A Simplified Path Tracing Architecture

Figure 1: A path traced image with of a scene with millions of triangles with full lighting in a participating medium.

Abstract

While the basic path tracing algorithm was developed for simple surfaces and well-defined luminaires, modern scenes contain complex luminaires, surfaces and participating media. This has resulted in add-ons for the path tracing algorithm. We rethink the path tracing architecture in the context of this complexity, and show that an even simpler architecture than classical path tracing works well. This consists of using simple directional sampling without shadow rays, and treating participating media as probabilistic surfaces.

Keywords: path tracing, participating media

1 Introduction

While many realistic rendering algorithms have been developed over the last twenty years, most are tailored toward relatively simple scenes that have more lighting complexity than geometric complexity. However, there has been an exponential explosion of the geometric complexity of computer graphics models, and there is no reason to expect this explosion to slow. The first realistic rendering algorithm able to deal with such geometric complexity was *path tracing* [Kajiya 1986]. However, the elegance of the simple path tracing algorithm is often lost in robust implementations because of special cases, such as small luminaires, high dynamic range environment maps, and participating media. In this paper we reexamine path tracing with such complexities in mind. The key to our improvements is that we have a higher sampling budget than Kajiya could afford[1], and show that we can regain the elegance of the original algorithm by careful directional sampling, and by treating media as probabilistic surfaces.

[1]Even with the reduced computational requirements of a path tracer with an illumination cache, hundreds to thousands of samples per pixel are still needed for good image quality. For example, for the film *Shrek 2*, hundreds of rays per pixel were used on average, with 2000 rays per pixel used in some frames [Tabellion and Lamorlette 2004]. Similarly, the *Kilauea* rendering project used over 4000 rays per pixel in some scenes [Kato and Saito 2002].

2 Path Tracing

In any algorithm that uses path tracing, the transport equation is evaluated at any point $\mathbf{x}$ hit by a ray [Kajiya 1986]. The radiance L of a point $\mathbf{x}$ viewed from direction ω is:

$$L(\mathbf{x},\omega) = \int \rho(\mathbf{x},\omega,\omega')\, L_f(\mathbf{x},\omega')\, cos\theta' d\omega', \qquad (1)$$

where L_f is the field radiance, and ρ is the BRDF (bidirection reflectance distribution function) [Jensen 2001; Dutre et al. 2003; Pharr and Humphreys 2004]. This integral can be approximated by Monte Carlo integration [Cook 1986] with a suitable random ω' with density $p(\omega')$:

$$L(\mathbf{x},\omega) \approx \frac{\rho(\mathbf{x},\omega,\omega')\, L_f(\mathbf{x},\omega')\, cos\theta'}{p(\omega')}. \qquad (2)$$

This gives rise to simple code:

```
function radiance( ray (a + sω) )
if (ray hits surface at point x) then
    generate random ray r = (x + tω') with ω' ~ p
    return ρ(x, ω, ω')(ω' · N)radiance(r)/p(ω')
else
    return background(ω)
```

The simple path tracer outlined above is not efficient (i.e. converges slowly) because it has deep ray trees, which inhibits stratification and thus hurts the rate of convergence [Mitchell 1996]. One way to deal with this is to only allow one diffuse scattering event. This can work well in some scenes, particularly outdoors (e.g. Figure 1, [Tabellion and Lamorlette 2004]).

Our implementation uses a shader architecture [Hanrahan and Lawson 1990; Slusallek and Siedel 1995] so that each material can behave according to its own rules. This allows us to easily add specialized surface shading algorithms to our path tracer.

