density (e.g. for medical 3D scans), in which case a preprocessing step [Lev88] is necessary. If the volume is defined as a ternary function, sampling is straightforward. If the volume is defined as a 3D array of voxels, the value must be found through interpolation (usually trilinear interpolation [Lev88], but other methods such as monte carlo interpolation [HHCM21] are possible as well). When interpolating color, c must be weighted with its associated opacity [WMG98].

3.4. Direct Illumination

Until now, we have ignored scattering in the ray marching algorithm. In the following, we describe an approach to simulate single scattering (that is all light rays that are scattered only once) developed by Kajiya and Von Herzen [KVH84] based on the work of Blinn [Bli82]. This works as a two-step process. The first step is computed on a volume $\mathcal{V}$ containing opacity and albedo information and on a set of light sources $\{l_1,\dots,l_m\}$. Based on $\mathcal{V}$, one voxelized volume $\mathcal{V}'_k$ is created for every light source l_k in the following manner: For a voxel $\vec{x}$ in $\mathcal{V}'_k$ and light source l_k the amount of light $\vec{x}$ receives from l_k is calculated by attenuating the emmited light εl_k by the transmittance $\tau(\vec{x},l_k) = \int_{\vec{x}}^{l_k}(1-\alpha(\vec{x'}))d\vec{x'}$, which can be computed as described above. The albedo $a(\vec{x})$ regulates how much of the arriving light is reflected at $\vec{x}$. Thus, the color of $\vec{x}$ is

$$c_k(\vec{x}) = a(\vec{x})\tau(\vec{x},l_k)\varepsilon(l_k) \tag{18}$$

In other words, the volume $\mathcal{V}'_k$ contains information about the amount of light that is reflected from l_k at every voxel.

The second step works very similar to ray marching as described by equation 17, with the only difference being the calculation of $c(i)$. For this, the values of $c_k(p_i)$ needs to be known for all k, which can be calculated by sampling and interpolating in $\mathcal{V}'_k$. $c_k(p_i)$ is the amount of light reflected at p_i, but only a portion is reflected towards ω (the direction to the camera). Thus, by scaling all c_k by the phase function and summing up their contribution, we can define $c(i)$ as

$$c(i) = \sum_{1\leq k\leq m} f_p(\omega,\omega_k,p_i)c_k(p_i) \tag{19}$$

with ω_k being the direction from p_i to l_k (ω_k is different for every l_k, which is why the different c_k have to be stored seperately).

This approach could be generalized to calculate multi-scattering (that is light that needs two or more scattering events to reach the camera), but this would very quickly become too costly. Therefore, when rendering images with global illumination, more sophisticated algorithms are necessary[REFORMULATE! + SOURCES]. [-expensive, mention optimization algorithms, transition to mcrt, mention bias]

4. Monte Carlo Ray Tracing and Global Illumination

A common technique in classical ray tracing to generate unbiased images with global illumination is the so-called monte carlo ray tracing (MCRT) approach which solves the rendering equation by stochastically sampling it (as opposed to uniform sampling used by the ray marching algorithm described above).

In this chapter, we describe a typical MCRT, explain certain techniques necessary for its formulation, and describe further improvements to the MCRT.

4.1. General Monte Carlo Algorithm

We describe a typical MCRT such as that used by Hofman et al. [CITE!]. The algorithm starts like[simiilar to?] the already mentioned by casting a ray from the camera p_0 in direction $-\omega_0$. Then, a distance d_0 is sampled stochastically and so-called path vertex p_1 is created. From p_1, a new direction ω_1 is sampled. This process is repeated until a light source is hit or until the path gets terminated(usually by Russian roulette). Furthermore, at every path vertex p_i, next event estimation is done by casting a shadow ray towards a randomly sampled point on a light source, called l_i.[ADD: Then, the light contributions along the path are properly weighted and summed up to compute the final result.] An illustration of the process can be seen in figure 6.

