Volumetric Ray Tracing

Matthias Eberhardt[1]

[1]Ostbayerische Technische Hoschschule Regensburg, Germany

Abstract

Volumetric ray tracing comprises different techniques to visualize volumetric data by sending rays within the volume and sampling it at various points. In this paper, we aim to introduce the reader to the mathematical theory of volumetric ray tracing, as well as to provide a detailed description of three basic algorithms that are used to render ambient, direct and global lighting respectively. Furthermore, we also provide a short description of some more speciliazed monte carlo ray tracer. At last, we provide the reader with a summary and references to related topics.

1. Introduction

In Computer Graphics, objects usually are represented as a set of geometric primitives [KB04] (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 5) [Lev88]. We begin our paper by introducing the reader to the mathematical formalisms of volumetric ray tracing in chapter 2. In chapter 3, we make certain assumptions about the mathematical model that allow us to describe simple volume rendering algorithms by foregoing global illumination. Afterwards, we introduce a monte carlo ray tracer, capable of rendering global illlumation, as well as mathematical methods necessary for its formulation. We then present a selection of specialized variants of the monte carlo ray tracer in chapter 5. In the final chapter 6, we draw a comparison and provide the reader with references to further work.

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 [Gla89]. That means for a point x on the plane we cast a ray in direction $-\omega$ and calculate the amount of light x receives from direction ω, called $L(x,\omega)$. To do this, we find the closest intersection point of the ray with a piece of geometry, y, from where a certain light intensity $L_e(y,\omega)$ is transported in direction ω, either through

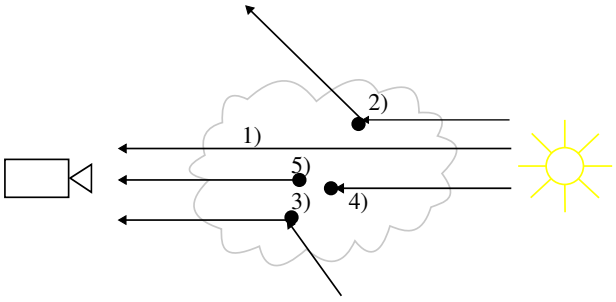

Figure 1: An overview of the possible interactions a light ray can experience in a participating medium. The light might travel through the medium unimpeded (1, e.g. in a clear atmosphere), might be out or in-scattered (2 and 3, e.g. in clouds), or absorbed (4, e.g. in smoke). Additionally, light might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

reflection or emission (The method for calculating L_e is a topic of classical raytracing [Gla89] and will not be further evaluated in this paper, and we assume L_e to be known). Thus, we get the equation

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

However, this equation assumes that the light has no interactions between x and 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 consider are absorption, emission, in-scattering, and out-scattering [Max95].

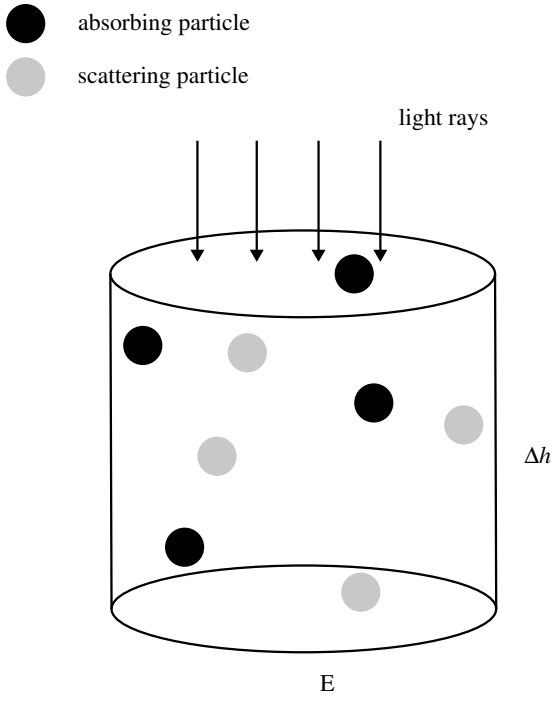

Figure 2: Image adapted from Max [Max95].

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

2.1. Absorption and Out-Scattering

At first, let us consider only out-scattering and absorption, which (like in-scattering and emission) occur due to tiny particles suspended in the volume [NGD*06] (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 [NGD*06] (similar to how detailed surface structures can be modeled by microfacets [Bli77] in surface ray tracing).