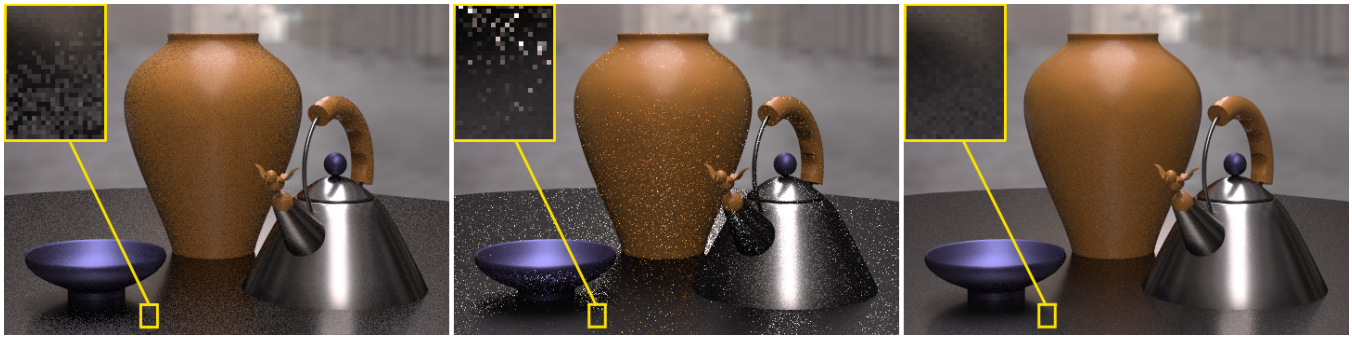

Figure 2: Using a mixture density. Left: Sampling the BRDF. Middle: Sampling the environment map. Right: Using a mixture density with weighting coefficients 0.75 for the BRDF portion and 0.25 for the environment map. All images are 289 samples per pixel and have approximately the same run times.

3 Directional Sampling

The key to improving the efficiency of our path tracer is to choose a "good" density function p. Most path tracers choose to partition illumination into two separate integrals corresponding to "direct" and "indirect" lighting [Shirley et al. 1996]. One problem with such a separation is that it is not clear where high dynamic range (HDR) environment maps [Unger et al. 2003] or bright caustics fit into such a partition. We sidestep this issue by using a variant of *multiple importance sampling* [Veach and Guibas 1995], where a collection of densities $p_1, p_2, \ldots p_m$ are combined to form a single density. We simply use a *mixture density* that is a weighted average of the p_i. We note that this is equivalent to a special case of the "single sample model" of Veach and Guibas, but we find the mixture density formulation much more intuitive as it allows us to apply Equation 2 directly. To our knowledge, this mixture density formulation is novel for a path tracer, as is the resulting implementation.

Multiple importance sampling would require an estimator for each density we would like to use. The mixture density methodology instead uses a single improved density with only one estimator. This results in cleaner code for the mixture density model. It also allows us to easily add new densities to describe elements such as caustics without requiring additional estimators; we simply redistribute the weighting coefficients.

In practice, one p_i is related to the BRDF, one describes directions toward luminaires, and one is proportional to the background luminance. When the background is an HDR environment map, the p_i related to the background can be sampled according to intensity weighted solid angle [Chiu et al. 1996]. Sampling the product of the illumination field and the BRDF would be even better [Burke 2004], but it is not clear how to efficiently incorporate such a strategy into a renderer that uses many samples per pixel.

Some care must be taken to evaluate p correctly. For a 1D example, if we choose a sample from the one dimensional density $(p_1 + p_2)/2$ using a stratified seed R, we do the following:

if $R < 0.5$ **then**
  $R = 2R$
  $x =$ generateSample(p_1, R)
else
  $R = 2(R - 0.5)$
  $x =$ generateSample(p_2, R)
$p = 0.5p_1(x) + 0.5p_2(x)$

Note that we need to be able to evaluate the values of both p_1 and p_2 regardless of which distribution the sample is taken from. This matters for cases where the p_i have overlapping non-zero regions (e.g., sampling the BRDF and the environment map). Some efficiency is gained if *generateSample* returns both the sample x and the value $p_i(x)$.

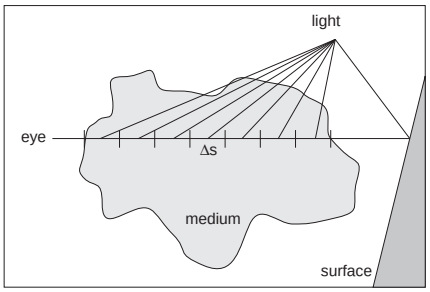

Figure 3: A traditional ray-march through a medium, and at each step a shadow ray is also marched.

