Volumetric Path Tracing

Steve Marschner
Cornell University
CS 6630 Spring 2012, 8 March

Using Monte Carlo integration is a good, easy way to get correct solutions to the radiative transfer equation. It is not fast (at least not as naïvely implemented), especially for inhomogeneous or highly scattering media, but it is easy to get the right answer (plus random noise) for complex cases where there pretty much exists no other way to get an artifact-free solution.

Deriving the path tracing algorithm pretty much boils down to applying the Monte Carlo integration framework to the integral form of the volume rendering equation that we derived in the radiative transport lectures. As usual, the key to developing correct algorithms is to carefully write down the integrals involved and construct the Monte Carlo estimators systematically.

The integral form of the equation we need to solve, in the notation I've been using in lecture, is:

$$L(x,\omega) = \int_x^y \tau(x,x') \Big[\underbrace{\sigma_s(x') \int_{4\pi} f_p(x',\omega,\omega')\, L(x',\omega')\, d\omega'}_{\text{inscattering}} + \underbrace{\epsilon(x',\omega)}_{\text{emission}} \Big] dx' + \underbrace{\tau(x,y)\, L_e(y,\omega)}_{\text{attenuation}}$$

The MC approach is to identify the integrand, choose a point from a suitable pdf, and compute the ratio:

$$\frac{f}{p} = \tau(x,x') \Big[\sigma_s(x')\, f_p(x',\omega,\omega')\, L(x',\omega') + \frac{1}{4\pi}\, \epsilon(x',\omega) \Big] \Big/ p(x',\omega')$$

Since we will usually choose x' first, then choose ω' conditioned on that choice, we can write this as a nested estimator that estimates the ray integral using an estimate for the scattering integral:

$$\frac{f}{p} = \tau(x,x') \left[\frac{\sigma_s(x')\, f_p(x',\omega,\omega')\, L(x',\omega')}{p(\omega')} + \epsilon(x',\omega) \right] \Big/ p(x')$$

Let's apply this first in some special cases. First we will look at the case for homogeneous, gray (wavelength-independent) media, for which is *much* easier to get a working implementation.

1. Emission only

Here $\tau(x, x') = 1$, and $\sigma_s = \sigma_a = 0$. The integral is just

$$L(x,\omega) = \int_x^y \epsilon(x',\omega)\, dx'$$

If we use uniform sampling with respect to distance along the ray, we have the estimator:

$$g = \epsilon(x',\omega)\, \|x-y\|$$

If the emission is homogeneous, then the answer is simple: $\varepsilon\|x-y\|$. The estimator is constant, because p is proportional to f—this is "perfect" importance sampling, causing a zero-variance estimator. (Of course, it also requires having the answer in closed form in the first place!)

2. Homogeneous, absorption only.

Here $\varepsilon = \sigma_s = 0$, and $\tau(\mathbf{x}, \mathbf{y}) = \exp(-\sigma_a\|\mathbf{x}-\mathbf{y}\|)$.

Since there is no scattering or emission, the integral along the line goes away and we are left with the simple answer:

$$L(\mathbf{x}, \omega) = \tau(\mathbf{x}, \mathbf{y})\, L_e(\mathbf{y}, \omega).$$

There's no integral, so there's no Monte Carlo.

3. Emission + homogeneous absorption.

This is the first case where things become nontrivial. Here $\sigma_s = 0$ so the remaining terms are:

$$L(x,\omega) = \int_x^y \tau(x,x')\, \epsilon(x',\omega)\, dx' + \tau(x,y)\, L_e(y,\omega).$$

For the integral,

$$f = \tau(x,x')\ \epsilon(x',\omega)$$

and for uniform sampling

$$g = \|x-y\|\, e^{-\sigma_a\|x-x'\|}\, \epsilon(x',\omega).$$

This will work, but depending on the density of the medium, it can be mighty inefficient. (Exercise: sketch g as a function of arc length along the ray. What happens to the variance as the medium becomes denser?) Once there is attenuation, uniform sampling is no longer a good strategy, but a very simple importance sampling strategy is available: we can importance sample the attenuation.

