Basics of physically-based rendering

Conference Paper · November 2012

DOI: 10.1145/2407783.2407785

4 authors, including:

Peter Shirley
NVIDIA
233 PUBLICATIONS **13,769** CITATIONS

SEE PROFILE

Basics of Physically-based Rendering

Peter Shirley R. Keith Morley Peter-Pike Sloan Chris Wyman
NVIDIA

1 Introduction

These are notes on the underlying fundamentals of *physically-based rendering* prepared for SIGGRAPH Asia 2012. When the first author started in computer graphics there was no real internet, engineering was funded by the cold war rather than consumer spending, and a ray tracer took days or weeks to run. SIGGRAPH course notes were just as different: they were the source of most understandable practical information about rendering. Now there are many good books (some evolved from SIGGRAPH course notes), hundreds of good web pages dense with information, and thousands of bad web pages. The problem has changed from a lack of information to *too much* information of variable quality! Thus these course notes are meant as a trail guide through the literature rather than another detailed discussion.

First, there is an important issue discussed only rarely but understood by almost all rendering people. Physically-based rendering usually has one of three very different goals, and the endeavor is very different for each of them:

predictive rendering where the results have to match the scene if you really build it. Used in design and simulation applications.

plausible rendering where it needs to look plausibly real, but can be wrong. Used in entertainment and visualization applications.

visually rich rendering where it needs the visual richness of reality, but can be highly stylized.

We will stick mainly with the fundamentals of predictive rendering in these notes. By fundamentals we mean mainly physics and math, as opposed to algorithms. Predictive rendering is not more important or better than the other two types, but like learning to draw realistically before becoming a stylized artist, learning the predictive arts help form a firm foundation before learning the art of breaking rules.

2 Reading

There are finally several serious rendering books on the market. If you are serious about rendering as a career, buy them all. But start with the most recent one: *Physically Based Rendering* [9]. Also, you'll need to deal with images, and this recent book is the place to start: *High Dynamic Range Imaging* [10].

Once you get through those books, start looking at what is being taught at various universities. Many people have their courses online. Another great resource is realtimerendering.com. Despite its name, it is probably also the best physically based rendering web site as well. Available there is Glassner's tome [3] which is fundamental and thus still quite useful including great explanations on physics, radiometry, math, and algorithms.

After that, hit the papers. This is much easier than it used to be due to sites such as Google Scholar. A key feature of scholar is that it shows you all papers that have referenced a paper. So you can easily work forwards in time in the literature hunt. Also, it gives the number of citations, which gives you a good idea of what everybody is reading. You will get an edge in your knowledge by reading good papers nobody else reads, but it's good to be culturally literate and read the "big" ones.

3 Light

The basic rendering problem is given a geometric model with material properties, produce an image. An image usually means assuming some sort of camera model. In graphics we usually make a pretty simple model of the camera as most applications are more about what something "looks like" than what a specific photograph of it looks like. But implicitly this usually means some sort of idealized camera like we wish we had in real life. Ironically the cameras lack of flaws can itself make the image look unrealistic because we are used to the effects of imperfections of real cameras showing up in an image.

A camera usually has some lens system and some sensor. The sensor itself has sensor elements often called pixels, and these pixels respond differently to different directions and wavelengths of light. This immediately gets at a central problem in rendering: how do we think of and measure light? That usually leads to a discussion of *radiometry* which gives us the analogs of length and mass but for light. The authors view radiometry as a necessary evil, but we always do what we can to avoid it altogether, and when pressed, we don't understand its details very well.

What we do understand pretty well is what is often called a *photon-like particle* or *photon* for short. Graphics people almost never mean some weird quantum-mechanical wave packet when they say "photon". They mean a very very small ball-bearing like energy packet that carries some amount of energy at some wavelength (or collection of wavelengths). This is sometimes called the *corpuscular* model of light in physics. For the rest of these notes we assume the photon has a single wavelength, some amount

of energy, and perhaps linear polarization. It also has a position at a given time and a direction of propagation, so in sum:

$$\text{photon} = \{\text{energy}, \text{wavelength}, \text{polarization}, \text{time}, \text{position}, \text{direction}\}$$

Note again this is a simplification of a real photon, and may carry much more energy than a real photon. It's probably more appropriate to call it a particle (or ball bering!), so keep in mind this is *not* a photon as is meant in physics.