This mixture density architecture can be extended recursively. The density for BRDFs can be a mixture density of a diffuse and a specular lobe. The density for area lights can weight nearby lights more heavily. A density over known caustics could weight directions according to an approximation of subtended solid angle. This emphasizes that any improvement in the weighting coefficients, even through approximations, will be beneficial in improving efficiency.

An example of the power of this technique is shown in Figure 2. Here we use brute force importance sampling of the environment map and the BRDFs, using models and BRDF data from Lawrence et al. [2004]. The mixture coefficients were chosen manually, but in theory could be chosen according to heuristics. In addition to dealing with sampling problems, the directional sampling method turns out to produce a very clean code architecture; not having to deal with the partition of direct and indirect lighting makes the code a direct implementation of Equation 2.

4 Participating Media

One of our key simplifications over traditional path tracing involves participating media. The basic idea is to treat the participating medium as a probabilistic scatterer/absorber rather than as an attenuating body. It is tempting to modify Equation 1 to account for local interaction with a volume

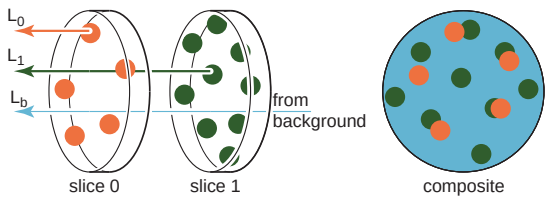

Figure 4: Two thin slices of a participating medium and a background color. When looking up the "tube" there are three possibilities for what is seen as shown by the arrows: a particle in slice 0, a particle in slice 1, or the background.

over some distance Δs but this would not be compatible with surfaces. Instead we note that the concept of transparency can be made probabilistic; rather than a path through a volume having 20% opacity, there is a 20% chance that the ray "hits" the volume. This change in architecture allows surfaces and volumes to utilize the same interface.

The traditional traversal of a volume is shown in Figure 3. This is expensive because of the ray marching for shadow rays; $O(n^2)$ time is required since each of the $O(n)$ steps along the viewing ray generates a shadow ray that also has $O(n)$ steps. Although this idea has been used for bidirectional ray tracing [Lafortune and Willems 1996], photon tracing [Jensen 2001] and Metropolis Light Transport [Pauly et al. 2000], we are unaware of its use in a traditional path tracer.

In Figure 4 we see two thin slices of a medium that contain scattering/absorbing particles, as well as a background color. These slices are "thin" in that it is assumed that there is no overlap between particles for a given slice. If we think of compositing from front to back (traversing from the eye), the color of the ray is:

$$L = L_0 \sigma_t^0 \Delta s + (1 - \sigma_t^0 \Delta s) L_0^+, \tag{3}$$

where L_0 is the radiance of the particles in slice 0, σ_t^i is extinction coefficient (the probability per unit length of interaction with particles) in slice i, Δs is the thickness of the slice, and L_0^+ is the radiance seen along the ray after slice 0. Note that this is valid only if $\sigma_t^0 \Delta s$ is small. Similarly,

$$L_0^+ = L_1 \sigma_t^1 \Delta s + (1 - \sigma_t^1 \Delta s) L_b. \tag{4}$$

This can be generalized to a large number of slices, as is done in typical renderings. The radiance of particles L_i in a slice can be computed using a variant of Equation 1,

$$L_i(\mathbf{x}, \omega) = \Lambda(\mathbf{x}) \int p(\mathbf{x}, \omega, \omega')\, L_f(\mathbf{x}, \omega')\, d\omega', \tag{5}$$

where p is the phase function, and $\Lambda(\mathbf{x}) = \sigma_s / \sigma_t$ is the single scattering albedo (ratio of scattering to extinction coefficients) at a given point $\mathbf{x}$ [Pharr and Humphreys 2004]. This can then be sampled with a density function over the whole sphere in a way very similar to Equation 2.

Instead of using ray marching, we implement transparency with a probabilistic intersection routine. A ray traveling through the medium will either "hit" a particle and scatter, or it will pass through the medium unaffected (Figure 5). Although this results in more randomness for a given ray, it allows us to implement a ray-volume intersection using an $O(n)$ algorithm with the same interface as a ray-surface intersection.