Attenuation in the homogeneous medium is $\tau(s) = \exp(-\sigma_t s)$ where s is arc length along the ray. We would like to find a pdf $p(s)$ that is proportional to $\tau(s)$. To make it a pdf we just need to normalize it. The normalization depends upon the domain over which we normalize it—let me start by assuming the ray never exits the medium, so the domain of integration is $[0, \infty]$. Normalizing τ to make a pdf:

$$p(s) = \frac{\tau(s)}{\int_0^\infty \tau(t)dt} = \frac{\exp(-\sigma_t s)}{[-\exp(-\sigma_t t)/\sigma_t]_{t=0}^{\infty}} = \sigma_t \exp(-\sigma_t s)$$

Then, integrating p from 0 to s we obtain $P(s)$, the cumulative distribution function (cdf):

$$P(s) = \int_0^s p(s')ds' = \left[-\exp(-\sigma_t s')\right]_0^s = 1 - \exp(-\sigma_t s)$$

Recall the procedure for choosing a sample according to p is to take a random number ξ that is uniform over the interval [0,1] and solve $P(s) = \xi$. In this case that leads to

$$\begin{aligned} \xi &= 1 - \exp(-\sigma_t s) \\ s &= -\frac{\ln(1-\xi)}{\sigma_t} \\ \text{or equiv:} \quad s &= -\ln \xi / \sigma_t \end{aligned}$$

(since $1 - \xi$ has the same distribution as ξ.)

Using this pdf leads to the estimator:

$$g(\mathbf{x}') = \frac{f(\mathbf{x}')}{p(\mathbf{x}')} = \frac{\tau(\mathbf{x}, \mathbf{x}')\varepsilon(\mathbf{x}')}{\sigma_t \tau(\mathbf{x}, \mathbf{x}')} = \frac{\varepsilon(\mathbf{x}')}{\sigma_t}$$

This will perform a lot better than uniform random sampling in relatively dense media, and because it makes the estimator so simple to compute, it is an elegant choice. The procedure looks like:

function radiance_estimator($\mathbf{x}$, ω):
    ξ = rand()
    s = –ln ξ / σ_t
    return $\varepsilon(\mathbf{x} - s\omega)$ / σ_t

Of course when the medium doesn't continue indefinitely along the ray, we'd like to choose a point according to $\tau(s)$ but restricted an interval $[0, s_{\max}]$. This is an easy generalization of the previous result (Exercise: The difference is in the normalization factor. Work it out.) but isn't quite so elegant. Since, if the ray ends somewhere, we also need to include the radiance from behind the medium, a simple trick is to generate s and then compare it with $s_{\max}$. The probability that $s > s_{\max}$ is $1 - \mathrm{P}(s_{\max})$, which is $\exp(-\sigma_t s_{\max})$: exactly the attenuation that needs to be applied to the radiance from behind the medium. So if we just return this background radiance whenever we attempt to choose a point past the end, the expected value is correct, leading to the following tweaked procedure:

function radiance_estimator($\mathbf{x}$, ω):
    $s_{\max}$, L_0 = trace_ray()
    ξ = rand()
    s = –ln ξ / σ_t
    if $s < s_{\max}$ **then**
        return $\varepsilon(\mathbf{x} - s\omega) / \sigma_t$
    else
        return L_0

Another way to interpret the sampling procedure is that we select a random attenuation ξ and then find the distance s for which $\tau(s) = \xi$.

4. Emission + (inhomogeneous) absorption

In this situation the integral to be computed is still:

$$L(\mathbf{x}, \omega) = \int_{\mathbf{y}}^{\mathbf{x}} \tau(\mathbf{x}', \mathbf{x}) \varepsilon(\mathbf{x}') d\mathbf{x}'$$

except this time, when we open the black box called τ, we find:

$$L(\mathbf{x}, \omega) = \int_{\mathbf{y}}^{\mathbf{x}} \exp\left(-\int_{\mathbf{x}'}^{\mathbf{x}} \sigma_t \right) \varepsilon(\mathbf{x}') \, d\mathbf{x}'$$

We can approach this using the same importance sampling scheme as above, but it is no longer so simple to compute the attenuation. Instead of just evaluating an exponential, we have to somehow sample proportional to the spatially varying attenuation coefficient along the ray. One way to do this is using numerical integration: compute attenuation using some very simple quadrature rule—for example, the trapezoid rule, or even just the Riemann sum—on a regularly spaced set of samples along the ray. Establishing a suitable spacing requires knowing something about how fast the attenuation coefficient can vary in the volume. Another way uses a clever random sampling scheme; both are discussed below.

So assuming we have some scheme for sampling the integral, we can make an estimator from it:

$$g = \frac{\tau(\mathbf{x}, \mathbf{x}') \varepsilon(\mathbf{x}', \omega)}{p(\mathbf{x}')}$$

but again, it is crucial to importance sample, which no longer can be done using the simple computation for homogeneous media.

