Distribution Ray Tracing

Brian Curless
CSE 557
Fall 2013

James L. Hasiya, "The Rendering Equation."


"Distributed Ray Tracing," Computer Graphics
Carpenter.

Robert L. Cook, Thomas Porter, Luren
Academic Press, 1989. [In the lab.]
A. Glassner, An Introduction to Ray Tracing.

Further Reading:

Shihley, 13.11, 14.1-14.3

Required:
Mirror-like fashion...

All of this assumes that inter-reflection behaves in a

Pixel anti-aliasing

\[ \int \frac{1}{xp(x)} \]

Pixel anti-aliasing

No anti-aliasing

Pixel anti-aliasing
BRDF's can be quite sophisticated...

Here's a plot with \( \omega_n \) held constant.

Distribtuion Function (BRDF).

This function is called the Bi-directional Reflectance:

\[
\frac{I(\omega_o \rightarrow \omega_i)}{I(\omega_o \rightarrow \omega_n)} = \frac{I(\omega_i \rightarrow \omega_o)}{I(\omega_i \rightarrow \omega_n)}
\]

\( \rho \)

The mapping function \( \gamma \) is often written in terms of \( \omega_n \) and \( \omega_i \):

\[
\gamma = \begin{bmatrix}
\left( \frac{\|\mathbf{L} + \mathbf{I}\|}{\mathbf{L} + \mathbf{I}} \cdot \mathbf{N} \right)^2 + \left( \mathbf{K} \cdot \mathbf{I} \right)^2
\end{bmatrix} \mathbf{L} = \mathbf{1}
\]

Viewing directions:

Illumination model are a mapping from light to viewing directions:

The diffuse+ specular parts of the Blinn-Phong BRDF
\[ u \omega \rho (W \cdot u \omega ) ( ^{1}n_{o} \omega \leftarrow u \omega ) \int ( ^{1}n_{o} \omega )^{H} = ( ^{1}n_{o} \omega ) \int \]

or, written more generally:

\[ \rho (W \cdot L) (\Lambda \leftarrow L) \int (\Lambda )^{H} = (\Lambda ) \int \]

Coming in from all directions, which can be added up:

We can now think of the BRDF as weighting light being reflected. Suppose \( \Lambda = L \) and \( u \omega = ^{1}n_{o} \omega \). That means we can take two equivalent views of

\[ ( u \omega \leftarrow ^{1}n_{o} \omega ) \int = ( ^{1}n_{o} \omega \leftarrow u \omega ) \int \]

BRDF's exhibit Helmholtz reciprocity:

Light reflection with BRDF's
Distributing rays over reflection directions gives:

For example:

and reflection aliasing, because we are under-sampling reflection approximation glossy surfaces introduces a kind of

The mirror-like form of reflection when used to

Simulating gloss and translucency
Reflection anti-aliasing

\[ u \cdot \omega \rho (N \cdot u \omega) \cdot \omega \leftarrow u \cdot \omega \cdot \mathcal{H} (u \cdot \omega) \]
Pixel and Reflection anti-aliasing
Full anti-aliasing... lots of nested integrals!

Computing these integrals is prohibitively expensive, especially after following the rays recursively. We'll look at ways to approximate high-dimensional integrals...
**Quadrature**

Evaluating an integral in this manner is called

$$\left( x \nabla \right) \int \sum_{u} \frac{u}{M} \approx f$$

and the summation becomes:

$$\frac{u}{M} = x \nabla$$

where we have sampled at times at spacing $\Delta x$. If these samples are distributed over an interval $w$, then

$$x \nabla \left( x \nabla \right) \int \sum_{u} \approx f$$

then we can approximate the integral by:

If $f(x)$ is not known analytically, but can be evaluated,

$$xp(x) \int = f$$

function

Let's say we want to compute the integral of a

**Approximating Integrals**
\[ \forall y + [x, y] = y + \forall x \]
\[ \forall [x, y] = y + \forall x \]

We can also show that for independent random variables \( x \) and \( y \):
\[ \forall [x, y] = \forall x \]
\[ \forall [x, y] = \forall x \]

\[ \forall [x, y] = \forall x \]
\[ \forall [x, y] = \forall x \]

Recall some of the "rules" of random variables:

- A function (non-negative, integrates to unity).

As distributed according to \( p(x) \), a probability density function is distributed according to \( p(x) \), a probability density function.