For the algorithm, the following quantities need to be known: The transmittance τ between the various path vertices and light

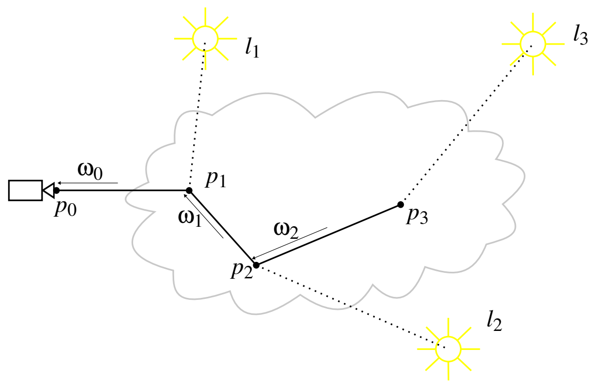

Figure 6: Image adapted from XXX(Compensated ray marching).

sources, the directions ω_i, and the locations of l_i and p_i. The sampling of ω_i and l_i works like in classical ray tracing, using multiple importance sampling[CITE] of the phase function and light distribution. Sampling p_i and estimating the transmittance require novel approaches not used in normal ray tracing, two of which are described below.

4.2. Delta Tracking

To calculate $p_{i+1} = p_i - \omega t_t$ from a given p_i and ω_i, the distance t_i between those two points must be sampled. Recall that $\tau(p_i, p_{i+1})$ describes the ratio of light arriving from p_{i+1} at p_i without colliding with a particle. Therefore, the probability that a collision occurs between p_i and p_{i+1} is

$$1 - \tau(p_i, p_{i+1}) \tag{20}$$

This can be used to importance sample the distance t_i by inverting equation 20 and applying the resulting function to ξ (a random variable uniformly sampled from $[0,1]$). However, τ is defined as $e^{\int_0^{t_i} \mu_t(s)ds}$, which can't be easily inverted if the medium is heterogenous (if μ_t is not constant). Delta tracking (also known as woodcock tracking) solves this problem by introducing so-called fictitious particles, which neither scatter nor absorb light rays. Their distribution μ_f is chosen so that the sum of μ_t and μ_f add up to a globally constant value, the so-called majorant $\bar{\mu}$. The original problem of finding a real collision is then reformulated to finding any collision (fictitious or real). Since $\bar{\mu}$ is a constant, equation 20 can easily be inverted and we get

$$t_i^1 = -\frac{ln(1-\xi)}{\bar{\mu}} \tag{21}$$

as the sampled distance to the next collision, which might be with a real or a fictitious particle (a so-called null collision). At position $p_i^1 = p_i - \omega_i t_i^1$, the probability of hitting a real particle is

$$p_{real\,collision} = \frac{\mu_t(p_i^1)}{\bar{\mu}} \tag{22}$$

Thus, we can decide if a real particle has been hit by sampling another ξ and comparing it to $p_{real\,collision}$. If ξ is smaller, a real particle has been hit, if not, there is a null collision. In this case, we repeat the process and continue to sample distances t_i^j (illustrated in figure 7) from the point p_i^{j-1} until a real collision at p_i^n is detected. The distance to the real collision is then the sum of all distances previously computed:

$$t_i = \sum_{1 \le j \le n} t_i^j \tag{23}$$

The position of the real collision is then assumed to be the path vertex p_{i+1}.

Delta tracking can also be used to estimate the transmittance τ between two points p' and $p + \omega t$. This is done by using delta tracking to find a real collision on the ray from p to p'. If this collision occurs before p', τ is assumed to be 0, if the collision occurs after p', τ is estimated as 1 (see figure 8). However, there exist other, more sophisticated methods for estimating transmittance, such as ratio tracking.

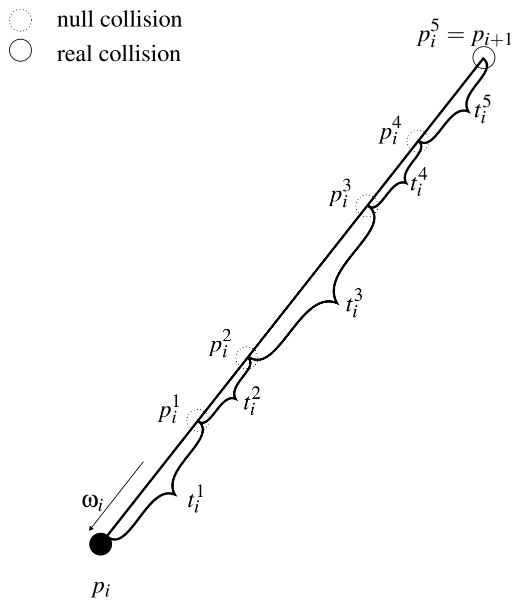

Figure 7: Delta tracking follows the ray in direction $-\omega_i$ and samples collision points. The first real collision is the position of the next path vertex. Image adapted from XXX(Compensated ray marching).