4.1. Sampling distances by quadrature: ray marching

Setting p to be proportional to τ as before is possible, but leads to a messy expression for P. However, generalizing the "find where attenuation is ξ" procedure is simple. We need to compute the distance at which $\tau = \xi$ (for a uniform random ξ), which can be done by modifying the numerical integrator so that it marches along the ray from the starting point, accumulating attenuation, until it just crosses ξ, then interpolates to find the sample point.

A simple Riemann integration procedure to find the attenuation between 0 and s is

function attenuation($s_{\max}$, h)
    $T = 0$
    for ($s = 0$; $s < s_{\max}$; s += h)
        $f = \sigma_t(s)$
        T += hf
    T –= $(s - s_{\max})\, f$
    return exp(–T)

To solve for $\tau(s) = \xi$, just alter the stopping condition:

function find_attenuation(τ_0, τ_1, s_0, h)
    $T = \ln(\tau_0)$
    $T_1 = \ln(\tau_1)$
    for ($s = s_0$; $T < T_1$; s += h)
        $f = \sigma_t(s)$
        T += hf
    s –= $(T - T_1) / f$
    return s

Of course these routines can be improved without much extra complexity by using the trapezoid rule.

The pdf that results from this procedure is

$$
\begin{aligned}
p(t) &= p(\xi)\left|\frac{d\xi}{dt}\right| = \left|\frac{d\tau}{dt}\right| \\
\frac{d\tau}{dt} &= \frac{d}{dt}\exp\left(-\int_0^t \sigma_t(\mathbf{x}(t'))dt'\right) \\
&= -\exp\left(-\int_0^t \sigma_t(\mathbf{x}(t'))dt'\right)\sigma_t(\mathbf{x}(t)) \\
\left|\frac{d\tau}{dt}\right| &= \tau(\mathbf{x},\mathbf{x}')\sigma_t(\mathbf{x}')
\end{aligned}
$$

(Note here that $d\tau/dt = d\xi/dt$.) If we use this pdf for importance sampling, the estimator becomes:

$$
g = \frac{\tau(\mathbf{x},\mathbf{x}')\varepsilon(\mathbf{x}',\omega)}{p(\mathbf{x}')} = \frac{\varepsilon(\mathbf{x}',\omega)}{\sigma_t(\mathbf{x}')}
$$

This all leads to the following algorithm for sampling an inhomogeneous medium:

function radiance_estimator($\mathbf{x}$, ω):
    $s_{\max}$, L_0 = trace_ray()
    ξ = rand()
    s = find_attenuation(ξ, 0, h)
    if $s < s_{\max}$ **then**
        return $\varepsilon(\mathbf{x} - s\omega) / \sigma_t(\mathbf{x} - s\omega)$

```
    else
        return L_0
```

The disappointing thing is that we can't just plug in a Monte Carlo estimator for the attenuation integral. If it was true that $E\{\exp(X)\} = \exp(E\{X\})$ then we'd be in good shape—but sadly it is not. This means that the only thing we can do is put in a sufficiently accurate numerical estimate of the integral: we can't count on statistical averaging to let us use a very noisy estimate.

We also can't generate an individual sample of s very quickly—we have to integrate along the ray from the eye to the sample point. This means that if we are planning to take many samples along the ray (which, of course, we will need to do), it would be terribly inefficient to handle them independently—in fact, it would be quadratic in the number of samples required.

To generate samples for many values of ξ together, let's assume that we have the list of random numbers $\xi_0, \ldots, \xi_{(n-1)}$ in hand *a priori*, and that they are sorted so that $\xi_0 < \xi_1 < \cdots < \xi_{(n-1)}$. To generate the many samples all at once, we can call the integration procedure multiple times, each time picking up from where it left off; you'll note I added the arguments τ and s_0 so that this is easy to do:

```
function radiance_estimator(x, ω, ξ-list):
    s_max, L_0 = trace_ray()
    S = 0
    n = ξ-list.length
    for (i = 0; i < n; i++)
        ξ = ξ-list[i]
        s = find_attenuation(ξ, s, s_max, h)
        if s < s_max then
            S += ε(x − sω) / σ_t(x − sω)
        else
            break
    return (S + (n − i)L_0) / n
```

(I hope this pseudocode is correct, but as it wasn't actually transcribed from working code there is some chance of bugs.) Here I am giving the samples that fall past $s_{\max}$ (all $n–i$ of them) the value L_0 in the average.

Note that it is easy to perform stratified sampling on the ray integral: you just use stratified random values for the ξ_i, for instance by splitting the interval up into n pieces and choosing one random number in each one:

```
function generate_ξs(n):
    for (i = 0; i < n; i++)
        ξs[i]  = (i + rand()) / n
    return ξs
```

This is easy to do—in fact, easier than generating a sorted list of independent random numbers—and will perform much better.