Let's say the position is a random variable, \( x \), which regularly is to distribute the Bernoulli stochastically.

An alternative to distributing the sample positions

A stochastic approach
In fact, the summation is itself another random variable. What is its expected value and variance? 

\[ \sum_{i=1}^{u} \frac{X_i}{u} \approx \mathbb{E}[X] \]

Samples:

We can now approximate \( \mathbb{E}[X] \) as the average of \( u \) samples.

A stochastic approach (con't)
\[ \frac{d}{dx} \mathcal{G}(x) \leq \int f(x) \, dx \]

\[ \mathcal{F}(x) = \int \left. \mathcal{G}(x) \right|_{x}^{\infty} \, dx \]

\[ \mathcal{E} = \int x \mathcal{P}(x) \, dx = \int (X \delta) \, dx \]

Getting back to our original problem of estimating an integral, can we choose \( \mathcal{G}(x) \) so that:

\[ x \mathcal{P}(x) d(x) \delta \int = [\mathcal{E}] \]

The expected value is:

\[ (X \delta) \]

Suppose now we have a function of a random variable, \( \mathcal{P}(x) \):

Integrals as expected values
and/or functions are under our control here?

We want a low variance estimate. What variables

\[
\left[ \left( \frac{P(x)}{f(x)} \right)^\frac{1}{2} u \right]^2 + \frac{1}{n} \left[ \sum_{i=1}^{n} \left( \frac{P(x)}{f(x)} \right)^\frac{1}{2} u_i \right]^2 = \left[ \frac{\int P(x) dX}{\int f(x) dX} \right] \approx \left[ \frac{\sum_{i=1}^{n} u_i}{\int f(x) dX} \right] \]

What is the variance of the estimate?

This procedure is known as Monte Carlo Integration.

\[
I = \left[ \int g(x) \right] \approx \sum_{i=1}^{n} \frac{1}{f(X)} u_i \approx f
\]

Estimate the integral as:

Thus, given a set of samples positions, \( x \), we can

Monte Carlo Integration
Choosing \( x \) from a uniform distribution:

with uniform sampling over an interval of width \( w \) (i.e.,

Suppose we now try to estimate the integral of \( f(x) \):

\[
\int_{-w}^{w} f(x) \, dx
\]

Suppose that the unknown function we are

Uniform sampling
This approach is called importance sampling. Will be large and choose \( p(x) \) based on those heuristics. Alternatively, we can use heuristics to guess where \( f(x) \) goes.

\[
\int f(x) g(x) \, dx = 1 \\
\int f(x) \, dx = 1 \\
Q = \left[ \frac{x}{f(x)} \right] \\
\frac{1}{f(x)} = P(x) = \frac{g(x)}{f(x)}
\]

Why don't we just do that?

A better approach, if \( f(x) \) is non-negative, would be to

Importance sampling
Then choose a sample from each bin. You can break it up into bins of equal probability. The idea is that, given your probability function:

stratified sampling

An improvement on importance sampling is stratified sampling.
\[ \left( \mathcal{d} \right) \cdot \frac{1}{\mathcal{u}} \sum_{i} u_{i} \cdot \mathcal{d}_{i} = \] 

Substituting back in:

\[ \left( \mathcal{d} \right) \cdot \frac{1}{\mathcal{u}} \sum_{i} u_{i} = \mathcal{d} \] 

Secondary rays:

For a given primary ray, its intensity depends on:

\[ \left( \mathcal{d} \right) \cdot \frac{1}{\mathcal{u}} = \mathcal{d} \] 

Ray's:
The intensity at a pixel is the sum over the primary rays:

We can think of this problem in terms of enumerated rays:

**Summing over Ray Paths**
Solution: Choose a small number of "good" paths.

Problem: Too expensive to sum over all paths.

So, we can see that ray tracing is a way to approximate a complex nested integral with a summation over ray paths (of arbitrary length).