We closely follow Max [Max95] in the derivation of the stochastical model: Consider a close-up look at a ray section traveling through the volume. This section is assumed as a cylinder with a base area E and a height Δh (see figure 2). Within it, 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 simplifies to $\rho_a A \Delta h$ and $\rho_s A \Delta h$, giving us the fractions of light 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

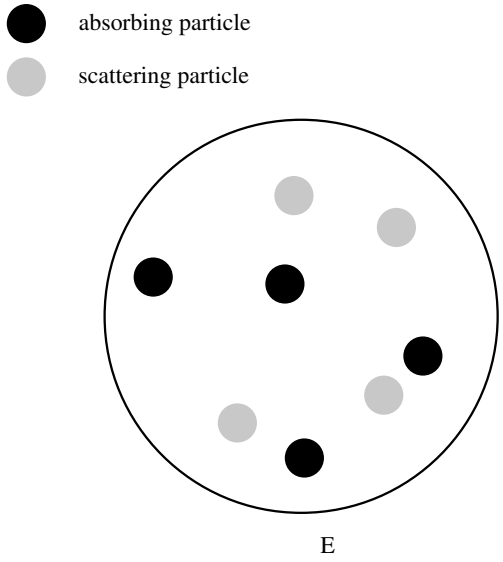

Figure 3: Top-down view of the cylinder. A fraction of the base area E, is occluded by absorbing and scattering particles. Image inspired by Max [Max95].

to $-(\rho_s A + \rho_a A)dh$. Formulated as a differential equation, this results in

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

As a shorthand, 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 simplifies to

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

which solution is

$$L(h) = L(0)e^{-\int_0^h \mu_t(s)ds} \tag{4}$$

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, the so-called transmittance τ . Thus, we can now specify how much L_e gets attenuated:

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

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

2.2. Emission and In-Scattering

Consider any arbitrary point u on the ray from x to y (see figure 4). At u, the particles of the medium emit some light towards

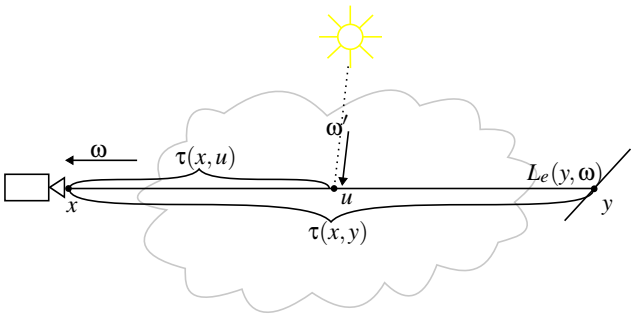

Figure 4: Simplified illustration of light transport in a participating medium. Light is transported from y to x in direction ω, but is attenuated by $\tau(x,y)$. During the travel, the light ray also receives light, such as at point u, which receives some light from ω', some of which is scattered towards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

ω [Max95], which we call $\varepsilon(u,\omega)$. In the cylindrical model, the probability of finding an absorbing particle at position u is $\mu_a(u)$. We assume that the particles responsible for absorption are also responsible for emission [Max95], meaning the amount of light emitted at u towards ω is $\mu_a(u)\varepsilon(u,\omega)$.

In-scattered light can arrive at u from every direction, therefore we need to integrate over the unit sphere Ω surrounding u [JC98]:

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

However, not all light arriving at u is scattered towards ω. The phase function $f_p(u,\omega,-\omega`)$ describes the probality that light hitting a particle at $u$ from $\omega`$ is reflected towards ω [JC98, CPP*05], analogous to the BRDF [Bli77] in surface ray tracing. Furthermore, the amount of light scattered towards ω also depends on the probability that in-scattering particles are present at u [JC98]. Assuming that in and out-scattering particles are the same results in the equation

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

describing the total amount of light being added to the ray at u. Not all of $L_e^s(u,\omega)$ arrives at x since the newly added light also needs to travel through the volume where it is attenuated by the transmittance $\tau(x,u)$. To get the total light intensity added to the ray, we sum up the light contribution of all points along the ray by integrating over it [ZRL*07]:

$$\int_x^y \tau(x,u)L_e^s(u,\omega)du \tag{9}$$

2.3. The Rendering Equation

Adding this to the light arriving from y gives the total amount of light arriving at x from ω.