3.1 Radiance

Now radiometry is how we take a bunch of those photons each with infinitesimal energy, and pretend it's a big continuous function. We could define a density field over all space, directions, times, and wavelengths, and we would get *radiance*, or more technically *spectral radiance*, but in practice people almost never bother to say "spectral". The function defining radiance everywhere is often called the *plenoptic function*. Sometimes linear polarization is added as a dimension, and sometimes full polarization with phase as well. There are many discussions of radiance in the literature, and all of them, including the ones we authored, are to us confusing. We instead emphasize three things about radiance that can be useful:

1. Radiance stays constant along a line.
2. Radiance is very similar to what we think of as "colors" that we see in a direction
3. Sensors measure moments of radiance
4. All radiometric quantities can be derived from radiance

Numbers 1 and 2 are related: photons don't lose energy as they travel, so there is no "inverse square rule" with radiance. That makes it especially nice to deal with in code. Numbers 3 and 4 are also related. Most radiometric quantities are themselves moments of radiance.

It is tempting to define radiance as a function defined over space:

$$\text{radiance} = L(\text{position}, \text{direction}, \text{wavelength}, \text{time})$$

here the symbol L is used for spectral radiance. This convention is semi-common across many fields and is because radiance is a cousin term to the photometric *luminance* and history made L a common choice. A confusing thing about this function is that it tends to be well-defined everywhere except at surfaces where reflection/refraction occurs. For this reason at surfaces we usually have two kinds of radiance:

Incoming radiance, often called *field radiance*, is what an insect on the surface sees coming into it, like an environment map.

Outgoing radiance, often called *surface radiance*, is what the viewer sees looking *at* a surface.

3.2 Moments of radiance

There are all sorts of radiometric quantities that are useful. These are usually *moments* of radiance, by which we mean the integral of radiance after being multiplied by some constant or function. The most common one is *irradiance*, at a point x, often denoted H and is the density of light from all directions hitting a surface:

$$H(x, \lambda) = \int_{\omega \text{ in}} L(x, \omega, \lambda) \cos\theta d\omega$$

Here the integral is over incoming directions ω, and θ is the angle between surface normal and ω.

The analogous outgoing quantity is the radiant exitance where the integral is over *outgoing* directions:

$$E(x, \lambda) = \int_{\omega \text{ out}} L(x, \omega, \lambda) \cos\theta d\omega$$

Sometimes to read papers you need to know what irradiance is, but for the most part you don't ever need to use it, and when you do, the thinking of it in terms of an integral of radiance is usually better.

3.3 Lambertian surfaces

Many surfaces are *matte*. These are usually approximated in graphics and many other disciplines by *Lambertian* surfaces. The Lambertian surface is thermodynamically impossible, but it is computationally and conceptually convenient and looks much like real matte surfaces in practice. It has the basic property that its radiance is linearly proportional to the total amount of light per unit area hitting it, and that radiance is the same for all viewing directions. This is typically the result of bouncing around so much in the surface that the"memory" of where the light comes from is lost. The "total amount of light per unit area hitting it" is the irradiance. The content of proportionality is sadly not the reflectance (sometimes called albedo) R. Instead there is an annoying π:

$$L(\lambda) = \frac{1}{\pi} E = \frac{R(\lambda)}{\pi} H$$

That π is there to enforce energy conservation, but it is the source of much confusion and many bugs.

3.4 Weighted averages of radiance

We can get around the π problem and having to learn the prevalence of confusing radiometry by formulating everything in normalized moments of radiance. These are

also weighted averages of radiance. A normalized moment $M(\lambda)$ at a point is just

$$M(\lambda) = \int_{\omega} m(\omega) L(\omega, \lambda) d\omega$$

with the normalization constraint

$$\int_{\omega} m(\omega) d\omega = 1.$$

We can get rid of the π by folding it into a normalization term

$$\int_{\omega} \frac{\cos\theta}{\pi} d\omega = 1.$$

And radiance of a diffuse surface is then:

$$L(\lambda) = \int_{\omega} \frac{R(\lambda)\cos\theta}{\pi} L(\omega, \lambda) d\omega$$

Note the position x is implicit. It is often left off for convenience (or out of laziness).