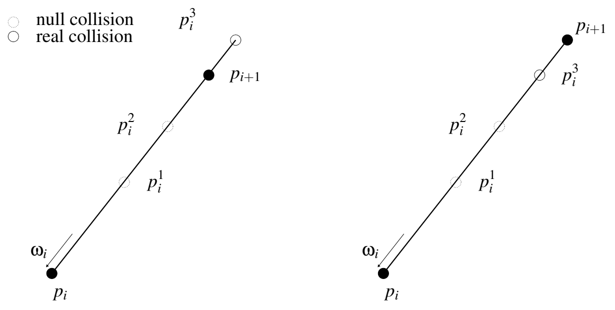

Figure 8: A figure that contains two subfigures

4.3. Ratio Tracking

Ratio tracking is used to estimate the transmittance between two already known points p and p'. This is done by calculating various collision points between p and p', as described above, and stops once a collision point behind p' is reached. At each collision point p^i, the value $\frac{\mu_f}{\bar{\mu}} = 1 - \frac{\mu_t}{\bar{\mu}}$ is saved. This fraction between fictitious particles and all particles describes the probability, that a light ray can pass p^i without real collision. The transmittance τ can then be estimated as

$$\tau(p,p') = \prod_{1 \le i \le n} (1 - \frac{\mu_t(p^i)}{\bar{\mu}}) \tag{24}$$

where n is the index of the last collision found before p'.

5. Selection of Various Monte Carlo Ray Tracing Algorithms

In this chapter, we present a brief overview of several modified MCRT algorithms alongside their strengths. For more details, we refer the reader to the original sources.

5.1. Metropolis Light transport

Metropolis light transport [VG70] works[REPLACE] by creating multiple paths from one initial path $\mathcal{P}_0$ consisting of several path vertices, by consecutively mutating path $\mathcal{P}_i$ to get $\mathcal{P}_{i+1}$. The mutations (such as adding or deleting vertices) might be rejected, if the new path does not contribute much light to the final image [VG70]. An extension to this algorithm for handling participating media has been proposed by Pauly et al. [PKK00].

5.2. Bidirectional Path Tracing

Bidirectional path tracing is a technique originally developed for surface rendering [LW98]. The algorithm works by not only tracing paths starting from the camera but also starting from the light (see figure 9). Each vertex of the camera path then sends shadow rays to each vertex on the light path. An extension of the algorithm for rendering scenes with participating media has been presented by Lafortune and Willems [LW96].

5.3. Photon Mapping

Photon mapping is a two pass process [JC98], conceptionally similar to the approach described in chapter 3.4. Originally developed for surface rendering [Jen96], its first step is the computation of two photon maps [JC98] (one for lower frequency global lighting, and one for light that travled on paths that were reflected by specular surfaces). In the second pass, the image is renderd by tracing paths from the camera into the scene and querying the photon maps for nearby photons. Jensen and Christensen [JC98] extended this method to participating media by introducing volumetric photon maps (which differ from the volumes $\mathcal{V}_k'$ described in 3.4, as they are not discrete and consider global illumination). These photon maps are computed by tracing rays from the light sources and storing the positions of interactions of those rays with the participating medium.