$$L(x,\omega) = \int_x^y \tau(x,u)L_e^s(u,\omega)du + \tau(x,y)L_e(y,\omega) \tag{10}$$

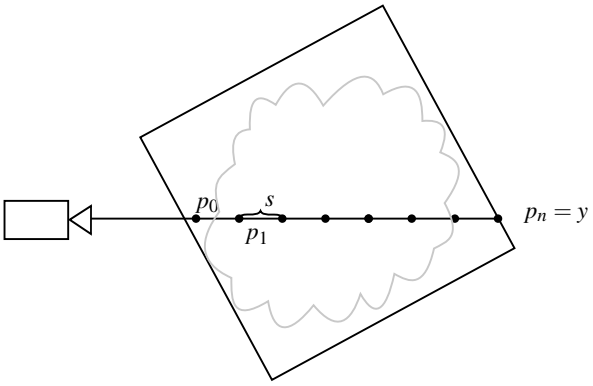

Figure 5: Illustration of the ray marching process. Image adapted from Levoy [Lev90].

This integral is analytically unsolvable in the general case [KVH84]. 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.

3.1. Simplified Rendering Equation

In the following, we ignore all scattering effects in the medium 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, 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 $\tau(x,y)L_e(y,\omega)$ therefore is omitted. Thus, we arrive at the equation

$$L(x,\omega) = \int_x^y \tau(x,u)\mu_a(u)\varepsilon(u,\omega)du \tag{11}$$

For this chapter, we assume that the medium is contained within a cuboid boundary [DH92, Lev90]. Since there is no geometry, the endpoint y is located where the ray leaves the boundary (see figure 5).

3.2. Discrete Approximation to the Simplified Equation

$L(x,\omega)$ 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 [Sha98], otherwise details are missed [NSJ14]. Sampling starts at

the first point within the volume, which has a distance of s_0 to x. The i-th sample point is described by

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

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

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

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

$$\tau(p_0, p_i)\mu_a(p_i)\varepsilon(p_i, \omega)s \tag{14}$$

for one line segment between p_{i-1} and p_i with $\tau(p_0, p_i)$.

$$\tau(p_0, p_i) = \prod_{1 \le j \le i} (\tau(p_{j-1}, p_j) \tag{15}$$

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 $c(i)$. $\tau(p_0, p_i)$ is the total attenuation (transparency) from p_0 to p_i, $\tau(p_{j-1}, p_j)$ is the attenuation from p_{j-1} to p_j and can be calculated under the assumption μ_t is constant between p_{j-1} and p_j by applying equation 5:

$$\tau(p_{j-1}, p_j) = e^{-s\mu_t(p_j)} \tag{16}$$

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 [DH92]. Inserting this in formula for $\tau(x, p_i)$ yields

$$\prod_{1 \le j \le i} (1 - \alpha(j)) \tag{17}$$

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

$$1 - \beta(i) = \prod_{1 \le j \le i} (1 - \alpha(j)) \tag{18}$$

where $\beta(i)$ is the accumulated opacity [DH92] between x and p_i. Using these results, we approximate the integral in equation 11 as the sum

$$L(x, \omega) = \sum_{1 \le i \le n} (1 - \beta(i))c(i) \tag{19}$$

which can be calculated in a single for loop [DH92].

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]). 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 (all light rays 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 $\{l_1, \ldots, l_m\}$. Based on $\mathcal{V}$, one discrete volume $\mathcal{V}_k'$ is created for every light source l_k in the following manner: For a voxel x in $\mathcal{V}_k'$ and light source l_k the amount of light x receives from l_k is calculated by attenuating the emmited light $\varepsilon(l_k)$ by the transmittance $\tau(x, l_k) = \int_x^{l_k}(1 - \alpha(x'))dx'$, which can be computed as described above in equation 17. The albedo $a(x)$ regulates how much of the arriving light is reflected at x. Thus, the color of x is

$$c_k(x) = a(x)\tau(x, l_k)\varepsilon(l_k) \tag{20}$$

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

The second step works very similar to ray marching as described by equation 19, 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 light intensity reflected at p_i, but only a portion is reflected towards ω. By scaling all c_k by the phase function and summing up their contribution, we define $c(i)$ as

$$c(i) = \sum_{1 \le k \le m} f_p(\omega, -\omega_k, p_i)c_k(p_i) \tag{21}$$

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

This algorithm is an improvment to the one described in chapter 3.2, but still suffers from problems such as increased computational complexity, inability to render volumes with high albedo [KVH84] and its bias [HHCM21]. While certain optimizations with respect to computational speed have been developed [Lev90, DH92], rendering photrealistic images requires more sophisticated techniques.