4 Rendering equation

The *rendering equation* was introduced to the graphics community in independent papers by Kajiya [7] and by Immelet al. [4] in 1986. It relates the radiance of a surface to the incident radiances coming in. Leaving off the lambdas and the point we are looking the equation is of the form:

$$L(\omega_o) = L_e(\omega_o) + \int_{\omega_i} \rho(\omega_i, \omega_o) L(\omega_i) \cos\theta d\omega_i$$

This basically reads *light comes in from all directions and is either absorbed or redirected*. The *BRDF* (bidirectional reflectance dsitribibution function) ρ depends on surface properties. Here i and o are *incoming* and *outgoing* (which correspond to field and surface radiances) and L_e is the emitted part, like from a light bulb.

4.1 BRDF

Now what is that BRDF function ρ? For example, what is it for a Lambertian surface? From the previous equations we can see it is R/π. It seems the π is something we just can't get away from. Note that the BRDF makes a weighted moment.

There are dozens of BRDFs in the literature, and almost all of them seem sort of the same: a diffuse term, and a specular highlight term. Most have a "fresnel effect" where the specular gets bigger and the diffuse smaller as θ goes near 90 degrees. There is no winner among the models. Pick one. A good choice is Ward's model, but be sure to get the modern versions of his model that have been improved [2].

4.2 Variants of the rendering equation

The rendering equation is written down in many forms. Key is to realize these differences are all notational. For example, Arvo [1] liked to fold the cosine into the measure (i.e., the projected area measure) and name radiance "f" which yields:

$$f(x) = g(x) + \int \rho(x, y) L(y) dy$$

He also found that too complicated and preferred the machinery of operators [1]:

$$f = g + Mf$$

That was all notation. Another set of equations does a change of variables to integrate over surfaces rather than directions. If we note that the solid angle of a small surface dA is

$$d\omega = \frac{dA \cos \theta'}{d^2}$$

where θ' is the angle the surface we see makes relative to us, and d is the distance to that surface, we can perform some algebra to get something like this:

$$L(x, \omega_o) = L_e(x, \omega_o) + \int_{x'} \rho(\omega_i, \omega_o) L(x', x - x') \frac{\cos \theta \cos \theta'}{\|x - x'\|^2} dA'$$

The key thing there is the integration is over all surfaces, and that can sometimes be a computationally better option.

Note that the rendering equation is an *integral equation* in that the function is defined in terms of an integral of itself. Many new algorithms attack the equation in its integral form where it is an integral over paths, so it is not "recursive".

4.3 Reversability

A key characteristic of the rendering equation is that it is *self-adjoint*. This means reversible, in the sense that we can reverse which way we send the light. A path tracer and a photon tracer are largely the same. This is one reason the BRDF formulation is good: it is reciprocal meaning we can exchange the order of incoming and outgoing directions and it is the same. There are some neat real world implications of this [11].

5 Smoke and fog

Thus far we've discussed surfaces. But the rendering equation, as is, does not apply to smoke and fog. The key thing is light doesn't just scatter at surfaces, but can scatter anywhere along a ray. And the BRDF does not apply; instead the *phase function* p

describes light scattering in a volume. The photon interpretation makes this straightforward. Along the light ray, there is some small probability the photon hits a particle and is absorbed or scattered. The probability dP of hitting a particle in a short distance dL is:

$$dP = a dL$$

where a is proportional to particle density, and this can vary with wavelength. For example, for air $a \propto 1/\lambda^4$ which is why the sky is blue. If there is an interaction, the light is scattered with probability Λ. That Λ is called many things but is basically a reflectance or albedo. For coal-dust it is near zero, and for steam it is near one. It varies with wavelength.

If the light interacts with a particle, and it is not absorbed then it scatters. It scatters with a distribution proportional to the (usually normalized) phase function $p(\omega_o)$. In practice, almost everybody uses either the isotropic constant $p = 1/(4\pi)$ or the Henyey-Greenstein function which is analogous to the Phong lighting model: it's a convenient stretchable lump.