Another way to look at this, which makes a lot of sense if evaluating ε is cheap, is that we should just use enough samples that the emission is lost.

4.2. Sampling distances by Woodcock tracking

More recently, another technique has become popular for sampling distances along a ray in an inhomogeneous medium. The idea comes from the neutron transport community in the 60s and has various names (delta tracking, pseudo-scattering); we'll call it Woodcock tracking in an effort to credit the first paper to write it down.

The idea is to take many steps as if the medium were denser than it is, and then randomly choose to return that distance or take another step. It's simplest to simply write pseudocode:

function sample_distance($\mathbf{x}$, ω, $\sigma_{\max}$):
    $s = 0$
    while true:
        s += –ln(1 – rand()) / $\sigma_{\max}$
        if (rand() < $\sigma_t(\mathbf{x} - s\omega) / \sigma_{\max}$)
            break
    return s

where $\sigma_{\max}$ is an upper bound for σ_t over the whole ray. It's not immediately obvious that this works. Certainly it does work if $\sigma_t(\mathbf{x}) = \sigma_{\max}(\mathbf{x})$ everywhere—it just takes one step, then returns. Certainly it stops in denser areas more often, since the random walk terminates with probability proportional to $\sigma_t(\mathbf{x})$, and it will never stop where $\sigma_t(\mathbf{x}) = 0$. And it seems plausible that the probability decreases with distance since more or larger steps are required to go farther. But it can indeed be proven to produce values of s with probability density

$$p(s) = \sigma_t(s) \exp\left(-\int_0^s \sigma_t(s')ds'\right) \quad \text{or} \quad p(\mathbf{x}') = \sigma_t(\mathbf{x}')\tau(\mathbf{x}, \mathbf{x}')$$

which is actually exactly the same density as produced by the ray marching algorithm in the previous section, except with no numerical error.

An intuitive explanation for why this works: imagine that, in addition to the particles that are really there, our medium also contains some particles that have no absorption and a perfectly forward-directed phase function, so that scattering from one of these particles is a no-op. Introduce just enough of these particles to "fill up" the volume to a constant attenuation $\sigma_{\max}$ everywhere. Since the scattering cross section is now constant, it's easy to importance sample again, and each time we interact with the medium we check to see whether we hit one of these imaginary particles, and if we did, we draw again (since the effect of scattering is just to continue the path).

Woodcock tracking usually leads to good performance compared to ray marching, particularly if the optimization of multiple samples per ray-march is not implemented. Its performance is sensitive to the "spikiness" of the density: if the max is much larger than the typical values, it will take a lot of small steps. This situation can be improved by breaking the volume into pieces (perhaps hierarchically) and computing separate maxima per piece.

5. Introducing the scattering term

We now know how to handle absorption and emission in the general case. The remaining issue is how to compute scattering. As we've observed before, outscattering behaves exactly like absorption, as far as attenuation is concerned. Similarly, inscattering behaves exactly like emission, as far as the integral along the ray is concerned. So we can use the method from the previous section but modify the emission by adding a Monte Carlo estimator for the scattering integral.

Estimating the scattering integral in a volume is just like estimating the reflection integral at a surface. It is an integral over the sphere, and we will choose directions to sample, thinking of them as points on the sphere.

$$\varepsilon_s(\mathbf{x}', \omega) = \sigma_s(\mathbf{x}') \int_{4\pi} f_p(\mathbf{x}', \omega, \omega') L(\mathbf{x}', \omega') d\omega'$$

The first thing to try is importance sampling by the phase function. (This means that whenever we propose a particular phase function, we also need to come up with a procedure for sampling it. There is a very simple procedure available for Henyey-Greenstein, the most popular phase function model out there.) Remember the phase function is normalized to be a probability distribution already, so in this case $p(\omega') = f_p(\mathbf{x}', \omega, \omega')$. Then the estimator is

$$\begin{aligned} g(\omega') &= \frac{f(\omega')}{p(\omega')} = \frac{\sigma_s(\mathbf{x}') f_p(\mathbf{x}', \omega, \omega') L(\mathbf{x}', \omega')}{p(\omega')} \\ &= \sigma_s(\mathbf{x}') L(\mathbf{x}', \omega') \end{aligned}$$

So all we need to do is trace a recursive ray to estimate $L(\mathbf{x}', \omega')$ and the process becomes:

1. Choose ξ in $[0, 1]$ and find the point $\mathbf{x}'$ on the ray for which $\tau(\mathbf{x}, \mathbf{x}') = \xi$.
2. Choose ω' according to the phase function.
3. Recursively estimate $L(\mathbf{x}', \omega')$ and compute the estimator $(\varepsilon(\mathbf{x}', \omega) + \sigma_s(x')\, L(\mathbf{x}', \omega')) / \sigma_t(\mathbf{x}')$.