4. Monte Carlo Ray Tracing and Global Illumination

A common technique in classical ray tracing to generate unbiased images with global illumination so-called monte carlo ray tracing (MCRT) approach [JLA*03] which estimates the rendering equation by stochastically sampling it.

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

4.1. General Monte Carlo Algorithm

We describe a typical MCRT such as that used by Hofman et al. [HHCM21]. The algorithm starts by casting a ray from the camera p_0 in direction $-\omega_0$. Then, a distance d_0 is sampled stochastically

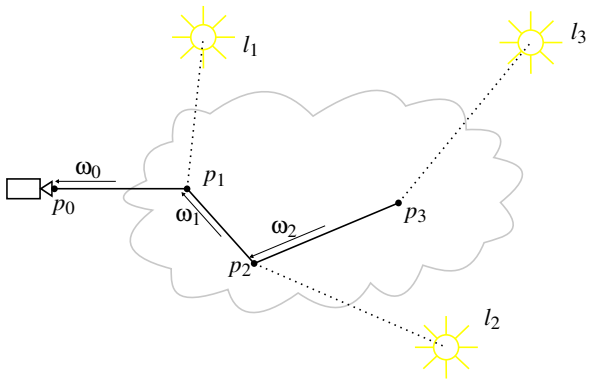

Figure 6: Illustration of monte carlo path tracing. Image inspired by Galtier at al. [GBC*13] and Lafortune and Willems [LW98]

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 [Kah55]). Furthermore, at every path vertex p_i, a so-called next event estimation [FHH*19] is done by casting a shadow ray towards a randomly sampled point on a light source l_i. Then, the light contributions along the path are properly weighted [VG95] 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 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 [VG95] 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 $p_{i+1} = p_i - \omega t_i$ 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 density [Wik23] that a collision occurs between p_i and p_{i+1} is

$$1 - \tau(p_i, p_{i+1}) \tag{22}$$

which is also the probability density of t_i. This can be used to importance sample [CC] the distance t_i by inverting equation 22 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. Delta tracking (also known as woodcock tracking) solves this problem by introducing so-called fictitious particles, which neither scatter nor absorb light rays [NSJ14]. 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 homogenous, equation 22 can easily be inverted [MK15] and we

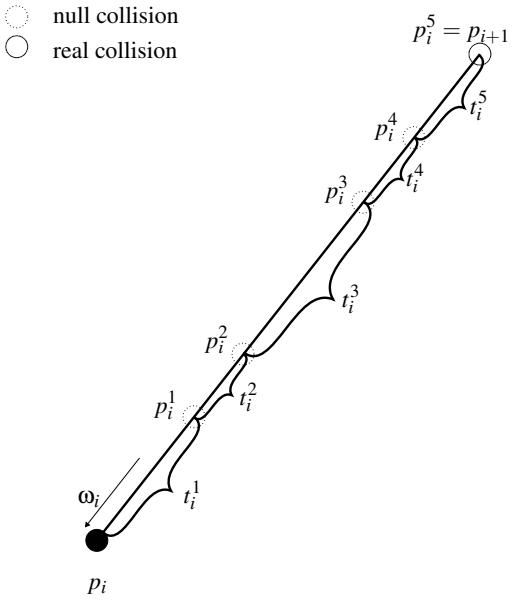

Figure 7: Delta tracking follows the ray in direction $-\omega_i$ and samples collision points. The first real collision is the position of the next path vertex. Image adapted from Galtier et al. [GBC*13].

get

$$t_i^1 = -\frac{ln(1-\xi)}{\bar{\mu}} \tag{23}$$

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

$$\mathbb{P}(\text{“}real\ collision\text{“}) = \frac{\mu_t(p_i^1)}{\bar{\mu}} \tag{24}$$

We decide if a real particle has been hit by sampling another ξ and comparing it to $\mathbb{P}(\text{“}real\ collision\text{“})$. If ξ is smaller, a real particle has been hit, if not, there is a null collision [NSJ14]. 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. Its distance to p_i is then the sum of all distances previously computed:

$$t_i = \sum_{1 \le j \le n} t_i^j \tag{25}$$

The real collision is then selected as the path vertex p_{i+1}.