6 Cameras and luminaries

Camera models can be very realistic and sophisticated but people are often sloppy here is we usually don't want a particular camera model but rather a "how a person sees it" perhaps with some lens effects thrown in. What people often store is the tristimulous XYZ so you don't lose the overall brightnesses of a scene. For example, is it night or day?

Luminaires, or light emitting objects, are in practice a pain. Almost nobody models a clear glass bulb around a blackbody filament as it would break most renderers and it is overkill for most applications. Almost all the time you can get away with a surface that has a uniform illumination coming from it, or some phony-like directional pattern.

If you have to do this "right" for real luminaries for architectural models, you are into the field of *illumination engineering* which has standards for luminaire specification. If you find yourself in this situation, there are two people that have worked extensively in both that area and computer graphics: Greg Ward-Larsen, and Ian Ashdown. Reading their works will get you most of the way there.

7 Mathematical tools

One of the appeals of physically based rendering is you get to use cool math. We give a brief summary here.

7.1 Monte Carlo and QMC

If you have an integral you can pick random points in the domain, evaluate the integrand at those points, and get an answer by an averaging process of those evaluations. For one random sample x_i with probability density function (pdf) p that gives:

$$I = \int f(x)dx \approx f(x_i)/p(x_i)$$

They way to think of the p correction factor is places where you densely sample x_i you need to count each sample less. The constants all are embedded in the normalized nature of pdfs.

For improve your result, simply average over a larger set of random x_i.

If, and this is an important if, the dimensionality of the integral is low (like 1-6), you can get better answers (less error) if you use non-random samples. One option is jittered or stratified samples. The other is semi-regular or QMC (quasi-Monte Carlo) samples. Note that the choice of p lowers your time constant whereas good samples lowers your time complexity (see Mitchell [8] for a nice discussion of this in 2D).

The catch is *the curse of dimensionality* which says that for high dimensional integrals (like 10 bounces in a cloud) that "good" samples don't help.

7.2 Density estimation and Metropolis

Density estimation is common in rendering, with *photon mapping* [6] being the prime example. Here a set of points has the right density (like the energy on a surface for "baking" where results are stored over the model for later lookup) and density estimation estimates the underlying density that generated those points. Recent work on photon beams [5] has added legs to this approach and emphasizes that being comfortable with math on paths is a good thing. Key here is we have some function $f(x) = kp(x)$ where p is a pdf (so non-negative with integral one). We have a bunch of samples that are distributed according to p. So we somehow blur the samples and try to get a reasonable approximation to p.

Metropolis is a way to take points and mutate them so their density is proportional to some function f. For the rendering equation, these points are *paths*. What makes this confusing is you need a measure defined over paths. And the paths are of different lengths. So a "point" in path space is a sequence of 3D points along a path, and this might be a 3 segment path or a 17 segment path. Bite the bullet and read some applied measure theory. There are no half-measures (pun intended) on learning this.

In a more general sense, Metropolis takes any set of points and any function $f(x) = kp(x)$ even if k is unknown, and uses mutations to "evolve" a set of points with density p. The constant k must be estimated by another means.

7.3 Functions on the sphere

Many of the functions used in rendering are represented on the sphere or hemisphere: BRDF's, phase functions for scattering, incident or outgoing radiance on a surface or in a volume, and distant environmental lighting. There are a several mathematical representations that are commonly used in the literature that make different trade-offs and they will be briefly discussed here. A nice technical report [13] that analyzes the sampling efficiency of various parameterizations of the sphere is worth looking at.

Spherical Harmonics (SH) are the spherical analog to the Fourier basis functions, while the general form in the literature represents complex functions, in computer graphics the real form of the basis functions are used. See [12] for detailed equations. The Spherical Harmonics form an orthogonal basis[1], which leads to an efficient way to both project signals into the basis and compute the integral of a product of two functions expressed in the basis. They are efficient at storing smooth signals, but can suffer from ringing artifacts and require a large number of coefficients to represent high frequency signals. The most important property they have is that the basis is closed under rotation, and there are efficient ways to rotate a signal just using the basis coefficients. This eliminates temporal artifacts that can happen with other basis. A special subset of the Spherical Harmonics, called Zonal Harmonics, represents functions that have circular symmetry around the Z axis. These are often used for phase functions in scattering, and have even more efficient rotation formula.