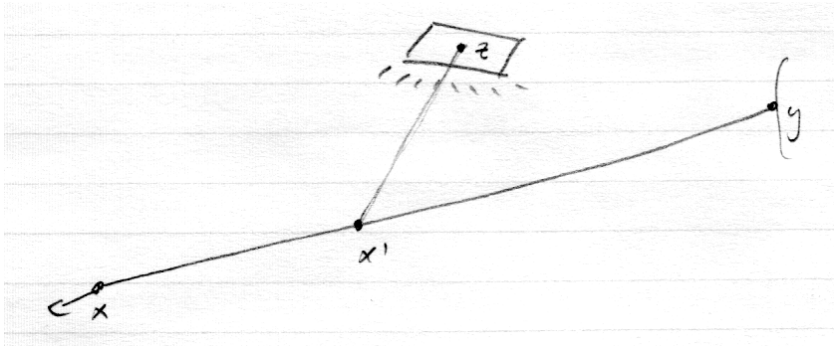
That is, we simply replace the statement starting S += … by the following:

ω' = sample_phase_function()
L_s = radiance_estimator($\mathbf{x} - s\omega$, ω', new_ξ_list())
S += $(\varepsilon(\mathbf{x} - s\omega) + \sigma_s L_s) / \sigma_t(\mathbf{x} - s\omega)$

Doing this leads to a path tracing algorithm very similar to a brute-force surface path tracer.

5. Direct lighting for volumes

We just developed the equivalent of BRDF-only sampling in path tracing: it bounces around in the medium and has to coincidentally escape and hit a light source in order to contribute any radiance to the image. This works OK as long as lighting is low frequency, but fails badly for small sources. The solution, just as with surface path tracing, is to introduce an importance sampling distribution that expects illumination to come from light sources.



The contribution to ray radiance just from direct illumination (aka. single scattering) is

$$L^1(\mathbf{x},\omega) = \int_{\mathbf{x}}^{\mathbf{y}} \tau(\mathbf{x},\mathbf{x}')\sigma_s(\mathbf{x}') \int_{4\pi} L^0(\mathbf{x}',\omega')f_p(\mathbf{x},\omega,\omega')d\omega'$$

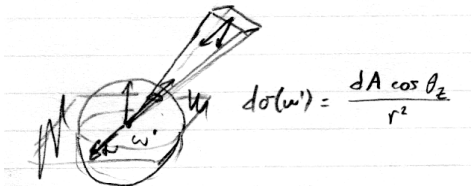
where L^0 is the reduced radiance coming directly from sources—attenuated, but not scattered:

$$L^0(\mathbf{x},\omega) = \tau(\mathbf{x},\mathbf{z})L_e^0(\mathbf{z},\omega); \quad \mathbf{z} = \psi(\mathbf{x},-\omega)$$

This looks like the surface illumination integral but with some extra attenuation and no cosine factor. With samples chosen proportional to the pdf $p(\omega')$ we have

$$g(\omega') = \frac{f(\omega')}{p(\omega')} = \frac{\sigma_s(\mathbf{x}')f_p(\mathbf{x}',\omega,\omega')L^0(\mathbf{x}',\omega')}{p(\omega')}$$

Suppose we choose $\mathbf{z}$ proportional to area, with density $p_A(\mathbf{z})$.



Just as with the case of surfaces illuminating surfaces, we need to convert this area-based pdf to a solid-angle-based pdf in order to use it in estimating this integral over solid angle.

$$p_\sigma(\omega') = p_A(\mathbf{z})\frac{|dA|}{|d\sigma|} = p_A(\mathbf{z})\frac{\|\mathbf{x}'-\mathbf{z}\|^2}{\langle\omega', n(\mathbf{z})\rangle}$$

This leads to the single scattering estimator

$$g(\omega') = \frac{f(\omega')}{p(\omega')} = \frac{\tau(\mathbf{x},\mathbf{z})\sigma_s(\mathbf{x}')f_p(\mathbf{x}',\omega,\omega')L_e^0(\mathbf{x}',\omega')\langle\omega', n(\mathbf{z})\rangle}{p_A(\mathbf{z})\|\mathbf{x}'-\mathbf{z}\|^2}$$

Now we have something that resembles a Kajiya-style path tracer. Adding multiple importance sampling and Russian Roulette requires straightforward generalizations of the methods used for surfaces.