4.3. Ratio Tracking

Ratio tracking [NSJ14] 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 found. At each collision point p^i, the value $1 - \frac{\mu_t}{(}p^i)\bar{\mu}$ is saved. This fraction between fictitious particles and all particles describes the probability, that a

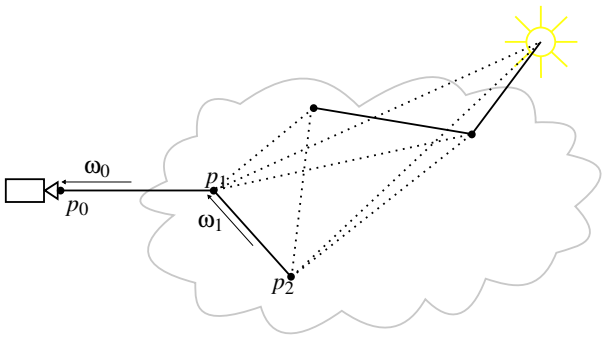

Figure 8: Illustration of bidirectional path tracing. Image adapted from Lafortune and Willems [LW96].

light ray can pass p^i without real collision [NSJ14]. The transmittance τ can then be estimated as

$$\tau(p,p') = \prod_{1\leq i\leq n} (1 - \frac{\mu_t(p^i)}{\bar{\mu}}) \tag{26}$$

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. 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. Bi-directional Path Tracing

Bi-directional 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]. The algorithm is particularly well suited for scenes with significant indirect illumination.

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.

5.4. Compensated Ray Marching

This algorithm, developed by Zhou et al. [ZRL*07] follows the same idea as photon mapping, but computes its photon distribution on an approximation of the original volume. In a preprocessing step, the original volume $\mathcal{V}$ is split into a volume $\widetilde{\mathcal{V}}$ and a residual volume $\mathcal{R} = \mathcal{V} - \widetilde{\mathcal{V}}$. $\widetilde{\mathcal{V}}$ is a low frequency approximation to $\mathcal{V}$ and $\mathcal{R}$ is its error. The global lighting is calculated on $\widetilde{\mathcal{V}}$ and is afterwards rendered using ray marching (similarily as in 3). During the ray marching $\mathcal{R}$ is used to compensate for errors in $\widetilde{\mathcal{V}}$.

6. Conclusion and Further Works

In this chapter, we provide the reader with an overview of various other approaches to volume rendering and classify the herre presented algorithms within that context. Furthermore, we draw a comparison between the presented algorithms and briefly discuss their strength and weaknesses.

6.1. Classification and Further Work

On the most basic level volume rendering can be divided into direct and indirect rendering [Wes91, HMC*00]. Indirect rendering requires a pre-processing step to create intermediate surface data, which then is rendered in a second step. Direct algorithms (which includes all algorithms presented in this paper) work directly on the volume data and can be further subdivided into feed forward and feed backward methods. Feed forward methods work by starting with data and determining how much each data point contributes to the final image [Gla89] (e.g. by tracing paths from the light towards the camera). Feed backwards algorithms work the opposite way. Of the algorithms presented in this paper photon mapping and bidirectional path tracing are hybrid methods while the others are feed backwards. Another possibility to classify rendering algorithms is between stochastic and deterministic ones. Of the presented examples, the ray marching and the compensated ray marching are deterministic, the others stochastic.

6.2. Comparison

In this chapter, we provide the reader with some context by Volume rendering is an extensive topic which comprises several approaches, which on the most basic level can be divided into direct and indirect rendering [Wes91, HMC*00]. Indirect rendering requires a pre-processing step to create intermediate surface data, which then is rendered in a second step. Direct algorithms (which

includes all algorithms presented in this paper) work directly on the volume data and can be further subdivided into feed forward and feed backward methods. Feed forward methods work by starting with data and determining how much each data point contributes to the final image [Gla89] (e.g. by tracing paths from the light towards the camera). Feed backwards algorithms work the opposite way. Of the algorithms presented in this paper photon mapping and bidirectional path tracing are hybrid methods while the others are feed backwards. Another possibility to classify rendering algorithms is between stochastic and deterministic ones. Of the presented examples, the ray marching and the compensated ray marching are deterministic, the others are stochastic.