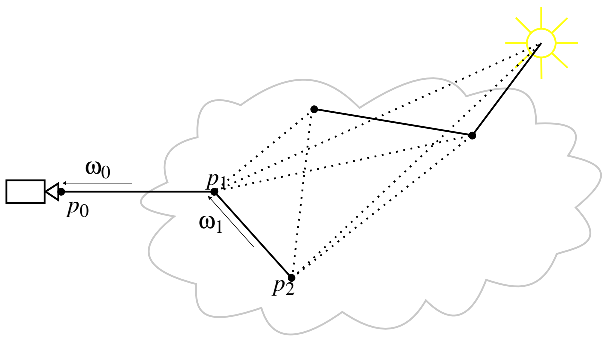

Figure 9: Illustration of bidirectional path tracing. Image adapted from Lafortune and Willems [LW96].

5.4. Compensated Ray Marching

6. Conclusion

References

[Bli82] Blinn J. F.: Light reflection functions for simulation of clouds and dusty surfaces. *SIGGRAPH Comput. Graph. 16*, 3 (jul 1982), 21–29. URL: https://doi.org/10.1145/965145.801255, doi:10.1145/965145.801255. 4

[DH92] Danskin J., Hanrahan P.: Fast algorithms for volume ray tracing. In *Proceedings of the 1992 Workshop on Volume Visualization* (New York, NY, USA, 1992), VVS '92, Association for Computing Machinery, p. 91–98. URL: https://doi.org/10.1145/147130.147155, doi:10.1145/147130.147155. 3, 4

[Gla89] Glassner A. S. (Ed.): *An Introduction to Ray Tracing*. Academic Press Ltd., GBR, 1989. 1

[HHCM21] Hofmann N., Hasselgren J., Clarberg P., Munkberg J.: Interactive path tracing and reconstruction of sparse volumes. *Proc. ACM Comput. Graph. Interact. Tech. 4*, 1 (apr 2021). URL: https://doi.org/10.1145/3451256, doi:10.1145/3451256. 4

[JC98] Jensen H. W., Christensen P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, Association for Computing Machinery, p. 311–320. URL: https://doi.org/10.1145/280814.280925, doi:10.1145/280814.280925. 2, 3, 6

[Jen96] Jensen H. W.: Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (Berlin, Heidelberg, 1996), Springer-Verlag, p. 21–30. 6

[KVH84] Kajiya J. T., Von Herzen B. P.: Ray tracing volume densities. *SIGGRAPH Comput. Graph. 18*, 3 (jan 1984), 165–174. URL: https://doi.org/10.1145/964965.808594, doi:10.1145/964965.808594. 1, 4

[Lev88] Levoy M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (1988), 29–37. doi:10.1109/38.511. 1, 3, 4

[Lev90] Levoy M.: Efficient ray tracing of volume data. *ACM Trans. Graph. 9*, 3 (jul 1990), 245–261. URL: https://doi.org/10.1145/78964.78965, doi:10.1145/78964.78965. 3

[LW96] Lafortune E. P., Willems Y. D.: Rendering participating media with bidirectional path tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (Berlin, Heidelberg, 1996), Springer-Verlag, p. 91–100. 1, 6

[LW98] LAFORTUNE E., WILLEMS Y.: Bi-directional path tracing. *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics' 93* (01 1998). 6

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108. doi:10.1109/2945.468400. 1, 2

[PKK00] PAULY M., KOLLIG T., KELLER A.: Metropolis light transport for participating media. In *Rendering Techniques 2000* (Vienna, 2000), Péroche B., Rushmeier H., (Eds.), Springer Vienna, pp. 11–22. 6

[VG70] VEACH E., GUIBAS L.: Metropolis light transport. *Computer Graphics (SIGGRAPH '97 Proceedings) 31* (02 1970). doi:10.1145/258734.258775. 6

[WMG98] WITTENBRINK C., MALZBENDER T., GOSS M.: Opacity-weighted color interpolation for volume sampling. In *IEEE Symposium on Volume Visualization (Cat. No.989EX300)* (1998), pp. 135–142. doi:10.1109/SVV.1998.729595. 4

[ZRL*07] ZHOU K., REN Z., LIN S., BAO H., GUO B., SHUM H.: *Real-Time Smoke Rendering Using Compensated Ray Marching*. Tech. Rep. MSR-TR-2007-142, September 2007. URL: https://www.microsoft.com/en-us/research/publication/real-time-smoke-rendering-using-compensated-ray-marching/. 3