Each triple \(ijkl\) corresponds to a ray path:

\[
\sum_{l} \sum_{j} \sum_{i} \sum_{k} u_l = \int \text{d}x \text{d}z \text{d}y \text{d}t
\]

We can incorporate tertiary rays next:

Summing over ray paths
We can visualize the span of rays we want to integrate over, within a pixel:

It - for purposes of illustration - as follows:

Let’s return to the glossy reflection model, and modify the model revisited...
Thus, we render with anti-aliasing as follows:

Reflection: tracing replaces the glossy reflection with mirror

Returning to the reflection example, Whitted Ray

Whitted Ray Tracing
Monte Carlo path tracing

Let's return to our original (simplified) glossy reflection model:

An alternative way to follow rays is by making random decisions along the way — a.k.a., Monte Carlo path tracing. If we distribute rays uniformly over pixels and reflection directions, we get:
Importance sampling:

The problem is that lots of samples are “wasted.”

Using again our glossy reflection model:

Let’s now randomly choose rays, but according to a probability that favors more important reflection directions, i.e., use importance sampling.
stratified sampling

Now let's restrict our randomness to within these zones, i.e. use stratified sampling. We can improve on this by splitting together. We still have a problem that rays may be cumped.

reflection into zones:
Tends to be less objectionable than aliasing artifacts. Introduce the solution (slightly granular images). This noise sampling patterns is that they actually inject noise One interesting side effect of these stochastic

called a jittered sampling pattern.
The jittered pattern on the right is also sometimes

![Diagram showing stratified vs. random sampling patterns.](image)

2D pixel (here 16 rays/pixel): Here we see pure uniform vs. stratified sampling over a
This approach was originally called "distributed Ray

<table>
<thead>
<tr>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera lens area</td>
</tr>
<tr>
<td>Light source area</td>
</tr>
<tr>
<td>Reflections and reflections</td>
</tr>
</tbody>
</table>

Sample:

- Provides additional effects by distributing rays to
- Replaces aliasing artifacts with noise.
- Uses non-uniform (interpolated) samples.

This method is called distribution ray tracing [Cook84].

Distribution Ray Tracing
A typical choice is numSubPixels = 5*5.

```
end function
end for
i:=(i’+1)%numSubPixels
end for
i:=(i’+1)%numSubPixels + ItraceRay(scene, p, d, id)
(d - s) ÷ normalize(d) ÷ p ÷ color ÷ pixelToWorld(itter(i’, id))
for each sub-pixel id in (i’)
do
0 ÷
for each pixel (i’, j’) in image do
function ItraceImage(scene):

pixel rays.
Each pixel records the average color of it’s itered sub-
ItraceImage() looks basically the same, except now
```

DRF Pseudocode
end function

return | → |

R' → | intersectedirection(N', p', material id)

shade( ) → |

function traceRay(scene', p', d', id):

opaque glossy surfaces:

Now consider traceRay(), modified to handle (only)

DRT pseudocode (cont'd)
(Quasi-Monte Carlo)

Pre-sampling glossy reflections
Distributing rays over light source area gives:

- Shadows
- Umbra
- Penumbra
- Occluder
- Light
- Surface
Q: How would we reduce blur?

In 3D, we can visualize the blur induced by the pinhole:

Aristotle'sCamera - "Camera obscura" - known to

The Pinhole Camera
Q: What happens as we continue to shrink the aperture?

Shrinking the pinhole
Shrinking the pinhole, cont'd

Diffraction

[Images of diffraction patterns with labels]
From the viewer:
We can equivalently turn this around by following rays:

Image plane:
We can think in terms of light heading toward the

The pinhole cameras revisited
How can we simulate a pinhole camera more accurately?

Given this flipped version: The pinhole camera, revisited.
where \( f \) is the focal length of the lens.

\[
\frac{1}{f} = \frac{1}{p} + \frac{1}{i}
\]

For a "thin" lens, we can approximately calculate where an object point will be in focus using the Gaussian lens formula.

"Lenses focus a bundle of rays to one point => can have larger apertures. Keep the image in focus. Pinhole cameras in the real world require small apertures to..."
The **depth of field** is a measure of how far from the object plane points can be before appearing "too blurry." Lenses do have some limitations. The most noticeable is the out of focus.
At in-focus depth (scaled up and positioned Image plane)

Aperture

The ideal center of projection: and then distributing rays over the aperture (instead of at the in-focus location, in front of the finite aperture.

We can model this as simply placing our Image plane.

Plane in focus

Image plane

Lens

Aperture

The in-focus plane:

Consider how rays flow between the Image plane and Simulating depth of field
Simulating depth of field, cont'd
In general, you can trace rays through a scene and keep track of their ids to handle all of these effects:
Distributing rays over time gives:

\[ \text{DRT to simulate motion blur} \]