References

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph. 11*, 2 (jul 1977), 192–198. URL: https://doi.org/10.1145/965141.563893, doi:10.1145/965141.563893. 2, 3

[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

[CC] CARTER L. L., CASHWELL E. D.: Particle-transport simulation with the monte carlo method. URL: https://www.osti.gov/biblio/4167844, doi:10.2172/4167844. 5

[CPP*05] CEREZO E., PÉREZ F., PUEYO X., SERON F. J., SILLION F. X.: A survey on participating media rendering techniques. *The Visual Computer 21*, 5 (Jun 2005), 303–328. URL: https://doi.org/10.1007/s00371-005-0287-1, doi:10.1007/s00371-005-0287-1. 3

[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

[FHH*19] FASCIONE L., HANIKA J., HECKENBERG D., KULLA C., DROSKE M., SCHWARZHAUPT J.: Path tracing in production: Part 1: Modern path tracing. In *ACM SIGGRAPH 2019 Courses* (New York, NY, USA, 2019), SIGGRAPH '19, Association for Computing Machinery. URL: https://doi.org/10.1145/3305366.3328079, doi:10.1145/3305366.3328079. 5

[GBC*13] GALTIER M., BLANCO S., CALIOT C., COUSTET C., DAUCHET J., EL HAFI M., EYMET V., FOURNIER R., GAUTRAIS J., KHUONG A., PIAUD B., TERRÉE G.: Integral formulation of null-collision monte carlo algorithms. *Journal of Quantitative Spectroscopy and Radiative Transfer 125* (2013), 57–68. URL: https://www.sciencedirect.com/science/article/pii/S0022407313001350, doi:https://doi.org/10.1016/j.jqsrt.2013.04.001. 5, 6

[Gla89] GLASSNER A. S. (Ed.): *An Introduction to Ray Tracing*. Academic Press Ltd., GBR, 1989. 1, 7

[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

[HMC*00] HUANG J., MUELLER K., CRAWFIS R., BARTZ D., MEISSNER M.: A practical evaluation of popular volume rendering algorithms. In *2000 IEEE Symposium on Volume Visualization (VV 2000)* (2000), pp. 81–90. doi:10.1109/VV.2000.10009. 6, 7

[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. 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

[JLA*03] JENSEN H., LECTURERS J., ARVO, DUTRE P., KELLER A., OWEN A., PHARR M., SHIRLEY P.: Monte carlo ray tracing siggraph 2003 course 44, 07 2003. 4

[Kah55] KAHN H.: *Use of Different Monte Carlo Sampling Techniques*. RAND Corporation, Santa Monica, CA, 1955. 5

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computers & Graphics 28*, 6 (2004), 801–814. URL: https://www.sciencedirect.com/science/article/pii/S0097849304001487, doi:https://doi.org/10.1016/j.cag.2004.08.009. 1

[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, 3, 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, 4

[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). 5, 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, 3

[MK15] MORGAN L., KOTLYAR D.: Weighted-delta-tracking for monte carlo particle transport. *Annals of Nuclear Energy 85* (08 2015). doi:10.1016/j.anucene.2015.07.038. 5

[NGD*06] NARASIMHAN S. G., GUPTA M., DONNER C., RAMAMOORTHI R., NAYAR S. K., JENSEN H. W.: Acquiring scattering properties of participating media by dilution. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 1003–1012. URL: https://doi.org/10.1145/1179352.1141986, doi:10.1145/1179352.1141986. 2

[NSJ14] NOVÁK J., SELLE A., JAROSZ W.: Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph. 33*, 6 (nov 2014). URL: https://doi.org/10.1145/2661229.2661292, doi:10.1145/2661229.2661292. 3, 5, 6

[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

[Sha98] SHANNON C.: Communication in the presence of noise. *Proceedings of the IEEE 86*, 2 (1998), 447–457. doi:10.1109/JPROC.1998.659497. 3

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

[VG95] VEACH E., GUIBAS L. J.: Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), SIGGRAPH '95, Association for Computing Machinery, p. 419–428. URL: https://doi.org/10.1145/218380.218498, doi:10.1145/218380.218498. 5

[Wes91] WESTOVER L.: Splatting: a parallel, feed-forward volume rendering algorithm. 6, 7

[Wik23] WIKIPEDIA: Probability density function — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Probability%20density%20function&oldid=1130726352, 2023. [Online; accessed 13-January-2023]. 5

[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, 6