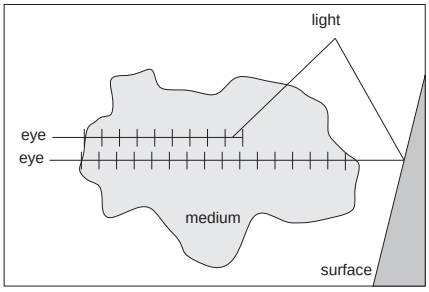

Figure 5: Two rays traverse a volume, and because of different random seeds, one hits the volume and one does not.

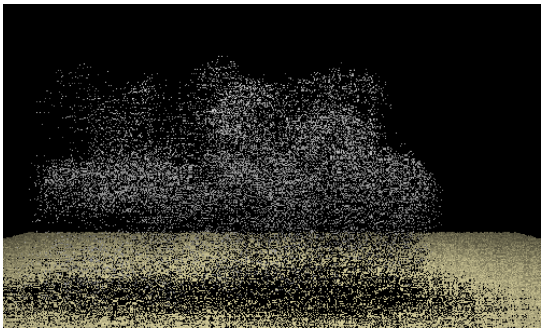

Figure 6: At one sample per pixel each pixel either hits the medium completely or is all background.

When a ray intersects the volume, we calculate shading as for a fully opaque surface. Using the phase function we compute a weighted average of field radiance. We then attenuate the weighted average using the single scattering albedo (Equation 5). This is exactly how surfaces are shaded, except that we use the phase function instead of the cosine weighted BRDF.

For a ray within a slice at $\mathbf{x}$ of thickness Δs the probability of interaction is $\sigma_t(\mathbf{x})\Delta s$. This gives rise to the code:

```
function volumehit( ray (a + sb) )
 // r() is random on [0,1], b is a unit vector
if (ray hits volume bounds) then
   find entry and exit distances s0 and s1
else
   return false
s = s0 + Δs/2        // center of first interval
while (s < s1) do
   x = a + sb
   if (r() < σt(x)Δs)  then
      return true, x, Λ(x), and p
   s = s + Δs
return false
```

The method above would not stratify between rays in the pixel that hit the volume. This can be solved by accumulating the probability of interaction along the ray by passing a random seed R that is selected from a stratified set of $1D$ samples on $[0, 1]$. For distance s along the ray, the probability P_i of having an intersection in or before step i is:

$$P_i = P_{i-1} + (1 - P_{i-1}) \sigma_t^i \Delta s. \tag{6}$$

This is just a cumulative probability distribution function, so it is well-suited to stratified sampling with a seed R. The while loop above becomes:

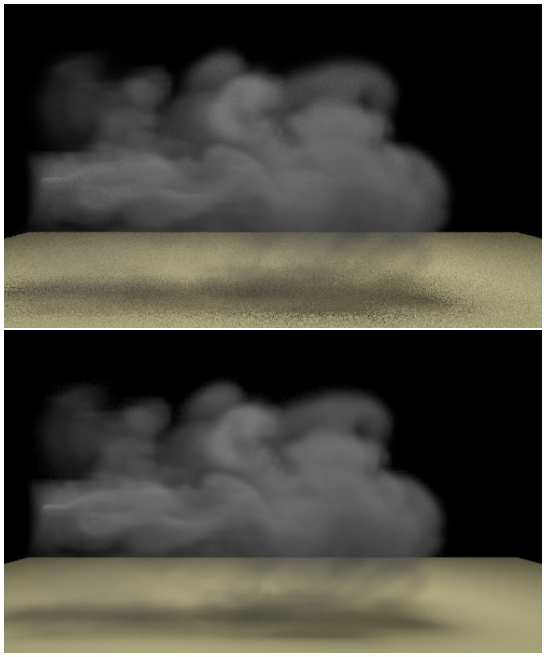

Figure 7: Top: traditional ray marching, 4 samples/pixel. Bottom: new probabilistic intersection, 400 samples/pixel. These images are computed in approximately the same time.