Wavelets, often on cube maps, perform well at capturing signals at multiple-frequencies, including the high frequencies that Spherical Harmonics struggle with. They have efficient formula for projection, evaluation and often generate sparse results, however there are a couple of downsides to using them. The differential solid angle at the center of a face is more than 5 times larger than a corner. This means that frequencies can leak in if you rotate a signal unless you carefully pre-filter based on the worst case solid angle. Translating a signal can also cause significant changes in the number of non-zero coefficients that are required to represent a signal at a given accuracy level, and flickering can occur when thresholding is used. They are still the basis of choice if you want to represent general all-frequency signals.

Spherical Radial Basis Functions (SRBFs), most often Gaussians, are trivial to rotate, have efficient formula for computing the integral of products and can also represent various frequencies. The one downside they have is that projecting a signal is much more difficult [14], since it involves non-linear optimization unlike wavelets or Spherical Harmonics.

8 Writing a renderer

There are a few "flavors" of rendering algorithm. Our breakdown of the big flavors is:

[1] $\int B_i(x)B_j(x)dx = \delta_{ij}$

path tracing: Sending Monte Carlo rays from the eye.

photon tracing: Sending Monte Carlo rays from the light.

bidirectional path tracing: Do path and photon tracing and do match-making between partial paths from each direction to get full light-to-eye paths.

Metropolis: Start with a set of seeds from any of these approaches (bidirectional is popular) and then mutate with Metropolis.

Radiosity: Do a full finite element solution and project the results to the screen.

There are many variants. For example, a photon tracer is open to blotchy, but one bounce of path tracing smooths this out. QMC is often used instead of random sampling. Image space blurring can be used to smooth results. Photon or path tracing can be run with textures on diffuse surfaces and baked into textures, or illumination can be baked into a few coefficients of spherical harmonics. The terms "backward" and "forward" are often used but be aware they are confusing, and most people thus avoid them today.

8.1 Debugging

Debugging physically based methods is difficult. Wrong pictures often look plausible. The first debugging case people often use is "the furnace". All surfaces are diffuse emitters with emitted radiance E. All surfaces are diffuse reflectors with albedo R. No matter what the geometric configuration, the final radiance is:

$$L = E(`1 + R + R^2 + R^3 + \cdots) = \frac{E}{1 - R}$$

Another good case is Cornell has the "Cornell Box" online with data and photos.

Finally, find a rendering buddy to compare images with!

References

[1] James Arvo. The role of functional analysis in global illumination. In *Rendering Techniques*, pages 115–126, 1995.

[2] D. Geisler-Moroder and A. Dür. A new ward brdf model with bounded albedo. In *Proceedings of Eurographics (Computer Graphics Forum)*, pages 1391–1398, 2010.

[3] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[4] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *Proceedings of the 13th annual conference*

on Computer graphics and interactive techniques, SIGGRAPH, pages 133–142, 1986.

[5] Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Trans. Graph.*, 30(1):5:1–5:19, 2011.

[6] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[7] James T. Kajiya. The rendering equation. In *SIGGRAPH*, pages 143–150, 1986.

[8] Don P. Mitchell. Consequences of stratified sampling in graphics. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 277–280, 1996.

[9] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[10] E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward, and K. Myszkowski. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Elsevier Science, 2010.

[11] Pradeep Sen, Billy Chen, Gaurav Garg, Stephen Marschner, Mark Horowitz, Marc Levoy, and Hendrik Lensch. Dual photography. In *Proceedings of SIGGRAPH)*, page 745 ÐÐ 755, 2005.

[12] P.P. Sloan. Stupid spherical harmonics (sh) tricks. In *Game Developers Conference*, 2008.

[13] J. Snyder and D. Mitchell. Sampling-efficient mapping of spherical images. Technical report, Microsoft Research, 2001.

[14] Yu-Ting Tsai and Zen-Chung Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph.*, 25(3):967–976, July 2006.