$P = 0$ *// the probability we hit before volume is zero*
while $(s < s_1)$ **do**
  $\mathbf{x} = \mathbf{a} + s\mathbf{b}$
  $P = P + (1 - P)\sigma_t(\mathbf{x})\Delta s$
  if $(R < P)$ **then**
    return true, $\mathbf{x}$, $\Lambda(\mathbf{x})$, and p
  $s = s + \Delta s$

In the case where σ_t is a constant (i.e. a uniform media), an analytic expression can be used to solve for s [Dutre et al. 2003]: $s = -\ln(1 - R)/\sigma_t$.

Figure 6 shows the new method with one sample per pixel, which demonstrates the probabilistic nature of our intersection routine: rays either hit the volume or pass through it completely. Figure 7 compares ray marching with our probabilistic intersection method. Note that for 4 samples per pixel, ray marching has mostly converged except for the shadow, which has variance because its samples are distributed across the light source. Our volume shading method runs in linear time, compared to the quadratic time required for traditional ray marching. This increase in efficiency allows us to utilize more samples per pixel with similar computational time. This higher sampling density leads to lower variance in other portions of the scene, such as shadows.

One concern with probabilistic intersection is that it assumes σ_t is a scalar constant. For some media (e.g., Rayleigh scattering air), σ_t varies with wavelength. In practice, we have not found color noise to be a dominant visual artifact.

5 Conclusion

We have presented a simplified architecture for path tracing. The system features intelligent sampling techniques based on mixture densities, and a single interface for both surfaces and volumes. We have developed a probabilistic intersection routine for participating media that is competitive with traditional ray marching, while allowing for efficient antialiasing of the scene.

References

Burke, D. 2004. *Bidirectional Importance Sampling for Illumination from Environment Maps*. Master's thesis, University of British Columbia.

Chiu, K., Zimmermann, K., and Shirley, P. 1996. The light volume: An aid to rendering complex environments. In *Eurographics Rendering Workshop*, 1–10.

Cook, R. L. 1986. Stochastic sampling in computer graphics. *ACM Transactions on Graphics 5*, 1, 51–72.

Dutre, P., Bekaert, P., and Bala, K. 2003. *Advanced Global Illumination*. AK Peters.

Hanrahan, P., and Lawson, J. 1990. A language for shading and lighting calculations. In *Proceedings of SIGGRAPH*, 289–298.

Jensen, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. AK Peters.

Kajiya, J. T. 1986. The rendering equation. In *Proceedings of SIGGRAPH*, 143–150.

Kato, T., and Saito, J. 2002. Kilauea - parallel global illumination renderer. In *Eurographics Workshop on Parallel Graphics and Visualization*, 7–16.

Lafortune, E. P., and Willems, Y. D. 1996. Rendering participating media with bidirectional path tracing. In *Eurographics Rendering Workshop*, 91–100.

Lawrence, J., Rusinkiewicz, S., and Ramamoorthi, R. 2004. Efficient brdf importance sampling using a factored representation. *ACM Transactions on Graphics 23*, 3, 496–505.

Mitchell, D. P. 1996. Consequences of stratified sampling in graphics. In *Proceedings of SIGGRAPH*, 277–280.

Pauly, M., Kollig, T., and Keller, A. 2000. Metropolis light transport for participating media. In *Eurographics Workshop on Rendering*, 11–22.

Pharr, M., and Humphreys, G. 2004. *Physically Based Rendering*. Morgan Kaufmann.

Shirley, P., Wang, C., and Zimmerman, K. 1996. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics 15*, 1 (Jan.), 1–36.

Slusallek, P., and Siedel, H.-P. 1995. Vision - an architecture for global illumination calculations. *IEEE Transactions on Visualization and Computer Graphics 1*, 1, 77–96.

Tabellion, E., and Lamorlette, A. 2004. An approximate global illumination system for computer generated films. *ACM Transactions on Graphics 23*, 3 (Aug.), 469–476.

Unger, J., Wenger, A., Hawkins, T., Gardner, A., and Debevec, P. 2003. Capturing and rendering with incident light fields. In *Eurographics Symposium on Rendering*, 141–149.

Veach, E., and Guibas, L. J. 1995. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of SIGGRAPH*, 419–428.